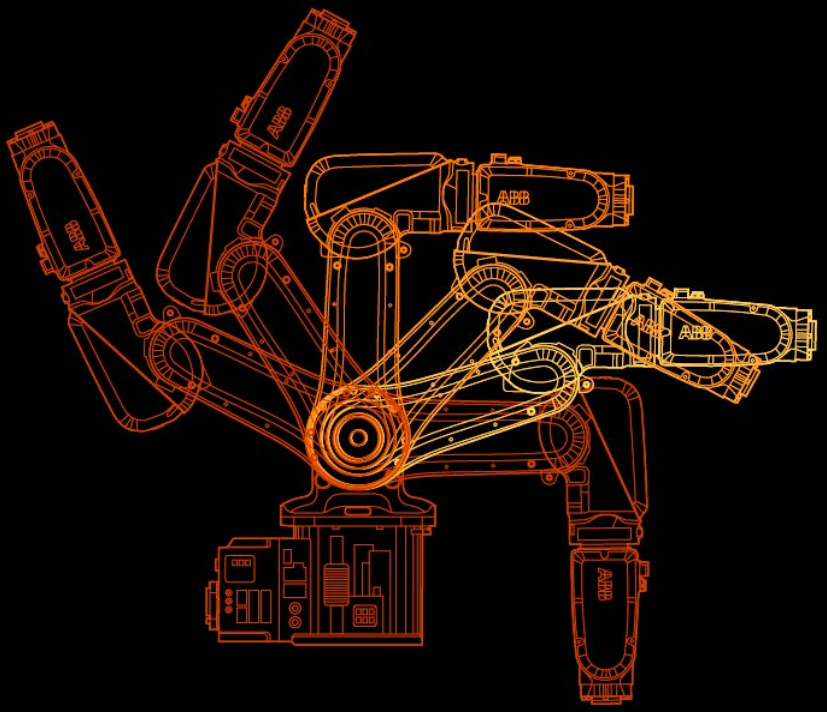


Manual de Usuario del Robot IRB120

Power and productivity
for a better world™

Manual Usuario Robot IRB120

Guía de Usuario

Tipos de Datos

Instrucciones

Funciones

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 permite la reproducción o la copia de este manual o cualquiera de sus partes indicando su origen.

Asea Brown Boveri SA
DM/Robotics
Área Formación
08192 Sant Quirze del Vallès

Manual Usuario Robot IRB120

Guía de Usuario

Tipos de Datos

Instrucciones

Funciones

Contenido

1. Seguridad

1.1. Acerca de este capítulo

Este capítulo describe los principios y procedimientos de seguridad que debe tener en cuenta al utilizar un robot o un sistema de robots.

1.2. Peligros asociados al sistema robot

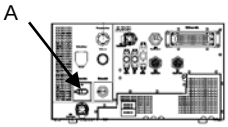
1.2.1. Peligros eléctricos

Descripción

El trabajo con tensiones elevadas es potencialmente letal. Las personas sometidas a altas tensiones pueden sufrir paros cardiacos, quemaduras u otras lesiones graves. Para evitar estos riesgos, no continúe con el trabajo sin eliminar el peligro de la forma detallada a continuación.

Manera de actuar

Paso	Acción	Figura
1	Apague el interruptor principal del controlador (A)	



1.2.2. Peligros por movimientos a alta velocidad

Descripción

Cualquier manipulador en movimiento es una máquina potencialmente letal. Durante el funcionamiento del manipulador, éste puede realizar movimientos inesperados y, en ocasiones, aparentemente irracionales. Sin embargo, todos los movimientos se realizan con una fuerza extraordinaria y pueden causar lesiones graves al personal y/o daños en los equipos que se encuentren cerca del área de trabajo del manipulador.

Manera de actuar

Paso	Acción	Información
1	Antes de intentar hacer funcionar el manipulador, asegúrese de que todos los *equipos de paro de emergencia* estén instalados y conectados correctamente. Asegúrese de que no haya ninguna persona dentro del área de trabajo del manipulador	Equipos de paro de emergencia, como puertas, trampillas de contacto, barreras fotoeléctricas, etc.

1.3. Sistemas de seguridad del robot

1.3.1. ¿Qué es un paro de emergencia?

Definición de paro de emergencia

Un paro de emergencia es un estado que tiene prioridad sobre cualquier otro control del robot, desconecta la alimentación de accionamientos de los motores del robot, detiene todas las partes móviles y desconecta la alimentación de cualquier función potencialmente peligrosa controlada por el sistema de robot.

Un estado de paro de emergencia significa que se desconecta toda la alimentación del robot excepto la de los circuitos de liberación manual de frenos. Debe realizar un procedimiento de recuperación para poder volver al funcionamiento normal.

Dispositivos de paro de emergencia

Los sistemas de robot cuentan con varios dispositivos de paro de emergencia que pueden ser accionados para disparar un paro de emergencia. Existen pulsadores de paro de emergencia tanto en la Unidad de Programación como en el módulo de control. Su robot también puede contar con otros elementos de paro de emergencia. Consulte la documentación de su centro de producción o su célula para saber cómo está configurado su sistema de robot.

Pulsador de paro de emergencia en la Unidad de Programación y en el Controlador

Un paro de emergencia es un estado que tiene prioridad sobre cualquier otro control del manipulador, desconecta la alimentación de accionamiento de los motores del manipulador, detiene todas las partes móviles y desconecta la alimentación de cualquier función potencialmente peligrosa controlada por el sistema de manipulador.

Un estado de paro de emergencia significa que se desconecta toda la alimentación del manipulador excepto la de los circuitos de liberación manual de frenos. Debe realizar un procedimiento de recuperación, es decir, restablecer el pulsador de paro de emergencia y presionar el botón Motors ON, para poder volver al funcionamiento normal.

Los sistemas de manipulador cuentan con varios dispositivos de paro de emergencia que pueden ser accionados para disparar un paro de emergencia. Existen pulsadores de paro de emergencia tanto en el FlexPendant como en el Controlador.

1.3.2. Recuperación de paros de emergencia

Descripción general

La recuperación después de un paro de emergencia es un procedimiento sencillo pero importante. Este procedimiento garantiza que el sistema de robot no sea puesto de nuevo en producción sin antes eliminar la situación peligrosa.

Rearme del bloqueo de los pulsadores de paro de emergencia

Todos los dispositivos de paro de emergencia basados en un pulsador tienen una función de bloqueo que debe ser liberada para poder eliminar el estado de paro de emergencia del dispositivo.

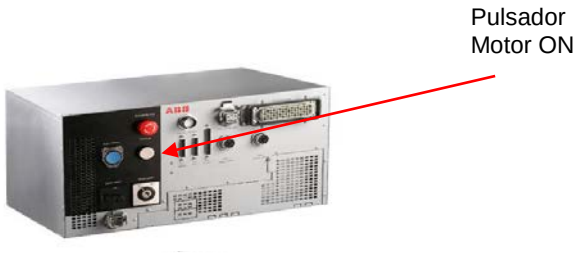
En muchos casos, esto se hace girando el pulsador de la forma marcada en el mismo, pero también hay dispositivos en los que es necesario tirar del pulsador para liberar el bloqueo.

Recuperación de paros de emergencia

	Acción
1	Asegúrese de que ya no exista la situación peligrosa que ha dado lugar al estado de paro de emergencia.
2	Busque y rearme el o los dispositivos que dieron lugar al estado de paro de emergencia.
3	Confirme el evento de paro de emergencia en los mensajes de errores.
4	Presione el pulsador Motor ON para rearmar el sistema desde el estado de paro de emergencia.

El pulsador Motor ON

El pulsador Motor ON está situado en el panel de control del controlador. En algunas instalaciones, el pulsador puede estar en un panel de control remoto en algún lugar diferente del controlador.



1.3.3. ¿Qué es un paro de seguridad?

Definición de paro de seguridad

Un paro de seguridad significa que sólo se desconecta la alimentación de los motores del robot. No cuenta con ningún procedimiento de recuperación. Para la recuperación en caso de un paro de seguridad, sólo es necesario restablecer la alimentación de los motores. El paro de seguridad se conoce también como Paro de Protección.

Tipos de paro de seguridad

Los paros de seguridad se activan a través de entradas de señales especiales en el controlador. Estas entradas están destinadas para su uso con dispositivos de seguridad tales como barreras fotoeléctricas, interruptores de interbloqueo en las puertas….

Paro de Seguridad	Descripción
Paro Automático (AS)	Desconecta la alimentación de los accionamientos cuando el sistema trabaja en modo Automático EXCLUSIVAMENTE.
Paro General (GS)	Desconecta la alimentación de los accionamientos cuando el sistema trabaja tanto en modo Automático como en Manual.
Paro Superior (SS)	Desconecta la alimentación de los accionamientos cuando el sistema trabaja tanto en modo Automático como en Manual. Pensado para equipos externos.

1.3.4. ¿En qué consiste la protección?

Definición

Se conoce como "protección" al conjunto de medidas basadas en el uso de elementos protectores que evitan la exposición de las personas a los riesgos que no pueden ser eliminados razonablemente ni reducidos suficientemente en el diseño.

Espacio protegido

El espacio protegido es el espacio que queda englobado por los elementos de protección. Por ejemplo, una célula de robot está protegida por la puerta de la célula y su dispositivo de interbloqueo.

Dispositivos de interbloqueo

Cada elemento de protección presente cuenta con un dispositivo de interbloqueo que, si se acciona, detiene el robot. La puerta de la célula del robot cuenta con un interbloqueo que detiene el robot al abrir la puerta. La única forma de reanudar el funcionamiento es cerrar la puerta.

Mecanismos de protección

Un mecanismo de protección está compuesto por un conjunto de elementos de protección conectados en serie. Cuando se activa un elemento de protección, la cadena se rompe y el funcionamiento de la máquina se detiene, independientemente del estado de los elementos de protección del resto de la cadena.

Supervisión de seguridad

Los mecanismos de paro de emergencia y protección están supervisados, de forma que el controlador puede detectar cualquier fallo y detener el robot hasta que el problema quede resuelto.

1.3.5. Más elementos de seguridad incorporados al robot

Selector de modo

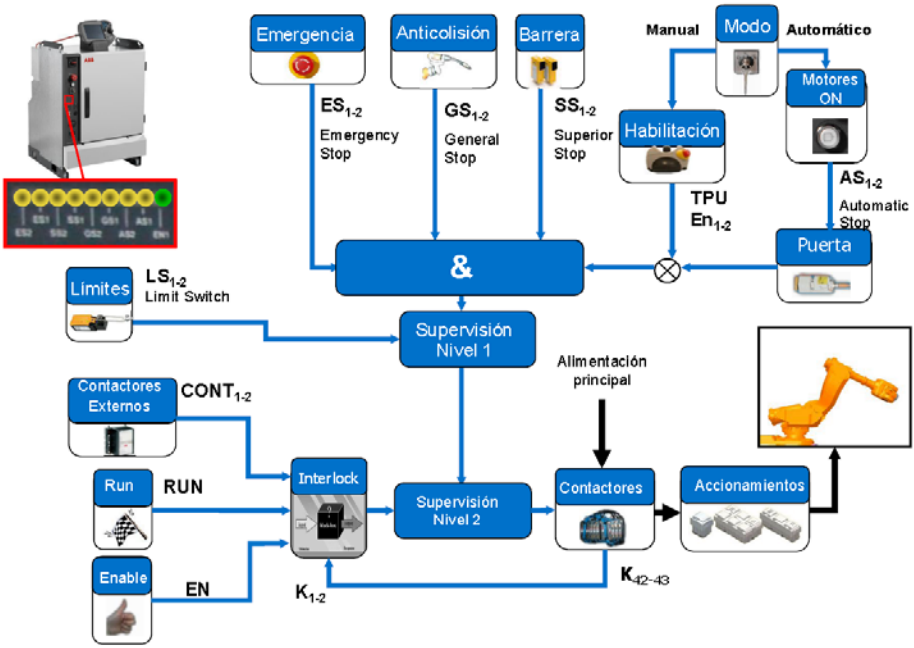
Es un dispositivo que se encuentra en el frontal del armario del controlador. Posee una llave que girándola permite cambiar el modo de funcionamiento del robot. Consulte la sección 2.4

El dispositivo de habilitación

Es una palanca incorporada en el lateral de la Unidad de Programación. En el **modo manual** los motores del robot son activados por el dispositivo. De esta forma, el robot sólo puede moverse siempre y cuando el dispositivo esté presionado.

El dispositivo de habilitación se ha diseñado de forma que sea necesario presionar la palanca sólo hasta la mitad de su recorrido para activar los motores del robot. Tanto en la posición en la que la palanca está presionada al máximo o liberada totalmente, el robot está inmovilizado.

1.3.6. Esquema general de la Cadena de Seguridades



2. Descripción General

2.1. Acerca del capítulo

Este capítulo contiene una descripción general de la unidad de programación, el controlador IRC5 y RobotStudio.

Un sistema de robot IRC5 básico suele componerse de un controlador, la Unidad de Programación y uno o varios robots o unidades mecánicas de otros tipos. En este manual se describe un sistema IRC5 básico.

2.2. ¿Qué es la Unidad de Programación?

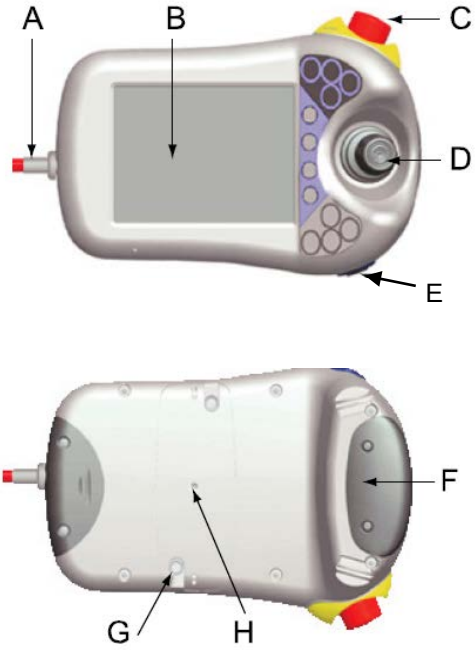
Descripción de la Unidad de Programación

La Unidad de Programación (denominada en ocasiones FPU o TPU) es un dispositivo que maneja muchas de las funciones relacionadas con el uso del sistema de robot, como ejecutar programas, mover el manipulador, crear y editar programas de aplicación, etc.

Se compone de elementos de hardware, como botones y un joystick, y de software y es un ordenador completo en sí. Se conecta al controlador a través de un cable con conector integrado.

Componentes principales

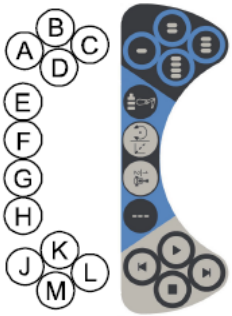
A continuación se enumeran las partes principales de la Unidad de Programación.



A	Cable conexión
B	Pantalla táctil
C	Pulsador de paro de emergencia
D	Joystick
E	Conector USB para los sticks de memoria para guardar o recuperar los programas
F	Dispositivo de habilitación
G	Puntero
H	Pulsador de Reset

Botones de hardware

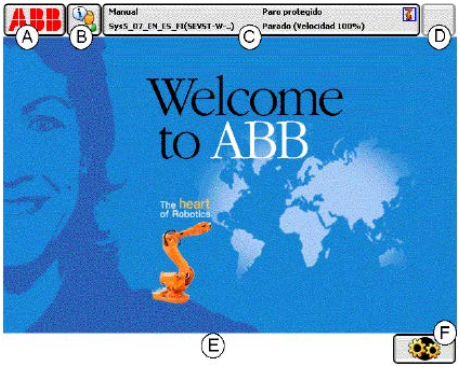
La Unidad de Programación cuenta con varios botones de hardware dedicados. Cuatro de ellos son programables.



A,B,C,D	Teclas programables.
E	Seleccionar una unidad mecánica: Unidad de Robot / Ejes Externos.
F	Botón acceso rápido a selección de movimientos: Reorientación / Lineal,
G	Botón acceso rápido a selección de movimientos: Ejes 1, 2,3 / Ejes 4, 5,6
H	Botón acceso rápido a selección de movimientos: Activar / Desactivar Incrementos.
K	Botón INICIAR. Inicia la ejecución del programa.
J	Botón RETROCEDER un paso. Ejecuta el programa una instrucción hacia atrás.
L	Botón AVANZAR un paso. Ejecuta el programa una instrucción hacia delante.
M	Botón DETENER. Detiene la ejecución del programa.

Elementos de la pantalla táctil

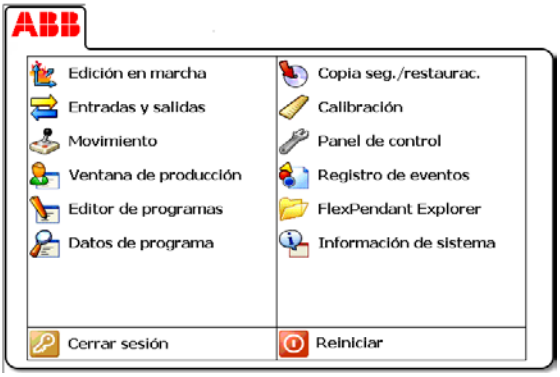
En la figura se muestran los elementos táctiles de la pantalla táctil de la Unidad de Programación.



A	Menú ABB
B	Ventana de operador
C	Barra de estado
D	Botón Cerrar
E	Barra de tareas
F	Menú de configuración rápida

Menú ABB:

Es el menú principal, permite visualizar las ventanas y aplicaciones principales.



Ventana de operador:

La ventana de operador muestra los mensajes del programa. Suelen aparecer cuando el programa necesita alguna respuesta por parte del usuario.

Barra de estado:

La barra de estado muestra información sobre el estado del sistema y permite acceso a los mensajes de error.

Botón Cerrar:

Al tocar el botón Cerrar se cierra la vista o aplicación que esté activa actualmente.

Barra de tareas:

Se pueden abrir varias vistas desde el menú ABB, pero solo se puede trabajar con una de ellas en cada momento. La barra de tareas muestra todas las vistas y aplicaciones abiertas y permite cambiar entre ellas.

Menú de configuración rápida:

El menú de configuración rápida contiene opciones sobre movimientos y ejecución de programas.

2.3. ¿Qué es un controlador IRC5?

Controlador IRC5

El controlador IRC5 contiene todas las funciones necesarias para mover y controlar el robot.

Las variantes básicas del controlador IRC5, son el Single Cabinet y el Compacto.

El controlador tiene dos partes: el módulo de control y el módulo de accionamientos.

- El módulo de control contiene todos los elementos electrónicos de control, como el ordenador principal, las tarjetas de E/S y la unidad de almacenamiento.
- El módulo de accionamiento contiene todos los elementos electrónicos de alimentación que proporcionan la alimentación a los motores del robot. Un módulo de accionamientos IRC5 puede contener los accionamientos de los seis ejes del robot y además dos o tres accionamientos para los ejes externos en función del modelo de robot.

Para controlar más de un robot con un mismo módulo de control (opción MultiMove), es necesario añadir un módulo de accionamientos más por cada robot adicional.

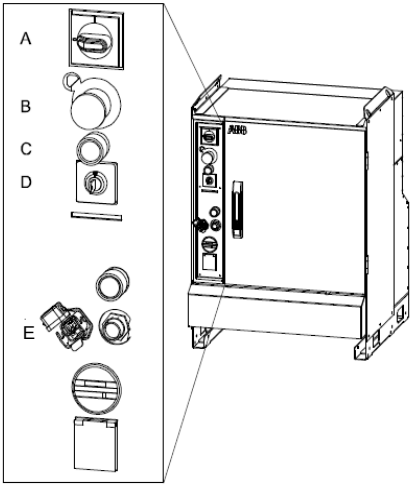
El controlador IRC5 compacto se utiliza con robots pequeños (IRB120, IRB1520, IRB1600, IRB360).

A

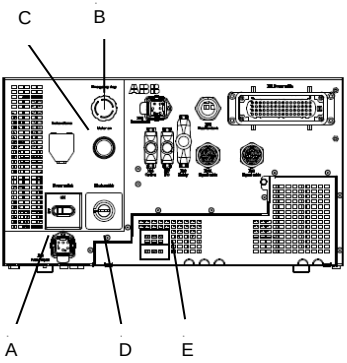
B

A	Armario sencillo *(Single Cabinet)*
B	Armario compacto

2.4. Panel de control



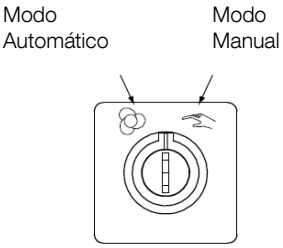
Panel de control
Armario single



Panel de control
Armario Compact

Parte	Descripción	Función
A	Interruptor principal	Interruptor de encendido/apagado para el apagado del sistema.
B	Pulsador paro emergencia	Pulsador de paro del robot cuando hay una emergencia. Desconecta la alimentación de los motores del robot.
C	Pulsador Motor ON	Pulsador petición motores ON y rearmar paro emergencia. Lámpara fija estado de motores ON, intermitente estado de motores OFF.
D	Selector de modo	Selector para trabajar en modo Automático o Manual.
E	Puerto Ethernet	Para conectarse con el sistema para tareas de Servicio, mediante la aplicación RobotStudio en modo Online.

Modos de Operación



Modo de operación	Descripción
Automático	• Modo de producción, se usa para ejecutar los programas. • El robot se moverá a la máxima velocidad programada. • El dispositivo de habilitación está desconectado, de forma que el robot pueda moverse sin intervención humana. • Todos los mecanismos de protección (GS, AS y SS) están activos durante el funcionamiento.
Manual	• Modo de programación. El robot sólo puede moverse a la velocidad reducida de 250 mm/s o menos. • El dispositivo de habilitación está conectado, de forma que necesita ser activado para que el robot pueda moverse. • Sólo el mecanismo de protección del AS se omite durante el funcionamiento.

2.5. ¿Qué es un robot o unidad mecánica?

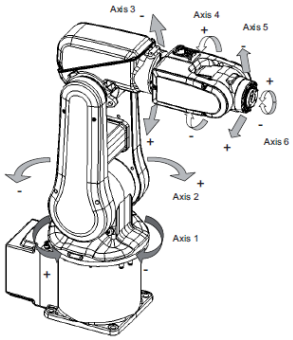
Descripción

El robot o unidad mecánica es el elemento del sistema, encargado de realizar las tareas de manipulación y movimiento para las que ha sido programado.

La configuración mecánica más común es la que tiene forma de brazo articulado y consta de un total de 6 ejes (hay mecánicas con cuatro ejes).

Su base puede ser instalada directamente sobre un elemento de apoyo anclado al suelo (robot fijo), o sobre una unidad mecánica móvil como un TRACK para dotar al robot de más movilidad (robot con desplazamiento sobre eje externo).

En la brida de sujeción del eje 6, es en donde se ubica la herramienta o útil con el que va a realizar las tareas el robot.



2.6. ¿Qué es el RobotStudio?

El RobotStudio es un software que se ejecuta en un PC destinada a la creación, programación y simulación offline de células robotizadas.

También tiene una funcionalidad online, si conectamos el PC al puerto Ethernet del controlador. Permite entre otras posibilidades, la creación, instalación y mantenimiento de Sistema, la programación y edición de los programas.

3. Navegación por la Unidad de Programación

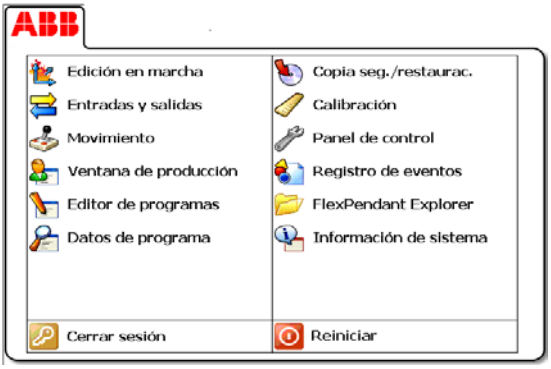
3.1. Acerca de este capítulo

Este capítulo facilitará el trabajo con la FlexPendant de una forma eficiente. Se describen los elementos de la pantalla táctil de la Unidad de Programación.

Todas las vistas del menú ABB, el elemento principal de la navegación, se describen en la descripción general con referencias a más detalles acerca de cómo usar sus funciones.

Además, este capítulo proporciona información acerca de procedimientos básicos, como por ejemplo el uso del teclado en pantalla para introducir texto o números, la forma de desplazarse y ampliar o reducir la imagen gráfica de la pantalla táctil y cómo usar la función de filtrado.

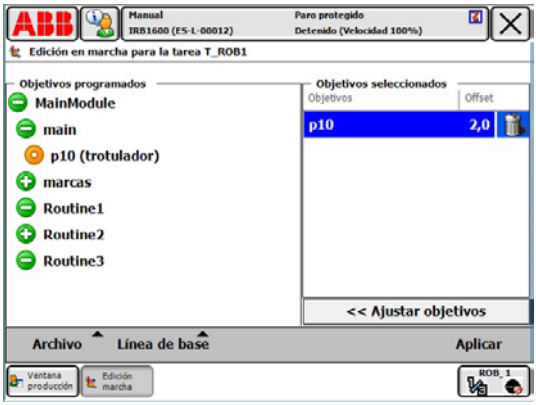
3.2. Menú ABB



3.2.1. HotEdit (Edición en marcha)

HotEdit

HotEdit es una función que permite la edición de posiciones programadas. Puede hacerse mientras el programa se está ejecutando. Es posible cambiar tanto las coordenadas como la orientación. La función de Edición en marcha sólo puede usarse con las posiciones del tipo robtarget.



Funciones disponibles en la Edición en marcha

Objetivos programados	Enumera todas las posiciones con nombre en una vista de árbol. Seleccione una o varias posiciones a ajustar tocando la flecha. Recuerde que si una posición determinada es usada en varios lugares de su programa, cualquier cambio realizado en el offset afectará a todos los lugares en los que se utiliza.
Objetivos seleccionados	Enumera todas las posiciones seleccionadas y su offset actual. Para eliminar una posición de la selección, tóquela y toque la papelera.
Archivo	Guarda y carga selecciones de posiciones a ajustar.

Línea de base	Se utiliza para aplicar o rechazar nuevos valores de offset de la línea de base, que contiene los valores de posición considerados actualmente como valores originales. Cuando esté conforme con su sesión de Edición en marcha y desee guardar los nuevos valores de offset como valores de posición originales, éstos se aplican a la línea de base. Los valores de línea de base anteriores para estas posiciones quedan ahora eliminados y no pueden ser restaurados.
Ajustar objetivos	Muestra valores para su ajuste: Sistema de coordenadas, modo de ajuste e incremento de ajuste. Seleccione sus opciones y utilice los iconos más y menos para especificar el ajuste de los objetivos seleccionados.
Aplicar	Toque Aplicar para aplicar los valores establecidos en la vista Ajustar objetivos. ¡Recuerde que con esto no cambian los valores de línea de base de las posiciones!

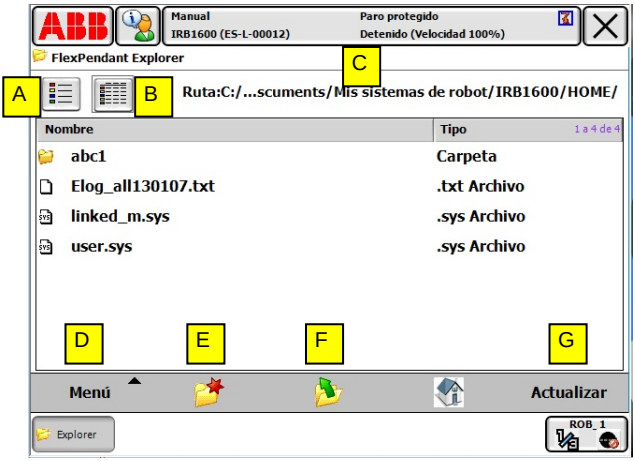
IMPORTANTE !!
Hotedit ofrece funcionalidades avanzadas por lo que debe utilizarse con cuidado. Se debe tener presente que los nuevos valores modificados se utilizará inmediatamente en la ejecución del programa una vez que se ha pulsado la tecla Aplicar

¡NOTA!
Si se ajusta una posición el nuevo valor se usará directamente. Si el ajuste se realiza cerca de los punteros PP o MP, puede resultar difícil de predecir en qué momento tendrá lugar un cambio de programa. Es importante saber en qué parte del programa se encuentra el robot antes de cambiar ningún valor mientras el programa se está ejecutando.

3.2.2. FlexPendant Explorer

Descripción general

FlexPendant Explorer es un administrador de archivos, similar al Explorador de Windows, que permite ver el sistema de archivos del controlador. Usted puede eliminar o trasladar archivos o carpetas, o cambiar su nombre.

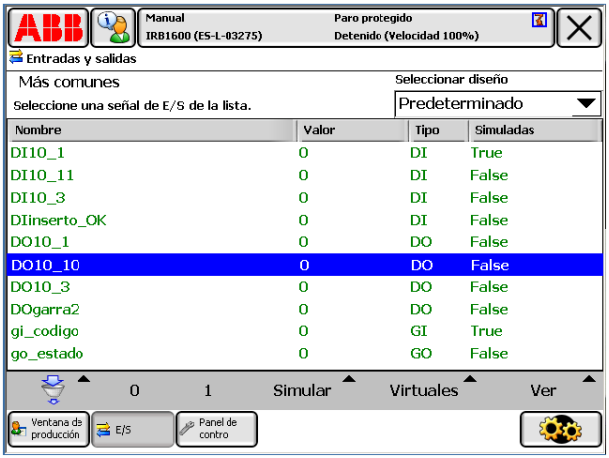


A	Vista Sencilla. Toque para ocultar el tipo en la ventana de archivos
B	Vista detallada. Toque para mostrar el tipo en la ventana de archivos
C	Ruta. Muestra las rutas de las carpetas.
D	Menú. Toque para mostrar las funciones para manejo de archivos
E	Nueva Carpeta. Toque para crear una nueva carpeta dentro de la carpeta actual.
F	Subir un nivel. Toque para pasar a la carpeta superior
G	Actualizar. Toque para actualizar los archivos y carpetas

3.2.3. Entradas y salidas, E/S

Descripción general

Las entradas y salidas, son señales utilizadas en el sistema de robot como por ejemplo la apertura o cierra de una pinza. La señales se configuran utilizando parámetros del sistema.



¿Qué es una señal?

Una señal de E/S es la representación de software lógica de:

- Entradas o salidas que se encuentran en una unidad de E/S de bus de campo que está conectada a un bus de campo dentro del sistema de robot (señal de E/S real).
- Una señal de E/S sin representación en ninguna unidad de E/S de bus de campo (señal de E/S virtual).

Mediante la especificación de una señal de E/S, se crea una representación lógica de la señal de E/S real o virtual. La configuración de la señal de E/S define los parámetros de sistema específicos de la señal de E/S que se usarán para controlar el comportamiento de la misma.

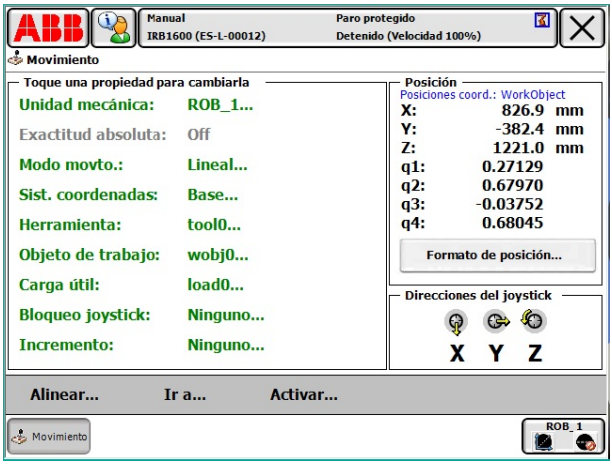
3.2.4. Movimiento

Descripción general

Las opciones para el control de movimiento, visualización de la posición del robot y dirección del joystick se encuentran en la ventana Movimiento. Las opciones más utilizadas están también disponibles en el menú de configuración rápida y en los nuevos pulsadores de acceso rápido a la selección de movimientos e incrementos.

Menú Movimiento

En la figura se muestran las funciones disponibles en el menú Movimiento:

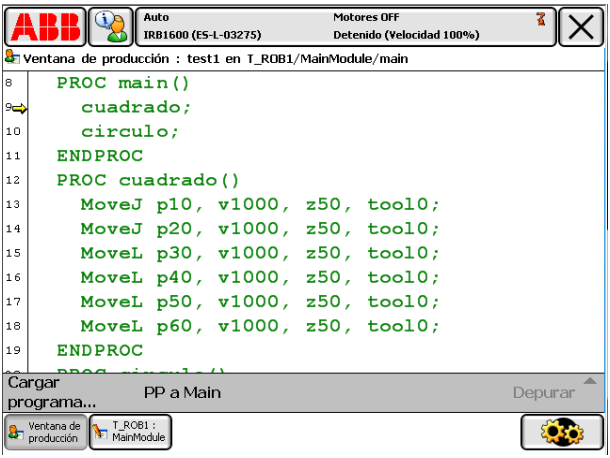


Botón	Función
Unidad Mecánica	Selección de la unidad mecánica activa
Exactitud absoluta	Exactitud absoluta: Off es el valor predeterminado. Si el robot cuenta con esta opción, aparecerá Exactitud absoluta: On
Modo Movto.	Selecciona el modo de movimiento
Sist. coordenadas	Selecciona el sistema de coordenadas
Herramienta	Selecciona la herramienta de trabajo
Objeto de trabajo	Selecciona el objeto de trabajo activo
Carga útil	Selecciona la carga útil activa
Bloqueo de joystick	Selecciona las direcciones de bloqueo del joystick.
Incremento	Selecciona los incrementos de movimiento
Posición	Visualiza la posición del eje en relación con el sistema de coordenadas seleccionado. Si los valores de la posición se muestran en rojo, indica que los valores no pueden ser reales, ya que los contadores de vueltas no están actualizados.
Formato de posición	Selecciona el formato de las posiciones
Direcciones del joystick	Muestra las direcciones actuales del joystick
Alinear	Alinear la herramienta actual con un sistema de coordenadas
Ir a	Mover el robot a una posición o un objetivo determinado.
Activar	Activar la unidad mecánica

3.2.5. Ventana de Producción

Descripción general

La ventana de producción se utiliza para ver el código del programa mientras éste se está ejecutando. Consulte la sección *Ejecución en producción en la página 106.*

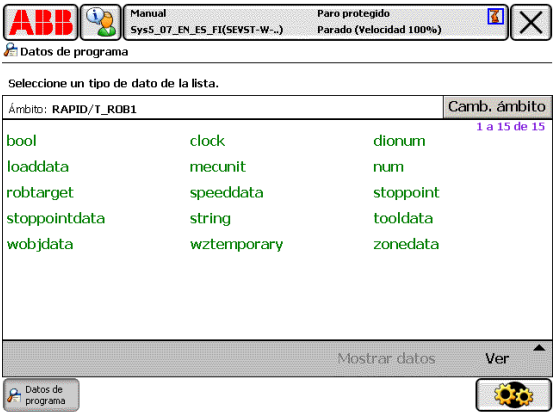


Cargar programa	Cargar un nuevo programa
PP a Main	Mover el puntero de programa al inicio de la rutina Main
Depurar	El menú Depurar sólo está disponible en el modo manual. En esta opción hay disponibles las siguientes opciones: **Modificar posición**, **Mostrar puntero de movimiento** **Mostrar puntero de programa** **Editar programa**

3.2.6. Datos de programa

Descripción general

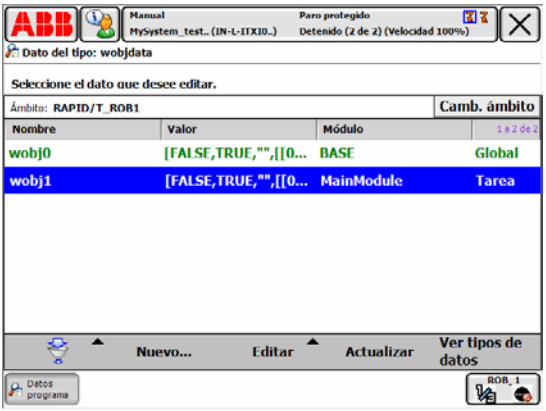
La vista Datos de programa contiene opciones de visualización y utilización de tipos de datos e instancias. Consulte el capítulo 2 de este manual.



Camb. ámbito	Cambia el ámbito de los tipos de datos de la lista
Mostrar datos	Muestra todas las instancias del tipo de dato seleccionado
Ver	Permite seleccionar los tipos de datos a visualizar: Todos los tipos de datos o solo los utilizados

Instancia de tipo de dato

A continuación se muestra una lista de instancias de un tipo de datos.

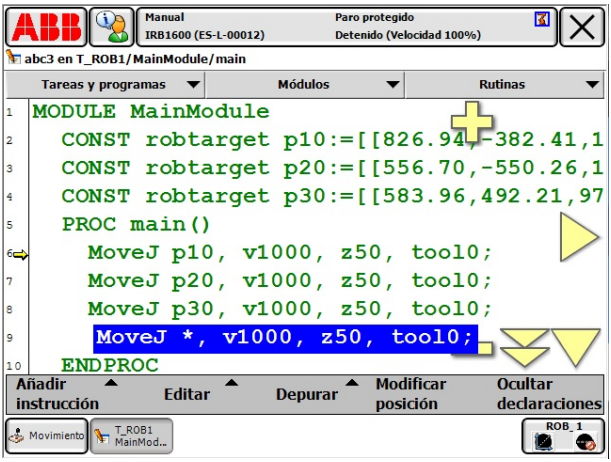


Filtro	Filtra las instancias
Nuevo	Crea una nueva instancia del tipo de dato seleccionado
Actualización	Actualiza la lista de instancias.
Editar	Edita los valores de la instancia seleccionada
Ver tipo de datos	Vuelve la menú **Datos de programa**

3.2.7. Editor de programas

Descripción general

El Editor de programas es donde se crean o modifican los programas. Consulte la sección *Programación en la página 81.*

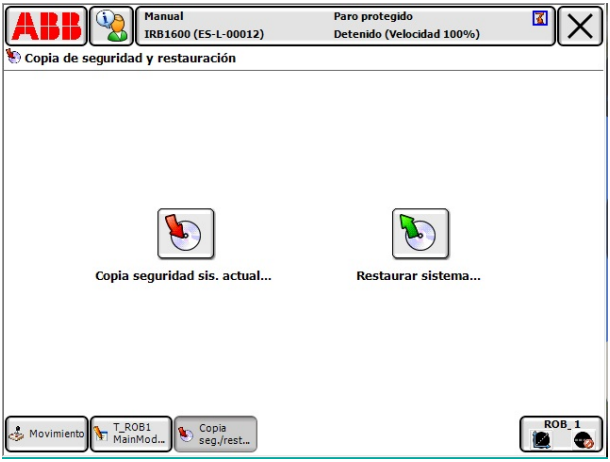


Tareas y programas	Menú para operaciones con programas
Módulos	Enumera todos los módulos
Rutinas	Enumera todas las rutinas
Añadir instrucción	Abre el menú de instrucciones
Editar	Abre el menú Editar
Depurar	Funciones para mover el puntero de programa, rutinas de servicio,…
Modificar posición	Véase 5.6.1 Modificación de posiciones
Ocultar declaraciones	Permite ocultar la sección de declaraciones de datos del programa de la visualización por la pantalla de la FlexPendant

3.2.8. Copia de seguridad y restauración

Descripción general

El menú Copia de seguridad y restauración se usa para realizar las copias de seguridad (backup) y la restauración de una copia de seguridad. Consulte *Copia de seguridad y restauración de sistemas en la página 115*

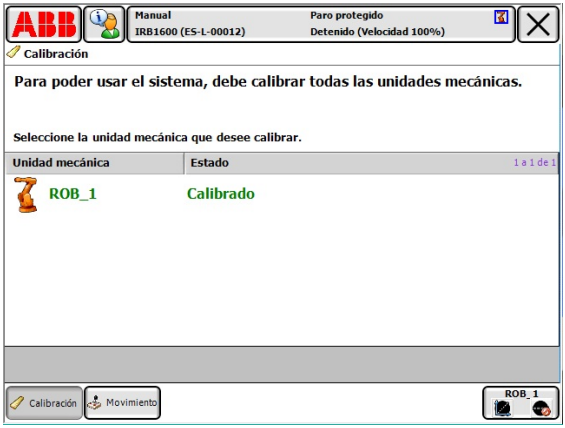


Copia seguridad sis. Actual…..	Permite realizar una copia de seguridad
Restaurar Sistema	Restaura una copia de seguridad

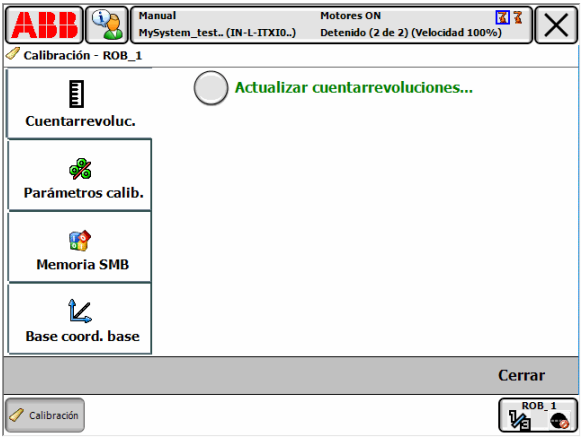
3.2.9. Calibración

Descripción general

La ventana Calibración se utiliza para calibrar las unidades mecánicas del sistema de robot (Actualizar contadores de vueltas del robot). Se visualizan todas las unidades mecánicas y el estado de calibración.



