Ejemplo de programación en RAPID.

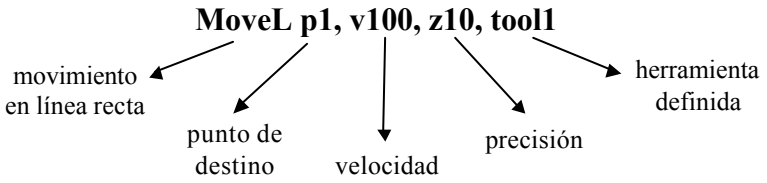
Instrucciones principales de RAPID

INSTRUCCIONES DE MOVIMIENTO

Existen 3 instrucciones de movimiento principales:

- **MoveL**: desplazamiento del extremo del robot hasta el punto indicado siguiendo una línea recta.
- **MoveC**: desplazamiento del extremo del robot hasta el punto indicado siguiendo un círculo.
- **MoveJ**: desplazamiento del extremo del robot hasta el punto indicado rápidamente, sin garantizar cuál es la trayectoria seguida (no hay coordinación de velocidad entre los distintos ejes del robot).

Veamos a continuación un ejemplo de una de estas instrucciones con sus parámetros:



Podemos ver cómo se pueden especificar tanto el punto al que queremos hacer llegar el extremo del robot como la **velocidad** del movimiento, la **precisión** y la **herramienta** que está utilizando el robot. Más adelante se verán estos parámetros en detalle.

Para comparar el funcionamiento de los tres tipos de movimiento descritos anteriormente, se ejecutarán en el robot las tres instrucciones siguientes, en cada uno de los casos partiendo de la posición de reposo del robot:

- **MoveL p1, v100, z10, tool1**
- **MoveC p0, p1, v100, z10, tool1**
- **MoveJ p1, v100, z10, tool1**

Nota: en la instrucción **MoveC** será necesario especificar, además del punto de destino **p1**, un punto de paso intermedio **p0** que servirá para definir la curvatura.

En cualquiera de los tres casos, la orientación del extremo del robot cambiará progresivamente desde la orientación inicial a la orientación especificada en la posición destino. En el caso de que ambas fueran iguales, la orientación no variaría durante todo el desplazamiento.

A continuación se explican algo más en detalle los parámetros de la instrucción:

Posición de destino 'p1' y posición intermedia 'p0':

Se trata de datos del tipo '**RobTarget**', que se explicará más adelante. En resumen, el dato **p1** o **p0** especifica tanto posición como orientación deseada para el robot de una forma unívoca.

Velocidad de movimiento 'v100':

Es un dato del tipo '**SpeedData'** que indica la velocidad deseada tanto de traslación como de rotación para el extremo del robot.

Precisión 'z10':

Es un dato del tipo '**ZoneData'** que especifica con qué precisión se debe alcanzar la posición solicitada antes de permitir que el robot se mueva hasta el punto siguiente.

Herramienta 'tool1':

Es un dato del tipo '**ToolData'** que describe las características de la herramienta acoplada al robot. Las dimensiones de la herramienta se utilizan para calcular las trayectorias respecto al punto central de la misma en lugar de calcularlas respecto al extremo del robot. Los datos relativos al peso y centro de gravedad de la herramienta se utilizan para adecuar los pares a ejercer por los motores del robot.

INSTRUCCIONES PARA LA UTILIZACIÓN DE ENTRADAS Y SALIDAS

Básicamente interesarán dos operaciones: comprobar el valor de una determinada **entrada** (por ejemplo, la información de un sensor de presencia para saber si hay una pieza lista para ser manipulada) o bien fijar el valor de una determinada **salida** (por ejemplo para activar una cinta transportadora una vez se ha depositado sobre ella la pieza o para activar la electroválvula de una pinza neumática cuando se desea agarrar un objeto).

Las instrucciones fundamentales para fijar el valor de las salidas son las siguientes:

- **Set**: fija el valor de una salida digital a 1.
- **Reset**: fija el valor de una salida digital a 0.
- **SetDO**: fija una salida digital a un valor simbólico (activado o desactivado).
- **SetAO**: fija el valor de una salida analógica.

