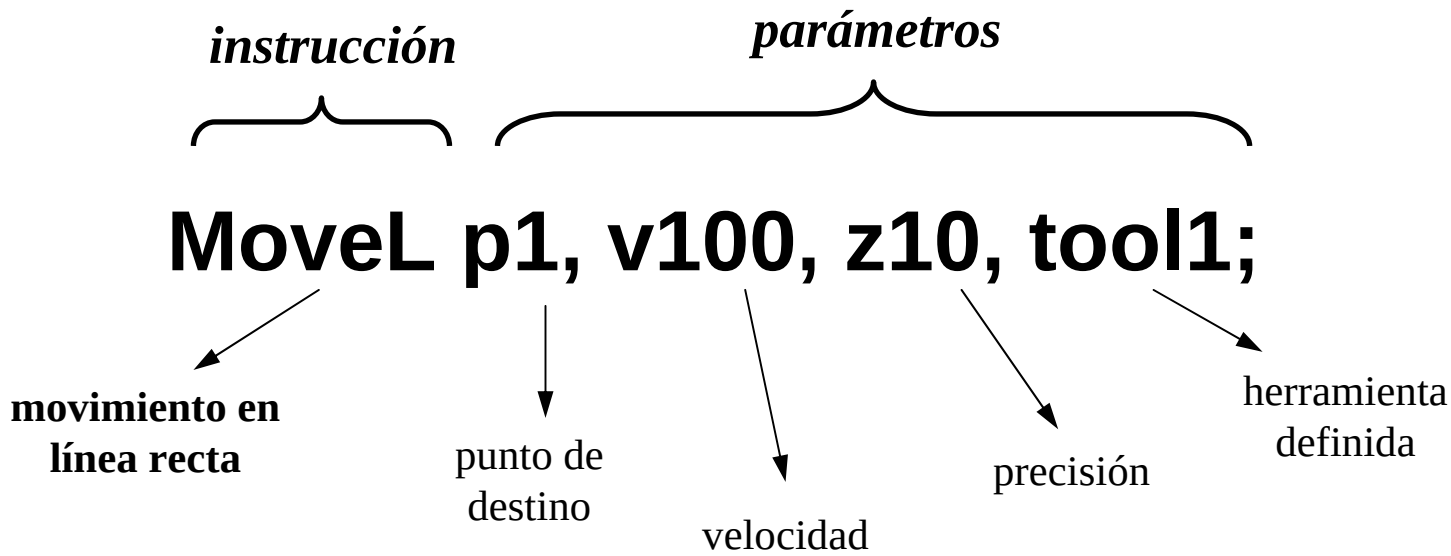
INTRODUCCIÓN A RAPID

Programación con el IRB-140

Características principales de RAPID

- Última generación (creado en 1994 por ABB).
- Muy similar a lenguajes de programación de propósito general de alto nivel (Basic, Pascal, C):
  - Estructurado a dos niveles: **módulos** por una parte y **rutinas y funciones** por otra.
  - Parámetros de las instrucciones: numéricos, referencias o expresiones.
  - Para cada dato se define su ámbito de utilización o **alcance**: local o global y su **persistencia**: variable, constante o persistente.

Ejemplo de instrucción RAPID



Instrucciones más utilizadas (I)

INSTRUCCIONES DE MOVIMIENTO

- **MoveL**: desplazamiento hasta el punto en línea recta.
- **MoveC**: desplazamiento describiendo un círculo.
- **MoveJ**: desplazamiento rápido, sin garantizar la trayectoria seguida (sin coordinación de velocidad entre los ejes del robot).

Ejemplos:

- **MoveL p1, v100, z10, tool1;**
- **MoveC p0, p1, v100, z10, tool1;** ***(ojo: dos puntos)***
- **MoveJ p1, v100, z10, tool1;**

Instrucciones más utilizadas(II)

Parámetros de las instrucciones de movimiento:

- **Posiciones deseadas ‘p1’ y ‘p0’:**

Dato tipo ‘**RobTarget’**:especifica posición y orientación unívocamente.

- **Velocidad ‘v100’:**

Dato tipo ‘**SpeedData’:** indica velocidad deseada de traslación y de rotación.

- **Precisión ‘z10’:**

Dato tipo ‘**ZoneData’**: indica la precisión con la que hay que alcanzar la posición pedida antes de continuar el movimiento hasta la posición siguiente.

- **Herramienta ‘tool1’:**

Dato tipo ‘**ToolData’**: describe la la herramienta acoplada al robot: dimensiones para calcular trayectorias más peso y momentos de inercia para calcular esfuerzos.