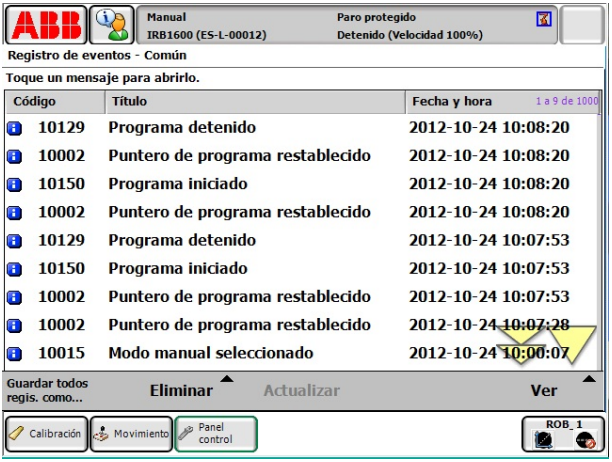
Opciones del menú Calibración



3.2.10. Registro de eventos

Descripción general

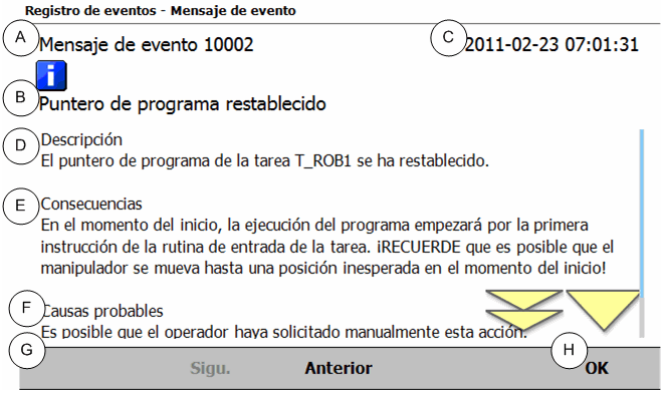
Con frecuencia, los sistemas de robot funcionan sin que tenga que estar nadie presente. La función de registro es una forma de almacenar información acerca de los eventos que han tenido lugar, como información de referencia futura y para facilitar la resolución de problemas. Consulte la sección *Manejo del registro de eventos en la página 112.*



Visualización de un mensaje	Toque el mensaje que se quiera visualizar. Véase la página siguiente
Desplazamiento por el mensaje y ampliación	Véase desplazamiento y zoom
Eliminar registro	Véase la sección ***Eliminación de los registros de eventos*** *en la página 113*
Guardar el registro	Véase la sección ***Guardado de los registros de eventos*** *en la página 114*

Un mensaje de registro de eventos

Cada entrada del registro de eventos se compone de un mensaje que describe el evento detalladamente y, normalmente, sugerencias para la resolución del problema.



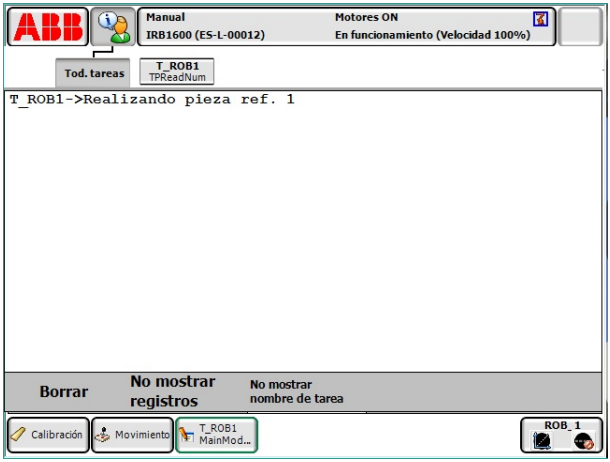
A	Número de evento. Todos los errores aparecen enumerados por su número.
B	Título del evento. Indica brevemente qué ha ocurrido.
C	Registro de hora del evento. Especifica exactamente cuándo se ha producido el evento.
D	Descripción. Una breve descripción del evento. Se ha diseñado para ayudarle a comprender las causas e implicaciones del evento.
E	Consecuencias. Una breve descripción de las consecuencias sobre el sistema, por ejemplo un cambio a otro modo de funcionamiento o un paro de emergencia, que ha tenido el evento determinado. Se ha diseñado para ayudarle a comprender las causas e implicaciones del evento.
F	Causas probables. Una lista de causas probables, enumeradas en orden de probabilidad.
G	Acciones recomendadas. Una lista de acciones correctoras recomendadas, basadas en las "Causas probables" especificadas anteriormente. Pueden ir de "Sustituya xx..." a "Ejecute el programa de prueba xx...". Es decir, puede contener acciones tanto para aislar el problema como para corregirlo.
H	Confirmar

3.3. Ventana de Operador

Descripción

La ventana de operador muestra los mensajes del programa. En los sistemas multitarea, los mensajes de todas las tareas se muestran en la misma ventana de operador. Si un mensaje requiere una acción, se mostrará una ventana separada para la tarea correspondiente.

Para abrir la ventana de operador, toque el icono que aparece a la derecha del logotipo de ABB en la barra de estado.



Borrar	Borra todos los mensajes
No mostrar registros	Oculta todos los mensajes
Mo mostrar nombre de tareas	Oculta los nombres de las tareas

3.4. Barra de Estado

La barra de estado muestra información acerca del estado actual del sistema, por ejemplo el modo de funcionamiento, el nombre del sistema y la unidad mecánica activa.

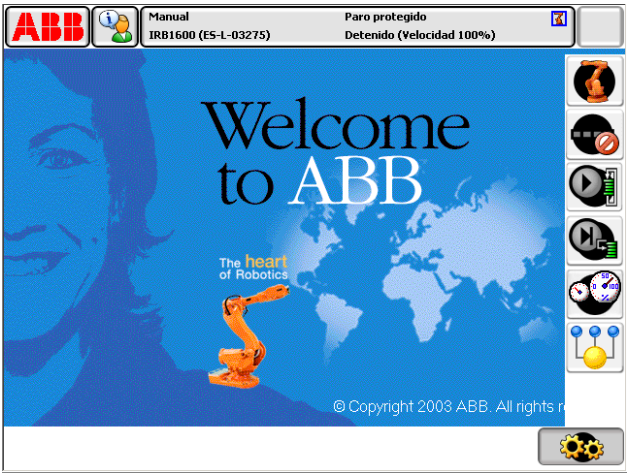
A	Ventana de operador
B	Modo de funcionamiento
C	Nombre del sistema (y nombre del controlador)
D	Estado del controlador
E	Estado de ejecución del programa
F	Unidades mecánicas. La unidad seleccionada (y cualquier unidad coordinada con la seleccionada) aparece resaltada por un recuadro. Las unidades activas se muestran con colores, mientras que las desactivadas tienen el color gris.

3.5. Menú de configuración rápida

Consideraciones generales

El menú de configuración rápida ofrece una forma más rápida de seleccionar y cambiar, entre otras cosas, las opciones de movimientos, en lugar de usar la ventana de **Movimientos**.

Cada opción del menú utiliza un icono para mostrar el valor de la opción seleccionada actualmente. Pulse el botón configuración rápida para ver los valores de las opciones disponibles.

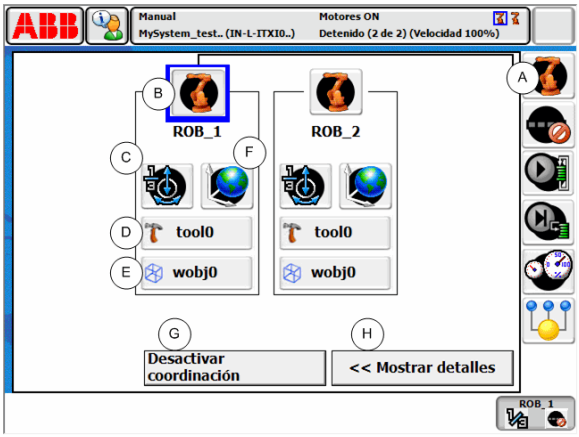


En esta sección se describen los botones del menú Configuración rápida.

A	Unidad Mecánica. *Consulte la página 37.*
B	Opciones de Incrementos. *Consulte la página 41*
C	Modos de ejecución. *Consulte la página 42*
D	Modos de ejecución paso a paso. *Consulte la página 43.*
E	Selección velocidad ejecución. *Consulte la página 44*
F	Seleccionar Tareas.

3.5.1. Menú Configuración rápida. Unidad mecánica

En el menú Configuración Rápida, toque Unidad mecánica y toque una unidad mecánica para seleccionarla.

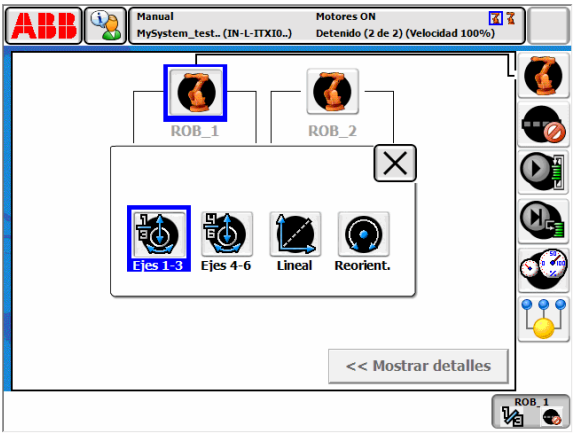


A	Botón de menú de unidad mecánica
B	Unidad mecánica, resaltando la unidad seleccionada.
C	Configuración de modos de movimiento (actualmente este modo de movimiento de los ejes 1 a 3)
D	Configuración de herramientas (actualmente seleccionada la herramienta tool0)
E	Configuración de los objetos de trabajo (actualmente seleccionado el objeto de trabajo wobj0)
F	Configuración del sistema de coordenadas (actualmente está seleccionadas coordenadas mundo).
G	Desactivar coordinación.
H	Mostrar detalles.

Importante
El menú Unidad Mecánica solo está disponible en el modo manual.

Configuración de modos de movimiento

Para ver o cambiar cualquier función de modo de movimiento, toque el botón de configuración de **modos de movimiento**. Esta configuración también está disponible en la ventana de movimientos.

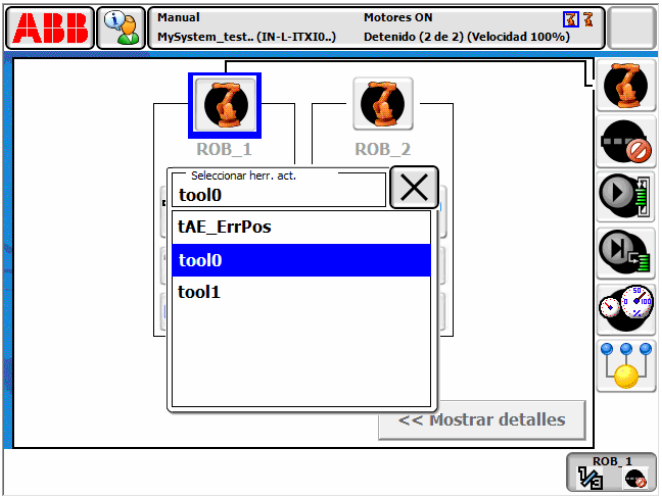


Esta pantalla nos permite seleccionar el modo de movimiento manual con el joystick:

- Ejes de 1 a 3
- Ejes de 4 a 6
- Lineal
- Reorientar

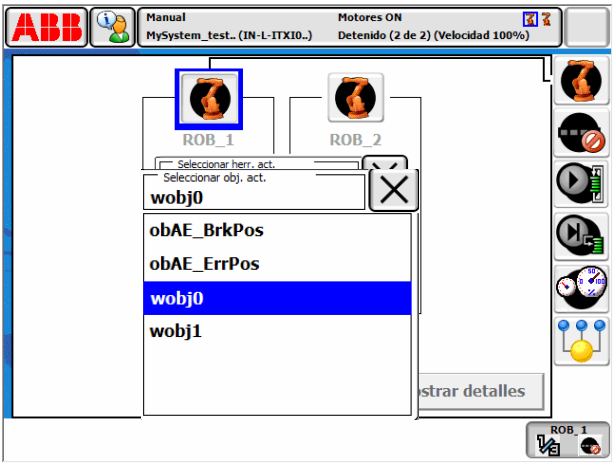
Configuración de herramientas

Para ver o cambiar las herramientas disponibles, toque el botón de configuración de **herramientas**. Esta configuración también está disponible en la ventana de movimientos.



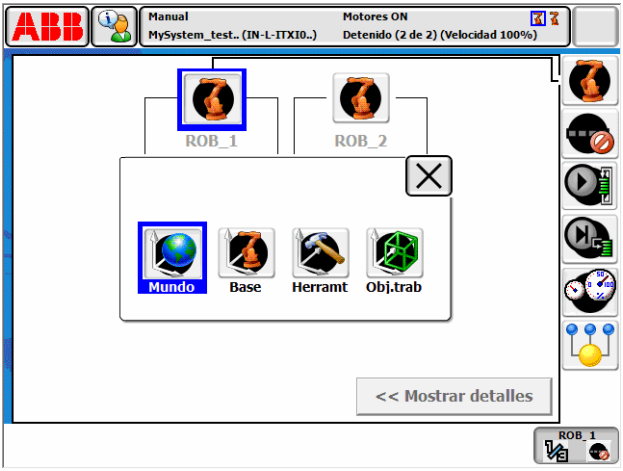
Configuración de los objetos de trabajo

Para ver o cambiar los objetos de trabajo disponibles, toque el botón de configuración de **objetos de trabajo**. Esta configuración también está disponible en la ventana de movimientos.



Configuración del sistema de coordenadas

Para ver o cambiar la funcionalidad del Sistema de coordenadas, toque el botón de configuración del **sistema de coordenadas**. Esta configuración también está disponible en la ventana de movimientos.

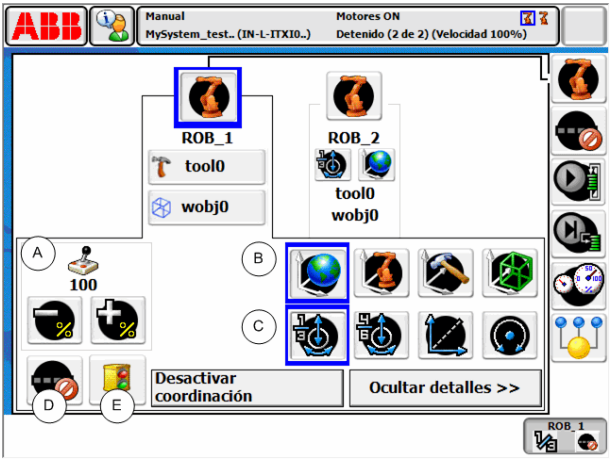


Seleccione un valor de sistema de coordenadas:

- Sistema de coordenadas mundo
- Sistema de coordenadas de la base
- Sistema de coordenadas de la herramienta
- Sistema de coordenadas del objeto de trabajo

Mostrar detalles

Toque **Mostrar detalles** para mostrar la configuración disponible para una unidad mecánica.



A	Configuración de redefinición de velocidad de movimiento (100% seleccionada actualmente))
B	Configuración del sistema de coordenadas (coordenadas mundo seleccionadas actualmente).
C	Configuración de modos de movimiento (modo de movimiento de los ejes de 1 a 3 seleccionado actualmente)
D	Activación o desactivación de incremento del usuario
E	Activación o desactivación de la supervisión de movimiento

Si cualquiera de los valores no está disponible, aparece tachado.

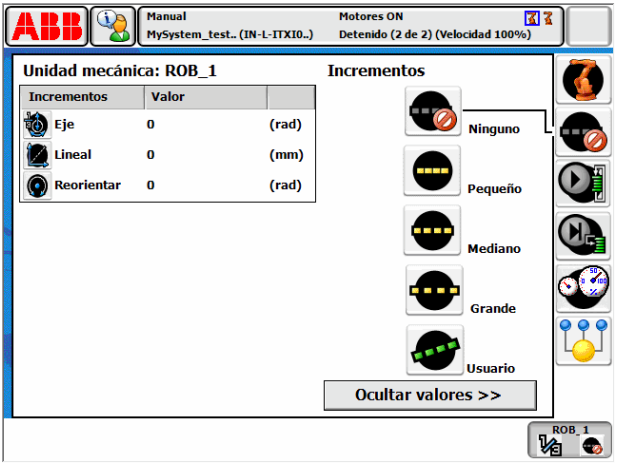
La configuración de modos de movimiento y sistema de coordenadas puede cambiarse tocando los botones.

Toque **Ocultar detalles** para volver a la pantalla básica.

3.5.2. Menú Configuración rápida. Incremento

Configuración de incrementos

La configuración de incrementos, también está disponible en la ventana de movimientos.



Ninguno	Sin incrementos
Pequeño	Movimientos pequeños
Mediano	Movimientos medianos
Grande	Movimientos grandes
Usuario	Movimientos definidos por el usuario
Mostrar Valores	Muestra los valores de los incrementos

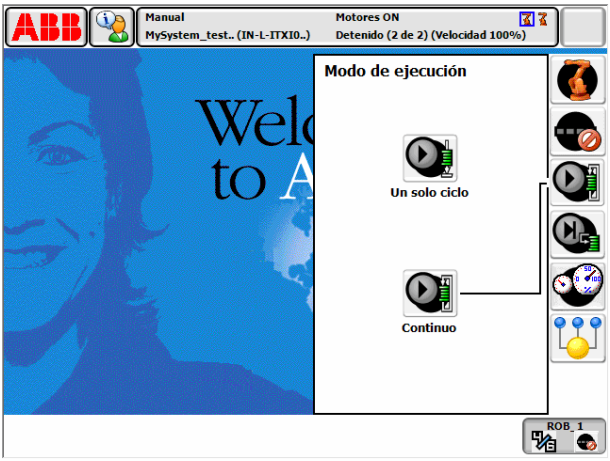
Importante

El menú Unidad Mecánica solo está disponible en el modo manual.

3.5.3. Menú Configuración rápida, Modo de Ejecución

Modo de ejecución

Mediante la configuración del modo de ejecución, se define si la ejecución del programa debe ejecutarse una sola vez y detenerse a continuación, o bien ejecutarse de forma continuada.

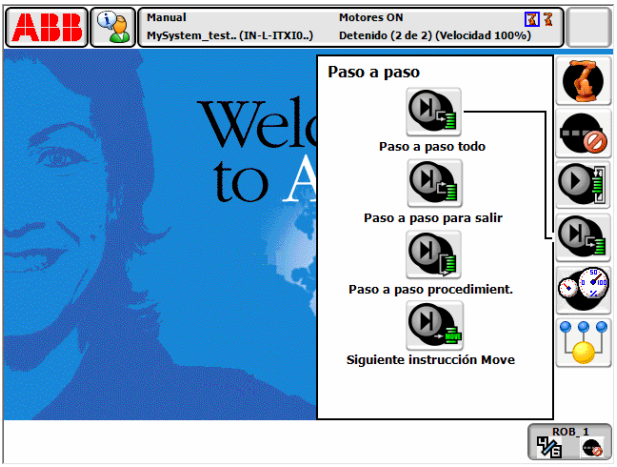


Un solo ciclo	Se ejecuta un solo ciclo y después se detiene la ejecución
Continuo	Se ejecuta continuamente

3.5.4. Menú Configuración rápida, Paso a paso

Paso a paso

Al definir el modo paso a paso, define cómo debe funcionar la ejecución paso a paso del programa.



Paso a paso todo	Entra en las rutinas a las que se llama y las ejecuta paso a paso.
Paso a paso para salir	Ejecuta el resto de la rutina actual y se detiene en la siguiente instrucción de la rutina desde la que se llamó a la rutina actual. No es posible usarlo en la rutina Main.
Paso a paso por procedimientos	Las rutinas a las que se llama se ejecutan en un paso.
Siguiente instrucción Move	Ejecuta el programa hasta la siguiente instrucción de movimiento. Se detiene antes y después de las instrucciones de movimiento, por ejemplo para modificar posiciones.

Limitaciones de la ejecución hacia atrás

La ejecución hacia atrás tiene algunas restricciones:

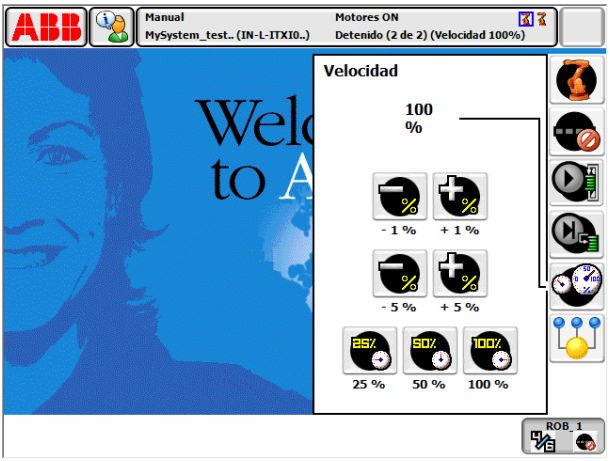
- El ejecutar paso a paso hacia atrás a través de una instrucción MoveC, la ejecución no se detiene en el punto circular.
- No es posible retroceder paso a paso hasta salir de una sentencia IF, FOR, WHILE y TEST.
- No es posible ejecutar paso a paso hacia atrás para salir de una rutina una vez alcanzado el principio de la rutina.
- Hay instrucciones que afectan al movimiento y que no pueden ejecutarse hacia atrás (por ejemplo, ActUnit, ConfL y PDispOn). Sl intenta ejecutar estas instrucciones hacia atrás, aparecerá un cuadro de alerta para informarle de que no es posible hacerlo.

3.5.5. Menú Configuración rápida, Velocidad

Velocidad

Los ajustes de velocidad se aplican al modo de funcionamiento actual. Sin embargo, si reduce la velocidad en el modo automático, el ajuste también se aplica al modo manual si cambia de modo.

Toque el botón **Velocidad** para ver o cambiar la configuración de velocidad. La velocidad de ejecución actual, respecto a la máxima, se muestra sobre los botones.



-1%	Reducir la velocidad de ejecución en pasos del 1%
+1%	Aumentar la velocidad de ejecución en pasos del 1%
-5%	Reducir la velocidad de ejecución en pasos del 5%
+5%	Aumentar la velocidad de ejecución en pasos del 5%
25%	Ejecutar a un cuarto de velocidad (25%)
50%	Ejecutar a la mitad de la velocidad (50%)
100%	Ejecutar a la máxima velocidad (100%)

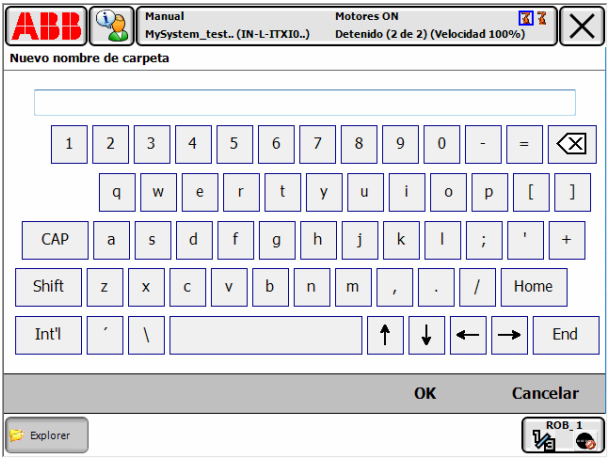
3.6. Procedimientos básicos

3.6.1. Utilización del teclado en pantalla

Utilización del teclado en pantalla

El teclado en pantalla se utiliza frecuentemente durante el manejo del sistema, por ejemplo para introducir nombres de archivo o valores de datos.

El teclado en pantalla funciona como un teclado convencional y permite situar el punto de inserción, escribir y corregir errores de escritura. Seleccione las letras, los números y los caracteres especiales para introducir sus textos o valores.



Utilización de otros caracteres

Es posible utilizar todos los caracteres occidentales, también en los nombres de usuarios y contraseñas. Para utilizar caracteres internacionales, toque el botón **Int'l** del teclado en pantalla.

Cómo cambiar el punto de inserción

Toque las teclas de flecha para cambiar el punto de inserción, por ejemplo al corregir errores de escritura.

Si tiene que mover…	a continuación, toque ….
Hacia atrás	←
Hacia adelante	→

Cómo borrar

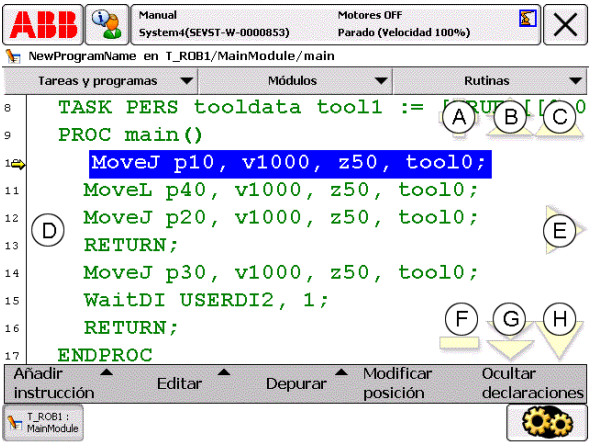
Pulse la tecla **Retroceso** (arriba a la derecha) para eliminar los caracteres que se encuentren a la izquierda del punto de inserción.

3.6.2. Desplazamiento y zoom

Descripción general

No todo el contenido de una pantalla estará visible de una sola vez. Para ver todo el contenido, puede:

- Desplazarse hacia arriba/hacia abajo (y en ocasiones hacia la izquierda/derecha)
- Ampliar o reducir (sólo disponible en el **Editor de programas**)



A	Ampliar (texto ampliado)
B	Desplazarse hacia arriba (el equivalente a la altura de una página)
C	Desplazarse hacia arriba (el equivalente a la altura de una línea)
D	Desplazarse hacia la izquierda
E	Desplazarse hacia la derecha
F	Reducir (texto reducido)
G	Desplazarse hacia abajo (el equivalente a la altura de una página)
H	Desplazarse hacia abajo (el equivalente a la altura de una línea)

4. Movimiento manual con joystick

4.1. Introducción al movimiento

¿En qué consiste el movimiento?

El movimiento consiste en el posicionamiento manual o el movimiento de los robots o los ejes externos con ayuda del joystick del FlexPendant. La palanca de mando o joystick de la unidad de programación se utiliza para posicionar el robot, bajo control manual, mientras se está programando. La palanca de mando puede utilizarse también para cambiar las posiciones de los ejes externos que están conectados al sistema.

La velocidad del movimiento correspondiente en el robot será proporcional a la deflexión de la palanca de mando.

IMPORTANTE:
El robot se mueve inmediatamente cuando se manipula la palanca de mando. Hay que asegurarse de que no hay nadie dentro del área de trabajo del robot antes de proceder a moverlo.

¿En qué situaciones es posible mover el robot o los ejes externos?

El movimiento solo se puede realizar en el modo manual. Es posible independientemente de qué vista se esté mostrando en la unidad de programación. Sin embargo, no es posible mover en modo automático ni durante la ejecución del programa.

Acerca de los modos de movimiento y los robots y ejes externos

El modo de movimiento y el sistema de coordenadas seleccionados determinan la forma en la que se mueve el robot.

En el *modo eje a eje* se mueven los ejes del robot realizando movimientos angulares. En este caso, el punto central de la herramienta no se mueve en línea recta.

En el *modo lineal*, el punto central de la herramienta se mueve realizando una línea recta en el espacio, paralela a los ejes del sistema de coordenadas seleccionado.

En el *modo reorientación* el punto central de la herramienta NO se mueve, los eje del robot se interaccionan para mantener el punto central de la herramienta estacionario, a pesar de los movimientos del robot.

Acerca de los modos de movimiento y los robots y ejes externos

Los ejes externos sólo pueden moverse *eje por eje*. Los eje externos pueden haber sido diseñado para movimiento lineal (transportador lineal) o para un movimiento giratorio (posicionadores de piezas de trabajo por ejemplo mesas de soldadura).

Los ejes externos no se ven afectados por el sistema de coordenadas seleccionado.

Acerca de los sistemas de coordenadas

El posicionamiento de un pasador en un orificio con una herramienta de pinza puede realizarse de forma muy sencilla en el sistema de coordenadas de la herramienta, siempre y cuando una de las coordenadas de ese sistema se encuentre en paralelo con el orificio. La realización de esta misma tarea en el sistema de coordenadas de la base puede requerir el movimiento en las coordenadas X, Y y Z, lo que hace mucho más difícil trabajar con precisión.

La selección del sistemas de coordenadas adecuados para el desplazamiento facilita el movimiento, pero no existe una respuesta simple o única a la pregunta de qué sistema de coordenadas debe utilizarse.

El sistema de *coordenadas de la base* tiene su punto cero en la base del robot, lo que hace que sus movimientos resulten predecibles. Por tanto, resulta útil a la hora de mover un robot de una posición a otra.

Un sistema de coordenadas determinado permite mover el punto central de la herramienta hasta la posición de destino con un número de movimientos de joystick menor que con otro sistema.

Condiciones como las limitaciones de espacio, los obstáculos o las dimensiones físicas de un objeto de trabajo o una herramienta también resultan útiles a la hora de hacer una valoración adecuada.

Encontrará más información acerca de los sistemas de coordenadas en la sección *¿Qué es un sistema de coordenadas? en la página 123*.

4.1.1. Selección del modo de movimiento

Descripción general

En la ventana de movimientos el área **Direcciones del joystick** muestra cómo los ejes del joystick se corresponden con los ejes del sistema de coordenadas seleccionado.

¡CUIDADO!
La visualización del área Direcciones no tienen como fin mostrar la dirección en la que se moverá la unidad mecánica. Intente siempre realizar los movimientos con desplazamientos reducidos del joystick, para comprobar las direcciones reales en las que se mueve la unidad mecánica.

Selección del modo de movimiento

En este procedimiento se describe cómo seleccionar un modo de movimiento.

Paso	Acción	Información
1	Seleccione menú **ABB** y a continuación **Movimiento**.	
2	Seleccione **Modo movto...**	
3	Pulse el modo que desee y seleccione **OK**.	El significado de las direcciones del joystick se muestra en el área **Direcciones del joystick** tras la selección.

Direcciones del joystick

El significado de las direcciones de joystick depende de qué modo de movimiento se haya seleccionado. Están disponibles los siguientes:

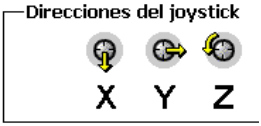
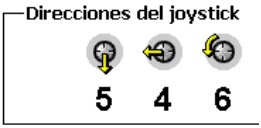
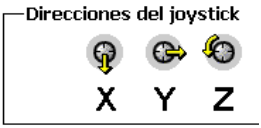
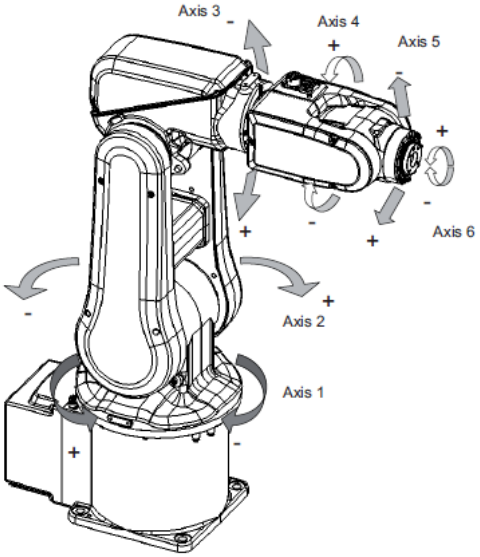
Modo de movimiento	Figura del joystick	Descripción
Lineal		El modo Lineal se describe en la sección *Movimiento Lineal en la página 57*.
Ejes de 1 a 3 (Predeterminado de los robots)		El modo de movimiento de ejes de 1 a 3 se describe en la sección *Movimiento eje por eje en la página 58*.
Ejes de 4 a 6		El modo de movimiento de ejes de 4 a 6 se describe en la sección *Movimiento eje por eje en la página 58*.
Reorientar		El modo Reorientar se describe en la sección *Selección de la orientación de la herramienta en la página 57*.

Figura de ejes del manipulador y direcciones de joystick

Los seis ejes de un manipulador genérico pueden ser movidos manualmente mediante las tres dimensiones del joystick, como se especifica a continuación.



Configuración predeterminada

Los modos de movimiento lineal y de reorientación tienen una configuración predeterminada para los sistemas de coordenadas y es válida para una unidad mecánica. Siempre se establece después de un reinicio. Si cambia el sistema de coordenadas de uno de estos modos de movimiento, el cambio se recordará hasta el reinicio siguiente (arranque en caliente)

Modo de movimiento	Sistema de coordenadas predeterminado
Lineal	Sistema de coordenadas de la base
Reorientación	Sistema de coordenadas de la herramienta

4.2. Sistemas de coordenadas

Sistemas de coordenadas

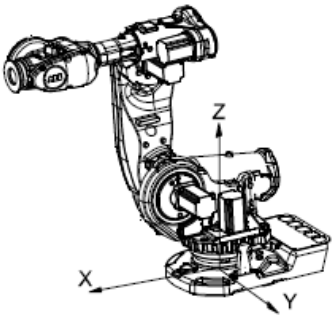
Un sistema de coordenadas define un plano o un espacio con ejes, partiendo de un punto fijo conocido como origen.

Las posiciones de robot se localizan mediante medidas a lo largo de los ejes de los sistemas de coordenadas.

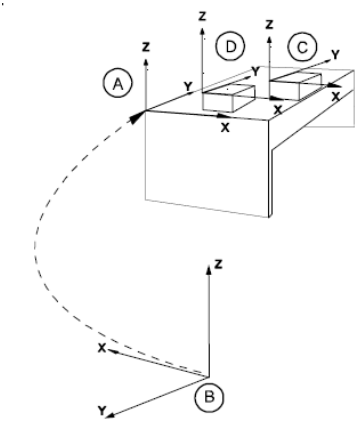
Los robots utilizan varios sistemas de coordenadas, cada uno de ellos adecuado para tipos concretos de movimientos o programaciones.

- El sistema de coordenadas de la base se encuentra en la base del robot. Es la forma más fácil de mover únicamente el robot de una posición a otra.
- El sistema de coordenadas del objeto de trabajo depende de la pieza de trabajo y con frecuencia es el más adecuado para la programación del robot.
- El sistema de coordenadas de la herramienta define la posición de la herramienta que utiliza el robot al alcanzar los objetivos programados.
- El sistema de coordenadas mundo define la célula de robot. Todos los demás sistemas de coordenadas dependen del sistema de coordenadas mundo, ya sea de forma directa o indirectamente. Resulta útil en los movimientos, los movimientos en general y el manejo de estaciones y células con varios robots o bien robots movidos por ejes externos.
- El sistema de coordenadas del usuario resulta útil a la hora de representar equipos que sostienen otros sistemas de coordenadas, por ejemplo objetos de trabajo

Coordenadas de la base

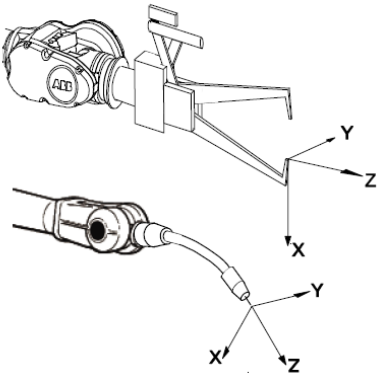


Coordenadas del objeto de trabajo

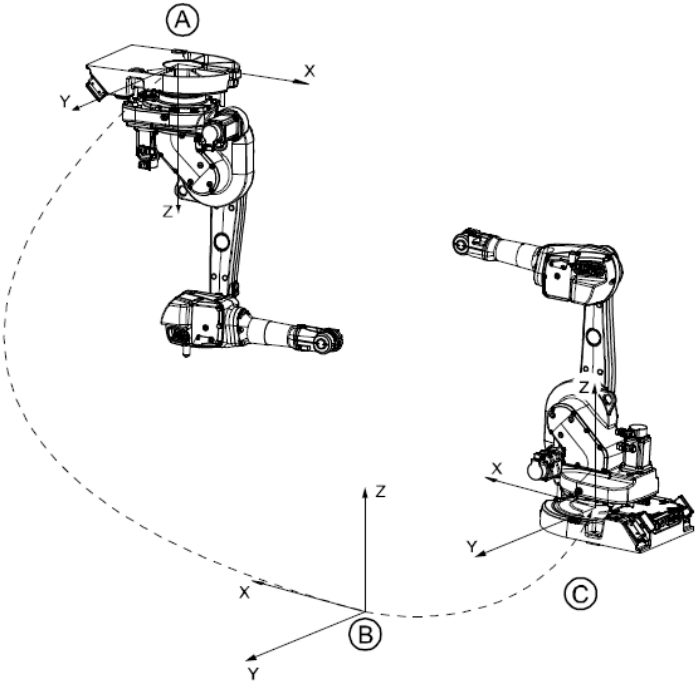


A	Sistema de coordenadas del usuario
B	Sistema de coordenadas mundo
C	Sistema de coordenadas del objeto de trabajo
D	Sistema de coordenadas del objeto de trabajo

Coordenadas de la herramienta



Coordenadas mundo



A	Sistema de coordenadas de la base
B	Sistema de coordenadas mundo
C	Sistema de coordenadas de la base

El uso del sistema de coordenadas mundo, se puede ver en el siguiente ejemplo: supongamos que tenemos dos robots, uno montado sobre el suelo y el otro invertido. El sistema de coordenadas de la base del robot invertido también estaría invertido, tal como se pude ver en la figura.
Si realiza un movimiento manual en el sistema de coordenadas de la base del robot invertido, será muy difícil predecir el movimiento. En cambio si utiliza el sistema de coordenadas mundo, como será el mismo para los dos robots, será mucho más fácil.

4.3. Restricciones en el movimiento

Movimiento de unidades mecánicas no calibradas

Las unidades mecánicas no calibradas sólo pueden ser movidas en modo eje a eje. Su área de trabajo no se comprueba.

Si la unidad mecánica no está calibrada, se muestra el texto "Unidad no calibrada" en el área de la Posición de la ventana de Movimiento.

Cuando el robot no está calibrado

¡CUIDADO!

Las unidades mecánicas cuya área de trabajo no sea controlada por el sistema de robot pueden moverse hasta posiciones peligrosas. Es necesario utilizar y configurar topes mecánicos para evitar riesgos para las personas o los equipos.

Movimiento con pesos de herramienta o carga útil no configurados

Si el peso de las herramientas o de las cargas útiles no está configurado, pueden producirse errores de sobrecarga durante los movimientos. Las cargas de los ejes externos controlados por elementos de software determinados (modelos dinámicos) sólo pueden ser definidas por programación.