Algunos ejemplos de utilización de estas instrucciones se ven a continuación:

Set out1

Fija el valor de la señal digital **out1** a uno.

Reset out3

Fija el valor de la señal digital **out3** a cero.

SetDO pinza, off

Fija el valor de la señal digital **pinza** a **off** (abre la pinza del robot).

SetAO out5, 4.3

Fija la señal analógica **out5** al valor **4.3**. No se trata de un valor físico (4.3V o 4.3mA) sino de un valor lógico cuya equivalencia física se debe definir en otro punto del programa.

Las instrucción principal que permite comprobar el valor de una entrada es la instrucción **WaitDI**, que hace que el robot espere hasta que la señal alcanza el valor deseado. Se ven un par de ejemplos:

WaitDI in3, 1

El robot esperará hasta que la señal digital **in3** tome el valor uno.

WaitDI pieza, 1

El robot esperará hasta que la señal digital **pieza** tome el valor uno.

También es posible comprobar el valor de una determinada señal de entrada con una instrucción de comparación normal, como muestran los siguientes ejemplos:

IF pieza = 1 THEN ...

Ejecuta las instrucciones siguientes si el valor de la señal digital pieza es igual a uno. La instrucción **IF THEN** de control de flujo se explicará mas adelante.

If par > 5.1 THEN ...

Ejecuta las instrucciones siguientes si el valor de la señal analógica **par** es mayor que **5.1** (se puede utilizar para realizar un control de esfuerzos, por ejemplo).

INSTRUCCIONES DE CONTROL DE FLUJO DE EJECUCIÓN

Son instrucciones muy similares a las utilizadas en cualquier lenguaje de programación; se enumeran a continuación:

- **IF THEN**: ejecuta una serie de instrucciones si se cumple una determinada condición.
- **FOR**: repite una sección del programa un determinado número de veces.
- **WHILE**: repite una sección del programa mientras se cumpla una condición dada.
- **TEST/CASE**: ejecuta diferentes instrucciones en función del valor de un dato (similar al switch/case del lenguaje C).
- **GOTO**: salto incondicional a un punto del programa.

VARIABLES Y EXPRESIONES

El lenguaje RAPID permite definir variables o datos de distintos tipos, como se verá más adelante. Con estas variables es posible crear expresiones aritméticas o lógicas mediante una serie de operadores bastante comunes (suma, producto, comparación, etc.). Detalles sobre el formato de estas expresiones y los tipos de datos sobre los que son aplicables se pueden encontrar en cualquier referencia del lenguaje de programación RAPID.

Principales tipos de datos en RAPID

A continuación se enumeran los principales tipos de datos que se pueden utilizar en RAPID y su utilización más común. El interés de esta lista es comprobar qué tipo de parámetros se manejan en la distintas instrucciones del robot.

ConfData: define las configuraciones del robot. Se utiliza para resolver las ambigüedades que se producen cuando una determinada posición y orientación es alcanzable con más de una combinación de valores para las articulaciones. Se especifican los cuadrantes en los que se encuentran los ejes del robot.

JointTarget: especifica una posición deseada para los ejes (articulaciones) del robot. Se utiliza cuando se quiere gobernar el robot directamente a través de sus articulaciones y no especificando posiciones y orientaciones.

LoadData: define las características de la carga que deberá manipular el robot (del objeto a agarrar con la pinza). Se utiliza para ajustar los pares a ejercer por los motores del robot.

MotSetData: define las características básicas de todos los movimientos a efectuar por el robot, siempre que no se especifique lo contrario. Estas características son: velocidad, aceleración, precisión, comportamiento en puntos singulares, etc.

Num: dato numérico. Se puede utilizar en cualquier expresión. Existe también el tipo **bool** para expresiones lógicas y el tipo **string** para expresiones de cadenas de texto.

Orient: permite especificar una orientación o una rotación, expresadas ambas como cuaternios.

