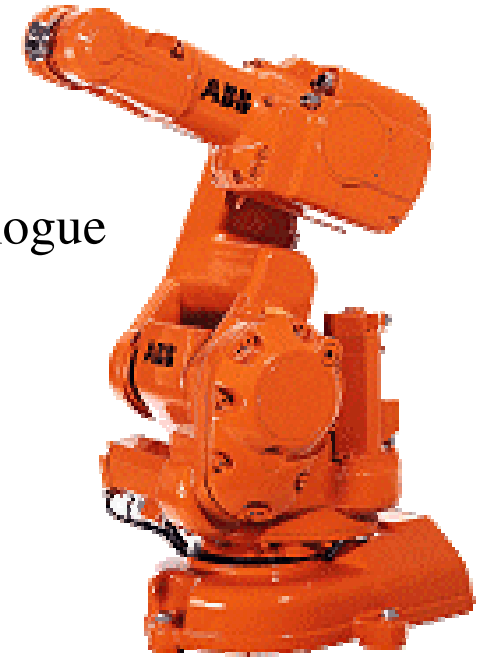
Automatización Industrial - II

56 – 10569 2º Cuatrimestre 2006

Práctica 3– Lunes y Miércoles 8 & 10 de Mayo 2006

Lenguaje RAPID

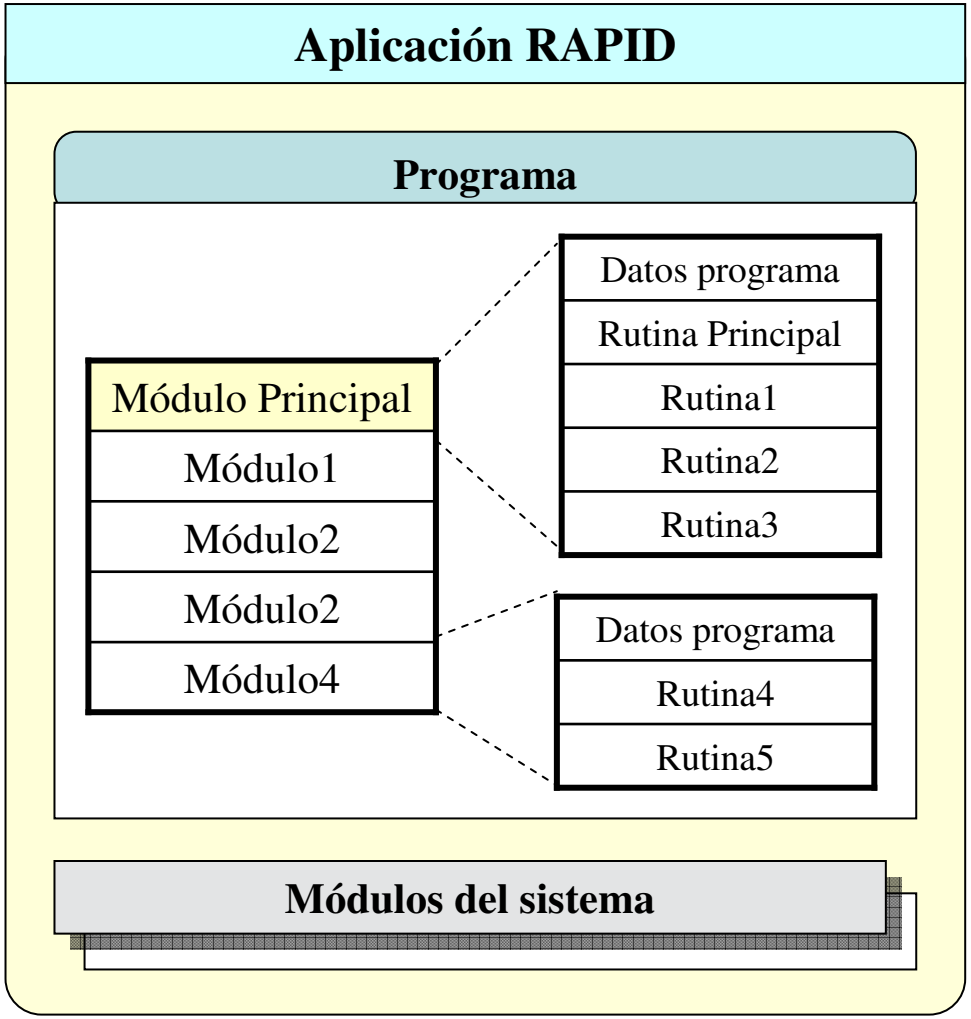
Robotics Application Programming Interactive Dialogue