4.4. Ajustes básicos para el movimiento

4.4.1. Selección del modo de movimiento

Modo de movimiento

Existen tres formas de seleccionar el modo de movimiento:

1 Con el botón **Cambiar modo de movimiento**.

2 Con la ventana de **movimiento** del menú **ABB**.

3 Con el menú **Configuración rápida**, **Unidad mecánica**

Selección del modo de movimiento con el botón de cambio

Pulse el botón **Cambiar el modo de movimiento de reorientación/lineal** para cambiar de modo de movimiento.

Selección del modo de movimiento en la ventana de movimiento

Utilice este procedimiento para seleccionar el modo de movimiento en la ventana de **movimiento**.

	Acción	Información
1	En el menú ABB, toque Movimiento	
2	Toque **Modo movto.**	
3	Toque el modo que desee y toque **OK**	El significado de las direcciones del joystick se muestra en el área **Direcciones del joystick** tras la selección

4.4.2. Selección de herramienta, objeto de trabajo y carga útil

Descripción general

Siempre resulta importante elegir la herramienta, objeto de trabajo o carga útil adecuados.

Resulta absolutamente esencial al crear un programa mediante movimientos hasta las posiciones de destino.

Si no lo hace, lo más probable es que dé lugar a errores de sobrecarga y/o posicionamiento incorrecto, ya sea durante el movimiento o al ejecutar el programa en producción.

Selección de herramienta, objeto de trabajo y carga útil

	Acción
1	En el menú **ABB**, seleccione **Movimiento** para ver las opciones de movimiento.
2	Seleccione **Herramienta**, **Objeto de trabajo** o **Carga útil** para mostrar listas de herramientas disponibles, objetos de trabajo o cargas útiles.
3	Seleccione la herramienta, el objeto de trabajo o la carga útil que desee, seguido de **OK**.

4.4.3. Modo de movimiento Reorientar.

La herramienta y el robot se mueven alrededor del punto central de la herramienta (TCP). El TCP no se moverá y se mantiene fijo en el espacio.

Las herramientas de soldadura al arco, fresado y dispensación deben estar orientadas con un ángulo determinado respecto de la pieza de trabajo para obtener los mejores resultados.

En la mayoría de los casos, la orientación de la herramienta es definida cuando el TCP ha sido movido ya hasta una posición determinada, por ejemplo el punto de inicio de la soldadura. Después de reorientar la herramienta, puede continuar moviéndose en un movimiento lineal para completar la trayectoria y las operaciones previstas.

El movimiento de reorientar es relativo respecto de la herramienta y del sistema de coordenadas seleccionado en ese momento.

Selección del movimiento de reorientar

	Acción
1	En el menú **ABB**, seleccione **Movimiento**.
2	Seleccione **Modo movto.** y a continuación, seleccione **Reorientar**, seguido de **OK**.
3	Es importante que tenga seleccionada la herramienta correcta, si no la ha seleccionado aún, seleccione a través de la ventana de movimiento.
4	Presione y mantenga presionado el dispositivo de habilitación para activar los motores de la unidad mecánica. Mueva el joystick. La orientación de la herramienta cambiará.

¡SUGERENCIA!
Utilice el menú **Configuración rápida** para seleccionar más rápido el modo de movimiento.

4.4.4. Modo de movimiento eje a eje

En el *modo* de movimiento *eje a eje,* se mueven los ejes del robot realizando movimientos angulares. En este caso, el punto central de la herramienta no se mueve en línea recta.

El robot reacciona a los movimientos del joystick incluso cuando el robot no está sincronizado.

Utilice el movimiento eje por eje cuando necesite:

- Alejar la unidad mecánica de posiciones peligrosas.
- Alejar un robot de singularidades.
- Posicionar los ejes de un robot para la calibración.

Selección del grupo de ejes con el botón de cambio

Selección del grupo de ejes en la ventana de movimiento

Utilice este procedimiento para seleccionar el grupo de ejes en la ventana de movimiento

	Acción
1	En el menú **ABB**, seleccione **Movimiento**
2	Seleccione **Modo movto.** y seleccione el grupo de ejes adecuado, grupo de ejes 1 a 3 o grupo de ejes 4 a 6.
3	Seleccione **OK** para completar la operación.
4	Presione el dispositivo de habilitación y mueva los ejes.

4.4.5. Modo de movimiento lineal

En el *modo* de movimiento lineal, se mueven los ejes del robot de tal manera que el TCP se mueve en movimientos rectos paralelos al sistema de coordenadas.

Sistemas de coordenadas para el movimiento

Para seleccionar el movimiento lineal con los distintos tipos de coordenadas desde la ventana de movimiento, seguir el siguiente procedimiento:

	Acción
1	En el menú **ABB**, seleccione **Movimiento** para ver las propiedades de movimiento.
2	Seleccione **Modo movto.** y a continuación **Lineal** seguido de **OK**. No necesita realizar este paso si ya ha seleccionado anteriormente el movimiento lineal.
3	El sistema por defecto selecciona coordenadas de la base del robot.
4	Presione y mantenga presionado el dispositivo de habilitación para activar los motores del manipulador.
5	Mueva el joystick para mover la unidad mecánica de la forma equivalente.

4.4.6. Bloqueo del joystick.

Es posible bloquear el joystick en direcciones concretas para impedir el movimiento de uno o varios ejes.

Por ejemplo, esto puede resultar útil al hacer ajustes detallados en las posiciones o al programar operaciones que sólo deben realizarse en la dirección de un eje concreto del sistema de coordenadas.

Recuerde que el bloqueo de los distintos ejes depende del modo de movimiento seleccionado actualmente.

¿Qué ejes están bloqueados?

Esta sección describe cómo comprobar qué direcciones del joystick están bloqueadas.

	Acción
1	En el menú **ABB**, seleccione **Movimiento** para ver las opciones de movimiento.
2	Seleccione **Bloqueo de joystick** para consultar las propiedades del joystick o compruebe las propiedades del área **Direcciones del joystick** que aparecen en la esquina derecha de la ventana. Los ejes bloqueados presentan un icono de candado.

Bloqueo del joystick en direcciones concretas

Esta sección describe cómo bloquear el joystick en direcciones concretas.

	Acción
1	En el menú **ABB**, seleccione **Movimiento** para ver las propiedades de movimiento
2	Seleccione **Bloqueo de joystick**. 
3	Seleccione el eje o los ejes del joystick que desee bloquear. Cada vez que seleccione en un eje, su estado cambia de bloqueado ha desbloqueado o viceversa.
4	Seleccione **OK** para realizar el bloqueo.

Desbloqueo de todos los ejes

Esta sección describe cómo desbloquear todos los ejes de las direcciones del joystick.

	Acción
1	En el menú **ABB**, seleccione **Movimiento**.
2	Seleccione **Bloqueo de joystick**.
3	Seleccione **Ninguno** y a continuación seleccione **OK**.

4.4.7. Movimiento incremental para posicionamientos exactos

Utilice el movimiento incremental para mover el robot en pasos pequeños, lo que permite un posicionamiento muy exacto.

Esto significa que cada vez que accione el joystick, el robot se mueve un paso (un incremento). Si mantiene accionado el joystick durante uno o varios segundos, se realiza una secuencia de pasos (a una velocidad de 10 pasos por segundo) mientras se mantenga accionado el joystick.

El modo predeterminado es el modo no incremental, en el que el robot se mueve continuamente al accionar el joystick.

Existen tres formas de seleccionar el tamaño del incremento:

- Con el botón **Activar/desactivar incrementos**.
- Con la ventana de **Movimiento** del menú **ABB**.
- Con el menú **Configuración rápida**, **Incrementos**.

Para usar el botón de activación/desactivación debe seleccionar primero el tamaño del incremento en la ventana de **movimiento** o en el menú **Configuración rápida**.

Selección de incrementos con el botón de incrementos

Definición del tamaño del movimiento incremental

En este procedimiento se detalla cómo especificar el tamaño del movimiento incremental.

	Acción
1	En el menú **ABB**, seleccione **Movimiento**.
2	Seleccione **Incremento**. 

3	Seleccione el modo de incremento que desee.
4	Seleccione **OK**.

Tamaños del movimiento incremental

Seleccione incrementos pequeños, medianos o grandes. También puede definir sus propios tamaños de movimiento incremental.

Incremento	Distancia	Angular
Pequeño	0,05 mm	0,005°
Mediano	1 mm	0,02°
Grande	5 mm	0,2°
Usuario		

4.4.8. Cómo leer la posición del robot.

Cómo se muestran las posiciones del robot

La posición se muestran como:

- *Movimiento de eje por eje*. El punto en el espacio, expresado en valor absoluto de cada eje, expresado en grados como el ángulo respecto de la posición de calibración.
- *Movimiento Lineal o reorientar*. El punto en el espacio, expresado en las coordenadas X, Y y Z del punto central de la herramienta expresada en milímetros más la orientación angular del punto central de la herramienta, expresada en Quaternions.

Cómo se muestran las posiciones de los ejes externos

Al mover un eje externo, sólo se muestra la posición del eje.

Las posiciones de los ejes lineales se expresan en milímetros como la distancia respecto de la posición de calibración.

Las posiciones de los ejes giratorios se expresan en grados como el ángulo respecto de la posición de calibración.

Ninguna posición mostrada

No se muestra ninguna posición si la unidad mecánica no está calibrada. En su lugar, aparece el texto "Unidad no calibrada".

Cómo leer la posición exacta

Este procedimiento describe cómo leer la posición exacta.

	Acción
1	En el menú **ABB**, seleccione **Movimiento**.
2	La posición se muestra en las propiedades del área **Posición** del lado derecho de la ventana.

Formato de la posición

La posición puede mostrarse en formatos diferentes. Toque **Formato de posición** para cambiar la configuración.

La Posición puede mostrarse en relación con las siguientes bases de coordenadas:

- Mundo
- Base
- Objeto de trabajo

El Formato de orientación puede configurarse como:

- Quaternion
- Ángulos Euler

El Formato de ángulo de posición puede configurarse como:

- Ángulos

La Unidad de ángulo de presentación puede configurarse como:

- Grados
- Radianes

5. Programación y pruebas

5.1. Antes de empezar a programar

Herramientas de programación

Puede usar tanto la unidad de programación como el RobotStudio para las tareas de programación. La unidad de programación es más adecuada para crear programas sencillos y la modificación de programas, por ejemplo posiciones, mientras que el RobotStudio es preferible para programaciones complejas.

Definición de herramientas, cargas útiles y objetos de trabajo

Defina las herramientas, las cargas útiles y los objetos de trabajo antes de empezar a programar. Siempre puede volver atrás y definir más objetos en otro momento, pero debe definir sus objetos básicos con antelación.

Definición de sistemas de coordenadas

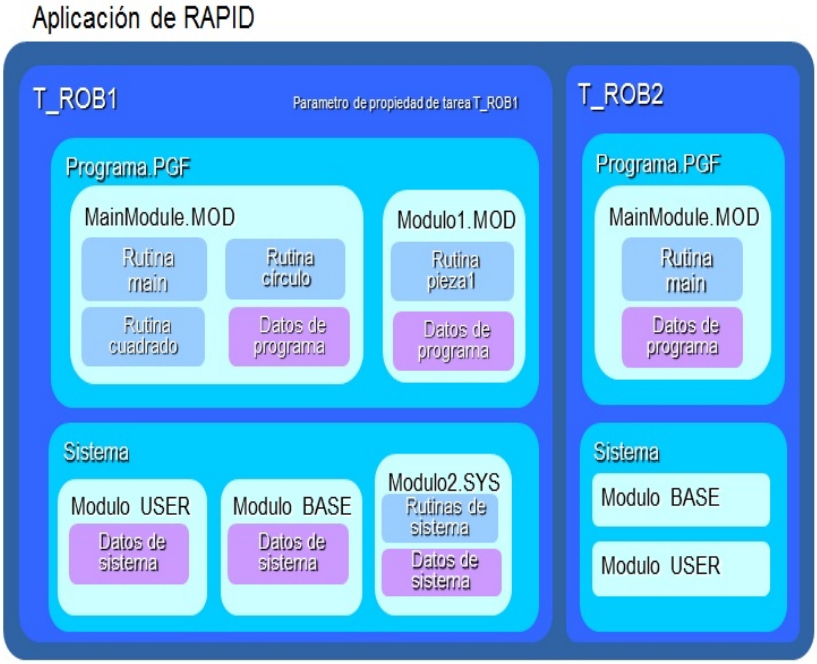
Asegúrese de que los sistemas de coordenadas mundo y de la base sean configurados correctamente durante la instalación de su sistema de robot.

Defina los sistemas de coordenadas de herramienta y de objeto de trabajo que necesite antes de comenzar a programar. A medida que añada más objetos de trabajo, también necesitará definir los sistemas de coordenadas correspondientes.

Verifique que la posición de calibración del robot sea correcta.

5.2. Concepto de programación

5.2.1. Estructura de una aplicación de RAPID



Partes

Parte	Función
Tarea	Cada tarea contiene un programa de RAPID y módulos de sistema destinados a realizar una función determinada. Cada aplicación de RAPID puede contener una tarea, y si tiene instalada la opción Multitarea, puede tener más de una tarea.
Parámetro de propiedad de tarea	Los parámetros de propiedades de tarea permiten establecer determinadas propiedades para todo el contenido de una tarea. Cualquier programa almacenado dentro de una tarea determinada asume las propiedades establecidas para la tarea.
Programa	Cada programa contiene uno o varios módulos de programa con código de RAPID para distintos fines, por ejemplo el modulo que se crea por defecto es el módulo principal “Main Module”. Cada programa debe tener una rutina main definida como ejecutable

Módulo de programa	Cada módulo de programa contiene datos y rutinas para una finalidad determinada. El programa está dividido en módulos principalmente para ofrecer una mejor visión general y facilitar el manejo. Normalmente, cada módulo representa una acción concreta del robot. Los módulos de programa suelen ser escritos por el usuario. Todos los módulos de programa se eliminan al eliminar un programa de la memoria del controlador.
Rutina	Una rutina contiene conjuntos de instrucciones, es decir, define qué debe hacer el sistema de robot. Las rutinas también pueden contener los datos necesarios para las instrucciones.
Rutina **main**	Cada programa **debe** tener una rutina especial llamada “main” definida como punto de inicio de la ejecución del programa. De lo contrario, no es posible ejecutarlo.
Instrucción	Cada instrucción es una petición para que tenga lugar un evento determinado, por ejemplo "Mover el manipulador hasta un punto determinado", "Cambiar el valor de una salida digital determinada", “Esperar el valor de una entrada digital” .
Datos	Los datos son valores y definiciones establecidos en los módulos de programa o de sistema. Estos datos son utilizados por las instrucciones del mismo módulo o de varios módulos (su disponibilidad depende del tipo de dato).
Módulo de sistema	Cada módulo de sistema contiene datos y rutinas encaminadas a la realización de una función determinada. El programa está dividido en módulos principalmente para ofrecer una mejor visión general y facilitar el manejo. Normalmente, cada módulo representa a una acción concreta del robot o una operación similar. Todos los módulos de sistema **se conservan** al ejecutar un comando "Eliminar programa". Los módulos de sistema suelen ser desarrollados por el fabricante del robot o los creadores de la línea de producción.

5.2.2. Acerca de los punteros de programa y de movimiento

El puntero de programa

El puntero de programa (PP) indica la instrucción en la que se inicia el programa al presionar los botones Iniciar, Avanzar y Retroceder del FlexPendant.

La ejecución continúa desde la instrucción en la que se encuentra el puntero de programa. Sin embargo, si el cursor se mueve a otra instrucción mientras el programa está parado, es posible mover el puntero de programa hasta la posición del cursor (o puede mover el cursor al puntero de programa) y reanudar la ejecución desde ese punto.

El puntero de programa se muestra como una flecha de color amarillo a la izquierda del código del programa mostrado en el Editor de programas y en la ventana de producción.

El puntero de movimiento

El puntero de movimiento indica la instrucción que está ejecutando actualmente el robot. Normalmente se trata de una o varias instrucciones que aparecen a continuación del puntero de programa, dado que el sistema ejecuta y calcula la trayectoria del robot más rápido de lo que se mueve el robot.

El puntero de movimiento se muestra como un pequeño robot a la izquierda del código del programa mostrado en el Editor de programas y en la ventana de producción.

El cursor

El cursor puede indicar una instrucción completa o cualquiera de los argumentos.

El cursor se muestra como un resalte de color azul en el código del programa que se muestra en el Editor de programas.

Editor de programas

Si cambia del Editor de programas a otra vista y vuelve al editor, el Editor de programas mostrará la misma parte del código siempre y cuando no se haya movido el puntero de programa. Si el puntero de programa se mueve, el Editor de programas muestra el código en la posición del puntero de programa.

Este mismo comportamiento se aplica a la ventana de producción.

5.3. Tipos de datos

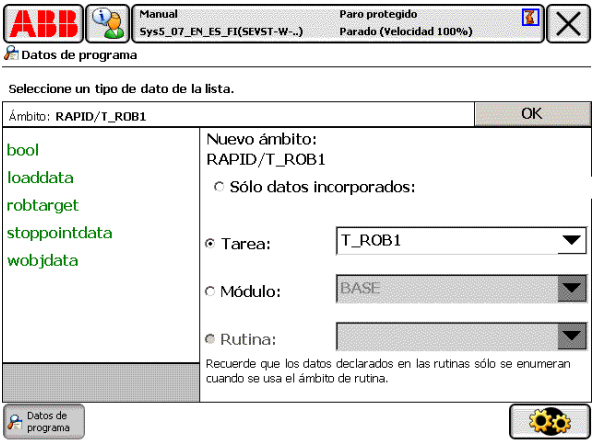
5.3.1. Visualización de los datos de tareas, módulos o rutinas concretos

Descripción general

Es posible ver conjuntos de tipos de datos mediante la selección de un ámbito específico.

Visualización de los datos de tareas, módulos o rutinas concretos

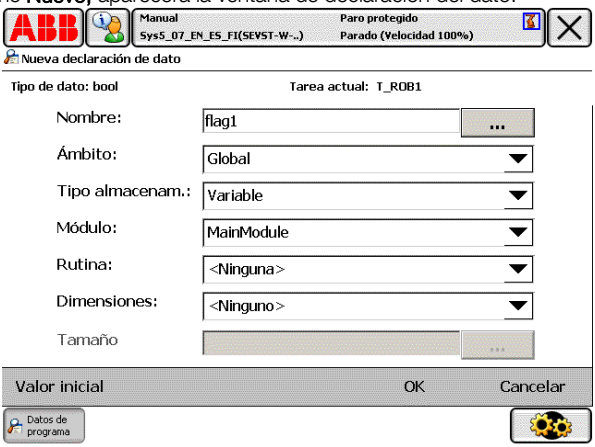
En esta sección se detalla cómo ver las instancias de dato de módulos o rutinas concretos.

Paso	Acción
1	En el menú **ABB**, seleccione **Datos de programa**.
2	Seleccione **Cambiar ámbito**. Aparece la pantalla siguiente: 
3	Seleccione el ámbito deseado, mediante las opciones: • Sólo datos incorporados: Muestra todos los tipos de datos utilizados por un sistema concreto. • Tarea: Muestra todos los tipos de datos utilizados por una tarea concreta. • Módulo: Muestra todos los tipos de datos utilizados por un módulo concreto. • Rutina: Muestra todos los tipos de datos utilizados por una rutina concreta.
4	Seleccione **OK** para confirmar la opción seleccionada.
5	Seleccione el tipo de dato y pulse **Mostrar datos** para ver sus instancias.

5.3.2. Creación de un nuevo dato

Creación de una nueva instancia

En esta sección se detalla cómo crear nuevas instancias de datos de los tipos de datos.

Paso	Acción
1	En el menú **ABB**, seleccione **Datos de programa**. Aparece una lista con todos los tipos de datos disponibles.
2	Seleccione el tipo de dato que desee crear, por ejemplo **bool** y seleccione **Mostrar datos**. Aparece una lista con todas las instancias del tipo de dato seleccionado.
3	Seleccione **Nuevo,** aparecerá la ventana de declaración del dato. 
4	Seleccione "**...**" a la derecha de **Nombre** para definir el nombre de la instancia de dato.
5	Seleccione el menú **Ámbito** para definir la accesibilidad de la instancia de dato. Seleccione: • **Global** • **Local** • **Tarea**
6	Seleccione **Tipo de almacenamiento** para seleccionar el tipo de memoria utilizado para el dato. Seleccione: • **Constante** el valor de una constante únicamente podrá ser cambiado manualmente, ejemplo datos robtarget. • **Variable** el valor de una variable podrá ser cambiada tanto manualmente como por el programa. Su valor inicial será activado automáticamente cuando: se carga un programa o cuando se realiza un PP a main, ejemplo datos num. • **Persistente** un dato persistente se describe como una variable cuyo valor inicial es constantemente actualizado. De esta forma su valor no será cambiado cuando se arranque el programa desde el principio. Si se guarda el programa el nuevo valor inicial será almacenado, ejemplo datos tooldata.

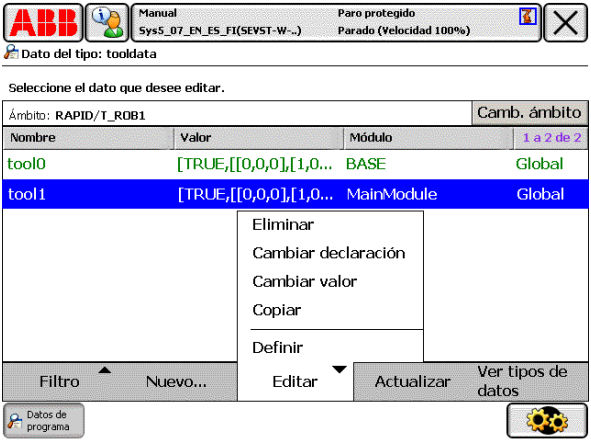
7	Seleccione el menú **Módulo** para seleccionar el módulo donde almacenar el dato.
8	Seleccione el menú **Rutina** para seleccionar la rutina donde almacenar.
9	Si desea crear un dato de tipo matriz, seleccione el menú **Dimensiones** y cambie el número de dimensiones en la matriz, de 1 a 3.
10	Seleccione **OK** para terminar.

5.3.3. Edición del valor de un dato

Descripción general

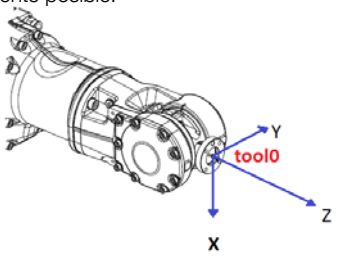
En esta sección se describe la forma de ver y editar el valor de un dato, además de eliminar, modificar la declaración y copiar o definir un dato.

En esta sección se detalla cómo visualizar los valores disponibles de un tipo de dato.

	Acción
1	En el menú **ABB**, seleccione **Datos de programa**.
2	Seleccione el tipo de dato que desee ver y seleccione **Mostrar datos**.
3	Seleccione el dato que desee editar y seleccione **Editar**. 
4	En función de qué desee hacer con el dato, tiene las opciones siguientes: • Seleccione **Eliminar** para eliminar el dato. • Seleccione **Cambiar declaración** para modificar la declaración del dato. • Seleccione **Cambiar valor** para modificar el valor de un dato. • Seleccione **Copiar** para copiar un dato. • Seleccione **Definir** para poder definir un dato, sólo disponible para herramientas y objetos de trabajo.

5.3.4. Creación de una herramienta

La posición del robot y sus movimientos siempre están referidos a su sistema de coordenadas de la herramienta, es decir, al TCP (punto central de herramienta) y a la orientación de la herramienta. Para obtener el mejor rendimiento, es importante definir el sistema de coordenadas de la herramienta lo más correctamente posible.



El punto central de la herramienta predeterminada (tool0) se encuentra en el centro de la brida de sujeción del robot y tiene un sistema de coordenadas predefinido, tal como el de la figura.

Un sistema de coordenadas de la herramienta puede ser definido manualmente o bien utilizando el robot como elemento de medida. Las definiciones manuales se utilizan cuando se disponga de datos precisos sobre las dimensiones de la herramienta o cuando se deban realizar cambios de poca importancia.

Creación de una herramienta nueva

Al crear una herramienta se crea un dato del tipo tooldata. La nueva herramienta tiene valores predeterminados para el peso, centro de gravedad y otros, que deben modificarse para poder usar la herramienta.

	Acción
1	En el menú **ABB**, seleccione **Movimiento**.
2	Seleccione **Herramienta** para ver la lista de herramientas disponibles.
3	Seleccione **Nuevo...** para crear una nueva herramienta. 
4	Seleccione **OK**.

En el caso en que el usuario desee salvar los datos en otro módulo, deberá seleccionar el campo **Módulo** y seleccionar el nombre del módulo en que se deban salvar los datos, por ejemplo si se almacena en el módulo del sistema User se podrá utilizar en más de un programa.

Valores de declaración de herramientas

Si desea cambiar...	...entonces...	Recomendación
El nombre de la herramienta	Seleccione el botón "**...**"que aparece junto al nombre.	Las herramientas reciben el nombre por defecto de tool1, tool2,… este nombre se puede cambiar por uno más descriptivo, como t_pinza o t_antorcha. Si cambia el nombre en la declaración de una herramienta después de que se hace referencia a ella en algún lugar del programa, deberá cambiar también todos los lugares en los que se use la herramienta.
El ámbito	Seleccione el ámbito que desee en el menú.	Las herramientas deben ser siempre globales para que estén disponibles desde todos los módulos del programa.
El tipo de almacenamiento	-	Los datos de herramienta deben ser siempre persistentes.
El módulo	Seleccione el nombre del módulo en que se deban salvar los datos.	MainModule si se va a utilizar en el programa solamente. Módulo User si se quiere utilizar en más de un programa.

La herramienta creada no puede usarse hasta definir los datos de la herramienta (coordenadas del TCP, peso, etc.). Esto puede hacerse ejecutando la rutina de servicio LoadIdentify o mediante la edición manual de los valores.

Solo el peso de la herramienta debe ser especificado. La carga útil manipulada por la misma deberá ser especificada mediante la instrucción Gripload.

5.3.5. Definición del sistema de coordenadas de la herramienta

Para definir el sistema de coordenadas de la herramienta, necesita en primer lugar un punto de referencia fijo (recomendamos una varilla acabada en punta que disponga de una buena fijación) situada dentro del área de trabajo del robot. Entonces, el usuario se moverá a cuatro posiciones del robot con distintas orientaciones (por lo menos), lo más cerca posible al punto de referencia fijo anterior. Estas posiciones se denominan Punto 1, 2, 3 y 4.

Para definir la orientación completa de una herramienta, se deberá mover una posición en el eje Z deseado y una posición en el eje X deseado al punto de elongación, estos puntos de elongación deberá tener la misma orientación que el último punto almacenado. Estas posiciones se denominan puntos de elongación Z y X.

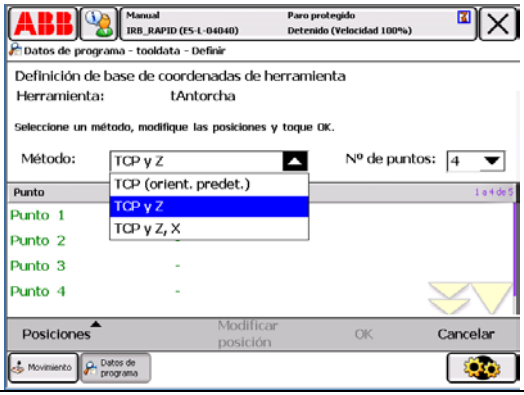
Métodos disponibles

Todos los métodos requieren que defina las coordenadas cartesianas del punto central de la herramienta. Estos métodos ofrecen distintas posibilidades de configuración y definición de la orientación.

Método	Si desea...
TCP (orientación predeterminada)	mantener la orientación actual
TCP y Z	cambiar la orientación en el eje Z
TCP y Z , X	cambiar la orientación en los ejes Z y X

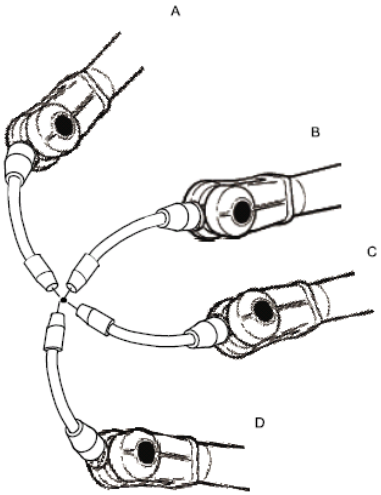
Como seleccionar un método

	Acción
1	En el menú **ABB**, seleccione **Datos de programa**.
2	Seleccione **tooldata** y **Mostrar datos** para ver la lista de herramientas disponibles.
3	Seleccione la herramienta que desee definir y seleccione **Editar**.
4	En el menú, seleccione **Definir** y aparece la ventana de definición del sistema de coordenadas.
5	Seleccione el método que prefiera en el menú emergente **Método**

	
6	Seleccione el número de puntos de aproximación en el menú emergente **Nº de puntos**. Normalmente basta con 4 puntos. Si elige más puntos para obtener un resultado más exacto, debe poner el mismo cuidado al definir cada uno de ellos.
7	Definir la base de coordenadas de la herramienta tal como se explica en el siguiente apartado

Definición de la base de coordenadas de la herramienta si hemos seleccionado el método TCP (orient. Predet.)

En este procedimiento se describe cómo definir el punto central de la herramienta en coordenadas cartesianas.



	Acción	Información
1	Mueva el robot hasta una posición adecuada, A, para el primer punto de aproximación.	Utilice incrementos pequeños para posicionar con exactitud la punta de la herramienta lo más cerca posible del punto de referencia.

2	Seleccione **Modificar posición** para definir el punto.	
3	Repita los pasos 1 y 2 con cada punto de aproximación que desee definir, posiciones B, C y D.	Intente que los puntos sean los más diferentes entre ellos. Si son parecidos los valores calculados no serán adecuados.

¿ Es suficientemente bueno el resultado obtenido ?

La ventana de diálogo **Resultado de cálculo** muestra el resultado calculado de la definición de la base de coordenadas de la herramienta. Es necesario confirmar que acepta el resultado antes de que pueda tener efecto en el controlador. La alternativa es repetir la definición de la base de coordenadas para conseguir un mejor resultado. El resultado **Error medio** es la distancia media de los puntos de aproximación con respecto al TCP (punto central de la herramienta) calculado. **Error máximo** Es el error máximo existente entre todos los puntos de aproximación.

Resulta difícil afirmar qué resultado es aceptable. Depende de la herramienta, el tipo de robot, etc. que esté utilizando. Normalmente, un error medio de algunas décimas de milímetro es un buen resultado. Si el posicionamiento ha sido realizado con una exactitud razonable, el resultado será correcto.

Dado que el robot se usa como máquina de medición, el resultado también depende de dónde se haya hecho el posicionamiento del área de trabajo del robot. Puede encontrarse una variación del TCP real de hasta un par de milímetros (en los robots grandes) entre las definiciones realizadas en distintas partes del área de trabajo. La repetitividad de cualquier calibración posterior del TCP aumentará por tanto si se realizan cerca de las precedentes. Recuerde que el resultado es el TCP óptimo del robot en esta área de trabajo, teniendo en cuenta cualquier discrepancia del robot en la configuración existente.

Comprobación si la base de coordenadas ha sido definida correctamente

Una forma habitual de comprobar que la base de coordenadas de la herramienta ha sido definida correctamente es realizar una prueba de reorientación una vez completada la definición. Seleccione el modo de movimiento de reorientación y el sistema de coordenadas de la herramienta y mueva el robot. Verifique que la punta de la herramienta permanezca muy cerca del punto de referencia seleccionado a medida que se mueve el robot.

5.4. Objetos de trabajo (wobjdata)

5.4.1. Creación de un objeto de trabajo

La creación y definición de un nuevo objeto de trabajo es similar al procedimiento utilizado anteriormente en la creación de los datos de herramienta. El nuevo dato será del tipo de dato "wobjdata". Una vez creado, el sistema de coordenadas del objeto de trabajo es ahora idéntico al sistema de coordenadas Wobj0. Consulte también *¿Qué es un objeto de trabajo? en la página 122*.

Creación de un objeto de trabajo

Cuando se crea, el sistema de coordenadas del objeto de trabajo es idéntico al sistema de coordenadas mundo.

	Acción
1	En el menú **ABB**, seleccione **Movimiento**.
2	Seleccione **Objeto de Trabajo** para ver la lista de objetos de trabajo disponibles.
3	Seleccione la herramienta que desee definir y seleccione **Editar**.
4	Seleccione **OK**.

Valores de declaración de los objetos de trabajo

Si desea cambiar...	...entonces...	Recomendación
El nombre del objeto de trabajo	Seleccione el botón "**...**"que aparece junto al nombre.	Los objetos de trabajo reciben el nombre por defecto de wobj1, wobj2,… este nombre se puede cambiar por uno más descriptivo, como w_utillaje. Si cambia el nombre en la declaración de un objeto de trabajo después de que se hace referencia a él en algún lugar del programa, deberá cambiar también todos los lugares en los que se use el objeto de trabajo.
El ámbito	Seleccione el ámbito que desee en el menú.	Los objetos de trabajo deben ser siempre globales para que estén disponibles desde todos los módulos del programa.
El tipo de almacenamiento	-	Las variables de objeto de trabajo deben ser siempre persistentes.
El módulo	Seleccione el nombre del módulo en que se deban declarar los datos.	MainModule si se va a utilizar en el programa solamente. Módulo User si se quiere utilizar en más de un programa.

5.4.2. Definición del sistema de coordenadas del objeto de trabajo

Descripción general

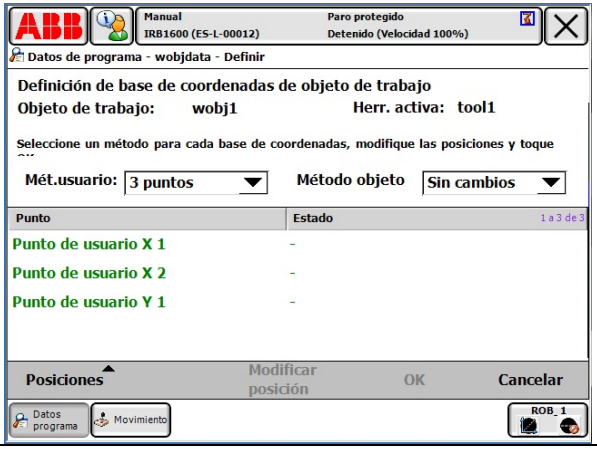
La definición de un objeto de trabajo, implica que se usa el robot para apuntar a su ubicación. Esto se hace mediante la definición de tres posiciones: dos en el eje X y una en el eje Y.

Los objetos de trabajos contienen en su definición dos bases de coordenadas: la base de coordenadas del usuario ("Uframe" dependiente de la base de coordenadas mundo) y la base de coordenadas del objeto ("Oframe" dependiente de la base de coordenadas del usuario).

A la hora de definir un objeto de trabajo, podemos definir las dos bases de coordenadas o solamente una de ellas. Usualmente suele definirse la base de coordenada del usuario.

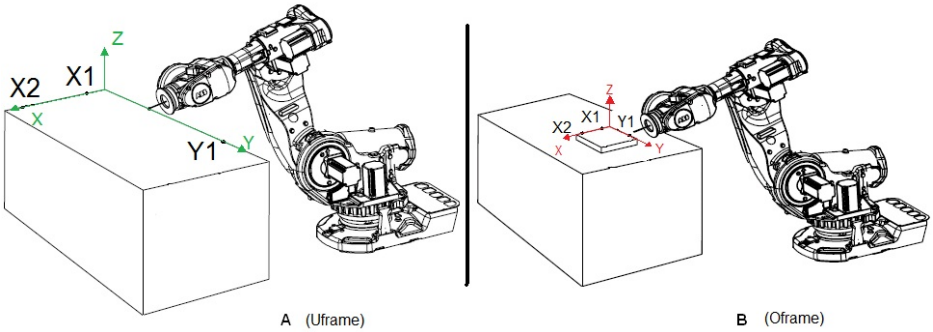
Selección de un método

En este procedimiento se describe cómo seleccionar el método. Recuerde que sólo podremos definir y modificar los objetos de trabajo creados por el usuario, por lo que no podremos modificar el objeto de trabajo predeterminado, wobj0.

	Acción
1	En el menú **ABB**, seleccione **Datos de programa**.
2	Seleccione **Wobjdata** y **Mostrar datos** para ver la lista de objetos de trabajo disponibles.
3	Seleccione el objeto de trabajo que desee definir y seleccione menú **Editar**.
4	En el menú, seleccione **Definir**.
5	Seleccione un método en el menú **Método de usuario** o el menú **Método de objeto**. 

Definición de la base de coordenadas del usuario y del objeto

En esta sección se detalla cómo definir la base de coordenadas del usuario (A) y la base de coordenadas del objeto (B).



El eje X pasará por los puntos X1-X2 y el eje Y pasará por Y1.

	Acción	Información
1	Seleccione la base de coordenadas que desee definir: **Método de usuario** o **Método de objeto** y seleccione la opción **3 puntos**.	
2	Presione el dispositivo de habilitación y mueva el robot hasta el primer punto X1.	Es recomendable que la distancia entre X1, X2 y Y1 sea grande lo que permite obtener una definición más exacta.
3	Seleccione el punto en la lista.	
4	Seleccione **Modificar posición** para definir el punto.	
5	Repita los pasos del 2 al 4 con los puntos, X2 y Y1.	

5.5. Programación

5.5.1. Manejo de programas

Descripción general

En esta sección se detalla cómo realizar el manejo normal de los programas de robot existentes:

- Crear un nuevo programa.
- Cargar un programa en memoria de trabajo.
- Guardar un programa en disco duro (Hd0A) u otra unidad (USB).
- Cambiar el nombre de un programa.
- Eliminar un programa de la memoria de trabajo.

Cada tarea solo puede contener *un* solo programa. Los procedimientos siguientes describen un sistema con una sola tarea.

Acerca de los archivos de programa

Al guardar un programa en el disco duro del controlador, se guarda de forma predeterminada en el directorio HOME de la carpeta del sistema, a no ser que se indique otra ubicación.

El programa se guarda como una carpeta que tiene el mismo nombre del programa y que contiene: los archivos de los módulos de programa con la extensión .mod y un archivo con el nombre del programa con la extensión .pgf.

Al cargar un programa, se abre la carpeta del programa y se selecciona el archivo pgf.

Al cambiar el nombre de un programa, se cambia el nombre de la carpeta y del archivo del programa.

Cuando se guarda un programa cargado que ya estaba guardado en el disco duro, no es necesario abrir la carpeta de programa existente. En su lugar, debe guardar la carpeta de programa de nuevo y sobrescribir la versión anterior, o bien cambiar el nombre del programa.

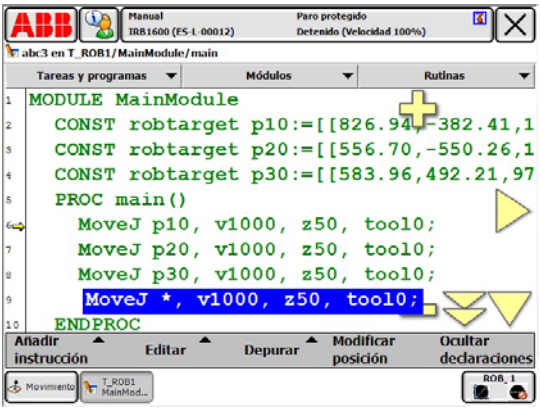
Creación de un programa nuevo

La forma de crear un nuevo programa cuando hay un programa disponible se detalla a continuación.

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione el acceso **Tareas y programas**.
3	Seleccione **Archivo** y a continuación **Nuevo programa**. Aparece una ventana de diálogo de advertencia. • Seleccione **Guardar** para guardar el programa de la memoria de trabajo. • Seleccione **No guardar** para cerrar el programa sin guardarlo, es decir, para eliminarlo de la memoria de trabajo. • Seleccione **Cancelar** para dejar cargado el programa.
4	Como pasos siguientes, añada instrucciones, rutinas o módulos.

Carga de un programa existente

En esta sección se describe como cargar un programa creado anteriormente.