Pos: permite expresar una posición mediante sus coordenadas cartesianas X, Y, Z.

Robjoint: expresa la posición en grados de cada uno de los ejes (articulaciones del robot).

RobTarget: define la posición y orientación deseada para el extremo del robot, incluyendo los datos de configuración necesarios para que no existan ambigüedades. También especifica la posición deseada para los ejes externos, si éstos existieran.

SpeedData: define la velocidad a la que debe efectuarse un movimiento. Esta velocidad se define por separado para la traslación, la rotación y el movimiento de los ejes externos.

StopPointData: especifica la precisión a alcanzar en el punto final de una trayectoria. Se establece también un tiempo límite para alcanzar el destino con la precisión pedida.

ToolData: describe las características de una determinada herramienta: pinza de agarre, boquilla de soldadura, etc. Las dimensiones de la herramienta se utilizan para calcular las trayectorias respecto al punto central de la misma en lugar de calcularlas respecto al extremo del robot. Los datos relativos al peso y centro de gravedad de la herramienta se utilizan para adecuar los pares a ejercer por los motores del robot.

ZoneData: especifica la precisión a alcanzar en un punto que no necesariamente debe ser el final de una trayectoria (es válido en cualquier caso).

Modo de introducir los programas al IRB140

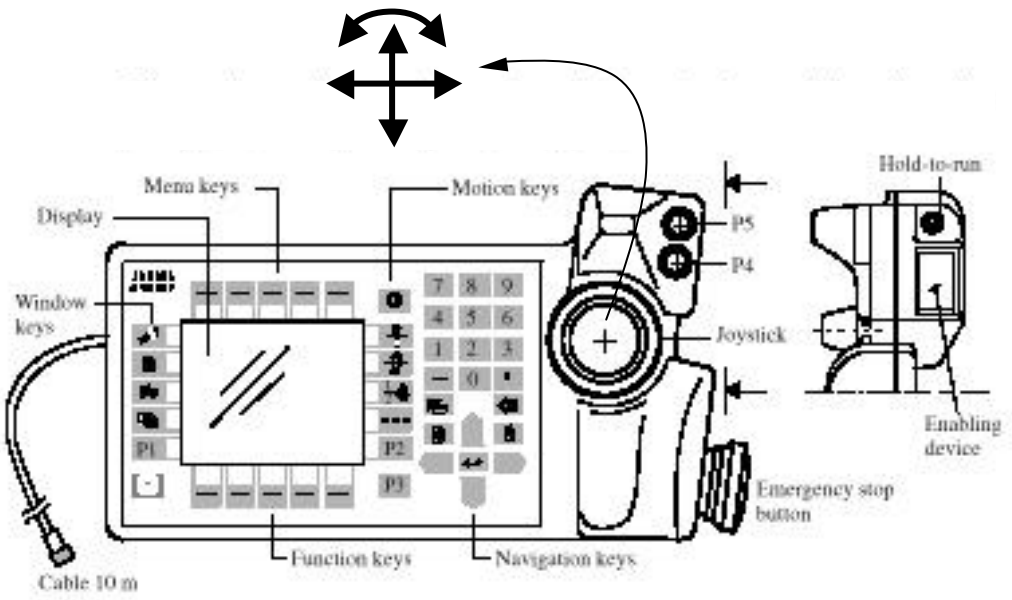
Existen dos formas de introducir programas al IRB140:

- generando el programa directamente a través de la paleta de programación
- escribiendo el programa en un PC y transfiriéndolo vía disquete o por cualquier otro medio.

Veremos a continuación más en detalle cada uno de estos dos métodos:

DESDE LA PALETA DE PROGRAMACIÓN

Antes de nada, veamos cuáles son los botones y mandos presentes en la paleta de programación.



El mando principal es un **joystick** con tres grados de libertad, tal y como se ve en la figura:

- Desplazamiento vertical.
- Desplazamiento horizontal.
- Giro.