ABB

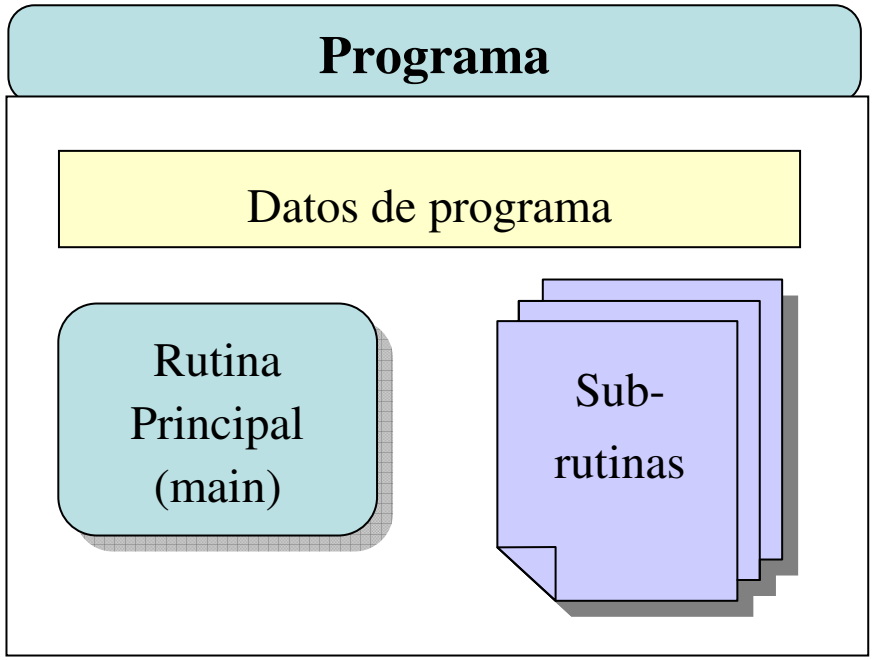
Lenguaje RAPID

- **Estructura del lenguaje**
  - RAPID es un leguaje de programación textual de alto nivel desarrollado por la empresa ABB.
  - Una aplicación RAPID consta de un programa y una serie de módulos del sistema.



Lenguaje RAPID

Programa RAPID



- El programa es una secuencia de instrucciones que controlan el robot y en general consta de tres partes:
  - Una rutina principal (main):

    Rutina donde se inicia la ejecución.
  - Un conjunto de sub-rutinas:

    Sirven para dividir el programa en partes más pequeñas a fin de obtener un programa modular.
  - Los datos del programa:

    Definen posiciones, valores numéricos, sistemas de coordenadas, etc.

Lenguaje RAPID

Programa RAPID

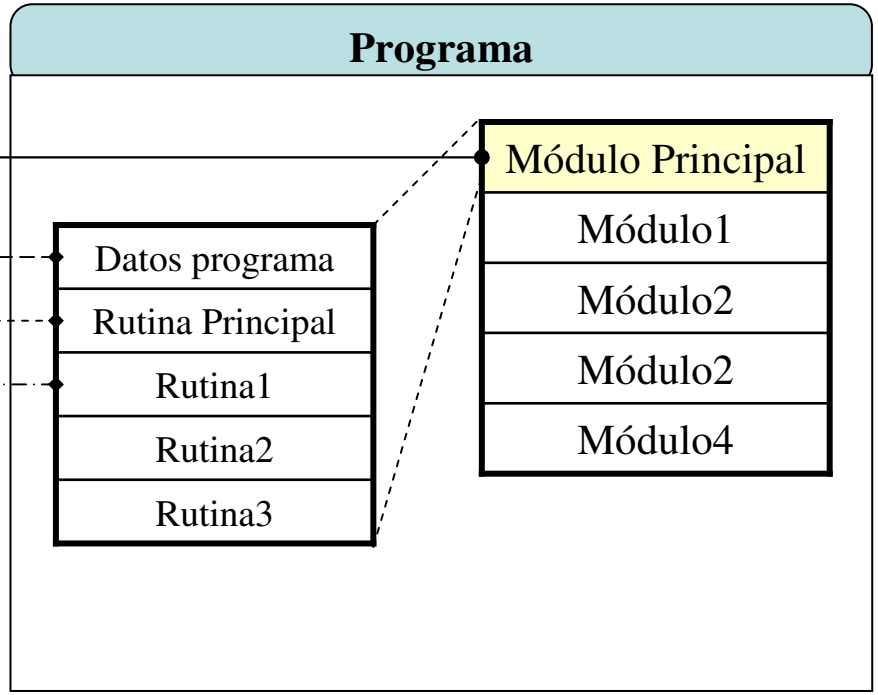
```
%%%
VERSION:1
LANGUAGE:ENGLISH
%%%
MODULE Module1

  declaraciones

  PROC proc1()
     . . .
  ENDPROC

  PROC main()
     . . .
     . . .
  ENDPROC

ENDMODULE
```



Lenguaje RAPID

- **Programa RAPID**

```
%%%
VERSION:1
LANGUAGE:ENGLISH
%%%
MODULE EJEMPLO
  CONST robtarget A:=[[0,0,0],[0,0,0,0],[0,-1,0,0], [9E+09,...]];

  CONST tooldata pinza:= [TRUE, [[0,0,0],[1,0,0,0]],
                                    [0,[0,0,0],[1,0,0,0],0,0,0]];
  PROC cerrar_pinza()
    Set spinza;
  ENDPROC

  PROC coger_pieza()
    MoveJ B1,v100,z5,pinza;
    MoveL B,v80,fine,pinza;
    cerrar_pinza;
  ENDPROC
```