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione el acceso **Tareas y programas**.
3	Seleccione menú **Archivo** y opción **Cargar programa**. Si ya había un programa cargado en memoria, aparece una ventana de diálogo de advertencia. • Seleccione **Guardar** para guardar el programa que está cargado. • Seleccione **No guardar** para cerrar el programa que está cargado sin guardarlo, es decir, para eliminarlo de la memoria de programas. • Seleccione **Cancelar** para dejar cargado el programa.
4	Seleccione en las unidades de almacenamiento el archivo de programa que desee cargar (archivos del tipo .pgf). A continuación, seleccione **OK**. El programa se carga en memoria y se muestra en la pantalla. 

Guardar un programa

En esta sección se describe cómo guardar en el disco duro del controlador el programa que está cargado en memoria en este momento.

Los programas cargados se guardan automáticamente en la memoria de programas, pero el hecho de guardarlo en el disco duro constituye una precaución adicional.

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione el acceso **Tareas y programas**.
3	Seleccione **Archivo** y seleccione **Guardar programa como...**.
4	Utilice el nombre propuesto o seleccione "**...**" para abrir el teclado en pantalla e introducir un nuevo nombre. A continuación, seleccione **OK**.

Cambio de nombre de un programa cargado

En esta sección se describe como cambiar el nombre de un programa cargado

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione **Tareas y programas**.
3	Seleccione **Archivo** y seleccione **Cambiar nombre de programa**. Aparece un teclado en pantalla.
4	Utilice el teclado en pantalla para introducir el nuevo nombre del programa. A continuación, seleccione **OK**.

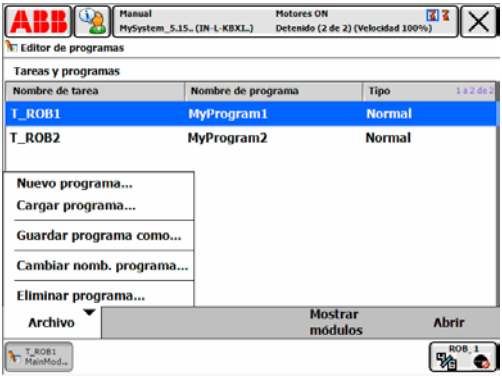
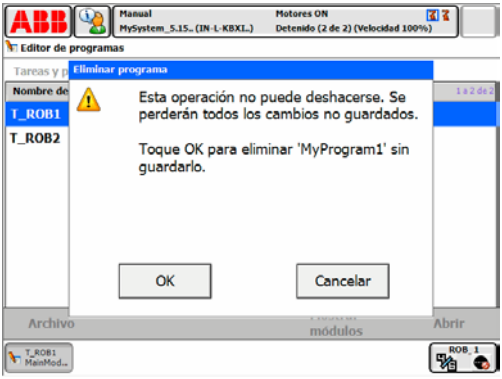
5.5.2. Eliminación de programas de la memoria

La eliminación de un programa no lo borra del disco duro del controlador (hd0a), sino únicamente de la memoria de programas.

Al cambiar de un programa a otro, el programa utilizado anteriormente se elimina de la memoria de programas, pero no del disco duro si está guardado en él.

Eliminación de programas de la memoria

En esta sección se detalla como eliminar programas de la memoria de programas

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione el acceso **Tareas y programas**.
3	Seleccione menú **Archivo** 
4	Seleccione **Eliminar programa…** 
5	Seleccionar **OK.**

5.5.3. Eliminación de programas del disco duro

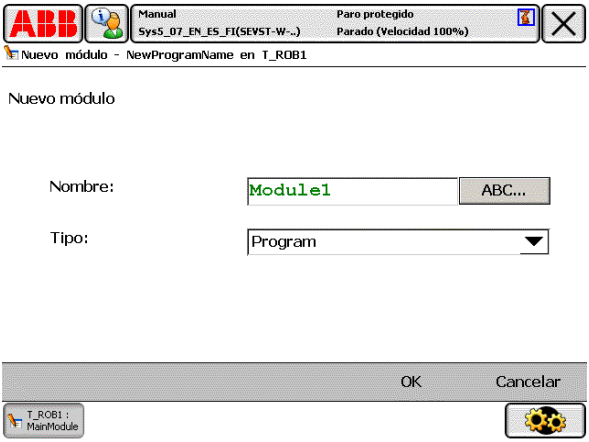
Los programas se eliminan con ayuda de la Unidad de Programación: En el menú **ABB**, seleccione **FlexPendant Explorer**. Al eliminar programas del disco duro del controlador (hd0a), el programa que está cargado actualmente en la memoria de programa no se ve afectado.

5.5.4. Manejo de módulos

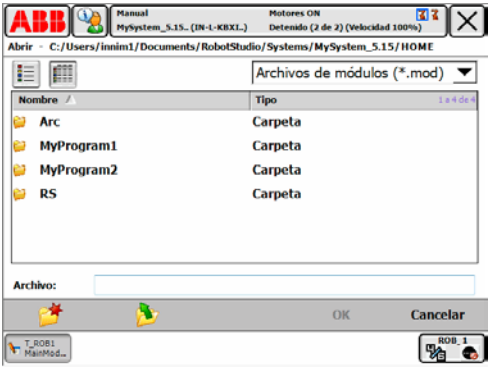
En esta sección se detalla cómo manejar los módulos, es decir:

- Crear un nuevo módulo.
- Cargar un módulo en memoria, creado anteriormente.
- Guardar un módulo en disco duro u otra unidad.
- Cambiar el nombre de un módulo.
- Eliminar un módulo de memoria.

Creación de un nuevo módulo

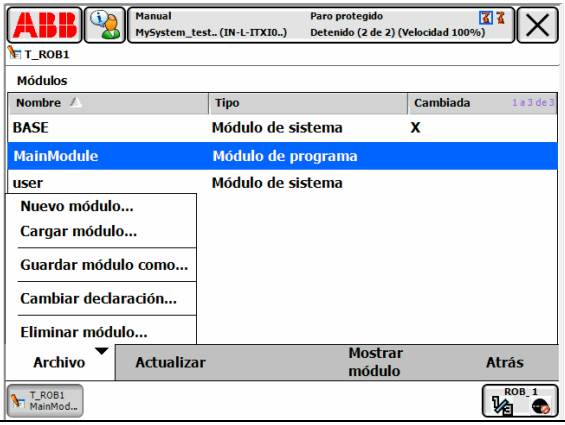
	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione el acceso **Módulos**.
3	Seleccione menú **Archivo** y opción **Nuevo módulo** 
4	Seleccione **ABC...** y utilice el teclado en pantalla para introducir el nombre del nuevo módulo. Seleccione **OK** para cerrar el teclado en pantalla.
5	Seleccione qué tipo de módulo desea crear: • Programa • Sistema A continuación, seleccione **OK**. Las diferencias existentes entre los tipos de módulos se describen en la sección *Estructura de una aplicación de RAPID en la página 66.*

Carga de un módulo creado anteriormente

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione el acceso **Módulos**.
3	Seleccione **Archivo** y a continuación **Cargar módulo**.  Busque y seleccione el módulo que desea cargar.
4	Seleccione **OK** para cargar el módulo seleccionado.

Guardado de un módulo

En esta sección se describe como guardar un módulo

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione **Módulos** y seleccione el módulo que quiere guardar
3	Selección **Archivo** y a continuación **Guardar módulo como…** 

4	Seleccione el nombre de archivo propuesto y utilice le teclado en la pantalla para introducir el nombre del módulo, y a continuación seleccione **OK**.
5	Utilice la herramienta de búsqueda de archivos para indicar donde desea guardar el módulo. La ubicación predeterminada es el disco duro. A continuación seleccione **OK**, y el módulo se guardará.

Cambio de nombre de un módulo

En esta sección se describe como cambiar el nombre de un módulo

1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione **Módulos**
3	Selección **Archivo** y a continuación **Cambiar nombre de módulo…** Aparece el teclado en pantalla.
4	Utilice el teclado en pantalla para introducir el nuevo nombre del módulo. Seleccione **OK**

Eliminación de un módulo

En esta sección se describe como eliminar un módulo de la memoria. Si el módulo está guardado en el disco, no se borrará del disco.

1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione **Módulos** y seleccione el módulo que desee eliminar.
3	Selección **Archivo** y a continuación **Eliminar módulo…** Aparece una ventana de dialogo.
4	Seleccione **OK** para eliminar el módulo sin guardarlo.

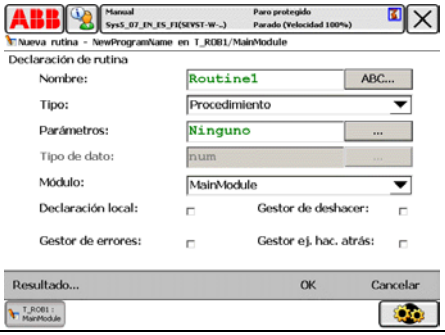
5.5.5. Manejo de rutinas

En esta sección se detalla cómo manejar las rutinas del programa, es decir:

- Crear una nueva rutina.
- Crear una copia de una rutina para duplicarla.
- Cambiar la declaración de una rutina
- Eliminar una rutina.

Creación de una rutina nueva

Se describe como crear una nueva rutina, modificar su declaración y su adición a un módulo.

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione **Rutinas**.
3	Seleccione menú **Archivo** y a continuación **Nueva rutina**. Se crea una nueva rutina, que se muestra con los valores de declaración predeterminados. 
4	Seleccione **ABC...** y utilice el teclado en pantalla para introducir el nombre de la nueva rutina. A continuación, seleccione **OK**.
5	Seleccione el tipo de rutina: • Procedimiento: Se describe como un número de instrucciones que realizan un trabajo específico, como por ejemplo la soldadura de una pieza. • Función: Se utiliza para las rutinas que nos devuelve un valor, y sirven por ejemplo para desplazar una posición. • Rutina TRAP: Se utiliza para las rutinas que permiten el tratamiento de interrupciones.
6	¿Necesita utilizar parámetros? En caso POSITIVO, seleccione "**...**", en caso NEGATIVO, continúe el proceso
7	Seleccione el módulo en que desee añadir la rutina.
8	Seleccione la casilla **Declaración local** si desea que la rutina sea local. Las rutinas locales sólo pueden usarse en el módulo seleccionado.
9	Seleccione **OK**.

Creación de una copia de una rutina

Esta sección se describe como crear una copia de una rutina

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione **Rutinas**.
3	Seleccione la rutina para resaltarla
4	Seleccione **Archivo** y **Copiar Rutina** Aparece la nueva rutina. El nombre de la nueva rutina es el mismo que tenía la rutina original pero con el sufijo *Copiar.*
5	Haga los cambios necesarios en las declaraciones de la nueva copia de la rutina. A continuación seleccione **OK**.

Modificar la declaración de una rutina

Esta sección se describe como modificar la declaración de una rutina.

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione **Rutinas**.
3	Seleccione la rutina para resaltarla
4	Seleccione **Archivo** y **Cambiar declaración**
5	Haga los cambios necesarios en las declaraciones de la rutina. A continuación seleccione **OK**.

Eliminar una rutina

Esta sección se describe como eliminar una rutina.

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione **Rutinas**.
3	Seleccione la rutina para resaltarla
4	Seleccione **Archivo** y **Eliminar rutina…**.
5	Seleccione: • **OK** para eliminar la rutina sin guardar ninguno de los cambios que haya hecho en ella. • **Cancelar** para deshacer los cambios sin eliminar la rutina

5.5.6. Manejo de instrucciones

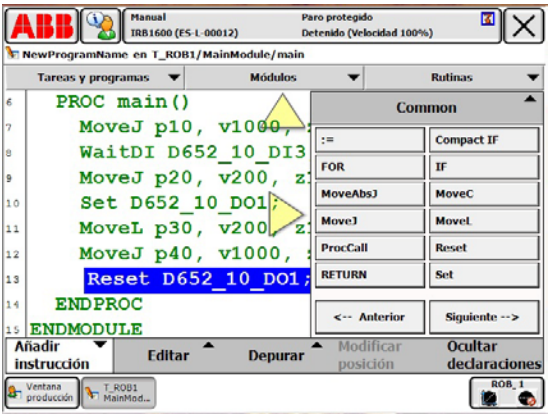
Los programas de RAPID se componen de instrucciones. Por ejemplo, una instrucción puede mover el robot, activar una señal de E/S o escribir un mensaje para el operador.

Operaciones de deshacer y rehacer

A la hora de editar programas en el Editor de programas, puede deshacer y rehacer hasta tres pasos. Esta función está disponible en el menú Edición.

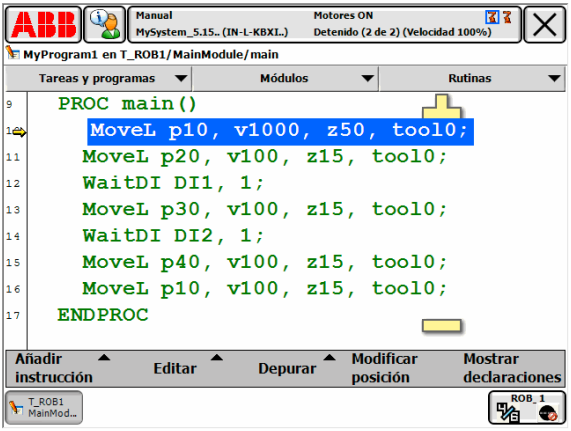
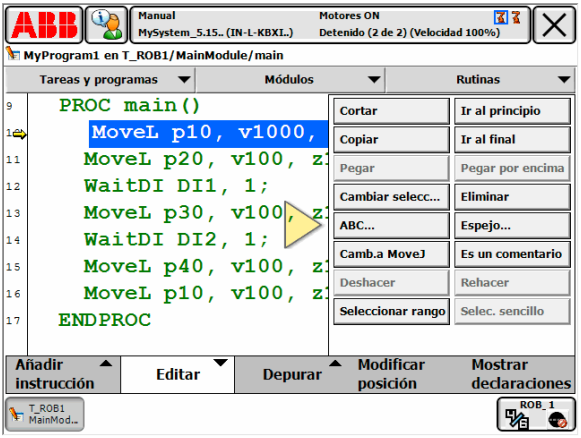
Cómo añadir instrucciones

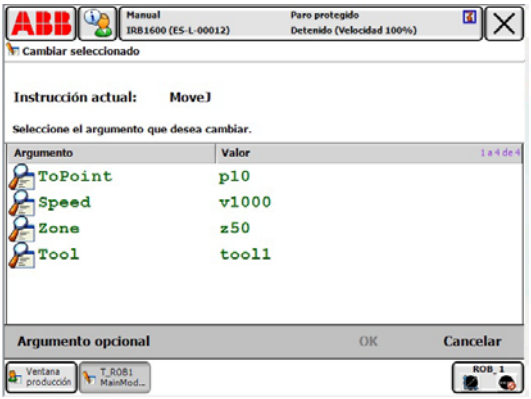
En esta sección se describe como añadir instrucciones

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Para resaltarla, seleccione la instrucción debajo de la cual desee añadir una nueva instrucción.
3	Seleccione **Añadir instrucción**, aparece una categoría de instrucciones.  Encontrará un gran número de instrucciones, divididas en varias categorías. La categoría por omisión es **Common** (Comunes), que contiene las instrucciones más comunes.
4	Seleccione **Common** para ver una lista con las categorías disponibles. También puede seleccionar **Anterior/Siguiente** en la parte inferior de la lista de instrucciones para pasar a la categoría anterior o siguiente.
5	Seleccione la instrucción que desee añadir y la instrucción se añade al programa.

Edición de argumentos de instrucciones

En esta sección se describe como editar los argumentos de las instrucciones

	Acción
1	Seleccione la instrucción que desee editar 
2	Seleccione **Editar** 

3	Seleccionar **Cambiar selecc..** En función del tipo de instrucción, los argumentos tienen tipos de datos diferentes. Utilice el teclado en pantalla para cambiar los valores de cadena o continúe en los pasos siguientes para otros tipos de datos o instrucciones con varios argumentos. 
4	Continuación del paso anterior, o si usa la segunda opción se abrirá la siguiente ventana: 
5	Seleccione una de las opciones del dato actual y seleccione **OK.**

Recomendación !!
Al tocar dos veces una instrucción, se inicia automáticamente la opción **Cambiar selecc..** Al tocar dos veces un argumento de la instrucción, se inicia automáticamente el editor de argumentos**.**

Copiado y pegado de instrucciones o argumentos

En esta sección se describe como pegar instrucciones o argumentos.

	Acción
1	Seleccione el argumento o la instrucción que desee copiar. Para seleccionar más de una fila: seleccione la primera fila, seleccione **Seleccionar rango** en el menú **Editar** y a continuación toque la última fila.
2	Seleccione **Editar** y después **Copiar**.
3	Sitúe el cursor en la instrucción o argumento sobre el cual desea pegar la instrucción o el argumento. A continuación, seleccione **Pegar**.

Como cortar una instrucción

En esta sección se describe como cortar una instrucción.

	Acción
1	Seleccione la instrucción que desee cortar. Para seleccionar más de una fila: seleccione la primera fila, seleccione **Seleccionar rango** en el menú **Editar** y a continuación toque la última fila.
2	Seleccione **Editar** y después **Cortar**.

Cambio del modo de movimiento de una instrucción de movimiento

En esta sección se describe como cambiar el modo de movimiento de una instrucción de movimiento.

	Acción
1	Seleccione la instrucción de movimiento que desee modificar. A continuación, seleccione **Editar**.
2	Seleccione **Cambiar a MoveJ** o **Cambiar a MoveL**.

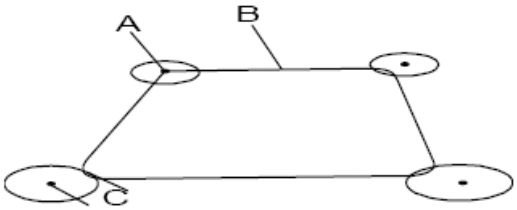
Convertir instrucciones en comentarios

Es posible convertir instrucciones en comentarios, de forma que se omitan durante la ejecución del programa. Seleccione la instrucción, a continuación seleccione en el menú **Editar**, **Es un comentario.**

5.5.7. Ejemplo: Cómo añadir instrucciones de movimiento

Descripción general

En este ejemplo creará un programa sencillo que hace que el robot describa un cuadrado. Necesita cinco instrucciones de movimiento para completar este programa.



A	Primer punto.
B	Dato de velocidad de movimiento del robot v50 = velocidad 50 mm/s.
C	Zona z50 = (50 mm).

Cómo añadir instrucciones de movimiento

En esta sección se describe como añadir instrucciones de movimiento

	Acción	Información
1	Mueva el robot hasta el primer punto.	Muévase de izquierda-derecha/arriba-abajo del joystick para describir un cuadrado.
2	En el Editor de programas, seleccione **Añadir instrucción**.	
3	Seleccione **MoveL** para insertar una instrucción.	
4	Repita la operación con las cuatro posiciones siguientes del cuadrado.	
5	Para las instrucciones primera y última: Seleccione la zona de precisión z50 y cámbiela utilizando el menú **Editar,** a continuación cambie el valor, seleccionando **fine**, Seleccione **OK**.	

Resultado

La secuencia de programa debe parecerse a la siguiente:

```
PROC main()
    MoveL *, v50, fine, tool0;
    MoveL *, v50, z50, tool0;
    MoveL *, v50, z50, tool0;
    MoveL *, v50, z50, tool0;
    MoveL *, v50, fine, tool0;
ENDPROC
```

5.6. Programación avanzada

5.6.1. Modificación de posiciones

Descripción general
Las posiciones son datos del tipos de dato robtarget.

Las posiciones pueden ajustarse con ayuda de la función HotEdit, que permite introducir valores de offset a través de un teclado en pantalla. El valor de offset se utiliza junto con el valor de posición original.

Las posiciones pueden ser modificadas mediante la función “Modificar posición” del Editor de programas o la ventana de producción, que permite modificar la posición programada por la nueva posición.

¡CUIDADO!
El cambio de las posiciones programadas puede alterar significativamente el patrón de movimientos del robot.

Asegúrese siempre de que todos los cambios resulten seguros tanto para los equipos como para el personal.

5.6.2. Modificación de posiciones en el Editor de programas o la Ventana de producción

Descripción general de la modificación de posiciones en el Editor de programas

Para modificar posiciones con el Editor de programas, el sistema debe estar en el modo manual.
A la hora de modificar posiciones moviendo el robot hasta la nueva posición, puede ejecutar previamente paso a paso el programa hasta las posiciones que desee modificar o moverse directamente a la nueva posición y utilizando la función **Modificar posición** cambiar la posición correspondiente de la instrucción.

Modificación de posiciones

En este procedimiento se describe cómo modificar posiciones, ya sea ejecutando paso a paso hasta las posiciones o seleccionando un movimiento.

	Acción	Información
1	En el menú **ABB**, seleccione **Editor de programas**.	
2	Detenga el programa si se está ejecutando.	
3	Si prefiere *ejecutar paso a paso*, ejecute el programa paso a paso hasta la posición que desee cambiar. Asegúrese de que esté seleccionado el argumento de posición correcto. Si selecciona una instrucción de movimiento, verifique en la ventana de **Movimientos** que tenga seleccionados el mismo objeto de trabajo y la misma herramienta que se utilizan en la instrucción.	En el modo de ejecución paso a paso, si la instrucción o la llamada al procedimiento tienen más de un argumento de posición, continúe con el modo paso a paso hasta alcanzar los distintos argumentos.
4	Mueva el robot hasta la nueva posición.	
5	Si utiliza el método con movimiento, seleccionar el argumento de posición que desee cambiar.	
6	Seleccione la función **Modificar posición** en el Editor de Programas. En la ventana de producción, seleccionar **Depurar** y a continuación **Modificar Posición** Aparece una ventana de diálogo de confirmación.	
7	Seleccione **Modificar** para usar la nueva posición, o **Cancelar** para conservar la original.	
8	Repita los pasos del 3 al 7 para cada argumento de posición que desee cambiar.	

Limitaciones

El botón **Modificar posición** del Editor de programas está desactivado hasta que seleccione un argumento de posición.

¡Atención!
Si cambia una posición con nombre, todas las demás instrucciones que utilicen esa posición se verán afectadas.

5.6.3. Cómo mover el robot hasta la posición programada

Posiciones

Los programas de robot suelen contener posiciones programadas. El robot puede moverse automáticamente hasta una posición programada con ayuda de una función de la ventana de Movimientos.

El robot se moverá a una velocidad de 250 mm/s.

¡PELIGRO!
Al mover el robot automáticamente, el brazo del robot puede moverse sin ninguna advertencia previa. Asegúrese de que no haya nadie dentro del espacio protegido y de que no haya ningún objeto entre la posición actual y la posición programada.

Cómo mover el robot hasta la posición programada

	Acción	Información
1	En el menú **ABB**, seleccione **Movimiento**.	
2	Asegúrese de que esté seleccionada la unidad mecánica correcta **ROB_1** y seleccione **Ir a...**.	
3	Seleccione una posición programada de la lista.	Si tiene muchas posiciones programadas, puede usar un filtro para reducir el número de posiciones visibles.
4	Presione y mantenga presionado el dispositivo de habilitación y seleccione y mantenga presionado el botón **Ir a**. Ahora el robot se mueve directamente de la posición actual hasta la posición programada. Asegúrese de que no haya ningún objeto en la trayectoria.	

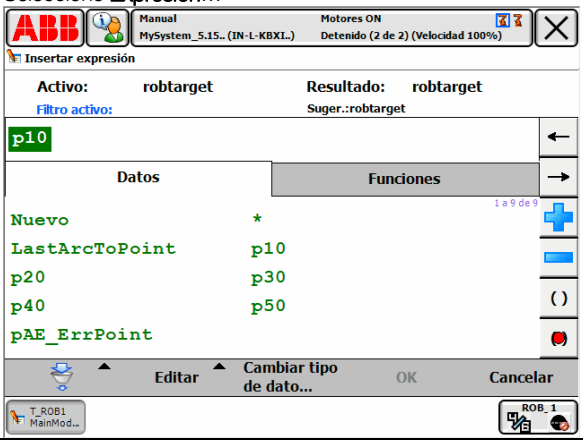
5.6.4. Edición de expresiones y argumentos de instrucciones

Expresiones

Una expresión especifica la evaluación de un valor. Por ejemplo, puede usarla en las situaciones siguientes:

- Como una condición de una instrucción IF
- Como un argumento en una instrucción
- Como un argumento en la llamada a una función

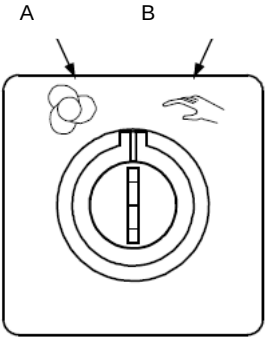
Edición de expresiones

	Acción
1	En el **Editor de programas**, seleccione la expresión que desee editar y seleccione el menú **Editar**.
2	Seleccione **Cambiar selección** y seleccione el argumento que desee cambiar.
3	Seleccione **Expresión**… 
4	Edite la longitud de la expresión, utilizando los botones de la derecha: • Flechas “izquierda / derecha”: Retroceder y avanzar en la expresión. • “+” Para añadir más elementos a la expresión. • “-“ Para eliminar elementos de la expresión. • “()” Para añadir paréntesis alrededor de la expresión seleccionada. • “(o)” Para eliminar los paréntesis.
5	Seleccione: • **Nuevo** para crear un nuevo dato, es decir, añadir una declaración de dato no utilizada anteriormente. • **Cambiar tipo de dato…** para cambiar de vista y modificar el tipo de dato. • **ABC** muestra el teclado en pantalla.
6	Seleccione **OK** para confirmar la expresión.

5.7. Pruebas

5.7.1. Modos de Funcionamiento

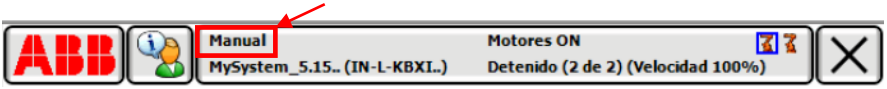
Selector de modo de funcionamiento



A	Modo Automático.
B	Modo Manual

Visualización del modo actual en la Unidad de Programación

En la FlexPendant, el modo de funcionamiento actual se puede ver en la barra de estado, tal como se puede ver en la siguiente figura:



5.7.2. Acerca del modo automático

¿Qué es el modo automático?

En el modo automático, la función de seguridad del dispositivo de habilitación está puenteada para que el manipulador pueda moverse sin intervención humana.

El modo automático es el modo de funcionamiento en el que el sistema de control del robot funciona de acuerdo con el programa de la tarea, con medidas de protección funcionales. Este modo permite controlar el manipulador, por ejemplo, mediante las señales de E/S del controlador. Es posible utilizar una señal de entrada para iniciar y detener un programa de RAPID y otra para activar los motores del manipulador.

¡AVISO!
Antes de seleccionar el modo automático, cualquier protección suspendida deberá ponerse de nuevo en pleno funcionamiento.

¿Qué es el modo automático?

En el modo automático suelen realizarse las siguientes tareas.

- Inicio y detención de procesos.
- Carga, inicio y detención de programas de RAPID.
- Devolución del manipulador a su trayectoria al reanudar el funcionamiento tras un paro de emergencia.
- Copia de seguridad del sistema.
- Restauración de copias de seguridad.
- Limpieza de herramientas.
- Preparación o sustitución de objetos de trabajo.
- Realización de otras tareas orientadas a procesos.

Limitaciones del modo automático

No se permite hacer movimientos en el modo automático. Puede haber otras tareas concretas que no deben realizarse en el modo automático.

5.7.3. Acerca del modo manual

¿Qué es el modo manual?

En el modo manual, el movimiento del manipulador se realiza bajo control manual. Es necesario presionar el dispositivo de habilitación para activar los motores del manipulador, es decir, para permitir el movimiento.

El modo manual se utiliza durante la programación y para la verificación de un programa.

Seguridad durante el modo manual

Durante el modo manual, el manipulador se maneja con personas a corta distancia. El manejo de un manipulador industrial es potencialmente peligroso y por tanto todo el manejo debe ser realizado de una forma controlada.

¿Qué es el modo manual a velocidad reducida?

En el modo manual a velocidad reducida, el movimiento está limitado a 250 mm/s. Es necesario presionar el dispositivo de habilitación para activar los motores del manipulador.

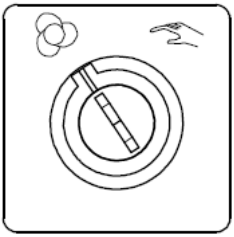
¡AVISO!
Siempre que sea posible, el modo manual de funcionamiento debe ejecutarse con todo el personal fuera del espacio protegido.

Tareas que suelen realizarse en el modo manual a velocidad reducida

En el modo manual a velocidad reducida suelen realizarse las siguientes tareas.

- Movimiento del manipulador para situarlo de nuevo en su trayectoria al reanudar el funcionamiento tras un paro de emergencia
- Corrección del valor de las señales de E/S tras situaciones de error
- Creación y edición de programas de RAPID
- Inicio, ejecución paso a paso y detención del programa, por ejemplo, durante la comprobación de un programa
- Ajuste de posiciones programadas

5.7.4. Cambio del modo manual al modo automático

	Acción	Información
1	Sitúe el selector de modo de funcionamiento de modo manual a modo automático	
2	Aparecerá una ventana en la que le pedirá que confirme el cambio de modo de funcionamiento en el controlador.	
3	Seleccione OK para cerrar la ventana. Si cambia el selector al modo manual, la ventana se cierra + automáticamente.	

5.7.5. Ejecución de una rutina determinada

Descripción general

Cuando se inicia un programa la ejecución se inicia desde el puntero de programa. Para iniciar la ejecución desde otra rutina, desplace el puntero de programa hacia la rutina.

Requisitos previos

Para ejecutar una rutina específica debe tener cargado el módulo que contiene la rutina y el controlador debe estar en modo manual y parado.

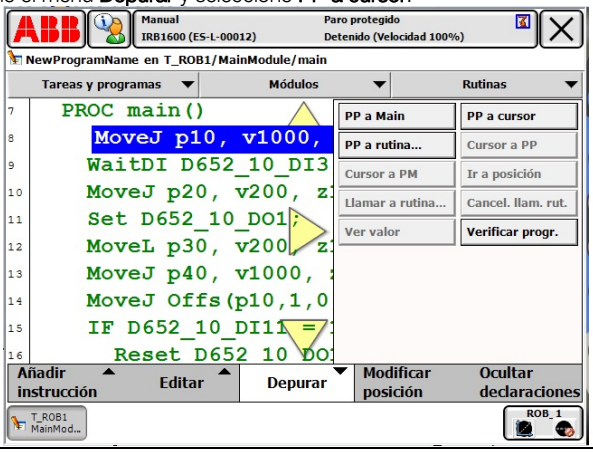
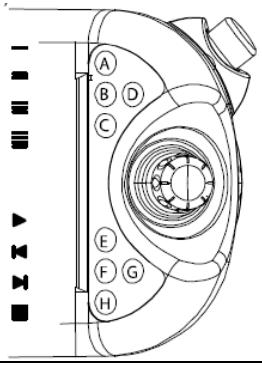
Ejecución de una rutina determinada

Este procedimiento se describe como ejecutar una rutina específica trasladando a ella el puntero de programa.

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione **Depurar** y a continuación **Mover PP a rutina** para situar el puntero de programa al principio de la rutina.
3	Presione el botón **Iniciar** de la Unidad de Programación.

5.7.6. Ejecución del programa a partir de una instrucción determinada

Para ejecutar una instrucción específica debe tener el puntero de programa en la rutina a la que pertenece la instrucción.

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione la instrucción del programa en la que desee iniciar la ejecución. Seleccione el menú **Depurar** y seleccione **PP a cursor**. 
3	**PELIGRO !!** ¡ Asegurarse de que no hay nadie dentro del área de trabajo del robot ¡
3	Presione el botón **Iniciar (▶).** El robot ejecutara la instrucción y empezará a moverse a la posición programada. 

6. Ejecución en producción

6.1. Arranque de programas

Arranque de programas

En este procedimiento se detalla cómo arrancar por primera vez un programa o cómo reanudar la ejecución de un programa que ha sido detenido anteriormente.

	Acción	Información
1	Compruebe que ha hecho todos los preparativos necesarios en el robot y en la célula de robot y que no haya ningún obstáculo dentro del área de trabajo del robot. ¡Asegúrese de que no haya nadie dentro de la célula de robot!	
2	Seleccione modo de funcionamiento en el controlador.	
3	Presione el pulsador Motores ON del controlador para activar el robot.	Pulsador de "Motores ON"  Selector de modo de funcionamiento
4	¿Tiene cargado un programa? En caso afirmativo, continúe en el paso siguiente. En caso negativo, cargue un programa.	
5	En caso necesario, seleccione el modo de funcionamiento y la velocidad en el menú de configuración rápida	
6	En el modo automático: 1 Presione el botón **Iniciar** del Flex-Pendant para iniciar el programa. En el modo manual: 1 Seleccione el modo de inicio. 2 Presione y mantenga presionado el dispositivo de habilitación. 3 Presione el botón **Iniciar** del Flex-Pendant para iniciar el programa.	

7	¿Se muestra la ventana de diálogo **Volver a trayectoria**? En caso afirmativo, devuelva el robot a la trayectoria con un método adecuado. De lo contrario, continúe.	
8	Si se muestra la ventana de diálogo **El cursor no coincide con el PP** toque **PP** o **Cursor** para seleccionar dónde debe iniciarse el programa. A continuación, pulse de nuevo el botón Iniciar.	

Reanudación de la ejecución tras cambios en el programa

Siempre puede reanudar un programa incluso si ha hecho cambios en él.

En el modo automático, es posible que aparezca una ventana de diálogo de advertencia para evitar que reinicie el programa si no conoce las posibles consecuencias.

	Si….	a continuación seleccione…
1	Está seguro de que los cambios que ha realizado no están en conflicto con la posición actual del robot y de que el programa puede continuar sin lesionar a nadie ni causar daños a otros equipos.	**Sí**
2	No está seguro de las consecuencias que podrían tener sus cambios y desea investigarlas con más detalle.	**No**

Reinicio desde el principio

Puede reiniciar los programas desde la ventana Producción o desde el Editor de programas.

Desde la **ventana de producción**:

	Acción
1	En el menú **ABB**, seleccione **Ventana de Producción**.
2	Seleccione **PP a main**
3	Para arrancar el programa, pulse el botón **Iniciar** de la unidad de programación

Desde el **Editor de programas:**

	Acción
1	En el menú **ABB**, seleccione **Editor de programas**.
2	Seleccione **Depurar**
2	Seleccione **PP a main**
3	Para arrancar el programa, pulse el botón **Iniciar** de la unidad de programación

6.2. Paro de programas

Paro de programas

	Acción
1	Compruebe que la operación en curso se encuentre en un estado que permita la parada.
2	Asegúrese de que no supone ningún peligro el paro del programa
3	Presione el botón **Detener** de la Unidad de Programación.

¡PELIGRO!
No utilice el botón Detener en situaciones de emergencia. Utilice el pulsador de paro de emergencia.
La detención de un programa con el botón Detener no significa que el robot deje de moverse inmediatamente.

6.3. Regreso del robot a la trayectoria

Sobre las trayectorias y zonas de retorno

Mientras se ejecuta un programa, el robot se encuentra en *trayectoria*, lo que significa que esta siguiendo la secuencia de posiciones deseada.

Si se detiene el programa, el robot sigue en la trayectoria a no ser que se cambie su posición. En este caso, se considera que está fuera de la trayectoria. Sin embargo, si el robot es detenido por un paro de emergencia o de seguridad, también puede quedar fuera de la trayectoria.

Si el robot detenido está dentro de la zona de retorno a la trayectoria es posible reanudar el programa y que el robot vuelva a la trayectoria y siga ejecutando el programa.

Recuerde que no hay ninguna forma de predecir el movimiento de retorno exacto del robot.

Regreso a la trayectoria

	Acción
1	Asegúrese de que no haya ningún obstáculo que obstruya el camino y que la carga útil o los objetos de trabajo estén situados correctamente.
2	En caso necesario, cambie el sistema al modo automático y presione el botón Motores ON del controlador para activar los motores del robot.
3	Presione el botón Iniciar del FlexPendant para reanudar la ejecución desde el punto en que se detuvo. Ocurrirá una de las cosas siguientes: • El robot o el eje vuelven lentamente a la trayectoria y la ejecución se reanuda. • Aparecerá la ventana de diálogo **Petición de recuperación**.
4	Si se muestra la ventana de diálogo **Petición de recuperación**, seleccione la acción adecuada.

Selección de la acción en la ventana Petición de recuperación

Si….	a continuación seleccione…
Desea volver a la trayectoria y continuar con el programa	Sí
Desea volver a la siguiente posición de objetivo y continuar con el programa	No
No desea continuar con el programa	Cancelar

7. Manejo de entradas y salidas, E/S

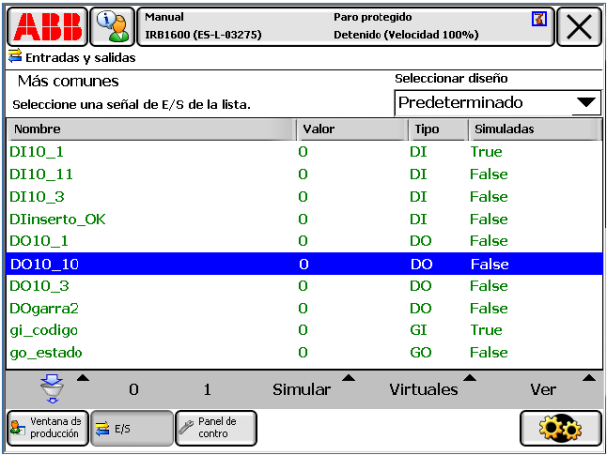
7.1. Entradas y salidas, E/S

Descripción general

La ventana de Entradas y Salidas se utiliza para visualizar y modificar los valores de las señales de E/S. las salidas pueden ser modificado su valor utilizando la opción **0** y **1**, en el caso de las entradas solo podrá ser modificado su valor si previamente son convertida en una señal simulada. Las señales se configuran utilizando los parámetros del sistema.

Visualización de señales

	Acción
1	En el menú ABB, seleccione **Entradas y salidas**. Aparece la lista de señales E/S más comunes.
2	Seleccione el menú **Ver** y a continuación seleccione la lista que desee visualizar de E/S, por ejemplo **todas las señales, entradas digitales, salidas digitales...**



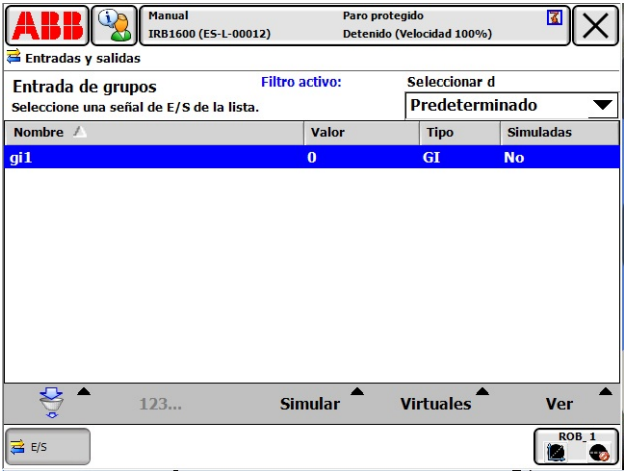
7.2. Simulación y cambio de valores de señales

Una señal puede ser convertida en una señal simulada y su valor puede ser cambiado.

	Acción
1	En el menú **ABB**, seleccione **E/S**. Aparece una lista con las señales más comunes.
2	Seleccione una señal.
3	Seleccione **Simular** para convertir la señal en una señal simulada. Seleccione **Eliminar simulación** para eliminar la simulación de la señal.
4	Seleccione **0 o 1** Para cambiar el valor de la señal. EN el caso de las señales analógicas, seleccione **123...** para cambiar el valor de la señal. Aparecerá el teclado numérico en pantalla. Introduzca el nuevo valor y seleccione **OK**.

7.3. Visualización de un grupo de señales

Visualización de un grupo de señales

	Acción
1	En el menú **ABB**, seleccione **E/S**. Aparece una lista con las señales más comunes.
2	En el menú **Ver**, seleccione **grupos entrada o grupos salidas**. 
3	Seleccione el grupo de señales de la lista y después **Propiedades**, y aparecerán las propiedades del grupo de señales.