Mediante estos tres grados de libertad será posible mover el robot de distintas formas:

- Asignar cada grado de libertad a los ejes cartesianos **X, Y, Z**y trasladar el extremo del robot a la posición que se desee.
- Asignar cada grado de libertad a los ángulos girados por la muñeca con respecto a los tres ejes coordenados y llevar de este modo la muñeca hasta la orientación deseada.
- Asignar cada grado de libertad del joystick a una articulación del robot y de este modo mover el robot articulación a articulación (sólo será posible mover 3 articulaciones simultáneamente).

El modo de movimiento elegido se selecciona mediante los botones de la paleta de programación.

Otro mando fundamental es el botón de **stop de emergencia**. Es un botón con enclavamiento que detiene inmediatamente el robot; se debe pulsar en caso de detectar un fallo en el comportamiento. Este botón es particularmente útil en las fases de desarrollo y prueba de programas.

El **botón de habilitación** (enabling device en la figura) es también común en las paletas de programación de cualquier robot. Es un dispositivo de seguridad con tres posiciones: el robot puede moverse mientras el botón se encuentre ligeramente pulsado; si el botón se suelta o se pulsa completamente el robot se detiene. De este modo se comprueba que el operario que maneja el robot no sufre una pérdida de atención en ningún momento.

El resto de los elementos de la paleta son el display donde se muestran todas las informaciones pertinentes y una botonera que permite realizar todas las operaciones de manejo del robot. Ambos elementos serán explicados según vaya siendo necesario su uso; a continuación se muestra un listado de los botones existentes.

Movimiento: Sirve para mover el robot.

Programa: Sirve para programar y para realizar las pruebas de los programas.

Entradas/Salidas: Sirve para operar de forma manual las señales de entrada y salida conectadas al robot.

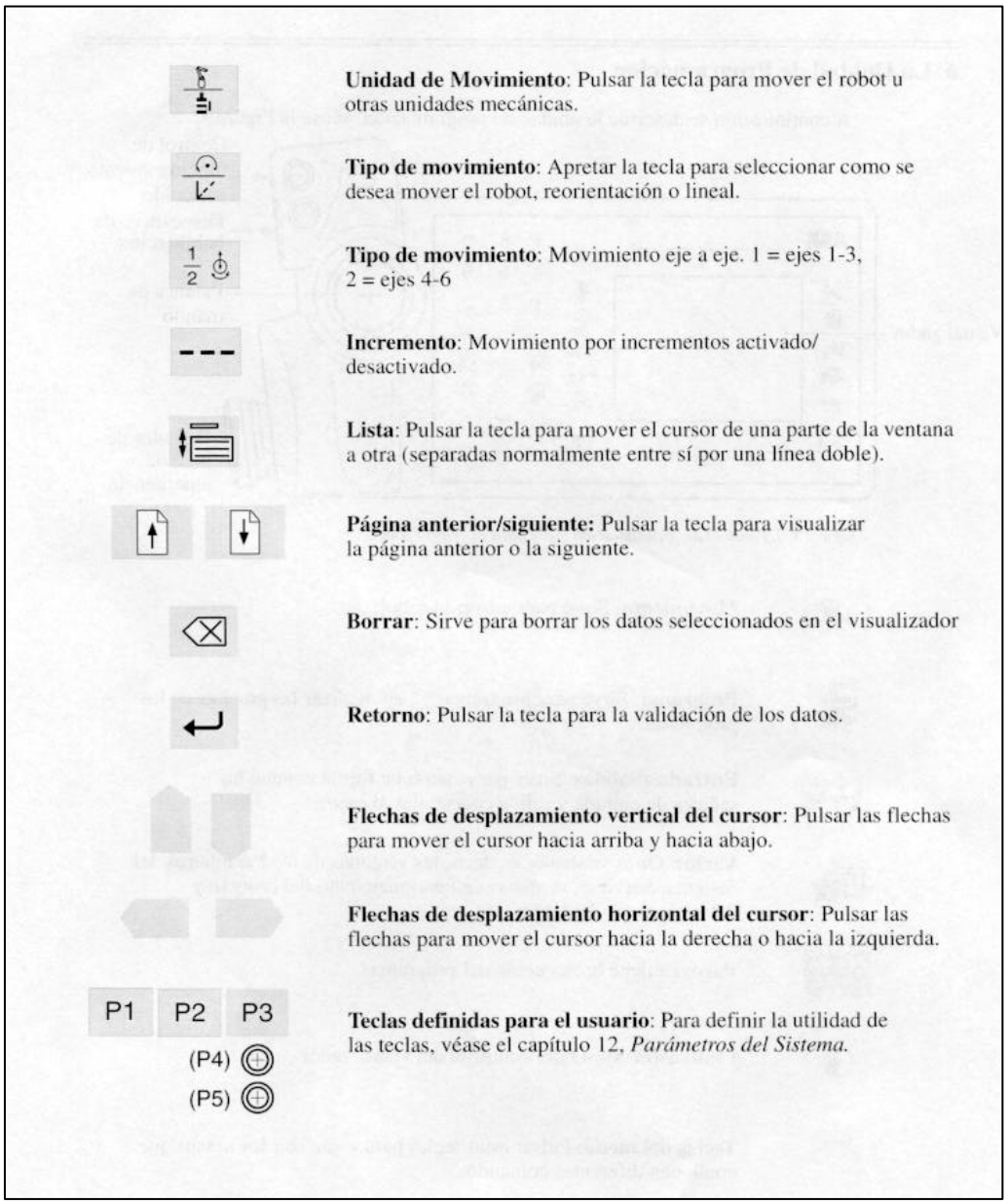
Varios: Otras ventanas, es decir, las ventanas de los Parámetros del Sistema, Servicio, ventanas de funcionamiento del proceso y Administrador de Archivos.