Instrucciones más utilizadas(III)

INSTRUCCIONES DE MANEJO DE E/S

- **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 (on/off, etc).
- **SetAO**: fija el valor de una salida analógica.
- **WaitDI**: espera hasta que una entrada tome un valor determinado

Posibles utilizaciones de las señales de E/S:

- comandar la válvula neumática que gobierna la pinza
- detener o poner en marcha una cinta transportadora
- saber si hay una pieza lista para ser procesada por el robot...

Instrucciones más utilizadas(IV)

Ejemplos:

- **Set out1** ***(fija el valor de la salida out1 a uno)***
- **Reset out3** ***(fija el valor de la salida out3 a cero)***
- **SetDO pinza, off** ***(fija el valor de la salida pinza a off)***
- **SetAO out5, 4.3** ***(fija la salida analógica out5 a 4.3) (ojo: ni V ni A)***
- **WaitDi pieza, 1** ***(espera hasta que la entrada pieza tome el valor 1)***

También es posible utilizar el valor de las entradas dentro de una expresión, como si se fueran variables:

- **IF pieza=1 THEN...** ***(una entrada digital controla el flujo de programa)***
- **IF par>1.5 THEN...** ***(idem con entrada analógica) (control esfuerzos)***

Instrucciones más utilizadas(V)

INSTRUCCIONES DE CONTROL DE FLUJO DE EJECUCIÓN

Las habituales en lenguajes de programación de alto nivel:

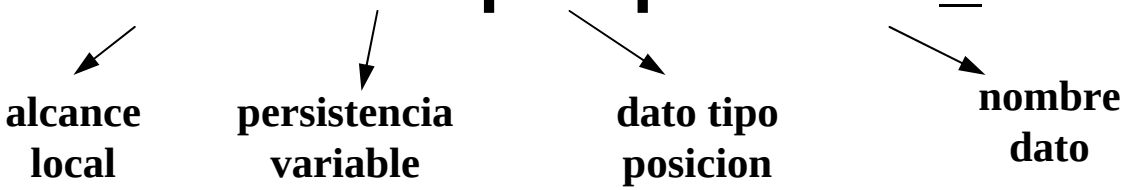
- **FOR ... ENDFOR**: repite una sección del programa un número fijo de veces.
- **WHILE ... ENDWHILE**: repite una sección del programa mientras se cumpla una cierta condición.
- **GOTO**: salto incondicional a una determinada instrucción.
- **IF ... THEN ... ELSE ... ENDIF**: ejecuta un conjunto de instrucciones sólo si se cumple una cierta condición.
- **TEST/CASE**: ejecuta unas u otras instrucciones en función del valor de un dato (similar al switch/case del lenguaje C).

Variables y expresiones en RAPID (I)

La variables y constantes en RAPID se denominan **datos.**

Definición de un dato: **alcance**, su **persistencia** y su **tipo:**

LOCAL VAR pos posicion_robot;



Asignación de valores a los datos: **operador :=**

posicion_robot := [0.0, 0.0, 0.0];

Variables y expresiones en RAPID (II)

Posibles ámbitos para un dato:

•**LOCAL**: sólo accesible desde la rutina o módulo donde se declara

•**GLOBAL**: accesible desde otras rutinas o módulos.

Posibles persistencias para un dato:

•**VAR:** dato variable. Su valor puede ser modificado en el programa.

•**CONST:** dato constante. No permite ser modificado.

•**PERS:** dato persistente. Si se modifica en el programa permanece modificado para nuevas ejecuciones del mismo.

Variables y expresiones en RAPID (III)

TIPOS DE DATOS BÁSICOS

Num: dato numérico (entero o real).

Bool: valor lógico (TRUE o FALSE).

String: cadena de caracteres.