8. Manejo del registro de eventos y errores- Copias de Seguridad.

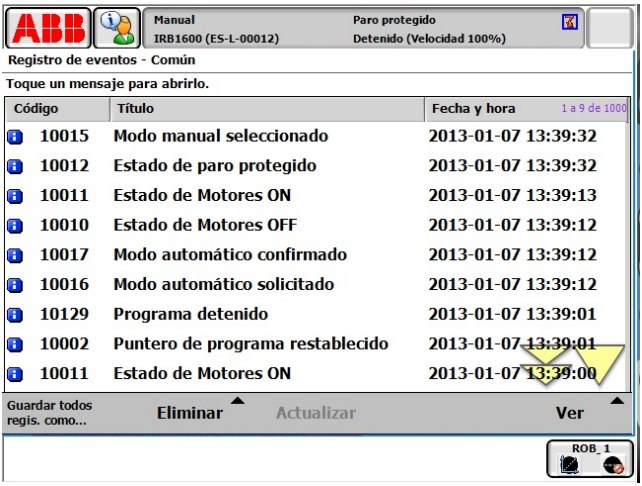
8.1. Acceso al registro de eventos y errores.

Registro de eventos y errores

Abra el registro de eventos y errores para:

- Ver todas las entradas de eventos actuales.
- Estudiar entradas de eventos concretas en detalle.
- Manejar las entradas de eventos del registro, por ejemplo para guardarlas o eliminarlas.

Apertura y cierre del registro de eventos

	Acción
1	Seleccione la **barra de estado** y aparecerá la ventana d e estado.
2	Seleccione **registro de eventos**. 
3	Si el contenido del registro no cabe en una sola pantalla, puede desplazarlo.
4	Seleccione una entrada del registro para ver el mensaje del evento.
5	Seleccione de nuevo la barra de estado para cerrar el registro.

8.2. Eliminación de los registros de eventos

¿En qué casos es necesario eliminar los registros de eventos?

Con frecuencia, la eliminación de entradas del registro suele ser una buena forma de rastrear posibles fallos, dado que se eliminan las entradas antiguas y no significativas, que ya no están relacionadas con el problema que está intentando resolver.

Eliminación de todas las entradas del registro

	Acción
1	Seleccione la **barra de estado** para abrir el registro de eventos o desde el **menú ABB** seleccione **registro de eventos**.
2	En el menú **Ver**, seleccionar **Común**
3	Seleccione **Eliminar** y a continuación **Eliminar todos los registros**. Aparece una ventana de diálogo de confirmación.
4	Seleccione **Sí** para borrar todos los registros o **No** para mantenerlo sin cambios.

Eliminación de las entradas del registro de una categoría determinada

	Acción
1	Seleccione la **barra de estado** para abrir el registro de eventos o desde el **menú ABB** seleccione **registro de eventos**.
2	En el menú **Ver**, seleccionar la categoría que desee eliminar
3	Seleccione **Eliminar** y a continuación **Eliminar registro**. Aparece una ventana de diálogo de confirmación.
4	Seleccione **Sí** para borrar todos los registros o **No** para mantenerlo sin cambios.

8.3. Guardado de los registros de eventos

¿En qué casos es necesario guardar los registros de eventos?

Debe guardar las entradas del registro si:

- Necesita vaciar el registro pero desea conservar las entradas actuales para su visualización posterior.
- Desea enviar los registros de eventos y errores a la asistencia técnica para solucionar un problema.
- Desea conservar los registros de eventos como material de referencia en el futuro.

¡NOTA!

El registro puede albergar hasta 20 entradas por categoría y hasta 150 entradas en la lista conjunta. Cuando el búfer se llena, las entradas más antiguas se sobrescriben y se pierden.

No existe ninguna forma de recuperar estas entradas de registro perdidas.

Guardado de todas las entradas del registro

	Acción
1	Seleccione la **barra de estado** para abrir el registro de eventos o desde el **menú ABB** seleccione **registro de eventos**.
2	Seleccione **Guardar todos los registros como**. Aparece la ventana de diálogo de archivo.
3	Si desea guardar el registro en otra carpeta, búsquela y ábrala.
4	En el cuadro **Nombre de archivo**, escriba un nombre para el archivo.
5	Seleccione **OK.**

8.4. Copia de seguridad y restauración de sistemas

8.4.1. ¿Qué se guarda en la copia de seguridad?

¿Qué se guarda?

La función de copia de seguridad guarda todos los parámetros del sistema, módulos de sistema y módulos de programa de un contexto.

Los datos se guardan en un directorio especificado por el usuario.

Este directorio se divide en cuatro subdirectorios, llamados Backinfo, Home, Rapid y Syspar.

El archivo System.xml también se guarda en el directorio ../backup (el directorio raíz) y contiene la configuración del usuario.

Carpeta	Descripción
Backinfo	Contiene archivos con información relacionada con el sistema del controlador.
Home	Copia de la carpeta con el mismo nombre situada en el disco duro del controlador.
Rapid	Carpeta donde se guardan todos los módulos de programa y sistema que están corriendo en la RAM del controlador.
Syspar	Carpeta donde se guardan todos los parámetros del controlador.

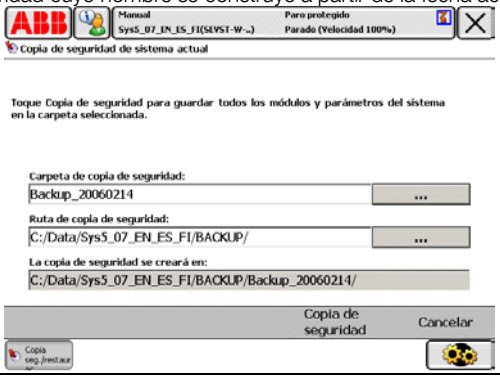
8.4.2. Copia de seguridad del sistema

¿En qué situaciones lo necesito?

ABB recomienda hacer una copia de seguridad:

- Antes de instalar una nueva versión de RobotWare.
- Antes de hacer cualquier cambio importante en las instrucciones y/o los parámetros, para permitir la recuperación de los valores anteriores.
- Después de hacer cualquier cambio importante en las instrucciones y/o los parámetros y comprobar los nuevos valores, para poder guardar los nuevos valores correctos.

Copia de seguridad del sistema

Paso	Acción
1	Toque el menú **ABB** y toque **Copia de seguridad y restauración**.
2	Toque **Copia de seguridad de sistema actual**. Aparece una pantalla que muestra la ruta seleccionada.
3	¿Es correcta la ruta de copia de seguridad mostrada? Sí es así: Toque **Copia de seguridad** para realizar la copia de seguridad en el directorio seleccionado. Se crea un archivo de copia de seguridad cuyo nombre se construye a partir de la fecha actual. Si no es así: Toque "**...**" a la derecha de la ruta de copia de seguridad y seleccione un directorio. A continuación, toque **Copia de seguridad**. Se crea una carpeta de copia de seguridad cuyo nombre se construye a partir de la fecha actual. 

8.4.3. Restauración del sistema

¿En qué situaciones lo necesito?

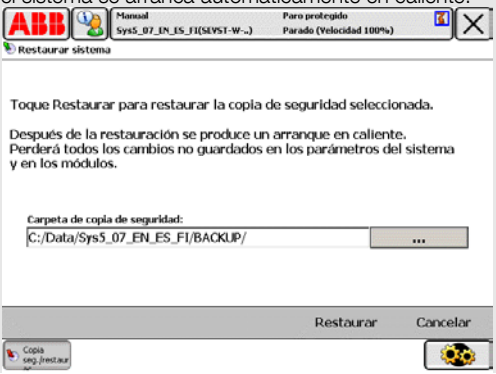
ABB recomienda hacer una restauración:

- Si sospecha que el archivo de programa está dañado.
- Si alguno de los cambios realizados en las instrucciones y/o los valores de los parámetros no ha dado buen resultado y desea recuperar los valores anteriores.

Durante la restauración, todos los parámetros de sistema se sustituyen y todos los módulos se cargan desde el directorio de la copia de seguridad.

El directorio Home se copia de nuevo al directorio HOME del nuevo sistema durante el arranque en caliente.

Restauración del sistema

Paso	Acción
1	En el menú **ABB**, toque **Copia de seguridad y restauración**.
2	Toque **Restaurar sistema.** Aparece una pantalla que muestra la ruta seleccionada.
3	¿Es correcta la carpeta de copia de seguridad mostrada? Sí es así: Toque **Restaurar** para realizar la restauración. Se realiza la restauración y el sistema se arranca automáticamente en caliente. Si no es así: Toque "**...**" a la derecha de la carpeta de copia de seguridad y seleccione un directorio. A continuación, toque **Restaurar**. Se realiza la restauración y el sistema se arranca automáticamente en caliente. 

9. Descripciones de términos y conceptos

9.1. ¿Qué es el sistema de robot?

Descripción

El concepto *sistema de robot* comprende el o los manipuladores, el módulo de control, el módulo de accionamiento y todos los equipos controlados por el controlador (herramienta, sensores, etc.).

Incluye todo el hardware y software necesario para hacer funcionar el robot.

El hardware y software específico de una aplicación, por ejemplo los equipos de soldadura por puntos, no se incluyen en este concepto.

9.2. ¿Qué son los robots, manipuladores y posicionadores?

Manipulador

El término *manipulador* es un término genérico para las unidades mecánicas que se utilizan para mover objetos, herramientas, etc. El término *manipulador* incluye tanto al robot como al posicionador.

Robot

Un *robot* es una unidad mecánica dotada de un TCP. El término robot no incluye el controlador.

Posicionador

Un *posicionador* es una unidad mecánica utilizada para mover un objeto de trabajo. Puede tener uno o varios ejes, normalmente no más de 3 ejes. Los posicionadores no suelen tener TCP.

Unidad mecánica

Una *unidad mecánica* puede ser movida, puede tratarse de un robot, un eje externo sencillo o un conjunto de ejes externos, por ejemplo un posicionador de dos ejes.

9.3. ¿Qué es una herramienta?

Herramienta

Una herramienta es un objeto que puede montarse directa o indirectamente sobre la brida de sujeción del robot, o montarse en una posición fija dentro del área de trabajo del robot.

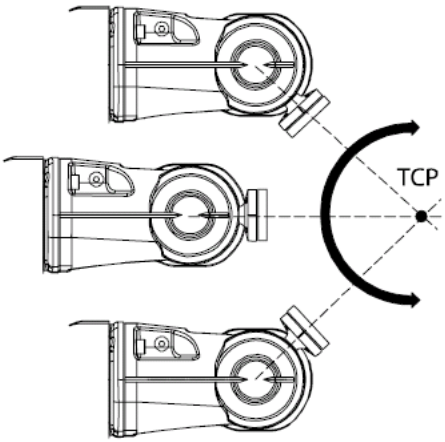
Todas las herramientas deben tener definido un TCP (punto central de la herramienta).

Cada herramienta que pueda ser utilizada por el robot debe ser medida y sus datos almacenados, para conseguir un posicionamiento exacto del punto central de la herramienta.

9.4. ¿Qué es el punto central de la herramienta?

Figura

La figura siguiente muestra el hecho de que el punto central de la herramienta (TCP) es el punto alrededor del cual se define la orientación de muñeca existente entre la herramienta y el manipulador.



Descripción

El punto central de la herramienta (TCP) es el punto respecto del cual se definen todas las posiciones del robot. Normalmente, el punto central de la herramienta se define respecto de una posición de la brida de sujeción del manipulador.

El punto central de la herramienta se mueve hasta la posición de destino programada. Este punto también constituye el origen del sistema de coordenadas de la herramienta.

El sistema de robot puede manejar varias definiciones de punto central de la herramienta a la vez, pero sólo puede estar activa una de ellas en cada momento.

Existen dos tipos básicos de puntos centrales de la herramienta: móvil o fijo.

Punto central de herramienta móvil

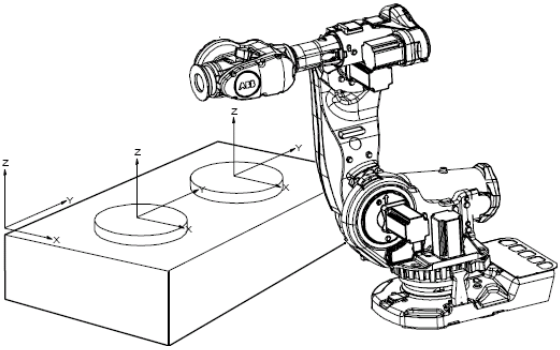
La inmensa mayoría de aplicaciones se basan en un punto central de herramienta móvil, es decir, un punto central de la herramienta que se mueve en el espacio junto con el manipulador.

Un punto central de herramienta móvil típico puede definirse respecto de otro punto, por ejemplo la punta de una pistola de soldadura al arco, el centro de una pistola de soldadura por puntos o el extremo de una herramienta de perfilado.

Punto central de herramienta fijo

En algunas aplicaciones se utiliza un punto central de herramienta fijo, por ejemplo cuando se utiliza una pistola de soldadura por puntos fija. En estos casos, el punto central de la herramienta puede definirse respecto del equipo fijo, en lugar de respecto del manipulador móvil.

9.5. ¿Qué es un objeto de trabajo?



Descripción

Un objeto de trabajo es un sistema de coordenadas que tiene asociadas determinadas propiedades específicas. Se utiliza principalmente para simplificar la programación durante la edición de programas debido a los desplazamientos asociados a tareas, objetos, procesos y otros elementos concretos.

El sistema de coordenadas del objeto de trabajo debe ser definido en dos bases de coordenadas: la base de coordenadas del usuario (dependiente de la base de coordenadas mundo) y la base de coordenadas del objeto (dependiente de la base de coordenadas del usuario).

Con frecuencia, los objetos de trabajo se crean para simplificar el movimiento a lo largo de las superficies del objeto. Puede tener creados más de un objeto de trabajo, de forma que debe decidir cuál debe usarse para el movimiento.

9.6. ¿Qué es un sistema de coordenadas?

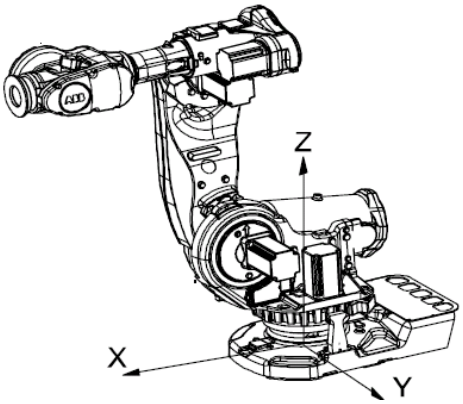
Descripción general

Un sistema de coordenadas define un plano o un espacio con ejes, partiendo de un punto fijo conocido como origen. Los objetivos y las posiciones de robot se localizan mediante medidas a lo largo de los ejes de los sistemas de coordenadas.

Los robots utilizan varios sistemas de coordenadas, cada uno de ellos adecuado para tipos concretos de movimientos o programaciones.

- El sistema de coordenadas de la base se encuentra en la base del robot. Es la forma más fácil de mover únicamente el robot de una posición a otra.
- El sistema de coordenadas del objeto de trabajo depende de la pieza de trabajo y con frecuencia es el más adecuado para la programación del robot.
- El sistema de coordenadas de la herramienta define la posición de la herramienta que utiliza el robot al alcanzar los objetivos programados.
- El sistema de coordenadas mundo define la célula de robot. Todos los demás sistemas de coordenadas dependen del sistema de coordenadas mundo, ya sea de forma directa o indirectamente. Resulta útil en los movimientos, los movimientos en general y el manejo de estaciones y células con varios robots o bien robots movidos por ejes externos.
- El sistema de coordenadas del usuario resulta útil a la hora de representar equipos que sostienen otros sistemas de coordenadas, por ejemplo objetos de trabajo.

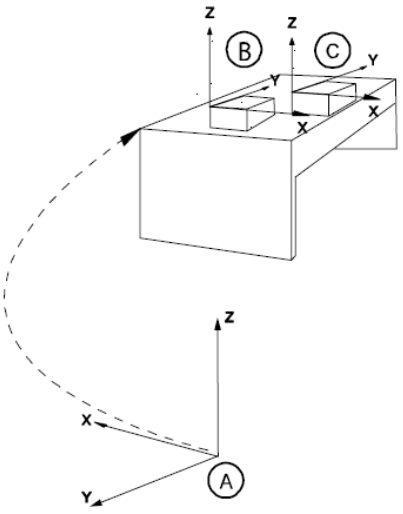
El sistema de coordenadas de la base



El sistema de coordenadas de la base tiene su punto cero en la base del robot, lo que hace que sus movimientos resulten predecibles en el caso de los robots con montaje fijo. Por tanto, resulta útil a la hora de mover un robot de una posición a otra. A la hora de programar un robot, suelen resultar más adecuados otros sistemas de coordenadas, como el sistema de coordenadas del objeto de trabajo.

Si se encuentra en pie delante del robot y realiza un movimiento en el sistema de coordenadas de la base y en un sistema de robot configurado de la forma normal, al mover el joystick hacia usted, el robot se mueve a lo largo del eje X, mientras que el movimiento del joystick hacia los lados hace que el robot se mueva a lo largo del eje Y. Al girar el joystick, el robot se mueve a lo largo del eje Z.

El sistema de coordenadas del objeto de trabajo



1	Sistema de coordenadas mundo
2	Sistema de coordenadas objeto de trabajo 1
3	Sistema de coordenadas objeto de trabajo 2

El sistema de coordenadas del objeto de trabajo se corresponde con la pieza de trabajo: Define el posicionamiento de la pieza de trabajo respecto del sistema de coordenadas mundo (o respecto de cualquier otro sistema de coordenadas).

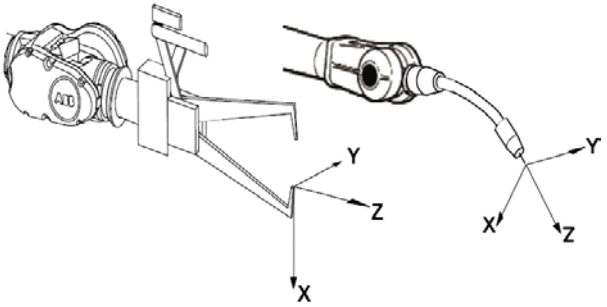
El sistema de coordenadas del objeto de trabajo debe ser definido en dos bases de coordenadas: la base de coordenadas del usuario (dependiente de la base de coordenadas mundo) y la base de coordenadas del objeto (dependiente de la base de coordenadas del usuario).

Un mismo robot puede tener varios sistemas de coordenadas de objetos de trabajo, ya sea para representar a varias piezas de trabajo diferentes o se trate de varias copias de una misma pieza de trabajo en ubicaciones diferentes.

Es en estos sistemas de coordenadas de objetos de trabajo donde se crean los objetivos y trayectorias durante la programación del robot. Con ello se consiguen un sinfín de ventajas:

- Al reposicionar el objeto de trabajo en la estación, sólo es necesario cambiar la posición del sistema de coordenadas del objeto de trabajo para que todas las trayectorias se actualicen a la vez.
- Permite el trabajo con objetos de trabajo movidos por ejes externos o tracks, dado que es posible mover la totalidad del objeto de trabajo junto con sus trayectorias.

El sistema de coordenadas de la herramienta



El sistema de coordenadas de la herramienta tiene su origen en el punto central de la herramienta seleccionada. Por tanto, define la posición y la orientación de la herramienta. El sistema de coordenadas de la herramienta se abrevia con frecuencia como TCP ("Tool Center Point", punto central de la herramienta).

El TCP es el punto movido por el robot hasta las posiciones programadas al ejecutar el programa.

Esto significa que si se cambia en una herramienta (la orientación del sistema de coordenadas de la herramienta), los movimientos del robot cambiarán de forma que el nuevo TCP alcance su objetivo.

Todos los robots tienen un sistema de coordenadas de herramienta predefinido, denominado *"tool0"* situado en el centro de la brida de sujeción del robot.

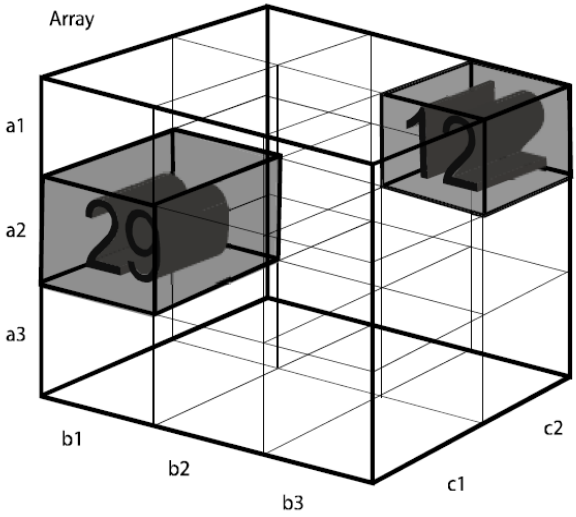
Al mover un robot, el sistema de coordenadas de la herramienta resulta útil para facilitar el cambio de la orientación de la herramienta durante la programación de movimientos.

¿Qué es una matriz de datos?

Una matriz de datos es un tipo especial de variable: las variables normales pueden contener un solo valor de dato, pero las matrices pueden contener varios.

Pueden describirse como tablas que pueden tener una o más dimensiones. Estas tablas pueden rellenarse con datos (por ejemplo valores numéricos, cadenas de caracteres o variables) para usarlos durante la programación o el manejo del sistema de robot.

A continuación se muestra un ejemplo de matriz tridimensional:



Esta matriz, denominada “Array”, se define con sus tres dimensiones, a, b y c. La dimensión a tiene tres filas, la b tiene tres filas (columnas) y la c tiene dos filas.

La matriz y su contenido pueden escribirse como `Array {a, b, c}`.

Ejemplo 1: `Array {2, 1, 1}=29`

Ejemplo 2: `Array {1, 3, 2}=12`

Manual Usuario Robot IRB120

Guía de Usuario

Tipos de Datos

Instrucciones

Funciones

Contenido

bool Valores lógicos

Bool se usa para los valores lógicos verdadero/falso.

Descripción

El valor del tipo de dato *bool* puede ser *TRUE (Verdadero)* o *FALSE (Falso)*.

Ejemplos

bflag1 := TRUE;
bflag1 tiene asignado el valor TRUE.

VAR bool bvalor1;
VAR num reg1;
.
IF reg1 > 100 bvalor1:=True;
bvalor1 tiene asignado el valor *TRUE* si *reg1* es mayor que *100* de lo contrario, le será asignado *FALSE*.

IF valor1 Set doSalida1;
La señal *doSalida1* será activada si *valor1* es *TRUE*.

valor1 := reg1 > 100;
valor2 := reg1 > 20 AND NOT valor1;
valor2 tiene asignado el valor *TRUE* si *reg1* se encuentra entre *20* y *100*.

Estructura

Bool es un dato de tipo atómico, lo que significa que no incluye otros tipos de datos.

clock Medida del tiempo

C*lock* se usa para la medida del tiempo. Las aplicaciones de los datos *clock* son su utilización como un cronómetro.

Descripción

Los datos del tipo *clock* almacenan una medida del tiempo en segundos y tienen una resolución de 0,01 segundos.

Ejemplo

VAR clock ckreloj1;
ClkReset ckreloj1;
El reloj, *ckreloj1*, es declarado y puesto a cero. Antes de utilizar *ClkReset*, *ClkStart*, *ClkStop* y *ClkRead*, se deberá declarar una variable de tipo *clock* en el programa.

Limitaciones

El tiempo máximo que puede ser almacenado en una variable de tipo *clock* es de aproximadamente 49 días (4.294.967 segundos). Las instrucciones *ClkStart*, *ClkStop* y *ClkRead* indican desbordamiento de reloj en el caso poco probable en que este hecho ocurra.

Un reloj deberá declararse siempre como una variable *VAR*, y no puede declararse como un tipo de variable persistente *PERS.*

Estructura

Clock es un tipo de dato atómico y tiene una estructura interna inaccesible.

Características

Clock es un tipo de datos sin valor y no puede ser utilizado en operaciones orientadas a valores.

confdata Datos de configuración del robot

Confdata sirve para definir las configuraciones de los ejes del robot.

Descripción

Todas las posiciones del robot están definidas y almacenadas utilizando coordenadas rectangulares. Al calcular las posiciones de los ejes correspondientes, pueden existir muchas veces dos o más soluciones posibles. Ello significa que el robot es capaz de alcanzar la misma posición, o sea, la herramienta es capaz de llegar a la misma posición y con la misma orientación, mediante diferentes posiciones y configuraciones de los ejes del robot.

Para determinar sin ambigüedad cual es la posición y orientación correcta, se deberá definir la configuración del robot utilizando los valores de cuatro ejes. Estos valores definen el cuadrante de la vuelta en donde están situados estos cuatro ejes del robot. Los cuadrantes están numerados según el orden 0, 1, 2, etc. (también pueden ser negativos). El número del cuadrante está asociado al ángulo del eje. Para cada eje, el cuadrante 0 es el primer cuarto, de 0 a 90°, en una dirección positiva partiendo desde la posición 0; el cuadrante 1 es el cuarto siguiente, de 90 a 180°, etc. El cuadrante -1 es el cuarto 0° a (-90°), etc. (véase la Figura 1).

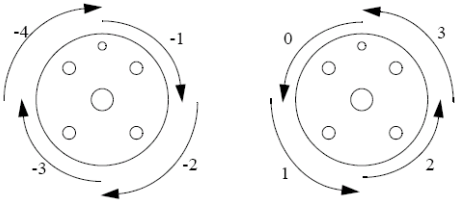


Figura 1 Los cuadrantes de configuración del eje 6.

Para los ejes lineales, el valor esta definido en el intervalo de 1 metro del eje. Para cada eje, el valor 0 es una posición entre 0 y 1 metro, 1 es una posición entre 1 y 2 metros, etc.... Para los valores negativos, -1 es una posición entre 0 y -1 metros, etc...Vease la figura 2

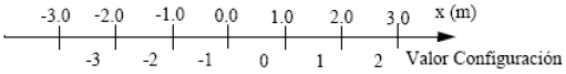


Figura 2 Valores de configuración para un eje lineal

Datos de configuración del robot para los modelos IRB140, 1600, 6600, 6650, 7600

Hay tres singularidades dentro del área de trabajo de estos robots.

cf1 se usa para el eje 1
cf4 se usa para el eje 4
cf6 no se usa para el eje 6.
cfx se usa para seleccionar uno de las 8 posibles configuraciones del robot numeradas desde el 0 hasta el 7. En la siguiente tabla se describe cada una de estas configuraciones en términos de como el robot se posiciona con respecto a las tres singularidades.

Cfx	Centro muñeca respecto al eje 1	Centro muñeca respecto eje inferior	Angulo eje 5
0	Delante	Delante	Positivo
1	Delante	Delante	Negativo
2	Delante	Detrás	Positivo
3	Delante	Detrás	Negativo
4	Detrás	Delante	Positivo
5	Detrás	Delante	Negativo
6	Detrás	Detrás	Positivo
67	Detrás	Detrás	Negativo

En las siguientes figuras se puede ver de como se puede conseguir la misma posición y orientación de la herramienta usando ocho diferentes configuraciones.

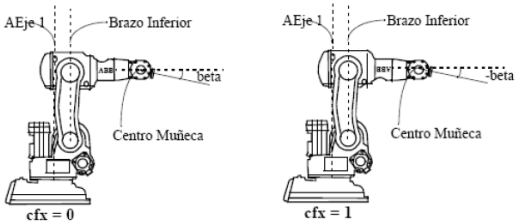


Figura 3 Ejemplo de configuración 0 y 1. Véase los diferentes signos del ángulo del eje 5.

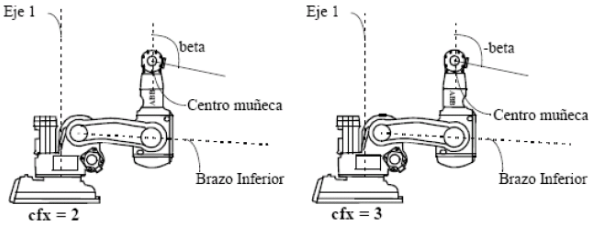


Figura 4 Ejemplo de configuración 0 y 1. Véase los diferentes signos del ángulo del eje 5

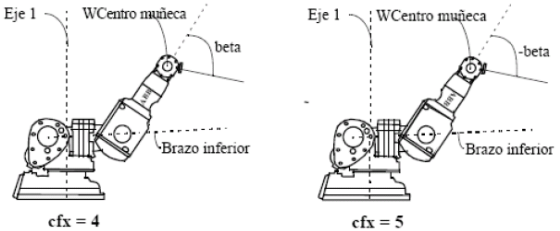


Figura 5 Ejemplo de configuración 0 y 1. Véase los diferentes signos del ángulo del eje 5

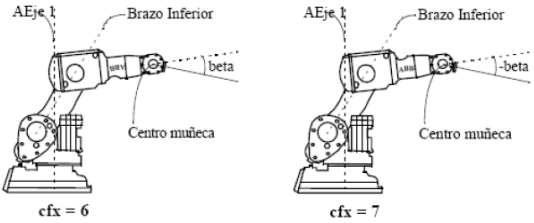


Figura 6 Ejemplo de configuración 0 y 1. Véase los diferentes signos del ángulo del eje 5

Componentes

cf1 **Tipo de dato:** *num*

Eje rotacional: El cuadrante utilizado por el eje 1, expresado con un número entero positivo o negativo.

Eje lineal: La medida correspondiente al intervalo del eje 1, expresado como un entero positivo o negativo.

cf4 **Tipo de dato:** *num*

Eje rotacional: El cuadrante utilizado por el eje 4, expresado con un número entero negativo o positivo.

Eje lineal: La medida correspondiente al intervalo del eje 4, expresado como un entero positivo o negativo.

cf6 **Tipo de dato:** *num*

Eje rotacional: El cuadrante utilizado por el eje 6, expresado con un número entero negativo o positivo.

Eje lineal: La medida correspondiente al intervalo del eje 6, expresado como un entero positivo o negativo.

cfx **Tipo de dato:** *num*

Eje rotacional: Para el robot IRB 140, 1600, 6600, 6650, 7600, la configuración actual del robot expresada por un entero comprendido entre 0 y 7.

Para el robot IRB5400: El cuadrante utilizado por el eje 6 expresado con un número entero negativo o positivo.

Para otros robots: El cuadrante utilizado por el eje 2, expresado con un número entero negativo o positivo.

Eje lineal: La medida correspondiente al intervalo del eje 2, expresado como un entero positivo o negativo.

Ejemplos

VAR confdata conf5 := [1, -1, 0, 0]
La configuración del robot *conf5* está definida de la siguiente manera:
La configuración del eje 1 del robot es el cuadrante *1*, es decir, 90-180º.
La configuración del eje 4 del robot es el cuadrante *-1*, es decir, 0-(-90º).
La configuración del eje 6 del robot es el cuadrante *0*, es decir, 0 – 90º.
La configuración del eje 5 del robot es el cuadrante *0*, es decir, 0 – 90º.

Estructura

< estructura de *confdata* >
< *cf1* de tipo *num* >
< *cf4* de tipo *num* >
< *cf6* de tipo *num* >
< *cfx* de tipo *num* >

jointtarget Datos de posición de los ejes

Jointtarget sirve para definir la posición a la que el robot y los ejes externos se moverán utilizando la instrucción *MoveAbsJ*.

Descripción

Jointtarget define la posición de cada uno de los ejes, tanto para el robot como para los ejes externos.

Componentes

robax **Tipo de dato:** *robjoint*

Son las posiciones de los ejes del robot expresadas en grados.
La posición de los ejes se define como la rotación en grados del eje correspondiente en una dirección positiva o negativa a partir de la posición de calibración del eje.

extax **Tipo de dato:** *extjoint*

La posición de los ejes externos.
La posición se define según las siguientes indicaciones para cada uno de los ejes (*eax_a*, *eax_b ... eax_f*):

- Para los ejes rotacionales, la posición es definida como la rotación, expresada en grados, a partir de la posición de calibración.
- Para los ejes lineales, la posición se define como la distancia, expresada en mm, a partir de la posición de calibración.

Los ejes externos *eax_a ...* son ejes lógicos. La relación existente entre el número del eje lógico y el número del eje físico se encuentra definida en los parámetros del sistema.
El valor 9E9 es el que se asigna a todos los ejes que no estén conectados. En el caso en que los ejes definidos en los datos de posición difieran de los ejes que están realmente conectados durante la ejecución del programa, se aplicará lo siguiente:

- En el caso en que la posición no esté definida en los datos de posición (valor 9E9) no se tendrá en cuenta el valor, si el eje está conectado y no activado. Pero si el eje está activado, el sistema generará un error.
- En el caso en que la posición este definida en los datos de posición y el eje todavía no esté conectado, no se tendrá en cuenta dicho valor.

Ejemplos

CONST jointtarget calib_pos := [[0, 0, 0, 0, 0, 0], [0, 9E9, 9E9, 9E9, 9E9, 9E9]];
La posición de calibración normal para un robot se encuentra definida en *calib_pos* mediante el tipo de dato *jointtarget.* La posición de calibración normal
0 (expresada en grados o en mm) también estará definida para el eje lógico externo *a*. En cambio, no están definidos los ejes externos *b, c, d, e, f.*

Estructura

< estructura de *jointtarget* >
< *robax* de *robjoint* >
< *rax_1* de *num* >
< *rax_2* de *num* >
< *rax_3* de *num* >
< *rax_4* de *num* >
< *rax_5* de *num* >
< *rax_6* de *num* >
< *extax* de *extjoint* >
< *eax_a* de *num* >
< *eax_b* de *num* >
< *eax_c* de *num* >
< *eax_d* de *num* >
< *eax_e* de *num* >
< *eax_f* de *num* >

loaddata Datos de carga

Loaddata sirve para describir las cargas que manipula el robot.
Los datos de carga suelen definir la carga útil (la carga de la pinza se define en la instrucción *GripLoad*) del robot, es decir, la carga soportada por el útil del robot. La carga de la herramienta (peso) está especificada en los datos de la herramienta (*tooldata*) que incluyen los datos de carga.

Descripción

Las cargas definidas sirven para determinar un modelo de dinámica del robot, de forma que los movimientos del robot puedan ser controlados de la mejor manera posible.

Es importante definir siempre la carga actual de la herramienta y cuando se use, la carga útil del robot. Definiciones incorrectas de los datos de carga pueden provocar una sobrecarga de la estructura mecánica del robot.

Si se especifican datos incorrectos de carga, se pueden generar las siguientes consecuencias:

Si el valor de los datos de carga especificados es mayor que el valor real de la carga;

- El robot no será utilizado en sus máximas posibilidades
- Perdida de precisión de la trayectoria incluyendo el riesgo de vibraciones.

Si el valor de los datos de carga especificados es menor que el valor real de la carga;

- Perdida de precisión de la trayectoria incluyendo el riesgo de vibraciones.
- Existe un riesgo de sobrecarga de la estructura mecánica

La carga útil será activada/desactivada utilizando la instrucción *GripLoad*.

Componentes

mass **Tipo de dato:** *num*

El peso de la carga en kg.

cog **Tipo de dato:** *pos*

El centro de gravedad de la carga de la herramienta será expresado utilizando el sistema de coordenadas de la muñeca. Si es una herramienta estacionaria, se referirá al centro de gravedad de la herramienta que sujeta el objeto de trabajo.

El centro de gravedad de una carga útil se definirá utilizando el sistema de coordenadas de la herramienta. Si se usa una herramienta estacionaria, entonces se utilizará el sistema de coordenadas del objeto.

aom **Tipo de dato:** *orient*

Carga de la herramienta (figura 1)
La orientación del sistema de coordenadas definido por los ejes inerciales de la carga de la herramienta. Expresado en el sistema de coordenadas de la muñeca como un cuaternión (q1, q2, q3 y q4). Si se usa una herramienta estacionaria, se referirá a los ejes inerciales de la herramienta que sujeta el objeto de trabajo.

La orientación del sistema de coordenadas de la carga de la herramienta debe coincidir con la orientación del sistema de coordenadas de la muñeca. **Deberá estar siempre definido como 1, 0, 0, 0.**

Carga útil (figura 2)
La orientación del sistema de coordenadas definido por los ejes inerciales de la carga útil. Expresado en el sistema de coordenadas de la herramienta como un cuaternión (q1, q2, q3 y q4). Si se usa una herramienta estacionaria, se utilizará un sistema de coordenadas del objeto.

La orientación del sistema de coordenadas de la carga útil debe coincidir con la orientación del sistema de coordenadas de la muñeca. **Deberá estar siempre activado en 1, 0, 0, 0.**

Debido a esta limitación, la mejor manera consiste en definir la orientación del sistema de coordenadas de la herramienta (tool frame) para que coincida con la orientación del sistema de coordenadas de la muñeca

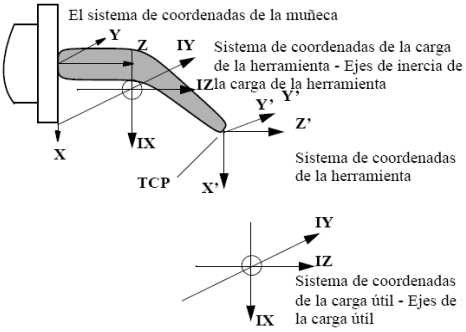


Figura 1 Restricción en la orientación del sistema de coordenadas de la carga de la herramienta y de la carga útil.

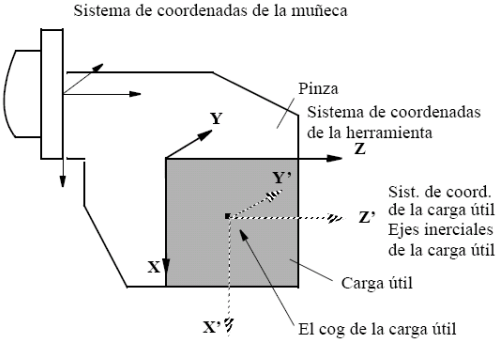


Figura 2 El centro de gravedad de la carga útil y los ejes de inercia.

ix Tipo de dato: *num*

El momento de inercia de la carga alrededor del eje x', en los ejes del sistema de coordenadas de la carga útil y de la carga de la herramienta, expresado en kgm2.

Una definición correcta de los momentos de inercia permitirá una utilización óptima de la planificación de la trayectoria y del control de los ejes. Ello puede ser muy importante cuando por ejemplo se manipulan grandes láminas de metal, etc. Si el momento de inercia para todos los componentes *ix, iy, iz* es equivalente a 0 kgm2, implica una carga puntual.

Normalmente los momentos de inercia sólo deberán ser definidos cuando la distancia entre la brida de montaje y el centro de gravedad sea menor que la dimensión de la carga (véase la Figura 3).

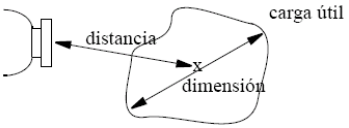


Figura 3 Normalmente, el momento de inercia deberá ser definido cuando la distancia sea menor que la dimensión de la carga.

iy Tipo de dato: *num*

El momento de inercia de la carga alrededor del eje y' expresado en kgm2. Para más información, véase *ix*.

iz Tipo de dato: *num*

El momento de inercia de la carga alrededor del eje z' expresado en kgm2. Para más información, véase *ix*.

Ejemplos

PERS loaddata pieza1 := [5, [50, 0, 50], [1, 0, 0, 0], 0, 0, 0];
Para definir la carga útil de la Figura 2 se deberá utilizar los siguientes valores:

- Peso de *5* kg.
- El centro de gravedad es x = *50*, y = *0*, z = *50* mm en el sistema de coordenadas de la herramienta.
- La carga útil es una carga puntual.

Set dopinza;
WaitTime 0.3;
GripLoad pieza1;
La activación de la carga útil, *pieza1*, se especifica en el mismo momento en que el robot coge la carga *pieza1*.

Reset dopinza;
WaitTime 0.3;
GripLoad load0;
La desconexión de una carga útil, se especifica en el mismo momento en que el robot deja la carga.