Paro: Detiene la ejecución del programa.

Contraste: Ajuste del contraste del visualizador.

Teclas del menú: Pulsar estas teclas para visualizar los menús que contienen diferentes comandos.

Teclas de función: Pulsar estas teclas para seleccionar los diferentes comandos directamente.



Una vez vista la paleta de programación, estudiaremos cuáles son los pasos a dar para la creación de un programa.

Primer paso: dar un nombre al programa

Se pulsará el botón **'programa'** de la botonera y se introducirá el nombre que se desea dar al programa.

Segundo paso: definición de herramientas y objeto de trabajo

Sólo será necesario si no se utilizan herramientas u objetos de trabajo previamente definidos en el sistema. Se verá más adelante cómo crear objetos del tipo **tooldata** (herramienta) o **loaddata** (objeto de trabajo o carga); por ahora consideraremos que se utiliza la herramienta definida por defecto en el sistema.

Tercer paso: Escritura de las instrucciones del programa

Los programas de gran extensión se estructuran en rutinas y módulos, de modo que sean más fáciles de leer. En nuestro caso, dado que los programas serán sencillos, obviaremos el paso de la creación de las rutinas y escribiremos las instrucciones directamente.

Las instrucciones se eligen de entre varias listas; normalmente bastará con utilizar la lista de instrucciones más comunes; pero si es necesario se puede abrir cualquier otra. Nosotros nos centraremos en las instrucciones de movimiento y en las instrucciones de control de entradas y salidas.

Las instrucciones de movimiento más comunes son **MoveL**, **MoveC** y **MoveJ**, tal y como se vieron anteriormente. Para introducir una cualquiera de ellas, caben dos posibilidades:

Posibilidad 1:

- Desplazar el robot hasta la posición y orientación deseadas como punto final de la instrucción de movimiento.
- Elegir la instrucción adecuada de la lista (por ejemplo, **MoveL**).
- Adecuar los parámetros:
  - **Posición destino**: se introducirá un asterisco **(*)** que indica que la posición destino es la posición en la que se encuentra el robot actualmente.
  - **Velocidad**: se elegirá alguna de las velocidades predefinidas: **v10**, **v50**, **v1000**, etc. donde la cifra indica la velocidad de traslación en mm/s. Para operaciones especiales será preciso definir una velocidad específica.
  - **Precisión**: se elegirá una de las precisiones predefinidas: **z10**, **z30**, **z150**, etc. donde la cifra indica el valor en mm. de la precisión (el valor **'fine'** supone una precisión de **0 mm**).
  - **Herramienta:** se elegirá en este caso la herramienta predefinida **tool0**. En caso de querer utilizar otra herramienta, habría que haberla definido previamente.