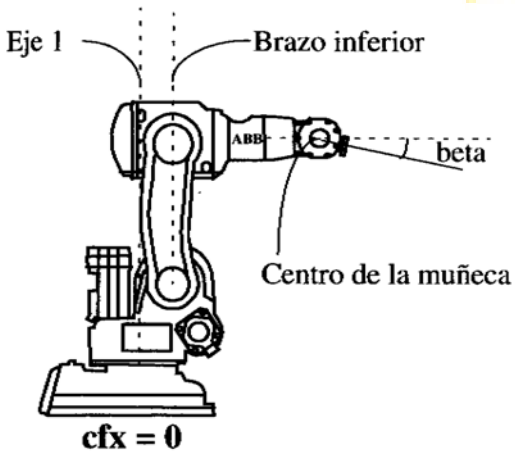
TIPOS DE DATOS PROPIOS DEL MANEJO DEL ROBOT

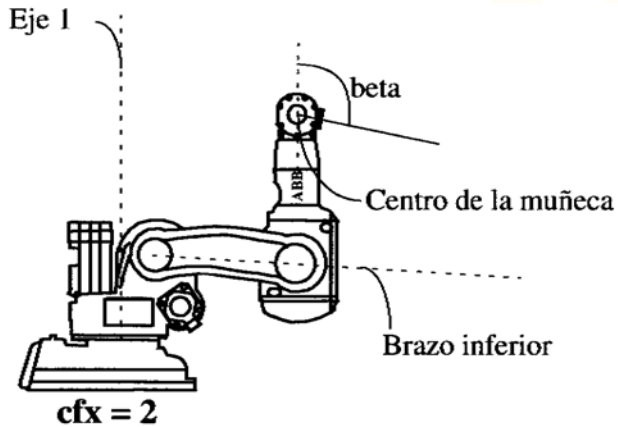
Pos: posición cartesiana (vector de 3 valores numéricos X, Y, Z)

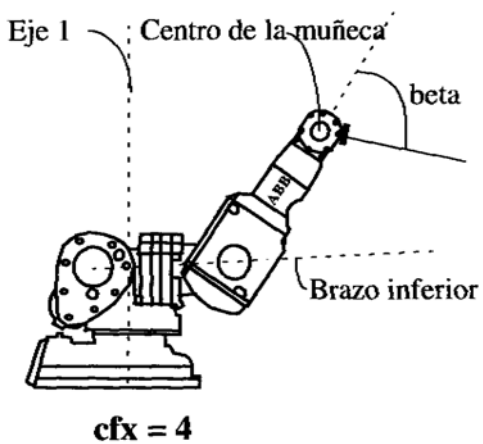
Orient: orientación expresada mediante un cuaternio (vector de 4 valores numéricos (Q1, Q2, Q3, Q4)

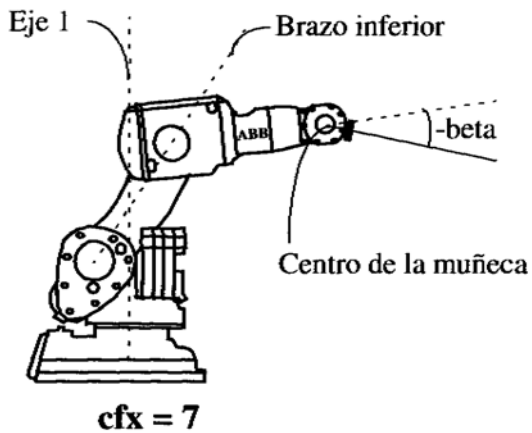
ConfData: configuración del robot. Indica los cuadrantes en los que se encuentran los ejes 1, 4, 6 y 5 del robot. Vector de 4 valores numéricos (conf1, conf2, conf3, conf4).

Configuración: resuelve ambigüedades



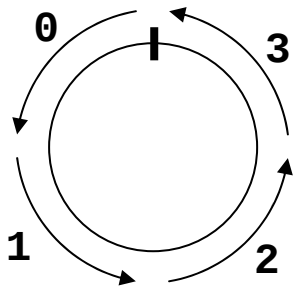






Cómo especificar la configuración

Para cada uno de los ejes 1, 4 y 6 se indica el cuadrante en el que se debe encontrar situado (ángulo girado desde 0º):




```
CONST confdata conf_1 := [0, 0, -1, 1];
```

La instrucción anterior indica una configuración 0 para el eje 1, 0 para el eje 4 y -1 para el eje 6. El último dato no se utiliza en el IRB-140.

Variables y expresiones en RAPID (IV)

TIPOS DE DATOS PROPIOS DEL MANEJO DEL ROBOT (cont)

ExtJoint: posición de los ejes externos. Vector de 6 valores numéricos (eje1 a eje6). El IRB-400 es capaz de controlar hasta 6 ejes externos.

RobTarget: indica un objetivo a alcanzar por el robot. Esto es:

- una posición (expresada como un dato **pos**: X, Y, Z).
- una orientación (dato **orient**: Q1, Q2, Q3, Q4).
- una configuración (dato **confdata**: conf1, conf2, conf3, conf4).
- una posición de los ejes externos (dato **extjoint**: eje1 a eje6).