Limitaciones

La carga útil deberá ser definida únicamente como una variable persistente (PERS) y no dentro de una rutina. Los valores utilizados serán entonces almacenados al guardar el programa en un disquete y volverán ser utilizados en el momento en que se vuelvan a cargar.

Los argumentos del tipo datos de carga (loaddata) en la instrucción *GripLoad* deberá ser un dato persistente entero (no un elemento de una matriz ni un componente de registro).

Datos predefinidos

La carga *load0* define una carga útil de un peso equivalente a 0 kg., es decir ninguna carga. Esta carga se utiliza como argumento de la instrucción *GripLoad* para desconectar una carga útil.

Siempre se podrá acceder a la carga *load0* desde el programa, sin embargo no podrá ser cambiada, ya que se encuentra definida en el módulo del sistema *BASE*.

PERS loaddata load0 := [0.001, [0, 0, 0.001], [1, 0, 0, 0],0, 0 ,0];

Estructura

< estructura de *loaddata* >
< *mass* de *num* >
< *cog* de *pos* >
< *x* de *num* >
< *y* de *num* >
< *z* de *num* >
< *aom* de *orient* >
< *q1* de *num* >
< *q2* de *num* >
< *q3* de *num* >
< *q4* de *num* >
< *ix* de *num* >
< *iy* de *num* >
< *iz* de *num* >

num Valores numéricos

Num se usa para los valores numéricos; por ejemplo, para los contadores.

Descripción

El valor de un tipo de dato *num* puede ser:

- un número entero; por ejemplo, -5,
- un número decimal; por ejemplo, 3,45.

También puede estar escrito de forma exponencial; por ejemplo, 2E3 (= 2*103 = 2000), 2,5E-2 (= 0,025).

Los números enteros entre -8388607 y +8388608 son siempre almacenados como números enteros exactos.

Los números decimales no son más que números aproximados y no podrán ser utilizados en comparaciones del tipo *es igual a* o *no es igual a.* En el caso de divisiones y operaciones que utilizan números decimales, el resultado será también un número decimal; es decir, no un número entero exacto.
Ej.:

```
a := 10.5;
b := 5;
IF a/b=2 THEN
```

Dado que el resultado de a/b no es un número entero, esta condición no se cumplirá nunca.

Ejemplo

```
VAR num reg1;
```

reg1 := 3;
A *reg1* se le asigna el valor *3*.

a := 10 DIV 3;
División entera donde *a* es el cociente de la división (=3).

Datos Predefinidos

La constante pi (π) ya esta definida en el módulo del sistema *BASE*.
CONST num pi := 3.1415926;

Las constantes EOF_BIN y EOF_NUM ya están definidos en el sistema
CONST num EOF_BIN:= -1;
CONST num EOF_NUM:= 9.998E36;

Limitaciones

El rango de los valores de los números esta limitado de acuerdo con la norma ANSI IEEE 754-1985 (simple precisión) para los formatos de coma flotante.

Estructura

Num es un dato de tipo atómico, lo que significa que no contiene otro tipo de dato.

orient Orientación

Orient se usa para las orientaciones (como la orientación de una herramienta) y las rotaciones (como la rotación de un sistema de coordenadas).

Descripción

La orientación se describe bajo la forma de un cuaternión formado por cuatro elementos: *q1*, *q2*, *q3* y *q4*. Para más información sobre lo que es un cuaternión y como se calcula, véase el apartado ¿Que es un cuaternión? más abajo.

Componentes

q1 **Tipo de dato:** *num*

Cuaternión 1.

q2 **Tipo de dato:** *num*

Cuaternión 2.

q3 **Tipo de dato:** *num*

Cuaternión 3.

q4 **Tipo de dato:** *num*

Cuaternión 4.

Ejemplo

```
VAR orient orientacion1;
.
orientacion1 := [1, 0, 0, 0];
```

La orientación *orientacion1* tiene asignado el valor q1=1, q2-q4=0; estos valores corresponden a ninguna rotación.

Limitaciones

La orientación debe ser normalizada, es decir que la suma de los cuadrados de los cuatro debe ser igual a 1:

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$$

¿Qué es un Cuaternión?

La orientación de un sistema de coordenadas (como la de una herramienta) puede ser descrita como una matriz rotacional que describe la dirección de los ejes del sistema de coordenadas respecto a un sistema de referencia (véase la Figura 1).

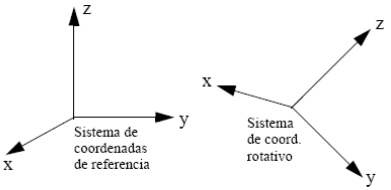


Figura 1 La rotación de un sistema de coordenadas está descrita mediante un cuaternión.

Los ejes de los sistemas de coordenadas rotativos (**x**, **y**, **z**) son vectores que pueden ser expresados en el sistema de coordenadas de referencia según lo siguiente:

x = (x1, x2, x3)
y = (y1, y2, y3)
z = (z1, z2, z3)

Esto significa que el componente x del vector x en el sistema de coordenadas de referencia será x1, el componente y- será x2, etc.
Estos tres vectores pueden ser puestos juntos en una matriz rotacional, donde cada uno de los vectores forma una columna:

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

Un cuaternión no es más que una forma más concisa de describir esta matriz rotacional; los saturninos se calculan basándose en los elementos de la matriz rotacional:

$$q1 = \frac{\sqrt{x_1 + y_2 + z_3 + 1}}{2}$$

$$q2 = \frac{\sqrt{x_1 - y_2 - z_3 + 1}}{2} \qquad \text{signo } q2 = \text{signo } (y3\text{-}z2)$$

$$q3 = \frac{\sqrt{y_2 - x_1 - z_3 + 1}}{2} \qquad \text{signo } q3 = \text{signo } (z1\text{-}x3)$$

$$q4 = \frac{\sqrt{z_3 - x_1 - y_2 + 1}}{2} \qquad \text{signo } q4 = \text{signo } (x2\text{-}y1)$$

Ejemplo 1

Una herramienta está orientada de forma que su eje Z' apunte recto hacia delante (en la misma dirección que el eje X del sistema de coordenadas en la base). El eje Y' de la herramienta corresponde al eje Y del sistema de coordenadas en la base (véase la Figura 3). ¿Cómo está definida la orientación de la herramienta en el dato de posición (robtarget)?

La orientación de la herramienta en una posición programada está normalmente 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 en la base es igual al sistema de coordenadas mundo. Así, la orientación es relativa con el sistema de coordenadas en la base.

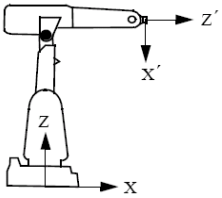


Figura 2 La dirección de una herramienta de acuerdo con el ejemplo 1.

Los ejes estarán relacionados de la siguiente forma:

x' = -**z** = (0, 0, -1)
y' = **y** = (0, 1, 0)
z' = **x** = (1, 0, 0)

Lo cual corresponde a la siguiente matriz rotacional:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

La matriz rotacional proporciona el cuaternión correspondiente:

$$q1 = \frac{\sqrt{0+1+0+1}}{2} = \frac{\sqrt{2}}{2} = 0{,}707$$

$$q2 = \frac{\sqrt{0-1-0+1}}{2} = 0$$

$$q3 = \frac{\sqrt{1-0-0+1}}{2} = \frac{\sqrt{2}}{2} = 0{,}707 \qquad \text{signo } q3 = \text{signo } (1+1) = +$$

$$q4 = \frac{\sqrt{0-0-1+1}}{2} = 0$$

Ejemplo 2

La dirección de la herramienta ha sido girada de 30o en los ejes X'- y Z'-respecto al sistema de coordenadas de la muñeca (véase la Figura 3). ¿Cómo estará definida la orientación de la herramienta en los datos de la herramienta?

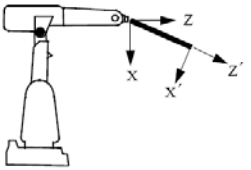


Figura 3 La dirección de la herramienta de acuerdo con el ejemplo2.

Los ejes estarán relacionados de la forma siguiente:

x' = (cos30o, 0, -sin30o)
y' = (0, 1, 0)
z' = (sin30o, 0, cos30o)

Lo cual corresponde a la siguiente matriz rotacional:

$$\begin{bmatrix} \cos 30° & 0 & \sin 30° \\ 0 & 1 & 0 \\ -\sin 30° & 0 & \cos 30° \end{bmatrix}$$

La matriz rotacional proporciona al cuaternión correspondiente:

$$q1 = \frac{\sqrt{\cos 30° + 1 + \cos 30° + 1}}{2} = 0{,}965926$$

$$q2 = \frac{\sqrt{\cos 30° - 1 - \cos 30° + 1}}{2} = 0$$

$$q3 = \frac{\sqrt{1 - \cos 30° - \cos 30° + 1}}{2} = 0{,}258819 \quad \text{signo } q3 = \text{signo } (\sin 30^{o} + \sin 30^{o}) = +$$

$$q4 = \frac{\sqrt{\cos 30° - \cos 30° - 1 + 1}}{2} = 0$$

Estructura

<estructura de *orient*>
<*q1* de *num*>
<*q2* de *num*>
<*q3* de *num*>
<*q4* de *num*>

robtarget Datos de posición

Robtarget sirve para definir la posición del robot y de los ejes externos.

Descripción

Los datos de posición sirven para definir la posición en las instrucciones de posicionamiento a donde deben moverse, tanto los ejes del robot y los ejes externos.

Dado que el robot es capaz de alcanzar una misma posición de varias formas diferentes, también se deberá especificar la configuración de los ejes. Esto servirá para definir los valores de los ejes en el caso de que sean ambiguos, por ejemplo:

- si el robot está en una posición hacia adelante o hacia atrás,
- si el eje 4 esta girado hacia abajo o hacia arriba,
- si el eje 6 esta en la vuelta positiva o negativa.

La posición será definida basándose en el sistema de coordenadas del objeto de trabajo, incluyendo cualquier desplazamiento de programa. Si la posición es programada con cualquier otro objeto de trabajo que el utilizado en la instrucción, el robot no se moverá de la forma esperada.

Asegurarse de que se está utilizando el mismo objeto de trabajo que el que se ha utilizado al programar las instrucciones de posicionamiento. De no seguirse esta recomendación, se pueden provocar daños al personal o al equipo.

Componentes

trans **Tipo de dato:** *pos*

La posición (x, y, z) del punto central de la herramienta expresada en mm.
La posición se especifica respecto al sistema de coordenadas del objeto utilizado, incluyendo el desplazamiento del programa. Si no se especifica ningún objeto de trabajo, se usará el sistema de coordenadas mundo.

rot **Tipo de dato:** *orient*

La orientación de la herramienta expresada bajo la forma de un quaternion (q1, q2, q3 y q4).

La orientación se especifica respecto al sistema de coordenadas del objeto utilizado, incluyendo el desplazamiento del programa. Si no se especifica ningún objeto de trabajo, se usará el sistema de coordenadas mundo.

robconf **Tipo de dato:** *confdata*

La configuración de los ejes del robot (cf1, cf4, cf6 y cfx). Esto se define bajo la forma del cuarto de vuelta donde esta el eje 1, del eje 4 y del eje 6. El primer cuarto de vuelta positivo, 0-90 o, está definido como 0.

Para más información, véase el tipo de dato *confdata*.

extax **Tipo de dato:** *extjoint*

La posición de los ejes externos.

La posición será definida como se indica a continuación para cada eje individual (*eax_a*, *eax_b* ... *eax_f*):

- Para los ejes rotacionales, la posición será definida como la rotación en grados a partir de la posición de calibración.
- Para los ejes lineales, la posición será definida como la distancia en mm a partir de la posición de calibración.

Los ejes externos *eax_a ...* son ejes lógicos. La forma en que el número de los ejes lógicos y el número de los ejes físicos están relacionados entre sí se encuentra definida en los parámetros del sistema.

El valor 9E9 será definido para los ejes que no están conectados. Si los ejes definidos en los datos de posición difieren de los ejes que están conectados en realidad para la ejecución del programa, se aplicará lo siguiente:

- Si la posición no está definida en los datos de posición (el valor 9E9) no se tendrá en cuenta el valor, si el eje está conectado y no está activado. Pero si el eje está activado, el sistema generará un error.
- Si la posición está definida en los datos de posición y que el eje todavía no está conectado, no se tendrá en cuenta el valor.

Ejemplos

CONST robtarget p15 := [[600, 500, 225,3], [1, 0, 0, 0], [1, 1, 0, 0],[11, 12.3, 9E9, 9E9, 9E9, 9E9]];

Una posición *p15* será definida de la siguiente manera:

- La posición del robot: *x = 600*, *y = 500*, *z = 225,3* mm en el sistema de coordenadas del objeto.
- La orientación de la herramienta en la misma dirección que el sistema de coordenadas del objeto.
- La configuración del eje del robot: eje 1 y 4 en el cuarto de vuelta 90-180º, el eje 6 en el cuarto 0-90º.
- La posición de los ejes externos lógicos, a y b, expresada en grados o en mm. (dependiendo del tipo de ejes). Los ejes c y f no están definidos.

```
VAR robtarget p20;
. . .
p20 := CRobT();
p20 := Offs(p20,10,0,0);
```

La posición *p20* esta definida en la misma posición que la posición actual del robot mediante la función *CRobT*. La posición será desplazada *10* mm en la dirección *x*.

Estructura

< estructura de *robtarget* >
< *trans* de *pos* >
< *x* de *num* >
< *y* de *num* >
< *z* de *num* >
< *rot* de *orient* >
< *q1* de *num* >
< *q2* de *num* >
< *q3* de *num* >
< *q4* de *num* >
< *robconf* de *confdata* >
< *cf1* de *num* >
< *cf4* de *num* >
< *cf6* de *num* >
< *cfx* de *num* >
< *extax* de *extjoint* >
< *eax*_a de *num* >
< *eax*_b de *num* >
< *eax*_c de *num* >
< *eax*_d de *num* >
< *eax*_e de *num* >
< *eax*_f de *num* >

speeddata Datos de velocidad

Speeddata sirve para especificar la velocidad a la que se moverán tanto los ejes externos como el robot.

Descripción

Los datos de velocidad definen la velocidad:

- a la que se mueve el punto central de la herramienta (TCP)
- de la reorientación de la herramienta,
- a la que se mueven los ejes externos rotacionales o lineales.

Cuando se combinan diferentes tipos de movimiento, a menudo hay una de las velocidades que limita todos los movimientos. La velocidad de los demás movimientos será reducida de tal manera que todos los movimientos acabarán su ejecución al mismo tiempo.

La velocidad será también restringida por la capacidad del robot. Esta será diferente según el tipo de robot y la trayectoria utilizada.

Componentes

v_tcp **Tipo de dato:** *num*

La velocidad del punto central de la herramienta (TCP) en mm/s. Si se utilizan ejes externos coordinados o una herramienta estacionaria, la velocidad será especificada respecto al objeto de trabajo.

v_ori **Tipo de dato:** *num*

La velocidad de reorientación del TCP expresado en grados/s. Si se utilizan ejes externos coordinados o una herramienta estacionaria, la velocidad será especificada respecto al objeto de trabajo.

v_leax **Tipo de dato:** *num*

La velocidad de los ejes externos lineales en mm/s.

v_reax **Tipo de dato:** *num*

La velocidad de los ejes externos rotacionales en grados/s.

Ejemplo

```
VAR speeddata vmedia := [ 1000, 30, 200, 15 ];
```

El dato de velocidad *vmedia* será definido con las siguientes características:

- *1000* 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 rotacionales.

```
vmedia.v_tcp := 900;
```
La velocidad del TCP será cambiada a *900* mm/s.

Datos predefinidos

Una serie de datos de velocidad están ya definidos en el módulo del sistema *BASE.*

Nombre	Vel. TCP	V. Orient.	Vel. Eje Ext. Lin.	Vel. Eje Ext. Rot.
v5	5 mm/s	500º/s	5000 mm/s	1000º/s
v10	10 mm/s	500º/s	5000 mm/s	1000º/s
v20	20 mm/s	500º/s	5000 mm/s	1000º/s
v30	30 mm/s	500º/s	5000 mm/s	1000º/s
v40	40 mm/s	500º/s	5000 mm/s	1000º/s
v50	50 mm/s	500º/s	5000 mm/s	1000º/s
v60	60 mm/s	500º/s	5000 mm/s	1000º/s
v80	80 mm/s	500º/s	5000 mm/s	1000º/s
v100	100 mm/s	500º/s	5000 mm/s	1000º/s
v150	150 mm/s	500º/s	5000 mm/s	1000º/s
v200	200 mm/s	500º/s	5000 mm/s	1000º/s
v300	300 mm/s	500º/s	5000 mm/s	1000º/s
v400	400 mm/s	500º/s	5000 mm/s	1000º/s
v500	500 mm/s	500º/s	5000 mm/s	1000º/s
v600	600 mm/s	500º/s	5000 mm/s	1000º/s
v800	800 mm/s	500º/s	5000 mm/s	1000º/s
v1000	1000 mm/s	500º/s	5000 mm/s	1000º/s
v1500	1500 mm/s	500º/s	5000 mm/s	1000º/s
v2000	2000 mm/s	500º/s	5000 mm/s	1000º/s
v2500	2500 mm/s	500º/s	5000 mm/s	1000º/s
v3000	3000 mm/s	500º/s	5000 mm/s	1000º/s
v4000	4000 mm/s	500º/s	5000 mm/s	1000º/s
v5000	5000 mm/s	500º/s	5000 mm/s	1000º/s
v6000	6000 mm/s	500º/s	5000 mm/s	1000º/s
v7000	7000 mm/s	500º/s	5000 mm/s	1000º/s
vmax	*)	500º/s	5000 mm/s	1000º/s
vrot1	0 mm/s	0º/s	0 mm/s	1º/s
vrot2	0 mm/s	0º/s	0 mm/s	2º/s
vrot5	0 mm/s	0º/s	0 mm/s	5º/s
vrot10	0 mm/s	0º/s	0 mm/s	10º/s
vrot20	0 mm/s	0º/s	0 mm/s	20º/s
vrot50	0 mm/s	0º/s	0 mm/s	50º/s
vrot100	0 mm/s	0º/s	0 mm/s	100º/s
vlin10	0 mm/s	0º/s	10 mm/s	0º/s
vlin20	0 mm/s	0º/s	20 mm/s	0º/s
vlin50	0 mm/s	0º/s	50 mm/s	0º/s
vlin100	0 mm/s	0º/s	100 mm/s	0º/s
vlin200	0 mm/s	0º/s	200 mm/s	0º/s
vlin500	0 mm/s	0º/s	500 mm/s	0º/s
vlin1000	0 mm/s	0º/s	1000 mm/s	0º/s

*) Velocidad máxima del TCP en función del tipo de robot La función del RAPID *MaxRobSpeed* devuelve este valor.

Estructura

< estructura de *speeddata* >
< *v_tcp* de *num* >
< *v_ori* de *num* >
< *v_leax* de *num* >
< *v_reax* de *num* >

string Cadena de Caracteres

String se usa para las cadenas de caracteres.

Descripción

Una cadena de caracteres está formada por una serie de caracteres (como máximo 80) incluidos entre comillas (").
Ej: "Esto es una cadena de caracteres".

En el caso en que se desee incluir comillas dentro de la cadena de caracteres, deberán repetirse dos veces.
Ej: "Esta cadena contiene un caracter""".

Si debe incluir una barra \ dentro de la cadena de caracteres, se debe escribir dos veces.
Ej: “Esta cadena contiene el caracter \\”

Ejemplo

```
VAR string texto;
.
texto := "limpieza boquilla1";
TPWrite texto;
```

El texto "limpieza boquilla1" será visualizado en la unidad de programación.

Limitaciones

Una cadena puede tener de 0 a 80 caracteres, incluyendo las comillas adicionales.

Una cadena puede contener cualquiera de los caracteres especificados en la norma ISO 8859-1 así como los caracteres de control (caracteres no contenidos en ISO 8859-1 con un código numérico entre 0-255).

Datos Predefinidos

Hay una serie de constantes de cadenas predefinidas disponibles en el sistema y que pueden utilizarse junto con funciones de cadena.

Nombre	Grupo de caracteres
STR_DIGIT	<digito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
STR_UPPER	<letra mayúscula> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï | 1) | Ñ | Ò | Ó | Ô | Õ | Ö | Ø | Ù | Ú | Û | Ü | 2) | 3)

STR_LOWER	<letra minúscula> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï | 1) | ñ | ò | ó | ô | õ | ö | ø | ù | ú | û | ü | 2) | 3) | ß | ÿ
STR_WHITE	<carácter en blanco> ::=

1) Letra eth islandesa.
2) Letra Y con acento agudo
3) Letra thorn islandesa.

Las siguientes constantes están definidas en el módulo de sistema BASE.

```
CONST string diskhome := “HOME:”;
! Para los sistemas antiguos S4C
CONST string ram1disk:= "ram1disk";
CONST string disktemp := “TEMP:”;
CONST string stEmpty:= "";
```

Estructura

El tipo de dato String es atómico, lo que significa que no esta compuesto por otros tipos de datos.

tooldataDatos de herramienta

Tooldata se usa para describir las características de una herramienta, como por ejemplo, una pistola de soldadura o una pinza.

Si la herramienta está fija en el espacio (si se trata de una herramienta estacionaria), se definirán los datos de herramienta normales correspondientes así como la pinza que sujeta el objeto de trabajo.

Descripción

Los datos de herramienta afectarán a los movimientos del robot en la siguiente medida:

- El punto central de la herramienta (TCP) se refiere a un punto que cumplirá con la trayectoria especificada y con la exigencia de velocidad. En el caso en que se reoriente la herramienta o si se usan ejes externos coordinados, solamente este punto seguirá la trayectoria deseada a la velocidad programada.
- En el caso en que se use una herramienta estacionaria, tanto la velocidad como la trayectoria programadas se referirán al objeto de trabajo.
- Las posiciones programadas se refieren a la posición del TCP utilizado y a la orientación respecto al sistema de coordenadas de la herramienta. Ello significa que si, por ejemplo, se ha reemplazado una herramienta porque está dañada, se podrá todavía utilizar el programa antiguo si se vuelve a definir el sistema de coordenadas de la herramienta.

Los datos de herramienta también se usan al mover el robot para:

- Definir el TCP que no debe moverse cuando se reorienta la muñeca.
- Definir el sistema de coordenadas de la herramienta con vistas a facilitar el movimiento o rotación de las direcciones de la herramienta.

Es importante definir siempre la carga actual de la herramienta y cuando se use, la carga útil del robot. Definiciones incorrectas de la carga pueden provocar una sobrecarga en la estructura mecánica del robot

Si se especifican datos incorrectos de carga, se pueden generar las siguientes consecuencias:

Si el valor de los datos de carga indicados es mayor que el valor real de la carga;
- El robot no será utilizado a sus máximas posibilidades
- Perdida de precisión de la trayectoria incluyendo el riesgo de vibraciones.

Si el valor de los datos de carga indicados es menor que el valor real de la carga;
- Perdida de precisión de la trayectoria incluyendo el riesgo de vibraciones.
- Existe un riesgo de sobrecarga de la estructura mecánica

Componentes

robhold **Tipo de datos:** *bool*

Define si el robot está sujetando la herramienta o no:

- *TRUE->* El robot está sujetando la herramienta.
- *FALSE ->* El robot no está sujetando la herramienta, es decir, se trata de una herramienta estacionaria.

tframe **Tipo de dato:** *pose*

El sistema de coordenadas de la herramienta, es decir:

- La posición del TCP (x, y, z) en mm, expresado en coordenadas de la muñeca.
- La orientación del sistema de coordenadas de la herramienta, expresado en el sistema de coordenadas de la muñeca bajo la forma de un cuaternión (q1, q2, q3 y q4). (Véase la Figura 1.)

Si se utiliza una herramienta estacionaria, la definición se realizará respecto al sistema de coordenadas del mundo.

Si la dirección de la herramienta no está especificada, el sistema de coordenadas de la herramienta y el sistema de coordenadas de la muñeca coincidirán.

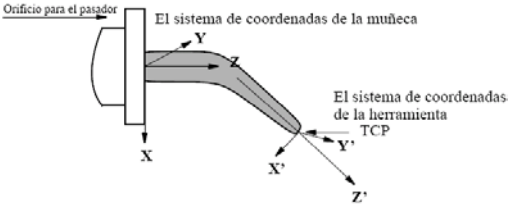


Figura 1 Definición del sistema de coordenadas de la herramienta.

tload **Tipo de dato:** *loaddata*

La carga de la herramienta, es decir:

- El peso de la herramienta en kg.
- El centro de gravedad de la herramienta (x, y, z) en mm, expresado en el sistema de coordenadas de la muñeca.
- La orientación del sistema de coordenadas de la carga de la herramienta expresado en el sistema de coordenadas de la muñeca, definiendo los ejes de inercia de la herramienta. La orientación del sistema de coordenadas de la carga de la herramienta, debe coincidir con la orientación del sistema de coordenadas de la muñeca. **Siempre debe ser 1,0,0,0.**
- El momento de inercia de la herramienta relativa a su centro de gravedad alrededor de los ejes de coordenadas de la carga de la herramienta en kgm2.

Si todos los componentes han sido definidos como 0 kgm2, la herramienta será considerada como si se tratara de una carga puntual.

Para más información, véase el tipo de dato *loaddata*.

Obsérvese que sólo se deberá especificar la carga de la herramienta. La carga útil manipulada por una herramienta se activa/desactiva mediante la instrucción *GripLoad*.

Ejemplos

PERS tooldata pinza := [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 de la Figura 1 se deberá definir utilizando los siguientes valores:

- El robot está sujetando la herramienta.
- El TCP está situado en el punto *223,1* mm en línea recta a partir del eje 6 y *97,4* mm a lo largo del eje X del sistema de coordenadas de la muñeca.
- Las direcciones X y Z de la herramienta están giradas 45° respecto al sistema de coordenadas de la muñeca.
- La herramienta pesa 5 kg.
- El centro de gravedad está situado en un punto a *75* mm en línea recta a partir del eje 6 y 23 mm a lo largo del eje X del sistema de coordenadas de la muñeca.
- La carga puede ser considerada como una carga puntual, es decir, sin ningún momento de inercia.

pinza.tframe.trans.z := 225.2;
El TCP de la herramienta, *pinza*, será redefinido a *225,2* en la dirección z.

Limitaciones

Los datos de herramienta (tooldata) deberán ser definidos únicamente como variables persistentes (*PERS)* y no deberán ser definidos dentro de una rutina. Los valores utilizados serán guardados, cuando el programa sea guardado y se recuperaran al cargar el programa.

El argumento del tipo datos de herramienta en cualquier instrucción de movimiento deberá ser un dato persistente entero, no un elemento de una matriz ni un componente de un registro.

Datos predefinidos

La herramienta *tool0* define el sistema de coordenadas de la muñeca y tiene como punto de origen el centro de la brida de montaje. Se podrá siempre acceder a *tool0* desde el programa, aunque no puedan ser cambiados (está en el módulo del sistema *BASE*).

PERS tooldata tool0 := [TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0, [0, 0, 0], [1, 0, 0, 0], 0, 0, 0]];

Estructura

< estructura de *tooldata* >
< *robhold* de *bool* >
< *tframe* de *pose* >
< *trans* de *pos* >
< *x* de *num* >
< *y* de *num* >
< *z* de *num* >
< *rot* de *orient* >
< *q1* de *num* >
< *q2* de *num* >
< *q3* de *num* >
< *q4* de *num* >
< *tload* de *loaddata* >

< *mass* de *num* >
< *cog* de *pos* >
< *x* de *num* >
< *y* de *num* >
< *z* de *num* >
< *aom* de *orient* >
< *q1* de *num* >
< *q2* de *num* >
< *q3* de *num* >
< *q4* de *num* >
< *ix* de *num* >
< *iy* de *num* >
< *iz* de *num* >

wobjdata Datos del objeto de trabajo

Wobjdata se usa para describir el objeto de trabajo que el robot está soldando, procesando, moviendo, etc.

Descripción

Si los objetos de trabajo han sido definidos en una instrucción de posicionamiento, la posición estará basada en las coordenadas del objeto de trabajo. Las ventajas de ello son las siguientes:

- Si los datos de posición son introducidos de forma manual, como en la programación off-line, los valores podrán obtenerse de un plano.
- Los programas podrán ser reutilizados rápidamente siguiendo los cambios realizados en la instalación del robot. Así, por ejemplo, si se mueve un utillaje, sólo se deberá volver a definir el sistema de coordenadas del usuario.
- Las variaciones respecto a la posición del objeto de trabajo podrán ser compensadas. Para ello, sin embargo, se necesitará algún tipo de sensor para saber la posición del objeto de trabajo.

Los datos del objeto de trabajo podrán ser utilizados también para el movimiento:

- El robot podrá ser movido en las direcciones del objeto de trabajo.
- La posición utilizada que se visualiza está basada en el sistema de coordenadas del objeto de trabajo.

Componentes

robhold Tipo de dato: *bool*

Define si el robot está sujetando o no el objeto de trabajo:
- *TRUE*-> El robot está sujetando el objeto de trabajo, es decir, que está utilizando una herramienta estacionaria.
- *FALSE* -> El robot no está sujetando el objeto de trabajo, es decir, que el robot está sujetando la herramienta.

ufprog Tipo de dato: *bool*

Define si se usa o no un sistema de coordenadas fijo del usuario:
- *TRUE* -> Sistema de coordenadas fijo del usuario.
- *FALSE*-> Sistema de coordenadas móvil del usuario, es decir, que se están usando ejes externos coordinados.

ufmec Tipo de dato: *string*

La unidad mecánica con la que los movimientos del robot están coordinados.
Sólo se especifica en el caso de sistemas de coordenadas móvil del usuario (*ufprog* es *FALSE*).
Está especificado con el nombre con el que está definido en los parámetros del sistema, por ejemplo, "orbit_a".

uframe **Tipo de dato:** *pose*

El sistema de coordenadas del usuario, es decir, la posición de la superficie de trabajo o el utillaje utilizados (véase la Figura 1):

- La posición del origen del sistema de coordenadas (x, y, z) en mm.
- La rotación del sistema de coordenadas, expresado como un cuaternión (q1,q2,q3,q4).

Si el robot está sujetando la herramienta, el sistema de coordenadas del usuario será definido en el sistema de coordenadas mundo.

Cuando se usan los ejes externos coordinados (*ufprog* es *FALSE*), el sistema de coordenadas del usuario será definido en los parámetros del sistema.

oframe **Tipo de dato:** *pose*

El sistema de coordenadas del objeto, es decir, la posición del objeto de trabajo utilizado (véase la Figura 1):

- La posición del origen del sistema de coordenadas (x, y, z) en mm.
- La rotación del sistema de coordenadas, expresado como un cuaternión (q1, q2, q3, q4).
- El sistema de coordenadas del objeto está definido en el sistema de coordenadas del usuario.

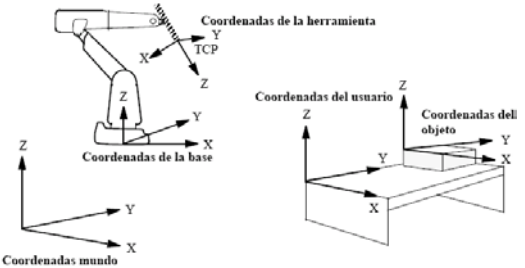


Figura 1 Los diferentes sistemas de coordenadas del robot (cuando el robot está sujetando la herramienta).

Ejemplo

PERS wobjdata obj2 :=[FALSE,TRUE,"", [[300, 600, 200],[1, 0, 0 ,0]],[[0, 200, 30], [1, 0, 0 ,0]]];
El objeto de trabajo de la Figura 1 se describe utilizando los siguientes valores:

- El robot no está sujetando el objeto de trabajo.
- Se usa el sistema de coordenadas fijo del usuario.
- El sistema de coordenadas del usuario no ha sido girado y las coordenadas de su origen están en x= *300*, y = *600*, z = *200* mm en el sistema de coordenadas del mundo.
- El sistema de coordenadas del objeto no ha sido girado y las coordenadas de su origen están en x= *0*, y= *200*, z= *30* mm en el sistema de coordenadas del usuario.

obj2.oframe.trans.z := 38.3; La posición del objeto de trabajo *obj2* se actualiza a *38,3* mm en dirección z.

Limitaciones

Los datos del objeto de trabajo (wobjdata) deberán ser definidos únicamente como variables persistentes (*PERS)* y no deberán ser definidos dentro de una rutina. Los valores utilizados serán guardados, cuando el programa sea guardado y se recuperaran al cargar el programa.

El argumento del tipo datos de herramienta en cualquier instrucción de movimiento deberá ser un dato persistente entero (no un elemento de una matriz ni un componente de un registro).

Datos predefinidos

Los datos del objeto de trabajo *wobj0* están definidos de forma que el sistema de coordenadas del objeto coincida con el sistema de coordenadas mundo. El robot no sujeta el objeto de trabajo. Siempre se podrá acceder a *Wobj0* desde el programa, pero nunca podrá ser cambiado (se encuentra almacenado en el módulo del sistema *BASE*).

PERS wobjdata wobj0 := [FALSE,TRUE,"", [[0, 0, 0], [1, 0, 0 ,0]],[[0, 0, 0], [1, 0, 0 ,0]]];

Estructura

< estructura de *wobjdata* >
< *robhold* de *bool* >
< *ufprog* de *bool*>
< *ufmec* de *string* >
< *uframe* de *pose* >
< *trans* de *pos* >
< *x* de *num* >
< *y* de *num* >
< *z* de *num* >
< *rot* de *orient* >
< *q1* de *num* >
< *q2* de *num* >
< *q3* de *num* >
< *q4* de *num* >
< *derame* de *pose* >
< *trans* de *pos* >
< *x* de *num* >
< *y* de *num* >
< *z* de *num* >
< *rot* de *orient* >
< *q1* de *num* >
< *q2* de *num* >
< *q3* de *num* >
< *q4* de *num* >

zonedata Datos de la zona

Zonedata se usa para especificar la precisión de una posición, es decir, a qué distancia de la posición programada deben estar los ejes antes de moverse hacia la posición siguiente.

Descripción

Una posición puede terminarse en un punto de paro o un punto de paso.

Un punto de paro significa que tanto el robot como los ejes externos deben alcanzar la posición especificada (parada) antes de que la ejecución del programa prosiga con la instrucción siguiente.

Un punto de paso significa que la posición programada nunca es alcanzada. En lugar de ello, la dirección del movimiento cambia antes de que se pueda alcanzar dicha posición.

Para cada posición, se podrá definir dos zonas distintas (áreas):

- La zona de la trayectoria del TCP.
- La zona extendida para la reorientación de la herramienta y de los ejes externos.

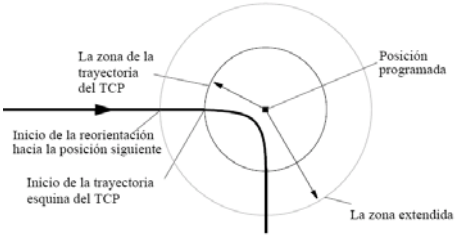


Figura 1 Las zonas de un punto de paso.

Las zonas funcionan de la misma manera durante el movimiento de los ejes, sin embargo, el tamaño de la zona puede diferir de algo respecto a la programada.

El tamaño de la zona nunca podrá ser mayor que la mitad de la distancia a la posición más cercana (hacia adelante o hacia atrás). En el caso de que sea mayor, el robot automáticamente la reducirá.

La zona de la trayectoria del TCP

Una trayectoria esquina (parabólica) es generada tan pronto como se alcanza el borde de la zona (véase la Figura 1).

La zona de reorientación de la herramienta

La reorientación se inicia en cuanto el TCP alcanza la zona extendida. La herramienta es reorientada de forma que la orientación sea la misma al abandonar la zona que si hubiera habido un punto de paro programado. La reorientación será más suave si se aumenta el tamaño de la zona y habrá menos riesgo de tener que reducir la velocidad para llevar a cabo la reorientación.

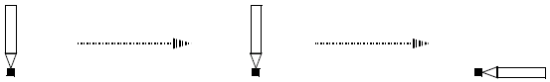


Figura 2 Tres posiciones han sido programadas; la última con una orientación de herramienta diferente.

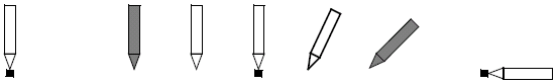


Figura 3 Si todas las posiciones fueran puntos de paro, la ejecución tendría este aspecto.

Figura 4 Si la posición central fuera un punto de paso, la ejecución del tendría este aspecto

La zona de los ejes externos

Los ejes externos empiezan a moverse hacia la siguiente posición tan pronto como el TCP alcanza la zona extendida. De esta forma, un eje lento puede acelerar antes y luego seguir ejecutando el programa de forma más regular.

Zona reducida

Con grandes reorientaciones de la herramienta o grandes movimiento de los ejes externos, la zona extendida del TCP e incluso la zona del TCP son reducidas automáticamente por el robot. La zona será definida con el tamaño relativo más pequeño basándose en los otros componentes de la zona y en el movimiento programado.

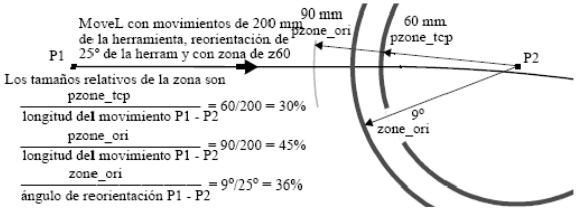


Figura 5 Ejemplo de zona reducida para la reorientación de la herramienta al 36% del movimiento debido a una zone_ori.

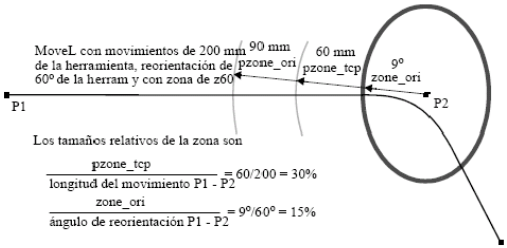


Figura 6 Ejemplo de zona reducida para la reorientación de la herramienta y la trayectoria del TCP al 15% del movimiento debido a una zone_ori.

Cuando los ejes externos están activos, afectarán los tamaños relativos de la zona, de acuerdo con las siguientes fórmulas:

$$\frac{\text{pzone\_eax}}{\text{longitud del movimiento P1 - P2}}$$

$$\frac{\text{zone\_leax}}{\text{longitud del movimiento máximo lineal del eje externo P1 - P2}}$$

$$\frac{\text{zone\_reax}}{\text{ángulo de reorientación máxima del eje ext. rotativo P1 - P2}}$$

NOTA: Si la zona del TCP resulta reducida a causa de una zone_ori, zone_leax o zone_reax, el planificador de trayectorias entra en un modo que permite gestionar el caso en que no hay movimiento del TCP. En el caso en que se produzca un movimiento del TCP cuando está en este modo, la velocidad no será compensada para la curva de la trayectoria en una zona esquina. Por ejemplo, ello producirá una reducción de velocidad del 30% en una esquina de 90 grados.

Componentes

finep **Tipo de dato:** *bool*

Define si el movimiento debe terminar como un punto de paro o de paso.

- *TRUE*-> El movimiento termina como un punto de paro. Los componentes restantes en los datos de zona no se utilizan.
- *FALSE* -> El movimiento termina como un punto de paso.

pzone_tcp **Tipo de dato:** *num*

El tamaño (el radio) de la zona del TCP en mm.

pzone_ori **Tipo de dato:** *num*

El tamaño de la zona (el radio) de la reorientación de la herramienta. El tamaño se definirá como la distancia del TCP al punto programado, en mm.

El tamaño deberá ser mayor que el valor correspondiente de *pzone_tcp*. Si se especifica un valor más bajo, el tamaño de la zona aumentará automáticamente, de la misma forma aumentará los valores de *pzone_tcp*.

pzone_eax Tipo de dato: *num*

El tamaño de la zona (el radio) de los ejes externos. El tamaño se definirá como la distancia del TCP al punto programado, en mm.