Posibilidad 2:

- Desplazar el robot hasta la posición y orientación deseadas como punto final de la instrucción de movimiento.
- Crear una variable de tipo **robtarget** con esa posición. Para ello se debe acceder a la ventana de datos desde el menú **'Ver'**, opción **'Datos'**. Deberemos asignar un nombre a la variable, por ejemplo, '**pos_fin'**.
- Elegir la instrucción adecuada de la lista (por ejemplo, **MoveL**).
- Adecuar los parámetros:
  - **Posición destino:** se introducirá el nombre de la variable recién creada, en nuestro caso '**pos_fin'.**
  - **Velocidad, Precisión** y **Herramienta**: se procederá como en el caso anterior.

Mediante cualquiera de las dos posibilidades, habrá quedado definida una instrucción de movimiento para el robot.

Para las instrucciones de control de las entradas y salidas emplearemos un procedimiento más sencillo. Bastará elegir la instrucción adecuada de entre la lista: **Set**, **Reset**, **SetDO**, **WaitDI**... todas ellas vistas con anterioridad. Los parámetros a fijar en este tipo de instrucciones se limitan al nombre de la señal a utilizar y al valor que se desea para ella o que se quiere comprobar en ella.

Con estos tipos de instrucciones vistos, ya es posible crear un programa para el robot que:

- Espere hasta tener una pieza lista para ser procesada.
- Se desplace hasta la posición de la pieza siguiendo la trayectoria deseada (instrucciones de movimiento).
- Agarre la pieza (activando una determinada señal digital).
- Desplace la pieza hasta el lugar deseado (instrucciones de movimiento).
- Suelte la pieza (desactivando la señal digital anterior).
- Ponga en marcha la cinta transportadora que debe llevarse la pieza una vez depositada por el robot (activando otra señal digital).

ESCRITURA DEL PROGRAMA EN UN PC

Una vez vista la forma de introducir los programas desde la paleta de programación, se estudiará la forma de escribir los programas en un PC y transferirlos posteriormente al robot vía disquete.
Un programa escrito en el PC debe tener un formato inteligible para el robot. Para ello se deberán seguir unas ciertas reglas de sintaxis en la escritura de las instrucciones y del conjunto del programa. Veremos a continuación como crear un programa para un ejemplo de movimiento sencillo como el descrito anteriormente.

Las líneas iniciales del programa deben ser, de acuerdo con la versión del software disponible, tal y como las que se muestran a continuación:

```
%%%
  VERSION:1
  LANGUAGE:ENGLISH
%%%
```

Tras estas líneas iniciales, se introducirá el código del programa; todo el englobado dentro de un determinado **módulo**; por lo tanto el código se englobará entre las dos líneas siguientes:

```
MODULE prueba
     ...
     código del programa
     ...
ENDMODULE
```

Se ha asumido que, en este ejemplo, el nombre dado al módulo principal ha sido '**prueba'**. Las primeras instrucciones que aparecerán en el programa serán definiciones de variables. Entre estas definiciones estarán las que hacen referencia a las herramientas a utilizar, las que se pretendan utilizar para realizar operaciones matemáticas o lógicas,

etc. Supongamos que en nuestro caso sólo pretendemos definir un dato, que especificará la herramienta con la que pensamos trabajar. En este caso la instrucción a introducir tendría un aspecto como el siguiente:

```
PERS tooldata pinza :=
[TRUE,[[0,0,0],[1,0,0,0]],[5,[9,0,9],[1,0,0,0],0.01,0.
01,0.01]];
```

Analicemos detenidamente la expresión anterior:

- La palabra clave **PERS** especifica el alcance de la variable o dato que deseamos crear. Existen tres posibilidades:
  - **LOCAL:** variable sólo accesible dentro del módulo donde se ha definido.
  - **GLOBAL:** variable accesible en todos los módulos del programa.
  - **PERS:** variable persistente: accesible en todos los módulos del programa y con valor no reinicializable al comenzar de nuevo los programas.