Ejes externos: diferentes aplicaciones

- Robot desplazable por un carril.
- Pinza controlada en posición.
- Herramienta especial (ejemplo: movimiento de atornillado, con traslación y rotación sincronizadas).

No confundir con las señales de salida: en este caso hay control en bucle cerrado.

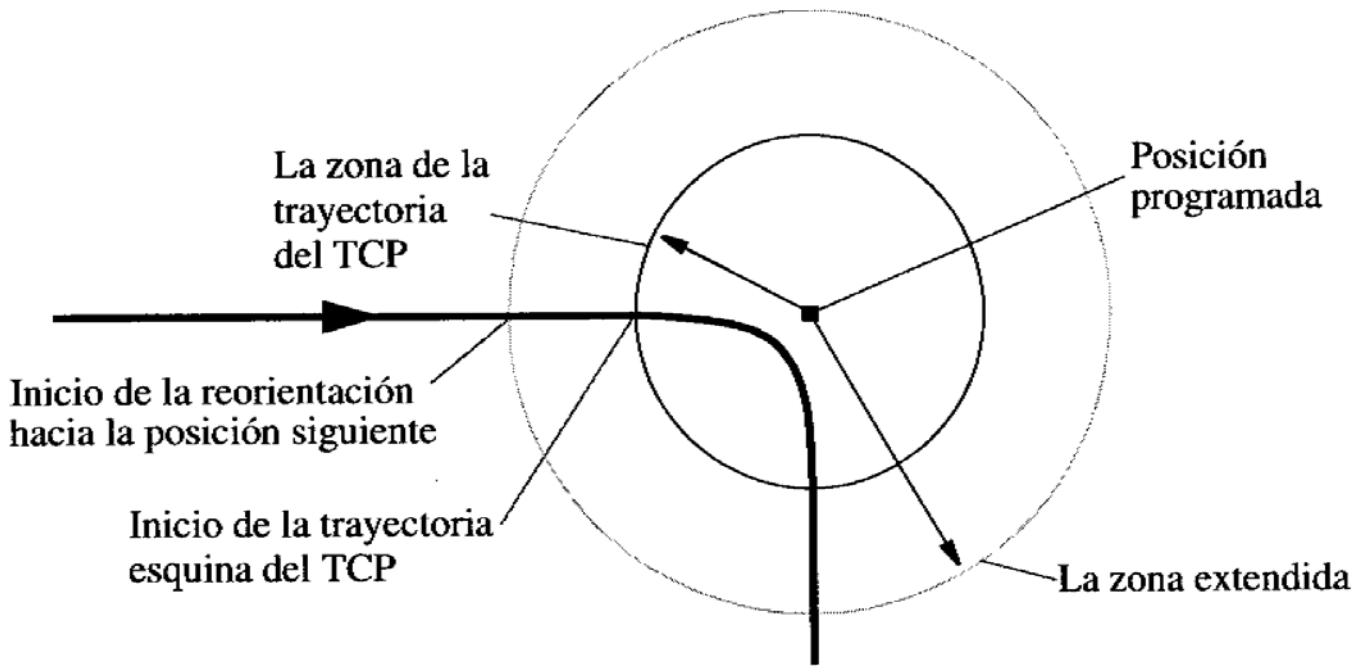
Variables y expresiones en RAPID (V)

TIPOS DE DATOS PROPIOS DEL MANEJO DEL ROBOT (cont)

SpeedData: velocidad para los movimientos. Existen valores predefinidos: **v100**, **v200**, **v1000, ...**. El valor indica velocidad de traslación en mm/s; también se especifica la vel. de rotación.

ZoneData: precisión con la que se debe alcanzar una posición antes de que el robot pueda dirigirse hacia la posición siguiente. Existen valores predefinidos: **z10**, **z20**, **z50**, **fine**, ... **.** El valor indica la máxima diferencia en mm. respecto del punto pedido. Se especifican más parámetros: orientaciones, ejes externos, ... También se distingue entre puntos de paso y puntos de paro.

Zona de precisión para un punto de paso



Diferencia punto de paso / punto de paro

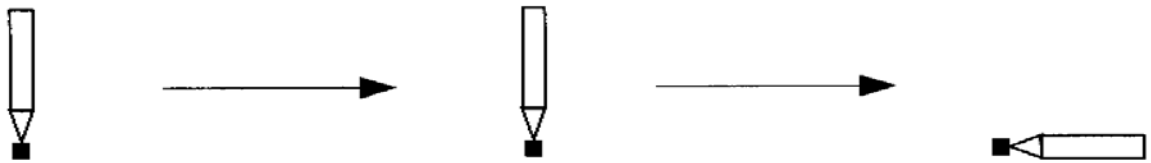


Figura 2 Tres posiciones han sido programadas; la última con una orientación de herramienta diferente.