El tamaño deberá ser mayor que el valor correspondiente de la *pzone_tcp*. Si se especifica un valor más bajo, el tamaño de la zona aumentará automáticamente de la misma forma aumentará los valores de *pzone_tcp*.

zone_ori Tipo de dato: *num*

El tamaño de la zona de la reorientación de la herramienta en grados. Si el robot está sujetando el objeto de trabajo, esto representará un ángulo de rotación para el objeto de trabajo.

zone_leax Tipo de dato: *num*

El tamaño de la zona de los ejes externos lineales en mm.

zone_reax Tipo de dato: *num*

El tamaño de zona de los ejes externos rotacionales en grados.

Ejemplos

VAR zonedata trayec := [FALSE, 25, 40, 40, 10, 35, 5];
El dato de zona *trayec* será definido con las siguientes características:

- El tamaño de zona de la trayectoria del TCP es de *25* mm.
- El tamaño de zona de la reorientación de la herramienta es de *40* mm (movimiento del TCP).
- El tamaño de zona de los ejes externos es de *40* mm (movimiento del TCP).

Sin embargo, si el TCP está parado, o si hay una reorientación más amplia, o si el movimiento de los ejes externos es mayor respecto a la zona, se aplicarán los siguientes valores:

- El tamaño de la zona de la reorientación de la herramienta es de *10* grados.
- El tamaño de la zona de los ejes externos lineales es de *35* mm.
- El tamaño de la zona de los ejes externos rotacionales es de *5* grados.

trayec.pzone_tcp := 40;
El tamaño de la zona de la trayectoria del TCP está ajustada a los *40* mm.

Datos predefinidos

Hay una serie de datos de zona predefinidos en el módulo del sistema *BASE*.

Puntos de paro

Nombre	
fine	0 mm

Puntos de paso

	movimiento del TCP			reorientación de la herramienta		
Nombre	trayec.TCP	Orientación	Ejes ext.	Orientación	Ejes lin.	Ejes rotat.
z0	0,3 mm	0,3 mm	0,3 mm	0,03 °	0,3 mm	0,03 °
z1	1 mm	1 mm	1 mm	0,1 °	1 mm	0,1 °
z5	5 mm	8 mm	8 mm	0,8 °	8 mm	0,8 °
z10	10 mm	15 mm	15 mm	1,5 °	15 mm	1,5 °
z15	15 mm	23 mm	23 mm	2,3 °	23 mm	2,3 °
z20	20 mm	30 mm	30 mm	3,0 °	30 mm	3,0°
z30	30 mm	45 mm	45 mm	4,5 °	45 mm	4,5 °
z40	40 mm	60 mm	60 mm	6,0 °	60 mm	6,0 °
z50	50 mm	75 mm	75 mm	7,5 °	75 mm	7,5 °
z60	60 mm	90 mm	90 mm	9,0 °	90 mm	9,0 °
z80	80 mm	120 mm	120 mm	12 °	120 mm	12 °
z100	100 mm	150 mm	150 mm	15 °	150 mm	15 °
z150	150 mm	225 mm	225 mm	23 °	225 mm	23 °
z200	200 mm	300 mm	300 mm	30 °	300 mm	30 °

Estructura

< estructura de *zonedata* >
< *finep* de *bool* >
< *pzone_tcp* de *num* >
< *pzone_ori* de *num* >
< *pzone_eax* de *num* >
< *zone_ori* de *num* >
< *zone_leax* de *num* >
< *zone_reax* de *num* >

Manual Usuario Robot IRB120

Funciones

Instrucciones

Tipos de Datos

Guía de Usuario

Contenido

Add Suma de un valor numérico

Add sirve para sumar o restar un valor a o de una variable numérica o persistente.

Ejemplos

Add reg1, 3;
Se suma 3 a *reg1*, es decir, reg1:=reg1+3.

Add reg1, -reg2;
Se resta *reg2* de *reg1*, es decir, reg1:=reg1-reg2.

Argumentos

Add Nombre ValorAñadir

Nombre **Tipo de dato:** *num*

El nombre de la variable o persistente que se desea cambiar.

ValorAñadir **Tipo de dato:** *num*

El valor que se desea sumar.

Sintaxis

Add
[Nombre ':='] < variable o persistente (**INOUT**) de *num* > ','
[ValorAñadir':='] < expresión (**IN**) de *num* > ';'

":=" Asignación de un valor

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

Ejemplos

reg1:= 5;
Se asigna el valor 5 a *reg1*.

reg1:= reg2 - reg3;
El valor resultante del cálculo *reg2-reg3* se asigna a *reg1.*

contador:= contador + 1;
contador se incrementa en1.

herram1.tframe.trans.x:= herram1.tframe.trans.x + 20;
El TCP de *herram1* se desplazará *20* mm en la dirección del eje X.

palet{5,8}:= Abs(reg1);
Al elemento {5,8} de la matriz *palet* se le asigna un valor igual al valor absoluto de la variable *reg1.*

Argumentos

Dato := Valor

Dato **Tipos de dato:** Todos

El dato al que se desea asignar un nuevo valor

Valor **Tipo de dato:** Igual que Dato

El valor deseado.

Limitaciones

El dato, cuyo valor debe ser cambiado, no puede ser ni:

- una constante
- un tipo de dato sin valor.

Los datos y valores deben tener tipos de datos similares, los mismos o equivalentes.

Sintaxis

(EBNF)
<asignación> ':=' <expresión> ';'
<asignación> ::=
<variable>
|<persistente>
|**<DOB>**

Clear Poner a cero una variable

Clear sirve para borrar el valor de una variable o persistente numérica, es decir, asignarle el valor 0.

Ejemplo

Clear reg1;
Se asigna el valor 0 a *Reg1*, es decir, reg1:=0.

Argumentos

Clear Nombre

Nombre Tipo de dato: *num*

El nombre de la variable o persistente que se desea poner a cero.

Sintaxis

Clear
[Nombre ':='] < variable o persistente (INOUT) de *num* > ';'

ClkReset Puesta a cero de un reloj usado para cronometraje

ClkReset sirve para poner a cero un reloj que funciona como un cronómetro.
Esta instrucción puede utilizarse antes de usar un reloj, para asegurarse de que está a 0.

Ejemplo

ClkReset ckreloj1;
El reloj *ckreloj1* será puesto a cero.

Argumentos

ClkReset Reloj

Reloj **Tipo de dato:** *reloj*

El nombre del reloj que se desea poner a cero.

Ejecución del programa

Cuando se resetea un reloj, éste se pondrá a 0.
Si el reloj está funcionando, se parará y luego se pondrá a 0.

Sintaxis

ClkReset
[Reloj ':='] < variable (**VAR**) de tipo *clock*> ';'

ClkStart — Puesta en marcha de un reloj usado para cronometraje

ClkStart sirve para poner en marcha un reloj que funciona como un cronómetro.

Ejemplo

ClkStart ckreloj1;
El reloj *ckreloj1* será arrancado.

Argumentos

ClkStart Reloj1

Reloj1 — Tipo de dato: *clock*

El nombre del reloj que se desea arrancar.

Ejecución del programa

Cuando se arranca un reloj, cuenta los segundos hasta que se para. Aunque se pare la ejecución del programa, el reloj sigue funcionando, aunque el acontecimiento que se deseaba cronometrar puede ya no ser válido. Por ejemplo, si el programa estaba cronometrando el tiempo de espera de una entrada, puede ocurrir que la entrada se haya activado mientras el programa estaba detenido, por lo que el programa no será capaz de "ver" el acontecimiento que ocurrió mientras estaba parado.

El reloj seguirá funcionando cuando se desconecte la alimentación del sistema siempre y cuando funcione la pila que alimenta el reloj de tiempo real del computador.
Cuando el reloj está funcionando, podrá ser leído, parado o puesto a cero.

Ejemplo

```
VAR clock ckreloj2;
....
ClkReset ckreloj2;
ClkStart ckreloj2;
WaitUntil dientrada1, 1;
ClkStop ckreloj2;
```

El reloj cronometará el tiempo que la entrada *dientrada1* tardará en pasar a 1.

Gestión de errores

Si un reloj llega a 4,294,967 segundos (49 días 17 horas 2 minutos 47 segundos) se produce un desbordamiento de lo registros y la variable del sistema ERRNO adquiere el valor ERR_OVERFLOW. Este error se puede gestionar con un Gestor de Errores.

Sintaxis

ClkStart
[Reloj ':='] < variable (**VAR**) de *clock*> ';'

ClkStop Paro de un reloj utilizado para cronometraje

ClkStop sirve para parar un reloj que funciona como cronómetro.

Ejemplo

ClkStop ckreloj1;
El reloj *ckreloj1* será parado.

Argumentos

ClkStop Reloj

Reloj **Tipo de dato:** *clock*

El nombre del reloj que se desea parar.

Ejecución del programa

Cuando se para un reloj, éste dejará de funcionar.
Si se para un reloj, este podrá ser leído, arrancado de nuevo o puesto a cero.

Gestión de errores

Si un reloj llega a 4,294,967 segundos (49 días 17 horas 2 minutos 47 segundos) se produce un desbordamiento de lo registros y la variable del sistema ERRNO adquiere el valor ERR_OVERFLOW. Este error se puede gestionar con un Gestor de Errores.

Sintaxis

ClkStop
[Reloj ':='] < variable (**VAR**) de *clock*> ';'

Compact IF Si se cumple una condición, entonces... (una instrucción)

Compact IF sirve cuando solo se debe ejecutar únicamente una instrucción si se cumple una condición.

En el caso de que se deban ejecutar diferentes instrucciones, según se cumpla o no una condición, se utilizará la instrucción *IF.*

Ejemplos

IF reg1 > 5 rutina1;
Si *reg1* **es mayor que** *5,* entonces se ejcuta la *rutina1*.

IF contador> 10 Set doCerrarPinza;
La señal *doCerrarPinza* se activará si el *contador> 10*.

Argumentos

IF Condición ...

Condición **Tipo de dato:** *bool*

La condición que debe cumplirse para que la instrucción pueda ejecutarse.

Sintaxis

(EBNF)
IF <expresión condicional> (<instrucción> | **<SMT>**) ';'

Decr Decremento en 1 de una variable numérica

Decr sirve para restar 1 a una variable o persistente numérica.

Ejemplo

Decr reg1;
Se restará 1de *reg1*, es decir, reg1:=reg1-1.

TPReadNum num_piezas, "Cuántas piezas deben ser procesadas? ";
WHILE num_piezas>0 DO
produc_pieza;
Decr num_piezas;
ENDWHILE

Se pide al usuario que introduzca el número de piezas que van a ser procesadas.
La variable *num_piezas* sirve para contar el número que quedan por procesar.

Argumentos

Decr Nombre

Nombre **Tipo de dato:** *num*

El nombre de la variable o persistente que se desea decrementar.

Sintaxis

Decr
[Nombre ':='] < variable o persistente (**INOUT**) de *num* > ';'

IF Si se cumple una condición, entonces... si no...

IF sirve cuando diferentes instrucciones deben ejecutarse según se cumpla o no una condición.

Ejemplos

IF reg1 > 5 THEN
Set doSalida1;
Set doSalida2;
ENDIF
Las señales *doSalida1* y *doSalida2* son activadas únicamente si *reg1* es mayor que *5*.

IF reg1 > 5 THEN
Set doSalida1;
Set doSalida2;
ELSE
Reset doSalida1;
Reset doSalida2;
ENDIF
Las señales *doSalida1* y *doSalida2* son activadas o desactivadas dependiendo si *reg1* es mayor que *5* o no.

IF contador > 100 THEN
contador := 100;
ELSEIF contador < 0 THEN
contador := 0;
ELSE
contador := contador + 1;
ENDIF
Si el valor inicial del *contador* está fuera del límite *0-100*, el valor que se le asigna al *contador* será el valor límite correspondiente. En caso contrario *contador* se incrementará en 1

Argumentos

IF Condición THEN ...
{ELSEIF Condición THEN ...}
[ELSE ...]
ENDIF

Condición — Tipo de dato: *bool*

La condición que debe cumplirse para que las instrucciones entre THEN y ELSE/ELSEIF puedan ejecutarse.

Ejecución del programa

Las condiciones son comprobadas siguiendo un orden secuencial, hasta que una de ellas se cumpla. La ejecución del programa continúa con las instrucciones asociadas a esa condición. Si no se cumple ninguna de las condiciones, la ejecución del programa continúa con las instrucciones que siguen a ELSE. Si se cumple más de una condición, sólo se ejecutarán las instrucciones que están asociadas con la primera de las citadas condiciones que se verifique.

Sintaxis

(EBNF)
IF <expresión condicional> **THEN**
<lista instrucciones>
{**ELSEIF** <expresión condicional> **THEN** <lista instrucciones> |
<EIF>}
[**ELSE**
<lista instrucciones>]
ENDIF

Incr Incremento en 1 de una variable numérica

Incr sirve para añadir 1 a una variable o persistente numérica.

Ejemplo

```
Incr reg1;
```
reg1 se verá incrementado en 1, es decir, reg1:=reg1+1.

```
WHILE distop_produc=0 DO
produc_piezas;
Incr piezas;
TPWrite "Nº de piezas producidas= "\Num:=piezas;
ENDWHILE
```

En cada ciclo el número de piezas producidas será actualizado en la pantalla de la unidad de programación. El proceso de producción continua su ejecución mientras no se active la señal *distop_produc.*

Argumentos

Incr Nombre

Nombre **Tipo de dato:** *num*

El nombre de la variable o persistente que se desea incrementar.

Sintaxis

```
Incr
   [ Nombre ':=' ] < variable o persistente (INOUT) de num > ';'
```

MoveAbsJ Movimiento del robot a una posición de ejes absoluta

MoveAbsJ (*Move Absolute Joint*) sirve para mover el robot a una posición de ejes absoluta, definida como posiciones de los ejes.

Se deberá utilizar esta instrucción únicamente en los siguientes casos:

- cuando el punto final es un punto singular
- para posiciones ambiguas, por ejemplo, para movimientos realizados con la herramienta por encima del robot.
- cuando se quiera llevar al robot a u a posición absoluta de los ejes independiente de la herramienta o workobject definido o desplazamiento de programa activo.

La posición final del robot, durante un movimiento realizado con *MoveAbsJ,* no se verá afectada por la herramienta dada, el objeto de trabajo determinado ni por un desplazamiento del programa activado. No obstante, el robot utiliza estos datos para calcular la carga, la velocidad del TCP, y la trayectoria en las esquina. Se podrán utilizar las mismas herramientas que las utilizadas en las instrucciones de movimiento.

El robot y los ejes externos se mueven a la posición siguiendo una trayectoria no lineal. Todos los ejes alcanzan la posición al mismo tiempo.

Esta instrucción solo se puede utilizar en la tarea *Main,* o en las tareas de movimiento en sistemas MultiMove.

Ejemplos

MoveAbsJ p50, v1000, z50, herram2;
El robot con la herramienta *herram2* se moverá, siguiendo una trayectoria de tipo no lineal, a la posición de ejes absoluta *p50*, con un dato de velocidad *v1000* y un dato de zona *z50*.

MoveAbsJ pCoger, v1000, fine, pinza3;
El robot con la herramienta *pinza3*, se moverá siguiendo una trayectoria de tipo no lineal, a un punto de paro que es almacenado como una posición de ejes absoluta en la instrucción.

Argumentos

MoveAbsJ [\Conc] APosEje [\ID] [NoEOffs] Velocidad [\V]
| [\T] Zona [\Z] [\InPos] Herramienta [\WObj]

[\Conc] Tipo de dato: *switch*

Las instrucciones lógicas siguientes se ejecutarán mientras el robot está en movimiento. El argumento sirve para acortar el tiempo de ciclo, cuando, por ejemplo, se realiza una comunicación con el equipo externo, siempre que no se requiera la sincronización con el movimiento.

Al utilizar el argumento \Conc, el número de instrucciones de movimiento que se ejecutan está limitado a 5. Si en una sección de programa se incluye la secuencia *StorePath-RestoPath*, no se permiten instrucciones de movimiento que contengan el argumento \Conc.

En el caso de omitir este argumento y la posición no es un punto de paro, la siguiente instrucción siguiente se ejecutará un poco tiempo antes de que el robot haya alcanzado la zona de precisión programada.

APosEje **Tipo de dato:** *jointtarget*

Es la posición del eje absoluta de destino del robot y de los ejes externos. Se define como una posición con nombre o almacenada directamente en la instrucción (marcada con un asterisco * en la instrucción).

[\ID] **Tipo de dato:** *identificador*

Este argumento se puede usar en sistemas MultiMove, si el movimiento es coordinado y sincronizado y no se pude utilizar en los otros casos.

El número especificado de id debe ser el mismo en todas las tareas del programa que deban cooperar entre si. Este número permite garantizar que los movimientos no se mezclaran durante la ejecución.

[\NoEOffs] **Tipo de dato:** *switch*

Si esta activado el argumento *NoEoffs*, el movimiento de la instrucción MoveAbsJ no se verá afectado por los offsets para los ejes externos.

Velocidad **Tipo de dato:** *speeddata*

Son los datos de velocidad que se aplican a los movimientos. Definen la velocidad del punto central de la herramienta, de la reorientación de la herramienta y de los ejes externos.

[\V] **Tipo de dato:** *num*

Este argumento sirve para especificar la velocidad del TCP en mm/s directamente en la instrucción, que substituirá a la velocidad correspondiente especificada en los datos de velocidad.

[\T] *(Tiempo)* **Tipo de dato:** *num*

Este argumento sirve para especificar el tiempo total expresado en segundos durante el cual el robot se mueve. Sustituye a los datos de velocidad de la instrucción.

Zona **Tipo de dato:** *zonedata*

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

[\Z] *(Zona)* **Tipo de dato:** *num*

Este argumento sirve para especificar la precisión de la posición del TCP del robot directamente en la instrucción. La longitud de la trayectoria esquina está expresada en mm, y sustituye a la zona correspondiente especificada en los datos de zona.

[\InPos] Tipo de dato: *stoppointdata*

Este argumento se usa para especificar los criterios de convergencia del TCP del robot en el punto de paro. Este dato del punto de paro sustituye al dato de zona especificado en el parámetro *Zona*.

Herramienta Tipo de dato: *tooldata*

La herramienta utilizada durante el movimiento. El TCP y la carga de la herramienta están definidos en los datos de herramienta. El TCP es el punto que se mueve a la posición de destino especificado.

[\WObj] *(Objeto de trabajo)* Tipo de dato: *wobjdata*

Es el objeto de trabajo utilizado durante el movimiento.

Este argumento podrá ser omitido si la herramienta es sujetada por el robot. No obstante, si el robot sujeta el objeto de trabajo, es decir, si la herramienta es estacionaria, o con ejes externos coordinados, entonces se deberá especificar este argumento.

En el caso de una herramienta estacionaria o de ejes externos coordinados, los datos utilizados por el sistema para decidir la velocidad y la trayectoria esquina del movimiento, se encuentran definidos en el objeto de trabajo.

Ejecución del programa

La herramienta se moverá a la posición de ejes absoluta de destino, con una interpolación de los ángulos de los ejes. Esto significa que cada eje se mueve a una velocidad de ejes constante y que todos los ejes alcanzan la posición final al mismo tiempo, lo que implica una trayectoria no lineal.

De forma general, el TCP se mueve aproximadamente a la velocidad programada.

La herramienta es reorientada y los ejes externos se mueven al mismo tiempo que el TCP. En el caso en que el sistema no sea capaz de alcanzar la velocidad programada para la reorientación o para los ejes externos, se reduce la velocidad del TCP.

Se genera una trayectoria esquina cuando la trayectoria continúa a una siguiente posición. En el caso en que se haya especificado un punto de paro en los datos de zona, la ejecución del programa continuará cuando el robot y los ejes externos hayan alcanzado la posición de ejes adecuada.

Ejemplos

MoveAbsJ p34, v2000\V:=2200, z40 \Z:=45, pinza3;
La herramienta, *pinza3*, se moverá siguiendo una trayectoria no lineal a una posición de ejes absoluta almacenada en la instrucción. El movimiento se llevará a cabo con los datos en *v2000* y *z40*, la velocidad del TCP es de 2200mm/s y el tamaño de zona del TCP *45* mm.

MoveAbsJ \Conc, *, v2000, z40, pinza3;

La herramienta, *pinza3*, se moverá siguiendo una trayectoria no lineal a una posición de ejes absoluta almacenada en la instrucción. Las instrucciones lógicas siguientes serán ejecutadas mientras el robot se está moviendo.

Gestión de errores

Cuando se va a ejecutar el programa, el sistema realiza una comprobación de que los argumentos Herram y \WObj no contienen datos contradictorios respecto a una herramienta móvil o estacionaria respectivamente.

Limitaciones

Un movimiento realizado con la instrucción *MoveAbsJ* no se verá afectado por un desplazamiento activo del programa, sin embargo se verá afectado por un offset activo para los ejes externos.

Para ser capaz de poder ejecutar hacia atrás una secuencia de programa en donde la instrucción *MoveAbsJ* esté implicada, y para evitar problemas con puntos singulares o áreas ambiguas, es esencial que las instrucciones siguientes cumplan ciertos requisitos, según se ve en la figura siguiente:

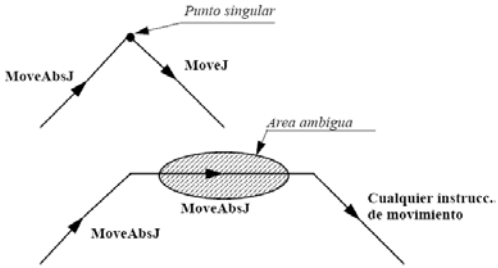


Figura 1 Limitación para la ejecución hacia atrás en un movimiento con la instrucción MoveAbsJ.

Sintaxis

MoveAbsJ
['\' Conc ',']
[APosEje ':='] < expresión (**IN**) de *jointtarget* > ','
[Velocidad ':='] < expresión (**IN**) de *speeddata* >
['\' V ':=' < expresión (**IN**) de *num* >]
| ['\' T ':=' < expresión (**IN**) de *num* >] ','
[Zona ':='] < expresión (**IN**) de *zonedata* >
['\' Z ':=' < expresión (**IN**) de *num* >] ','
[Herramienta ':='] < expresión (**PERS**) de *tooldata* >
['\' WObj ':=' < expresión (**PERS**) de *wobjdata* >] ';'

MoveC Movimiento circular del robot

MoveC (Move Circular) sirve para mover el punto central de la herramienta (TCP) de forma circular a una posición determinada. Durante el movimiento, la orientación suele permanecer constante respecto al círculo.

Esta instrucción solo se puede utilizar en la tarea *Main,* o en las tareas de movimiento en sistemas MultiMove.

Ejemplos

MoveC p1, p2, v500, z30, herram2;
El TCP de la herramienta, *herram2*, se moverá de forma circular a la posición *p2,* con un dato de velocidad de *v500* y un dato de zona de *z30*. El círculo se define a partir de la posición de arranque, el punto de círculo *p1* y el punto de destino *p2*.

MoveC pcir1, p15, v500, z40, pinza3 \WObj:=fijación;
El TCP de la herramienta, *pinza3*, se moverá de forma circular a la posición, *p15*, pasando por el punto de círculo *pcir1*. Estas posiciones están especificadas en el sistema de coordenadas del objeto para *fijación*.

MoveL p1, v500, fine, herram1;
MoveC p2, p3, v500, z20, herram1;
MoveC p4, p1, v500, fine, herram1;
Para realizar un círculo completo, las posiciones deben ser las que se indican en la Figura 1.

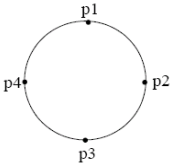


Figura 1 Un círculo completo se realiza mediante dos instrucciones MoveC.

Argumentos

MoveC [\Conc] PuntoCirculo AlPunto [\ID] Velocidad [\V] | [\T] Zona [\Z][\Inpos] Herramienta [\WObj] [\Corr]

[\Conc] **Tipo de dato:** *switch*

Las instrucciones lógicas siguientes se ejecutarán inmediatamente mientras el robot se esta moviendo. Este argumento se puede usar para evitar paros imprevistos, causados por la sobrecarga de la CPU, cuando se usan puntos de paso de esta manera se puede reducir el tiempo de ciclo. Este argumento también sirve para acortar el tiempo de ciclo cuando, por ejemplo, se está comunicando con el equipo externo y no se requiere ninguna sincronización entre el equipamiento externo y el movimiento del robot.

Utilizando el argumento *\Conc,* el número de instrucciones de movimiento que se ejecutarán, está limitado a 5. No se puede utilizar en una sección de programa que incluye *StorePath-RestoPath.*

Si se omite este argumento y AlPunto no es un punto de paro, la siguiente instrucción se ejecutará poco tiempo antes de que el robot haya alcanzado la zona programada.

Este argumento no se puede usar en movimientos coordinados sincronizados en sistemas MultiMove.

PuntoCírculo **Tipo de dato:** *robtarget*

Es el punto de círculo para el robot. El punto de círculo es una posición en el círculo, entre el punto de inicio y el punto final. Para obtener una mayor precisión, deberá estar situado aproximadamente a mitad de camino entre el punto de inicio y el punto final. Si está situado demasiado cerca del punto de inicio o del punto final, el robot puede generar un mensaje de aviso. El punto de círculo está definido como una posición con nombre o es almacenado directamente en la instrucción (marcado con un asterisco * en la instrucción). Las posiciones de los ejes externos no son utilizadas.

AlPunto **Tipo de dato:** *robtarget*

Es el punto de destino del robot y de los ejes externos. Está definido como una posición con nombre o es almacenado directamente en la instrucción (marcado con un asterisco * en la instrucción).

[\ID] **Tipo de dato:** *identificador*

Este argumento se puede usar en sistemas MultiMove, si el movimiento es coordinado y sincronizado y no se pude utilizar en los otros casos.

El número especificado de id debe ser el mismo en todas las tareas del programa que deban cooperar entre si. Este número permite garantizar que los movimientos no se mezclaran durante la ejecución.

Velocidad **Tipo de dato:** *speeddata*

Son los datos de velocidad que se aplican a los movimientos. Definen la velocidad del punto central de la herramienta, de la reorientación de la herramienta y de los ejes externos.

[\V] **Tipo de dato:** *num*

Este argumento sirve para especificar la velocidad del TCP en mm/s directamente en la instrucción, que substituirá a la velocidad correspondiente especificada en los datos de velocidad.

[\T] **Tipo de dato:** *num*

Este argumento sirve para especificar el tiempo total expresado en segundos durante el cual el robot se mueve. Sustituye a los datos de velocidad de la instrucción.

Zona **Tipo de dato:** *zonedata*

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

[\Z] Tipo de dato: *num*

Este argumento sirve para especificar la precisión de la posición del TCP del robot directamente en la instrucción. La longitud de la trayectoria esquina está expresada en mm, y sustituye a la zona correspondiente especificada en los datos de zona.

[\InPos] Tipo de dato: *stoppointdata*

Este argumento se usa para especificar los criterios de convergencia del TCP del robot en el punto de paro. Este dato del punto de paro sustituye al dato de zona especificado en el parámetro *Zona*.

Herramienta Tipo de dato: *tooldata*

La herramienta utilizada durante el movimiento.
El TCP y la carga de la herramienta están definidos en los datos de herramienta. El TCP es el punto que se mueve a la posición de destino especificado.

[\WObj] Tipo de dato: *wobjdata*

Es el objeto de trabajo utilizado durante el movimiento.

Este argumento podrá ser omitido si la herramienta es sujetada por el robot. No obstante, si el robot sujeta el objeto de trabajo, es decir, si la herramienta es estacionaria, o con ejes externos coordinados, entonces se deberá especificar este argumento.

En el caso de una herramienta estacionaria o de ejes externos coordinados, los datos utilizados por el sistema para decidir la velocidad y la trayectoria esquina del movimiento, se encuentran definidos en el objeto de trabajo.

[\Corr] *(Corrección)* Tipo de dato: switch

Si este argumento está presente, los datos de corrección introducidos en una entrada de corrección mediante la instrucción *CorrWrite* serán añadidos a la trayectoria y a la posición de destino.

Ejecución del programa

El robot y las unidades externas se moverán al punto de destino de la siguiente manera:

- El TCP de la herramienta se moverá de forma circular a la velocidad constante programada.
- La herramienta es reorientada a una velocidad constante desde la orientación de la posición de inicio a la orientación del punto de destino.
- La reorientación se lleva a cabo respecto a la trayectoria circular. Así durante el movimiento, la orientación permanecerá constante si la orientación respecto a la trayectoria es la misma en el punto de inicio y en el punto de destino (véase la Figura 2).

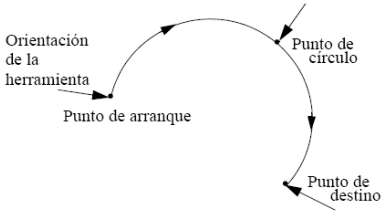


Figura 2 Orientación de la herramienta durante el movimiento circular.

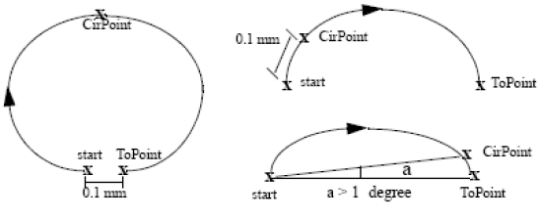
La orientación en el punto de círculo no es crítica; sirve únicamente para distinguir entre dos direcciones posibles de reorientación. La precisión de la reorientación sobre la trayectoria dependerá únicamente de la orientación en el punto de inicio y en el punto de destino.

En el caso en que la reorientación o los ejes externos no puedan alcanzar la velocidad programada, se reducirá la velocidad del TCP.

Se genera una trayectoria esquina cuando la trayectoria continúa a una siguiente posición. En el caso en que se haya especificado un punto de paro en los datos de zona, la ejecución del programa continuará cuando el robot y los ejes externos hayan alcanzado la posición de ejes adecuada.

Limitaciones

Hay algunas limitaciones en como se deben situar las posiciones *PuntoCirculo (CirPoint)* y *AlPunto (ToPoint)*



- Mínima distancia entre el inicio y *AlPunto (ToPoint)* es de 0,1 mm
- Mínima distancia entre el inicio y *PuntoCirculo (CirPoint)* es de 0,1 mm
- Mínimo ángulo entre *PuntoCirculo (CirPoint)* y *AlPunto (ToPoint)* desde el punto de inicio es de 1 °.

La precisión es baja cerca de los límites, así por ejemplo, si el punto de inicio y la posición *AlPunto(To Point)* están muy cerca una de otra, el fallo generado por la inclinación del circulo puede ser mayor que la precisión de los puntos que se han programado.

No se permite un cambio del modo de ejecución, de delante hacia atrás o viceversa, mientras el robot está parado en una trayectoria circular por lo que se generará un mensaje de error.

La instrucción *MoveC* (o cualquier otra instrucción que incluya un movimiento circular) no deberá nunca ser arrancada desde el principio, con el TCP entre el punto de círculo y el punto final. De lo contrario el robot no seguirá la trayectoria programada, sino que seguirá otra dirección comparada con la programada.

El usuario deberá asegurarse de que el robot puede alcanzar el punto de círculo durante la ejecución del programa y dividir el circulo en varios segmentos circulares si fuera necesario para poder realizar la trayectoria circular deseada.

Sintaxis

MoveC
['\' Conc ',']
[PuntoCírculo':='] < expresión (**IN**) de *robtarget* > ','
[AlPunto':='] < expresión (**IN**) de *robtarget* > ','
[Velocidad':='] < expresión (**IN**) de *speeddata* >
['\' V ':=' < expresión (**IN**) de *num* >]
[['\' T ':=' < expresión (**IN**) de *num* >] ','
[Zona ':='] < expresión (**IN**) de *zonedata* >
['\' Z ':=' < expresión (**IN**) de *num* >] ','
[Herramienta ':='] < persistente (**PERS**) de *tooldata* >
['\' WObj ':=' < persistente (**PERS**) de *wobjdata* >]
['\' Corr]';'

MoveJ Movimiento eje a eje del robot

MoveJ (Move Joint) sirve para mover el robot rápidamente desde un punto a otro cuando este movimiento no tiene que seguir una línea recta.

El robot y los ejes externos se moverán a la posición de destino siguiendo una trayectoria que no es lineal. Todos los ejes alcanzarán la posición de destino al mismo tiempo.

Esta instrucción solo se puede utilizar en la tarea *Main,* o en las tareas de movimiento en sistemas MultiMove.

Ejemplos

MoveJ p1, vmax, z30, herram2;
El punto central de la herramienta (TCP), *herram2*, se moverá siguiendo una trayectoria que no es lineal para alcanzar la posición, *p1*, con el dato de velocidad *vmax* y el dato de zona *z30*.

MoveJ p45, v2000\V:=2200, z40 \Z:=45, pinza3;
El TCP de la herramienta, *pinza3*, se moverá siguiendo una trayectoria que no es lineal, a una posición almacenada en la instrucción. El movimiento se llevará a cabo con los datos de *v2000* y *z40*; la velocidad y el tamaño de la zona del TCP son de *2200* mm/s y de *45* mm respectivamente.

MoveJ \Conc, pCoger, v2000, z40, pinza3;
El TCP de la herramienta, *pinza3*, se moverá siguiendo una trayectoria que no es lineal, a una posición almacenada en la instrucción. Las instrucciones lógicas siguientes se ejecutarán mientras el robot se mueve.

MoveJ pArranque, v2000, z40, pinza3 \WObj:=fijación;
El TCP de la herramienta, *pinza3*, se moverá siguiendo una trayectoria que no es lineal, a una posición, *pArranque.* Esta posición está especificada en el sistema de coordenadas del objeto para *fijación*.

Argumentos

MoveJ [\Conc] AlPunto [\ID] Velocidad [\V] | [\T] Zona [\Z] [\InPos] Herramienta [\WObj]

[\Conc] **Tipo de dato:** *switch*

Las instrucciones lógicas siguientes se ejecutarán inmediatamente. Este argumento sirve para acortar el tiempo de ciclo cuando, por ejemplo, se está comunicando con el equipo externo y no se requiere ninguna sincronización.

Utilizando el argumento *\Conc,* el número de instrucciones de movimiento que se ejecutarán, está limitado a 5. No se puede utilizar en una sección de programa que incluye *StorePath-RestoPath.*

Si se omite este argumento y AlPunto no es un punto de paro, la siguiente instrucción se ejecutará poco tiempo antes de que el robot haya alcanzado la zona programada.

Este argumento no se puede usar en movimientos coordinados sincronizados en sistemas MultiMove.

AlPunto **Tipo de dato:** *robtarget*

Es el punto de destino del robot y de los ejes externos. Está definido como una posición con nombre o es almacenado directamente en la instrucción (marcado con un asterisco * en la instrucción).

[\ID] **Tipo de dato:** *identificador*

Este argumento se puede usar en sistemas MultiMove, si el movimiento es coordinado y sincronizado y no se pude utilizar en los otros casos.

El número especificado de id debe ser el mismo en todas las tareas del programa que deban cooperar entre si. Este número permite garantizar que los movimientos no se mezclaran durante la ejecución.

Velocidad **Tipo de dato:** *speeddata*

Son los datos de velocidad que se aplican a los movimientos. Definen la velocidad del punto central de la herramienta, de la reorientación de la herramienta y de los ejes externos.

[\V] *(Velocidad)* **Tipo de dato:** *num*

Este argumento sirve para especificar la velocidad del TCP en mm/s directamente en la instrucción, que substituirá a la velocidad correspondiente especificada en los datos de velocidad.

[\T] *(Tiempo)* **Tipo de dato:** *num*

Este argumento sirve para especificar el tiempo total expresado en segundos durante el cual el robot se mueve. Sustituye a los datos de velocidad de la instrucción.

Zona **Tipo de dato:** *zonedata*

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

[\Z] *(Zona)* **Tipo de dato:** *num*

Este argumento sirve para especificar la precisión de la posición del TCP del robot directamente en la instrucción. La longitud de la trayectoria esquina está expresada en mm, y sustituye a la zona correspondiente especificada en los datos de zona.

[\InPos] **Tipo de dato:** *stoppointdata*

Este argumento se usa para especificar los criterios de convergencia del TCP del robot en el punto de paro. Este dato del punto de paro sustituye al dato de zona especificado en el parámetro *Zona*.

Herramienta **Tipo de dato:** *tooldata*

La herramienta utilizada durante el movimiento.
El TCP y la carga de la herramienta están definidos en los datos de herramienta.
El TCP sirve para decidir la velocidad y la trayectoria esquina del movimiento.

[\WObj] *(Objeto de trabajo)* **Tipo de dato:** *wobjdata*

Es el objeto de trabajo utilizado durante el movimiento.

Este argumento podrá ser omitido si la herramienta es sujetada por el robot.

No obstante, si el robot sujeta el objeto de trabajo, es decir, si la herramienta es estacionaria, o con ejes externos coordinados, entonces se deberá especificar este argumento.

En el caso de una herramienta estacionaria o de ejes externos coordinados, los datos utilizados por el sistema para decidir la velocidad y la trayectoria esquina del movimiento, se encuentran definidos en el objeto de trabajo.

Ejecución del programa

El TCP de la herramienta se moverá al punto de destino con una interpolación de los ángulos de los ejes. Ello significa que cada eje se moverá a una velocidad constante y que todos los ejes alcanzarán el punto de destino al mismo tiempo, lo cual originará una trayectoria que no es lineal, eso significa que el TCP se moverá aproximadamente a la velocidad programada (independientemente de si los ejes externos están coordinados o no). La herramienta será reorientada y los ejes externos se moverán al mismo tiempo que se mueve el TCP. En el caso en que la reorientación o los ejes externos no puedan alcanzar la velocidad programada, se reducirá la velocidad del TCP.

Una trayectoria esquina suele ser generada cuando el movimiento es transferido a la sección siguiente de una trayectoria. Si se especifica un punto de paro en los datos de zona, la ejecución del programa únicamente continuará cuando el robot y los ejes externos hayan alcanzado la posición adecuada.

Sintaxis

MoveJ
['\' Conc ',']
[AlPunto':='] < expresión (**IN**) de *robtarget* > ','
[Velocidad':='] < expresión (**IN**) de *speeddata* >
['\' V ':=' < expresión (**IN**) de *num* >]
| ['\' T ':=' < expresión (**IN**) de *num* >] ','
[Zona ':='] < expresión (**IN**) de *zonedata* >
['\' Z ':=' < expresión (**IN**) de *num* >] ','
[Herramienta':='] < persistente (**PERS**) de *tooldata* >
['\' WObj ':=' < persistente (**PERS**) de *wobjdata* >] ';'

MoveL Movimiento del robot en trayectorias rectas

MoveL sirve para mover el TCP de la herramienta a una posición de determinada siguiendo una trayectoria recta. Cuando el TCP debe permanecer estacionario, esta instrucción podrá ser utilizada también para reorientar la herramienta.

Ejemplo

MoveL p1, v1000, z30, herram2;
El TCP de la herramienta, *herram*2, se moverá de forma lineal a la posición *p1,* con el dato de velocidad *v1000* y el dato de zona *z30*.

MoveL *, v1000\T:=5, fine, pinza3;
El TCP de la herramienta, *pinza3*, se moverá de forma lineal a un punto fino almacenado en la instrucción (marcado con un asterisco *). El movimiento completo tardará 5 segundos en llevarse a cabo.

Argumentos

MoveL [\Conc] AlPunto [\ID] Velocidad [\V] | [\T] Zona [\Z] [\InPos] Herram [\WObj] [\Corr]

[\Conc] *(Concurrente)* **Tipo de dato:** *switch*

Las instrucciones lógicas siguientes se ejecutarán inmediatamente. Este argumento sirve para acortar el tiempo de ciclo cuando, por ejemplo, se está comunicando con el equipo externo y no se requiere ninguna sincronización.

Utilizando el argumento *\Conc,* el número de instrucciones de movimiento que se ejecutarán, está limitado a 5. No se puede utilizar en una sección de programa que incluye *StorePath-RestoPath.*

Si se omite este argumento y AlPunto no es un punto de paro, la siguiente instrucción se ejecutará poco tiempo antes de que el robot haya alcanzado la zona programada.

AlPunto **Tipo de dato:** *robtarget*

Es el punto de destino del robot y de los ejes externos. Está definido como una posición con nombre o es almacenado directamente en la instrucción (marcado con un asterisco * en la instrucción).

[\ID] **Tipo de dato:** *identificador*

Este argumento se puede usar en sistemas MultiMove, si el movimiento es coordinado y sincronizado y no se pude utilizar en los otros casos.