  Para los programas sencillos que crearemos no necesitaremos considerar el alcance de las variables; declararemos en general variables locales.
- **tooldata** especifica el tipo de dato que queremos crear, en este caso se trata de una definición de herramienta.
- **pinza** es el nombre que se desea dar a la variable y se elige arbitrariamente.
- **:=** es el operador de asignación en RAPID, permite dar valores a las variables.
- El resto de datos son los valores para cada uno de los campos del tipo de datos **tooldata**:
  - **TRUE** corresponde al campo **'robhold'** e indica si la herramienta está sujeta al robot o no.
  - El primer vector, **[0,0,0]** corresponde a la posición del punto origen de la herramienta con relación al extremo del robot, expresado en mm.
  - El segundo vector, **[1,0,0,0]** corresponde a la orientación de la herramienta con relación también al sistema de coordenadas situado en el extremo del robot. La orientación se expresa mediante un cuaternio.
  - El resto de datos tienen que ver con la masa y los momentos de inercia de la herramienta:
    - El primer valor, **5**, es el peso de la herramienta en kg.
    - El primer vector, **[9,0,9]**, indica la posición del centro de gravedad de la herramienta en mm.
    - El segundo vector, **[1,0,0,0]**, indica la orientación del sistema de coordenadas sobre el que se medirán los momentos de inercia de la herramienta. Debe tener SIEMPRE el valor **[1,0,0,0]** para indicar que este sistema de coordenadas coincide con el del extremo del robot.
    - Los tres restantes valores, **0.01,0.01,0.01** representan los momentos de inercia mencionados anteriormente y expresados en $kg{\cdot}m^2$.

Tras la declaración de variables, se encontrarían las rutinas y funciones constitutivas del módulo. Dado que el ejemplo a introducir es muy sencillo, esta estructuración no es necesaria, y todo el código se incluirá dentro de una única rutina, que necesariamente ha de llamarse **main** (rutina principal):

```
PROC main()
     ...
     instrucciones
     ...
ENDPROC
```

Las instrucciones a introducir serán básicamente movimientos y control de entradas y salidas digitales. Un ejemplo sería el siguiente:

```
! movimiento rapido hasta un punto proximo a la pieza
MOVEL pos1, v1000, z50, pinza;
! descenso hacia la pieza con velocidad lenta y alta
precision
MOVEL pos2, v50, z0.3, pinza;

! activacion de la pinza
SET do15;
! espera de 0.5 segundos para asegurar accion pinza
WAITTIME 0.5;

! movimiento rapido hasta un punto proximo al destino
MOVEL pos3, v1000, z50, pinza;
! descenso hacia el destino con velocidad lenta y alta
precision
MOVEL pos4, v50, z0.3, pinza;
```

Hay que resaltar algunos aspectos del ejemplo:

- Todas las líneas que comienzan con el carácter admiración **(!)** no son interpretadas por el robot, sencillamente sirven como comentario para facilitar la comprensión del programa.
- La instrucción **WaitTime** realiza una espera del número de segundos indicado antes de ejecutar la siguiente instrucción.
- Las variables **pos1**, **pos2**, **pos3** y **pos4**, deberían estar definidas e inicializadas a sus valores correspondientes. Son variables del tipo **robtarget**, por lo que deberían tener especificados los siguientes campos:
  - **Trans**: posición en coordenadas cartesianas **[x, y, z].**
  - **Rot**: orientación expresada como un cuaternio: **[q1, q2, q3, q4].**
  - **Robconf**: configuración de los ejes deseada (pueden existir ambigüedades si sólo se indican posición y orientación). Esta configuración se expresa mediante los cuadrantes de los ejes 1, 4 y 6 más un cuarto valor que no se utiliza en este modelo de robot.
  - **Extax**: indica la posición de los ejes externos **1** a **6** en caso de existir; si los ejes no están conectados (como es el caso) se utilizará el valor **9E9**.