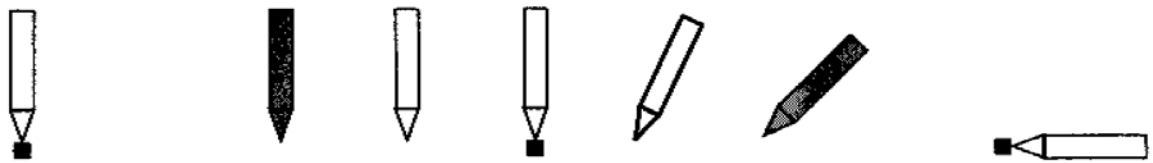


Figura 3 Si todas las posiciones fueran puntos de paro, la ejecución del programa tendría este aspecto.

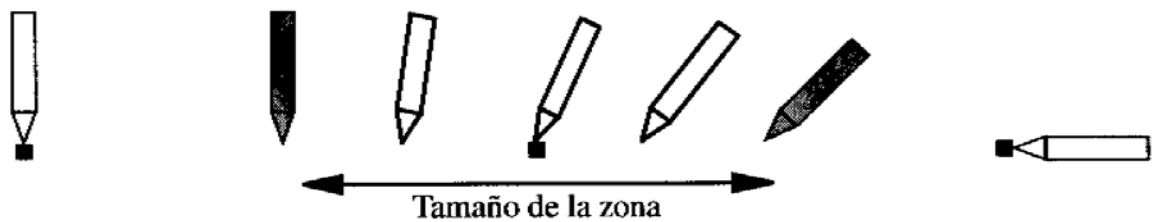


Figura 4 Si la posición del medio fuera un punto de paso, la ejecución del programa tendría este aspecto

Variables y expresiones en RAPID (VI)

TIPOS DE DATOS PROPIOS DEL MANEJO DEL ROBOT (cont)

ToolData: herramienta utilizada por el robot. Se especifica tanto la geometría (sistema de coordenadas de la herramienta) como la carga (peso, centro de gravedad y momentos de inercia).

LoadData: pieza manipulada por el robot. Se especifican el peso, el centro de gravedad y los momentos de inercia.