El número especificado de id debe ser el mismo en todas las tareas del programa que deban cooperar entre si. Este número permite garantizar que los movimientos no se mezclaran durante la ejecución.

Velocidad **Tipo de dato:** *speeddata*

Son los datos de velocidad que se aplican a los movimientos. Definen la velocidad del punto central de la herramienta, de la reorientación de la herramienta y de los ejes externos.

[\V] *(Velocidad)* **Tipo de dato:** *num*

Este argumento sirve para especificar la velocidad del TCP en mm/s directamente en la instrucción, que substituirá a la velocidad correspondiente especificada en los datos de velocidad.

[\T] *(Tiempo)* **Tipo de dato:** *num*

Este argumento sirve para especificar el tiempo total expresado en segundos durante el cual el robot se mueve. Sustituye a los datos de velocidad de la instrucción.

Zona **Tipo de dato:** *zonedata*

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

[\Z] *(Zona)* **Tipo de dato:** *num*

Este argumento sirve para especificar la precisión de la posición del TCP del robot directamente en la instrucción. La longitud de la trayectoria esquina está expresada en mm, y sustituye a la zona correspondiente especificada en los datos de zona.

[\InPos] **Tipo de dato:** *stoppointdata*

Este argumento se usa para especificar los criterios de convergencia del TCP del robot en el punto de paro. Este dato del punto de paro sustituye al dato de zona especificado en el parámetro *Zona*.

Herramienta **Tipo de dato:** *tooldata*

La herramienta utilizada durante el movimiento.
El TCP y la carga de la herramienta están definidos en los datos de herramienta.
El TCP es el punto que se mueve a la posición de destino especificado.

[\WObj] *(Objeto de trabajo)* **Tipo de dato:** *wobjdata*

Es el objeto de trabajo utilizado durante el movimiento.

Este argumento podrá ser omitido si la herramienta es sujetada por el robot. No obstante, si el robot sujeta el objeto de trabajo, es decir, si la herramienta es estacionaria, o con ejes externos coordinados, entonces se deberá especificar este argumento.

En el caso de una herramienta estacionaria o de ejes externos coordinados, los datos utilizados por el sistema para decidir la velocidad y la trayectoria esquina del movimiento, se encuentran definidos en el objeto de trabajo.

[\Corr] **(Corrección)** **Tipo de dato: switch**

Si este argumento está presente, los datos de corrección introducidos en una entrada de corrección mediante la instrucción *CorrWrite* serán añadidos a la trayectoria y a la posición de destino.

Ejecución del programa

El robot y los ejes externas se moverán a la posición de destino de la siguiente manera:

- El TCP de la herramienta se moverá de forma lineal a la velocidad programada cte.
- La herramienta es reorientada a intervalos iguales a lo largo de la trayectoria.
- Los ejes externos no coordinados se moverán a una velocidad constante para que puedan llegar al punto de destino al mismo tiempo que los ejes del robot.

En el caso en que la reorientación o los ejes externos no puedan alcanzar la velocidad programada, la velocidad del TCP será reducida.

Una trayectoria esquina se genera cuando el movimiento es transferido a la sección siguiente de una trayectoria. Si se especifica un punto de paro en los datos de zona, la ejecución del programa continuará cuando el robot y los ejes externos hayan alcanzado la posición adecuada.

Ejemplos

MoveL *, v2000 \V:=2200, z40 \Z:=45, pinza3;
El TCP de la herramienta, *pinza3*, se moverá siguiendo una trayectoria lineal, a una posición almacenada en la instrucción. El movimiento se llevará a cabo con los datos de *v2000* y *z40*; la velocidad y el tamaño de la zona del TCP son de *2200* mm/s y de *45* mm respectivamente.

MoveJ \Conc, *, v2000, z40, pinza3;
El TCP de la herramienta, *pinza3*, se moverá siguiendo una trayectoria lineal, a una posición almacenada en la instrucción. Las instrucciones lógicas siguientes se ejecutarán mientras el robot se mueve.

MoveJ pArranque, v2000, z40, pinza3 \WObj:=fijación;
El TCP de la herramienta, *pinza3*, se moverá siguiendo una trayectoria lineal, a una posición, *pArranque.* Esta posición está especificada en el sistema de coordenadas del objeto de *fijación*.

Sintaxis

MoveL
['\' Conc ',']
[AlPunto':='] < expresión (**IN**) de *robtarget* > ','
[Velocidad':='] < expresión (**IN**) de *speeddata* >
['\' V ':=' < expresión (**IN**) de *num* >]
| ['\' T ':=' < expresión (**IN**) de *num* >] ','
[Zona ':='] < expresión (**IN**) de *zonedata* >
['\' Z ':=' < expresión (**IN**) de *num* >] ','
[Herramienta ':='] < persistente (**PERS**) de *tooldata* >
['\' WObj ':=' < persistente (**PERS**) de *wobjdata* >]
['\' Corr]';'

ProcCall Llamada a una rutina

Una llamada a una rutina sirve para transferir la ejecución del programa a otro procedimiento.

Cuando el procedimiento ha sido ejecutado completamente, la ejecución del programa continúa con la instrucción que sigue a la llamada a la rutina.

Si se desea, se puede enviar una serie de argumentos a la rutina. Estos tendrán por misión controlar el comportamiento de la rutina y hacer que sea posible utilizar la misma rutina para diferentes aplicaciones.

Ejemplos

limpieza;
Llamada a la rutina *limpieza*.

mensajerror;
Set activar_lampErr;
..
PROC mensajerror()
TPWrite "ERROR";
ENDPROC
Se llamará a la rutina *mensajerror*. Cuando haya finalizado la ejecución de esta rutina, la ejecución del programa continuará con la instrucción que sigue a la llamada de la rutina, *Set activar_lampErr*.

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).

Rutina { Argumento }

Identificador

El nombre de la rutina que se desea llamar.

Argumento

Tipo de dato: De acuerdo con la declaración de la rutina.

Los argumentos de la rutina, de acuerdo con los parámetros de la rutina.

Limitaciones

Los argumentos de la rutina deben coincidir con sus parámetros:

- Todos los argumentos obligatorios deberán ser incluidos.
- Deberán ser colocados siguiendo el mismo orden.
- Deberán ser del mismo tipo de dato.
- Deberán ser del tipo correcto en relación al modo de acceso (entrada, varible o persistente).

Una rutina puede llamar a otra que, a su vez llama a otra, y así sucesivamente. Una rutina también puede llamarse a sí mismo, es decir, que realizará una llamada recursiva.

El número de niveles de rutina permitido dependerá del número de parámetros, pero por lo general se permiten más de 10 niveles.

Sintaxis

(EBNF)
<rutina> [<lista argumento>] ';'
<rutina> ::= <identificador>

PulseDO Generación de un pulso en una salida digital

PulseDO sirve para generar un pulso en una salida digital.

Ejemplos

PulseDO doSalida15;
Se generará un pulso de 0,2 s en la salida *doSalida15*.

PulseDO \PLength:=1.0, doConexión;
Se generará un pulso de 1 s en la señal *doConexión*.

Argumentos

PulseDO [\PLength] Salida

[\PLength] Tipo de dato: *num*

La duración del pulso en segundos (0,1 - 32 seg.).
Si se omite el argumento, se generará un pulso de 0,2 segundos.

Salida Tipo de dato: *signaldo*

El nombre de la salida en la que se deberá generar el pulso.

Ejecución del programa

Se generará un pulso con la duración indicada en la instrucción. (Véase la Figura 1).

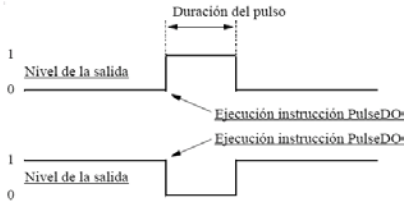


Figura 1 Generación de un pulso en una salida digital.

La instrucción siguiente se ejecutará después del inicio del pulso. En la salida seleccionada se generará un pulso sin afectar el resto de la ejecución del programa.

Limitaciones

La duración del pulso tiene una resolución de 0,01 segundos. Los valores programados que difieran de ésta serán redondeados.

Sintaxis

PulseDO
['\' PLength ':=' < expresión (**IN**) de *num* > ',']
[Salida ':='] < variable (**VAR**) de *signaldo* > ';'

Reset Puesta a cero de una salida digital

Reset sirve para poner una salida digital a cero.

Ejemplos

Reset doSalida15;
La señal *doSalida15* se pondrá a 0.

Reset dosold;
La señal *dosold* se pondrá a 0.

Argumentos

Reset Salida

Salida Tipo de dato: *signaldo*

El nombre de la salida digital que se desea poner a 0.

Ejecución del programa

El valor verdadero depende de la configuración de la señal. Si la señal ha sido invertida en los parámetros del sistema, la instrucción hará que la salida se ponga a 1.

Gestión de errores

En el caso de que no se establezca contacto con la unidad, se genera un error que se puede tratar en un gestor de errores.

La variable ERRNO adquiere el valor de ERR_NORUNUNIT

Sintaxis

Reset
[Salida ':='] < variable (**VAR**) de *signaldo* > ';'

Set Activación de una salida digital

Set sirve para colocar a uno el valor de una salida digital.

Ejemplos

Set doSalida15;
La salida *doSalida15* se pondrá a 1.

Set doSold;
La salida *dosSold* se pondrá a1.

Argumentos

Set Señal

Señal **Tipo de señal:** *signaldo*

El nombre de la salida que se desea poner a 1.

Ejecución del programa

El valor verdadero dependerá de la configuración de la señal. En el caso en que la señal haya sido invertida en los parámetros del sistema, la instrucción hará que el canal físico se ponga en cero.

Gestión de errores

En el caso de que no se establezca contacto con la unidad, se genera un error que se puede tratar en un gestor de errores.

La variable ERRNO adquiere el valor de ERR_NORUNUNIT

Sintaxis

Set
[Señal ':='] < variable (**VAR**) de *signaldo* > ';'

SetDO Cambio del valor de una salida digital

SetDO sirve para cambiar el valor de una salida digital, con o sin un retraso.

Ejemplos

SetDO do15, 1;
La salida *do15* se activará a *1.*

SetDO \SDelay := 0,2, dosold, 1;
La salida *dosold* pasará a 1 con un retraso de *0,2* s. La ejecución del programa no se verá alterada, ya que continuará en la instrucción siguiente.

Argumentos

SetDO [\SDelay][\Sync] Salida Valor

[\SDelay] Tipo de dato: *num*

Retrasa el cambio en el tiempo especificado en segundos (0,1 - 32 seg.). La ejecución del programa continúa directamente con la instrucción siguiente.

Una vez transcurrido el tiempo de retardo, la señal cambia sin afectar por ello el resto de la ejecución del programa. Si se omite el argumento, el valor de la señal cambia instantáneamente.

[\Sync] Tipo de dato: *switch*

Si se usa este argumento, la ejecución del programa no proseguirá hasta que la señal haya cambiado físicamente al valor indicado.

Si no se usa ni el argumento \Sync ni \SDelay, la señala se activará lo más rápido posible, pero la ejecución del programa proseguirá aunque la señal aun no tenga el valor indicado.

Salida Tipo de dato: *signaldo*

El nombre de la salida que se desea cambiar.

Valor Tipo de dato: *dionum*

El valor deseado de la señal. El valor se especifica como 0 o 1. En el caso de que se indique cualquier valor diferente de 0, el sistema lo toma como 1.

Ejecución del programa

El valor verdadero dependerá de la configuración de la salida. En el caso en que la salida haya sido invertida en los parámetros del sistema, el valor del canal físico será el opuesto.

Gestión de errores

En el caso de que no se establezca contacto con la unidad, se genera un error que se puede tratar en un gestor de errores.
La variable ERRNO adquiere el valor de ERR_NORUNUNIT

Sintaxis

SetDO
['\' SDelay ':=' < expresión (**IN**) de *num* > ',']
[Señal ':='] < variable (**VAR**) de *signaldo* > ','
[Valor ':='] < expresión (**IN**) de *dionum* > ';'

SetGO Cambio del valor de un grupo de salidas digitales

SetGO sirve para cambiar el valor de un grupo de salidas digitales, con o sin un retraso de tiempo.

Ejemplo

SetGO gogrup2, 12;
El grupo Go*grup2* se activará en *12*. Si Go*grup2* comprende 4 señales, por ejemplo, las salidas 6-9, las salidas 6 y 7 se pondrán a cero, mientras que la 8 y la 9 se activarán en 1.

SetGO \SDelay :=0,4, gogrup2, 11;
La señal *gogrup2* se activará en *10*. Si go*grup2* comprende 4 señales, por ejemplo, las salidas 6-9, las salida 8 se pondrá a cero, mientras que la 6, 7 y la 9 se activarán en 1, con un retraso de *0,4s*. La ejecución del programa no se verá alterada, ya que continuará en la instrucción siguiente.

Argumentos

SetGO [\SDelay] Señal Valor

[\SDelay] **Tipo de dato:** *num*

Retrasa el cambio del tiempo especificado en segundos (0,1 - 32 seg.).

La ejecución del programa continúa directamente con la instrucción siguiente.

Una vez transcurrido el tiempo de retraso, el valor de las salidas cambia sin afectar por ello el resto de la ejecución del programa.

Si se omite el argumento, el valor de la señal cambia directamente.

Señal **Tipo de dato:** *signalgo*

El nombre del grupo de salidas que se desea cambiar.

Valor **Tipo de dato:** *num*

El valor deseado para el grupo de salidas, que es un número entero positivo. El valor permitido depende del número de señales del grupo:

Nº de salidas	Valor permitido	Nº de salidas	Valor permitido
1	0 - 1	9	0 - 511
2	0 - 3	10	0 - 1023
3	0 - 7	11	0 - 2047
4	0 - 15	12	0 - 4095
5	0 - 31	13	0 - 8191
6	0 - 63	14	0 - 16383
7	0 - 127	15	0 - 32767
8	0 - 255	16	0 - 65535

Ejecución del programa

El valor programado se convertirá en un número binario sin signo. Este número binario será enviado al grupo de salidas con el resultado que las cada unas de las salidas individuales del grupo se pondrán a 0 o a 1. Debido a retrasos internos del propio sistema, el valor de una salida puede tener un valor indefinido durante un corto periodo de tiempo.

Gestión de errores

En el caso de que no se establezca contacto con la unidad, se genera un error que se puede tratar en un gestor de errores.

La variable ERRNO adquiere el valor de ERR_NORUNUNIT

Sintaxis

SetGO
['\' SDelay ':=' < expresión (**IN**) de *num* > ',']
[Señal ':='] < variable (**VAR**) de *signalgo* > ','
[Valor ':='] < expresión (**IN**) de *num* > ';'

Stop Paro de la ejecución del programa

Stop sirve para detener temporalmente la ejecución del programa.

La ejecución del programa también podrá ser detenida utilizando la instrucción *EXIT*. Esto, no obstante, sólo deberá realizarse si se ha terminado una tarea, o si se produce un error muy grave, ya que la ejecución del programa no podrá ser rearrancada después de un *EXIT*.

Ejemplo

```
TPWrite "Error en la línea de comunicaciones con el PC de supervisión";
Stop;
```

La ejecución del programa se detendrá después de que aparezca un mensaje en la unidad de programación.

Argumentos

Stop [\NoRegain]

[\NoRegain] **Tipo de dato:** *switch*

Este argumento especifica si para el siguiente arranque del programa en el modo manual, el robot y los ejes externos deben o no regresar a la posición donde se ha parado. En el modo automático el robot y los ejes externos siempre regresan a la posición de paro.

En el caso en que el argumento *NoRegain* esté activado, el robot y los ejes externos no regresarán a la posición de paro (en el caso de que se hayan sido movidos de dicha posición).

En el caso en que se omita el argumento y que el robot o los ejes externos se hayan movido de la posición de paro, el robot visualizará una pregunta en la unidad de programación.

Entonces, el usuario deberá contestar a la pregunta especificando si desea que el robot regrese a la posición del paro o no.

Ejecución del programa

La instrucción detendrá la ejecución del programa en cuanto el robot y los ejes externos alcancen la posición de destino programada en el movimiento que está realizando en el momento. La ejecución del programa podrá entonces ser rearrancada a partir de la siguiente instrucción.

Si hay una instrucción de *Stop* en alguna rutina de evento, la rutina será ejecutada desde el principio en el siguiente evento.

Si la instrucción *Stop* se usa en una tarea declarada Estática o Semiestática, su funcionamiento dependerá del valor del parámetro *TrustLevel*.

Ejemplo

```
MoveL p1, v500, fine, tool1;
TPWrite "Mover el robot a la posición esquina del pallet 1";
Stop \NoRegain;
p1_leido := CRobT();
MoveL p2, v500, z50, tool1;
```

La ejecución del programa se detiene con el robot situado en la posición *p1*. El usuario mueve el robot a *p1_leido*. En el rearranque de programa, el robot no regresará a *p1,* sino que almacenará la posición *p1_leido* y se moverá a *p2*.

Sintaxis

```
Stop';'
   [ '\' NoRegain ]';'
```

TEST Dependiendo del valor de una expresión...

TEST sirve cuando diferentes instrucciones deben ser ejecutadas dependiendo del valor de una expresión o de un dato.

En el caso en que no haya demasiadas alternativas, se puede usar también la instrucción *IF..ELSE.*

Ejemplo

```
TEST reg1
CASE 1,2,3 :
rutina1;
CASE 4 :
rutina2;
DEFAULT :
TPWrite "Elección ilegal";
Stop;
ENDTEST
```

Diferentes instrucciones serán ejecutadas dependiendo del valor de *reg1*. En el caso en que el valor sea 1-3, la *rutina1* será ejecutada. Si el valor es 4, la *rutina2* será ejecutada. De lo contrario, aparecerá un mensaje de error y la ejecución se detendrá.

Argumentos

TEST Dato Test {CASE Valor Test {, Valor Test} : ...}

[DEFAULT: ...] ENDTEST

Dato Test **Tipo de dato:** Todos

El dato o expresión con el que se desea que el valor test sea comparado.

Valor Test **Tipo de dato:** Igual que Dato Test

El valor que el dato test debe tener para que las instrucciones asociadas se ejecuten.

Ejecución del programa

El dato test será comparado con los valores test de la primera condición CASE. En el caso en que la comparación sea verdadera, las instrucciones asociadas se ejecutarán. Después de ello, la ejecución del programa continúa con la instrucción que sigue a ENDTEST.

Si la primera condición CASE no se cumple, se comprobará la siguiente condición CASE y así, sucesivamente. En el caso en que no se cumpla ninguna de las condiciones, las instrucciones asociadas con DEFAULT (si existe) serán ejecutadas.

Sintaxis

(EBNF)

TEST <expresión>
{(**CASE** <valor test> { ',' <valor test> } ':'
<lista instrucciones>) | **<CSE>** }
[**DEFAULT** ':' <lista instrucciones>]
ENDTEST
<valor test> ::= <expresión>

TPErase Borrado del texto visualizado en la FlexPendant

TPErase (Teach Pendant Erase) sirve para borrar el contenido de la pantalla de la unidad de programación táctil.

Ejemplo

```
TPErase;
TPWrite "Ejecución comenzada";
```

El contenido de la pantalla de la unidad de programación táctil será borrado antes de que aparezca el mensaje *Ejecución comenzada.*

Ejecución del programa

La pantalla de la unidad de programación táctil quedará libre de todo texto. La próxima vez que se introduzca texto, aparecerá en la línea superior de la misma.

Sintaxis

```
TPErase;
```

TPReadFK Lee las teclas de función

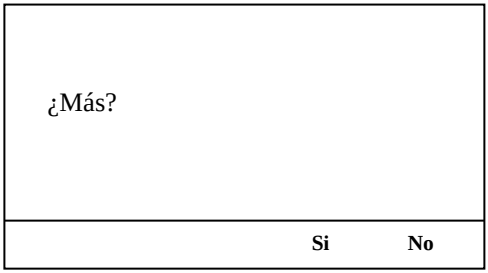
TPReadFK se utiliza para escribir un texto en las teclas de función y para determinar qué tecla de función se ha presionado.

Ejemplos

TPReadFK reg1, "¿Más?", stEmpty, stEmpty, stEmpty, "Sí", "No";

Se escribe el texto ¿Más? en la pantalla del FlexPendant y se activan las teclas de función 4y 5 usando las cadenas de texto Sí 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.



Argumentos

TPReadFK TPAnswer TPText TPFK1 TPFK2 TPFK3 TPFK4 TPFK5 [\MaxTime][\DlBreak] [\DOBreak] [\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

Texto de tecla de función
Tipo de dato: string
El texto que debe escribirse en tecla de función adecuada (con un máximo de 42 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]
Interrupción de entrada digital
Tipo de dato: signaldi
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.

[\DOBreak]
Interrupción de salida digital
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.

[\BreakFlag]
Tipo de dato: errnum
Una variable que contiene el código de error si se utilizan MaxTime, DIBreak o DOBreak.
Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes ERR_TP_MAXTIME, ERR_TP_ DIBREAK y ERR_TP_DOBREAK 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.

Gestión 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 produce una salida digital (parámetro \DOBreak) antes de una acción por parte del operador, la variable de sistema ERRNO cambia a ERR_TP_DOBREAK y la ejecución prosigue en el gestor de errores.

A continuación, estas situaciones pueden ser gestionadas en el gestor de errores.

Limitaciones

Evite usar un valor demasiado pequeño para el parámetro de tiempo límite \MaxTime si TPReadFK se ejecuta frecuentemente, por ejemplo en un bucle. Si lo hace, puede dar lugar a un comportamiento impredecible del rendimiento del sistema, por ejemplo una respuesta demasiado lenta de la unidad de programación.

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>]
['\'DOBreak':='<variable (VAR) of signaldo>]
['\'BreakFlag':='<var or pers (INOUT) of errnum>]';'
```

TPReadNum Lee un número del FlexPendant

TPReadNum se utiliza para leer un número del FlexPendant.

Ejemplos

TPReadNum reg1, "¿Cuántas unidades es necesario producir?";

Se escribe el texto ¿Cuántas unidades es necesario producir? en la pantalla del 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] [\DOBreak]

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]
Interrupción de entrada digital
Tipo de dato: signaldi
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.

[\DOBreak]
Interrupción de salida digital
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.

[\BreakFlag]

Tipo de dato: errnum

Una variable que contiene el código de error si se utilizan MaxTime, DIBreak o DOBreak.Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes ERR_TP_MAXTIME, ERR_TP_ DIBREAK y ERR_TP_DOBREAK 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.

Gestión 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 produce una salida digital (parámetro \DOBreak) antes de una acción por parte del operador, la variable de sistema ERRNO cambia a ERR_TP_DOBREAK y la ejecución prosigue en el gestor de errores.

Si no hay ningún cliente, por ejemplo un FlexPendant, que se encargue de la instrucción, la variable de sistema ERRNO cambia a ERR_TP_NO_CLIENT y la ejecución continúa en el gestor de errores. A continuación, estas situaciones pueden ser gestionadas en el gestor de errores.

Sintaxis

TPReadNum
[TPAnswer':='] <var or pers (**INOUT**) of num>','
[TPText':='] <expression (**IN**) of string>
['\'MaxTime':='<expression (**IN**) of num>]
['\'DIBreak':='<variable (**VAR**) of signaldi>]
['\'DOBreak':='<variable (**VAR**) of signaldo>]
['\'BreakFlag':='<var or pers (**INOUT**) of errnum>] ';'

TPWrite Escribe en el FlexPendant

TPWrite se utiliza para escribir texto en el FlexPendant. Es posible escribir el valor de determinados datos, además de texto.

Ejemplos

A continuación aparecen algunos ejemplos básicos de la instrucción TPWrite.

TPWrite "Ejecución iniciada";
El texto Ejecución iniciada se escribe en el FlexPendant.

TPWrite "Nº de piezas producidas="\Num:=reg1;
Por ejemplo, si reg1 contiene el valor 5, se escribe el texto Nº de piezas producidas=5 en el FlexPendant.

Argumentos

TPWrite String [\Num] | [\Bool] | [\Pos] | [\Orient]

String

Tipo de dato: string
La cadena de texto a escribir (con un máximo de 80 caracteres y 40 caracteres por fila).

[\Num]

Número
Tipo de dato: num
El dato cuyo valor numérico se desea escribir a continuación de la cadena de texto.

[\Bool]

Booleano
Tipo de dato: bool
El dato cuyo valor lógico se desea escribir a continuación de la cadena de texto.

[\Pos]

Posición
Tipo de dato: pos
El dato cuya posición se desea escribir a continuación de la cadena de texto.

[\Orient]

Orientación
Tipo de dato: orient
El dato cuya orientación 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, \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:

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, \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>]
| ['\'Orient':='<expression (**IN**) of orient>]';'

WaitDI Esperar hasta la activación de una entrada digital

WaitDI (Wait Digital Input) sirve para que el robot espere hasta que se active una entrada digital.

Ejemplos

WaitDI diEntrada4, 1;
La ejecución del programa continuará sólo después de que la entrada *diEntrada4* se haya puesto a 1.

WaitDI diestado_pinza, 0;
La ejecución del programa continuará sólo después de que la entrada *diestado_pinza* se haya puesto a 0.

Argumentos

WaitDI Señal Valor [\MaxTime] [\TimeFlag]

Señal **Tipo de dato:** *signaldi*

Es el nombre de la entrada.

Valor **Tipo de dato:** *dionum*

Es el valor deseado de la entrada.

[\MaxTime] **Tipo de dato:** *num*

Es el intervalo máximo de tiempo, en segundos, de espera permitido. En el caso de que haya transcurrido este intervalo de tiempo sin que se haya cumplido la condición, el sistema llamará al gestor de errores, si existe. La variable ERRNO adquiere el valor ERR_WAIT_MAXTIME que podrá ser utilizada para comprobar si ha transcurrido o no el intervalo máximo de tiempo especificado.

Si el sistema no dispone de ningún gestor de errores, la ejecución del programa se detendrá.

[\TimeFlag] **Tipo de dato:** *bool*

Este parámetro adquiere el valor TRUE si el tiempo máximo de espera ha transcurrido sin que la condición se haya cumplido. En el caso de que este parámetro esté incluido en la instrucción, no se considera un error que haya transcurrido el tiempo máximo de espera. Este argumento es ignorado si no se incluye en la instrucción el argumento *Maxtime*.

Ejecución del programa

En el caso en que el valor de la entrada sea el indicado cuando se ejecuta la instrucción, el programa continuará con la siguiente instrucción.

En el caso en que el valor de la entrada no sea el indicado, el robot entra en un estado de espera y cuando la entrada adopta el valor indicado, el programa continuará. El cambio es detectado mediante una interrupción interna, que proporciona una respuesta rápida.

Cuando el robot está esperando, el tiempo será supervisado, y en el caso en que exceda el valor máximo de tiempo, el programa continuará siempre y cuando esté especificado *TimeFlag*, o generará un error en el caso en que no lo esté. Si se especifica *TimeFlag*, adquirirá el valor TRUE cuando el tiempo máximo de espera haya sido excedido y tendrá el valor FALSE en caso contrario.

Sintaxis

WaitDI
[Señal ':='] < variable (**VAR**) de *signaldi* > ','
[Valor ':='] < expresión (**IN**) de *dionum* > ';'
['\'MaxTime ':='<expresión (**IN**) de *num*>]
['\'TimeFlag':='<variable (**VAR**) de *bool*>] ';'

WHILE Repetición de una instrucción mientras...

WHILE se utiliza cuando una serie de instrucciones deben ser repetidas mientras se vaya cumpliendo una condición determinada.

En el caso en que sea posible determinar por adelantado el número de repeticiones, se puede utilizar la instrucción *FOR*.

Ejemplo

```
WHILE reg1 < reg2 DO
...
reg1 := reg1 +1;
ENDWHILE
```

Repite las instrucciones del bucle WHILE - ENDWHILE mientras *reg1 < reg2*.

Argumentos

WHILE Condición DO ... ENDWHILE

Condición **Tipo de dato:** *bool*

La condición que se debe cumplir para que las instrucciones del bucle WHILE - ENDWHILE puedan ejecutarse.

Ejecución del programa

1. Se verifica la condición. Si la condición no se cumple, el bucle WHILE finaliza y la ejecución del programa continúa con la instrucción que sigue a la instrucción ENDWHILE. Si se cumple la condición, la ejecución del programa continua con las instrucciones contenidas dentro del bloque WHILE - ENDWHILE.

2. La condición se evalúa otra vez y se repite lo explicado en el punto 1.

3. El proceso continua hasta que el resultado de la evaluación es FALSO. Entonces la ejecución del programa continúa con la primera instrucción después del ENDWHILE.

Sintaxis

(EBNF)
WHILE <expresión condicional> **DO**
<lista instrucciones>
ENDWHILE

WaitTime Esperar durante un tiempo determinado

WaitTime sirve para que el robot espere un tiempo determinado. Esta instrucción puede utilizarse también para esperar hasta que el robot y los ejes externos se hayan parado.

Ejemplo

WaitTime 0.5;
La ejecución del programa espera 0,5 segundos.

WaitTime \InPos,0;
La ejecución del programa espera hasta que el robot y los ejes externos se hayan parado.

Argumentos

WaitTime [\InPos] Tiempo

[\InPos] **Tipo de dato:** *switch*

En el caso en que se utilice este argumento, tanto el robot como los ejes externos deberán haberse parado antes de que el tiempo de espera empiece a contar.

Este argumento solo se puede utilizar si la tarea controla cualquier tipo de unidades mecánicas.

Tiempo **Tipo de dato:** *num*

El tiempo, en segundos, en que la ejecución del programa debe esperar. El valor mínimo es de 0 s. y no tiene valor máximo. La resolución es de 0,001s.

Ejecución del programa

La ejecución del programa se detiene temporalmente durante un tiempo especificado, pero la gestión de las interrupciones y otras funciones similares, siguen activas.

En modo manual, si se selecciona el argumento \InPos y Tiempo es mayor de 3 s., se visualiza un cuadro de dialogo, para ofrecer la posibilidad de simular esta espera y continuar con la ejecución del programa. Si no se desea que aparezca este cuadro, se debe asignar el valor NO al parámetro de sistema SimulateMenu (Parámetros del sistema, Temas: Controlador, System Misc..

Limitaciones

El argumento *\Inpos* no podrá utilizarse junto con el servo suave (SoftServo).

Sintaxis

```
WaitTime
   ['\'InPos',']
   [Tiempo ':='] <expresión (IN) de num>';'
```

WaitUntil Esperar hasta el cumplimiento de una condición

WaitUntil sirve para esperar hasta que se cumpla una condición lógica; por ejemplo, el sistema podrá esperar hasta que se activen una o varias entradas.

Ejemplo

```
WaitUntil diEntrada4= 1;
```

La ejecución del programa continuará sólo después de que *diEntrada4* valga 1.

```
VAR bool tempmax;
WaitUntil diestado_pinza= 1\MaxTime := 60 \TimeFlag := tempmax;
IF tempmax:= TRUE THEN
TPWrite "No se ha recibido orden de arranque en el tiempo especificado";
ELSE
ciclo_sig;
ENDIF
```

Si la condición de entrada no se han cumplido en *60 s*egundos, aparecerá un mensaje de error en la pantalla de la unidad de programación. En el caso de que se cumpla, el programa continuará llamando a la rutina *ciclo_sig*

```
WaitUntil \InPos, diEntrada4=0 AND diEntrada6= 1;
```

La ejecución del programa esperará hasta que el robot se haya parado y que la entrada *diEntrada4* haya pasado a nivel 0 y la entrada *diEntrada6* valga 1.

Argumentos

WaitUntil [\InPos] Cond [\MaxTime] [\TimeFlag]

[\InPos] **Tipo de dato:** *switch*

En el caso en que se utilice este argumento, tanto el robot como los ejes externos deberán haberse parado antes de que se compruebe la condición. Este argumento solo se puede utilizar si la tarea controla unidades mecánicas.

Cond **Tipo de dato:** *bool*

Es la condición lógica que se debe esperar a que se cumpla.

[\MaxTime] **Tipo de dato: num**

Es el intervalo máximo de tiempo, en segundos, de espera permitido. En el caso de que haya transcurrido este intervalo de tiempo sin que se haya cumplido la condición, el sistema llamará al gestor de errores, si existe. La variable ERRNO adquiere el valor ERR-WAIT_MAXTIME que podrá ser utilizada para comprobar si ha transcurrido o no el intervalo máximo de tiempo especificado.

Si el sistema no dispone de ningún gestor de errores, la ejecución del programa se detendrá.

[\TimeFlag] **Tipo de dato: bool**

Este parámetro adquiere el valor TRUE si el tiempo máximo de espera ha transcurrido sin que la condición se haya cumplido. En el caso de que este parámetro esté incluido en la instrucción, no se considera un error que haya transcurrido el tiempo máximo de espera. Este argumento es ignorado si no se incluye en la instrucción el argumento *Maxtime*.

Ejecución del programa

Si la condición programada no se cumple con la ejecución de la instrucción *WaitUntil,* la condición será verificada de nuevo cada 100 ms.

Cuando el robot está esperando, el tiempo será supervisado, y en el caso en que exceda el valor máximo de tiempo, el programa continuará siempre y cuando esté especificado *TimeFlag*, o generará un error en el caso en que no lo esté. Si se especifica *TimeFlag*, adquirirá el valor TRUE cuando el tiempo máximo de espera haya sido excedido y tendrá el valor FALSE en caso contrario.

Limitaciones

El argumento *\InPos* no se puede utilizar conjuntamente con el SoftServo.

Sintaxis

WaitUntil
['\'InPos',']
[Cond ':='] <expresión (**IN**) de *bool*>
['\'MaxTime ':='<expresión (**IN**) de *num*>]
['\'TimeFlag':='<variable (**VAR**) de *bool*>] ';'

Manual Usuario Robot IRB120

Funciones

Instrucciones

Tipos de Datos

Guía de Usuario

Contenido

ClkRead Lectura de un reloj utilizado para el cronometraje

ClkRead sirve para la lectura de un reloj que funciona como un cronómetro.

Ejemplo

reg1:=ClkRead(reloj1);
El reloj *reloj1* será leído y el tiempo, en segundos, quedará almacenado en la variable *reg1*.

Valor de Retorno

El tiempo, en segundos, almacenada en el reloj. Resolución 0,01 segundos.

Argumento

ClkRead (Reloj)

Reloj **Tipo de dato:** *clock*

El nombre del reloj que se desea leer.

Ejecución del programa

Un reloj podrá ser leído tanto cuando está parado como cuando está en funcionamiento.

Una vez que se ha leído el reloj, se podrá volver a leer, volver a arrancar, parar o poner a cero.

Si el reloj se desborda, la ejecución del programa se detiene y se genera un mensaje de error.

Gestión de errores

Si el reloj funciona durante 4294967 segundos (49 días 17 horas 2 minutos 47 segundos) llegará a un estado de saturación y la variable del sistema ERRNO pasará a ERR_OVERFLOW.

El error podrá ser gestionado por el gestor de errores.

Sintaxis

ClkRead '('
[Reloj ':='] < variable (**VAR**) de *clock* > ')'
Una función con un valor de retorno del tipo *num*.

Offs Desplazamiento de una posición del robot

Offs sirve para añadir un offset o desplazamiento a una posición del robot.

Ejemplos

MoveL Offs(p2, 0, 0, 10), v1000, z50, herram1;
El robot se moverá a un punto que se encuentra a *10* mm de la posición *p2* (en la dirección z).

p1 := Offs (p1, 5, 10, 15);
La posición del robot *p1* será desplazada de *5* mm en la dirección x, de *10* mm en la dirección y, *15* mm en la dirección z.

Valor de Retorno

El dato de la posición desplazada.

Argumentos

Offs (Punto OffsetX OffsetY OffsetZ)

Punto **Tipo de dato:** *robtarget*

La posición que se desea desplazar.

OffsetX **Tipo de dato:** *num*

El desplazamiento en la dirección x.

OffsetY **Tipo de dato:** *num*

El desplazamiento en la dirección y.

OffsetZ **Tipo de dato:** *num*

El desplazamiento en la dirección z.

Ejemplo

```
PROC pallet (num fila, num columna, num distancia, PERS tooldata herram1,
PERS wobjdata wobj1)
VAR robtarget pospallet:=[[0,0,0],[1,0,0,0],[0,0,0,0], [9E9, 9E9, 9E9, 9E9, 9E9, 9E9]];
pospalett := Offs (pospalett, (fila-1)*distancia, (columna-1)*distancia, 0);
MoveL pospalett, v100, fine, herram1\WObj:=wobj1;
ENDPROC
```

Se ha creado una rutina para coger objetos de un palet. Cada palet ha sido definido como un objeto de trabajo (véase la Figura 1). La pieza que debe ser cogida (línea y columna) y la distancia entre los objetos se deberán proporcionar como parámetros de entrada.

Para aumentar el índice de líneas y columnas, se deberá salir de la rutina.

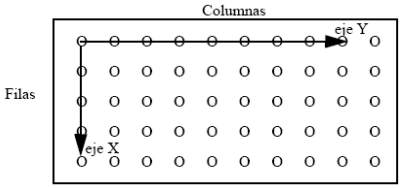


Figura 1 La posición y la orientación de un pallet se especifican definiendo un objeto de trabajo.

Sintaxis

Offs '('
[Punto ':='] <expresión (**IN**) de *robtarget*> ','
[OffsetX ':='] <expresión (**IN**) de *num*> ','
[OffsetY ':='] <expresión (**IN**) de *num*> ','
[OffsetZ ':='] <expresión (**IN**) de *num*> ')'
Una función con un valor de retorno del tipo de dato *robtarget*.

RelTool Ejecución de un desplazamiento relativo a la herramienta

RelTool (Relative Tool) sirve para añadir a una posición del robot, un desplazamiento y/o una rotación, expresados en el sistema de coordenadas de la herramienta.

Ejemplo

MoveL RelTool (p1, 0, 0, 100), v100, fine, herram1;
El robot se mueve a una posición que se encuentra a 100 mm del punto p1 en la dirección de la herramienta.

MoveL RelTool (p1, 0, 0, 0 \Rz:= 25), v100, fine, herram1;
La herramienta será girada 25o en torno al eje z.

Valor de retorno

La nueva posición con la suma de un desplazamiento y/o de una posible rotación, relativos a la herramienta utilizada.

Argumentos

RelTool (Punto Dx Dy Dz [\Rx] [\Ry] [\Rz])

Punto **Tipo de dato:** *robtarget*

Es la posición inicial 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*

Es el desplazamiento expresado en mm en la dirección x del sistema de coordenadas de la herramienta.

Dy **Tipo de dato:** *num*

Es el desplazamiento expresado en mm en la dirección y del sistema de coordenadas de la herramienta.

Dz **Tipo de dato:** *num*

Es el desplazamiento expresado en mm en la dirección z del sistema de coordenadas de la herramienta.

[\Rx] **Tipo de dato:** *num*

Es la rotación expresada en grados en torno al eje x del sistema de coordenadas de la herramienta.

[\Ry] Tipo de dato: *num*

Es la rotación expresada en grados en torno al eje y del sistema de coordenadas de la herramienta.

[\Rz] Tipo de dato: *num*

Es la rotación expresada en grados en torno al eje z del sistema de coordenadas de la herramienta.

En el caso en que se hayan especificado dos o tres rotaciones al mismo tiempo, ello se llevará a cabo en primer lugar en torno al eje x, luego en torno al nuevo eje y, para posteriormente realizarlo en torno al nuevo eje z.

Sintaxis

RelTool'('
[Punto':='] < expresión (**IN**) de *robtarget*>','
[Dx ':='] <expresión (**IN**) de *num*> ','
[Dy ':='] <expresión (**IN**) de *num*> ','
[Dz ':='] <expresión (**IN**) de *num*>
['\'Rx ':=' <expresión (**IN**) de *num*>]
['\'Ry ':=' <expresión (**IN**) de *num*>]
['\'Rz ':=' <expresión (**IN**) de *num*>]')'
Una función con un valor de retorno del tipo de dato *robtarget*.

Asea Brown Boveri, S.A.
DM / Robotics
C/ Illa de Buda, 55
08192 Sant Quirze del Vallès (Barcelona)
formacion.robot@es.abb.com
www.abb.es/robotics