Un ejemplo de definición de una de estas posiciones sería el siguiente:

```
CONST robtarget pos2 :=
[33.1,-61.8,64.9], [0.34,-0.42,0.42,-0.72], [-1,-1,-
1,1], [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09];
```

El programa completo quedaría al y como se muestra a continuación:

```
%%%
  VERSION:1
  LANGUAGE:ENGLISH
%%%

MODULE prueba

      PERS tooldata pinza :=
      [TRUE,[[0,0,0],[1,0,0,0]],[5,[9,0,9],[1,0,0,0],0.01,0.01,0.01]];

      CONST robtarget pos1 :=
      [[33.1,-61.8,54.9], [0.34,-0.42,0.42,-0.72], [-1,-1,-1,1],
      [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
      CONST robtarget pos2 :=
      [[33.1,-61.8,64.9], [0.34,-0.42,0.42,-0.72], [-1,-1,-1,1],
      [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
      CONST robtarget pos3 :=
      [[-27.2,53.4,-15.1], [0.27,0.32,-0.22,0.56], [-1,-1,-1,1],
      [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
      CONST robtarget pos4 :=
      [[-27.2,53.4,-25.1], [0.27,0.32,-0.22,0.56], [-1,-1,-1,1],
      [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

      PROC main()

            ! movimiento rapido hasta un punto proximo a la pieza
            MOVEL pos1, v1000, z50, pinza;
            ! descenso hacia la pieza con velocidad lenta y alta
            precision
            MOVEL pos2, v50, z10, pinza;

            ! activacion de la pinza
            SET do15;
            ! espera de 0.5 segundos para asegurar accion pinza
            WAITTIME 0.5;

            ! movimiento rapido hasta un punto proximo al destino
            MOVEL pos3, v1000, z50, pinza;
            ! descenso hacia el destino con velocidad lenta y alta
            precision
            MOVEL pos4, v50, z10, pinza;

      ENDPROC


ENDMODULE
```

4.5. Modo de ejecutar los programas en el IRB140

Deben considerarse dos posibilidades: la ejecución de programas en pruebas, durante la creación y depuración de los mismos; y la ejecución de programas en producción, una vez están perfectamente probados.

Para la ejecución en pruebas, el equipo ofrece muchas facilidades que permiten realizar las pruebas sin riesgos y detectar posibles errores en el programa. Las ayudas que ofrece el sistema son:

- Posibilidad de reducir la velocidad de todos los movimientos para la realización de pruebas: si el robot se mueve a velocidad reducida será más sencillo detenerlo ante un posible fallo.
- Posibilidad de ejecutar el programa paso a paso: se ven los resultados de la ejecución de cada instrucción antes de pasar a la siguiente.
- Posibilidad de ejecutar un solo ciclo de programa: en general, el robot ejecuta una secuencia de movimientos y acciones de modo repetitivo; para las pruebas es más razonable realizar un solo ciclo de trabajo.
- Posibilidad de ejecutar instrucciones hacia atrás: permite, en la mayor parte de los casos, volver al estado anterior a la ejecución de una determinada instrucción si se detecta que ha habido un error.
- Posibilidad de comenzar la ejecución en una instrucción determinada en lugar de en el principio del programa: esto permite probar rápidamente fragmentos del programa.

Todas estas opciones están accesibles desde la ventana '**Test del programa'**, a la que se llega con la opción '**Test'** del menú '**Ver'**.

Para el funcionamiento en producción, una vez perfectamente depurado el programa, se accederá a la ventana '**Producción'**, pulsando el botón **'Varios'** y eligiendo la opción '**Producción'**.