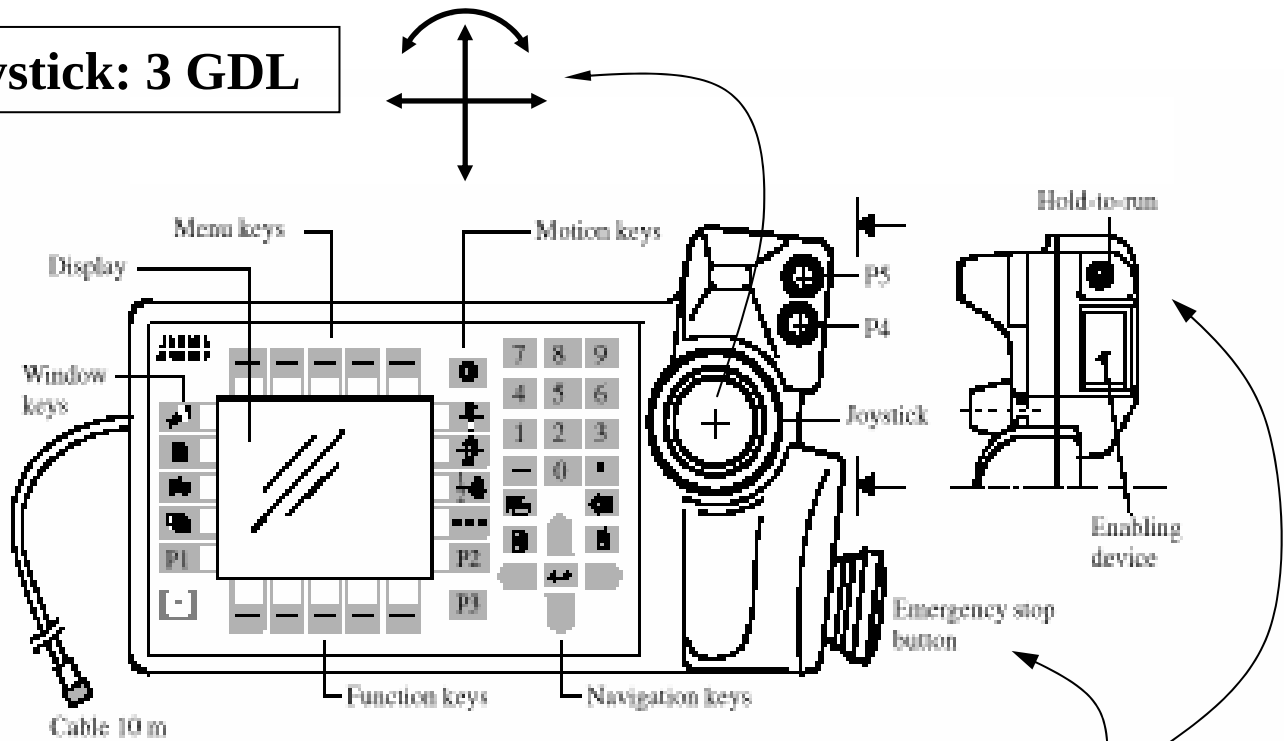
Introducción de programas en el IRB-140

DOS POSIBILIDADES

- Programación desde la paleta de programación.
  - Se lleva al robot a las posiciones deseadas manualmente (haciendo uso del joystick).
  - Se crea una instrucción de programa para cada posición indicada manualmente, especificando velocidad, precisión, etc.
  - Se añaden las instrucciones necesarias: control de E/S, etc.
  - Ventajas: no requiere ningún cálculo, pocas posibilidades de error.
- Escritura del programa en PC y transferencia al robot vía disquete.
  - Las posiciones se calculan analíticamente; con estos datos se escribe un programa utilizando el lenguaje RAPID.
  - Ventajas: facilidad para introducir modificaciones o revisar.

Programación desde la paleta (I)

Joystick: 3 GDL



2 dispositivos de seguridad

Programación desde la paleta (II)

Varias posibilidades para mover el robot con el joystick:

•Introduciendo traslaciones a lo largo de 3 ejes y giros sobre 3 ejes (cada eje se maneja con un grado de libertad del joystick).

- •respecto del sistema de coordenadas de la base del robot.
- •respecto del sistema de coordenadas de la herramienta u otro.

•Actuando directamente sobre las articulaciones del robot (máximo sobre 3 de los ejes simultáneamente).

Programación desde PC (I)

Cabecera de un programa RAPID:

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

Código del programa, siempre estructurado en módulos:

```
MODULE carga_CNC
  ...
  Cuerpo del programa
    ...
ENDMODULE
```

Por ejemplo, en este módulo se incluirían todos los datos e instrucciones correspondientes a la carga de una pieza en una máquina CNC.

Programación desde PC (II)

Dentro de un módulo: declaraciones de datos y procedimientos (rutinas):

```
MODULE carga_CNC
    VAR pos posic1 := [100, 200, 300];
    VAR pos posic2 := [200, 200, 100];...

    PROC rutina1
        ... instrucciones ...
    ENDPROC

    PROC rutina2
        ... Instrucciones ...
    ENDPROC

ENDMODULE
```

NOTA: Siempre debe existir la rutina main() que llama a las demás.

Ejemplo de programa completo (I)

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

MODULE movimiento_y_agarre

    !posición pedida: x=100, y=200, z=300
    CONST pos posic1 := [100, 200, 300];

    !orientación pedida, igual a la de la base del robot
    CONST orient ori1 := [1, 0, 0, 0];

    !configuración: cuadrantes 0, 0 y -1
    CONST confdata config1 := [0, 0, -1, 1];

    ! ejes externos: el valor 9E9 indica que no existen
    CONST extjoint ejes1 := [9E9, 9E9, 9E9, 9E9, 9E9, 9E9];

    !el objetivo para el robot incluye todos los datos:
    CONST robtarget rob1 := [posic1, ori1, config1, ejes1];
    ...
```


Ejemplo de programa completo (II)

```
...
PROC main()

    ! Espera a que la señal digital di5 esté activada
    ! (espera a tener una pieza disponible)
    WaitDI di5, 1;

    ! Se mueve a la posición programada el línea recta
    ! Se supone definida la herramienta tool1
    MoveL rob1, v200, fine, tool1;

    ! Activa la señal digital do3
    ! (activa la pinza del robot)
    Set do3;

ENDPROC

ENDMODULE
```