Lenguaje RAPID

- **Programa RAPID**

```
  PROC main()
    CONST dionum listo:=1;
    abrir_pinza;
    WHILE TRUE DO
      MoveJ A,v100,fine,pinza;
      WaitDI econtrol,listo;
      coger_pieza;
      MoveL B1,v80,z5,pinza;
      MoveJ D,v100,z100,pinza;
      MoveJ C1,v100,z5,pinza;
      MoveL C,v80,fine,pinza;
      abrir_pinza;
      MoveL C1,v80,z5,pinza;
    ENDWHILE
  ENDPROC
ENDMODULE
```

Lenguaje RAPID

Elementos básicos

Identificadores:

Permiten nombrar módulos, rutinas, datos y etiquetas.

Ejemplo:

```
MODULE nombre_módulo
PROC       nomre_rutina()
VAR pos    nombre_dato;nombre_etiqueta:
```

- El primer carácter es siempre una letra**.**
- Longitud máxima 16.
- Diferencia entre mayúsculas y minúsculas.

Palabras reservadas:

AND	BACKWARD	CASE	CONNECT	CONST	DEFAULT	DIV
DO	ELSE	ELSEIF	ENDFOR	ENDFUNC	ENDIF	ENDMODULE
ENDPROC	ENDTEST	ENDTRAP	ENDWHILE	ERROR	EXIT	FALSE
FOR	FROM	FUNC	GOTO	IF	INOUT	LOCAL
MOD	MODULE	NOSTEPIN	NOT	OR	PERS	PROC
RAISE	READONLY	RETRY	RETURN	STEP	TEST	THEN
TO	SYSMODULE TRAP	TRUE	VAR	VIEWONLY	WHILE	
WITH	XOR					

Lenguaje RAPID

- **Elementos básicos**

  - **Espacios y caracteres de fin de línea:**

  RAPID es un lenguaje sin formatos, en consecuencia los espacios pueden utilizarse en cualquier parte excepto en: identificadores, palabras reservadas, valores numéricos.

  Los identificadores , las palabras reservadas y los valores numéricos deberán estar separados entre sí por un espacio, un carácter de fin de línea o un tabulador

  - **Comentarios:**

  Sirven para facilitar la comprensión del programa, ocupan una línea entera comenzando con el símbolo !, finaliza con un carácter de fin de línea.

```
! Esto es un comentario
```

  - **Valores de cadena:**

  Secuencia de caracteres entre comillas.

```
"Esto es una cadena"
```

Lenguaje RAPID

Los Datos

- Los datos a manejar pueden ser definidos como:

  Constantes: `(CONS)` representen datos de un valor fijo a los que no se puede reasignar un nuevo valor.

  Variables: `(VAR)` son datos a los que se les puede asignar un nuevo valor durante la ejecución del programa.

  Persistentes: `(PERS)` se trata de variables en las que cada vez que se cambia su valor durante la ejecución del programa, también se cambia el valor de su inicialización.

Lenguaje RAPID

Los Datos

- Los datos se pueden definir según la cantidad de memoria que se necesita para almacenarlo:

Atómicos: En ellos solo se guarda un dato. No se puede dividir en otros más sencillos.

Registros: Es un tipo de dato en el que se guardan de una forma ordenada más de un dato. En lenguaje C sería un tipo de dato similar a las estructuras.

Lenguaje RAPID

Tipos de Datos: Atómicos

- **num:** Se usa para los valores numéricos, ya sean enteros o reales

Ejemplo:

```
VAR num flujo := 0;
flujo := 2.34;
```

Valores válidos: 5 0.37 0.1E-5 -12.34

- **bool:** Se usa para designar valores lógicos. (verdadero/falso)

Valores posibles: TRUE y FALSE.

```
VAR bool <identificador>:= <valor>
                <valor>: TRUE / FALSE
        <expresión lógica>
```

Ejemplo:

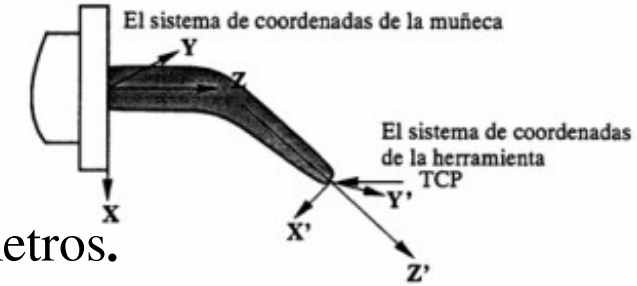
```
VAR bool abrir:=TRUE;
abrir:=FALSE;
abrir:= reg1 > 1;
```

- **string:** Se usa para guardar cadenas de caractéres, que pueden tener como máximo 80 incluidas las comillas “ que son las que delimitan la cadena.

Ejemplo:

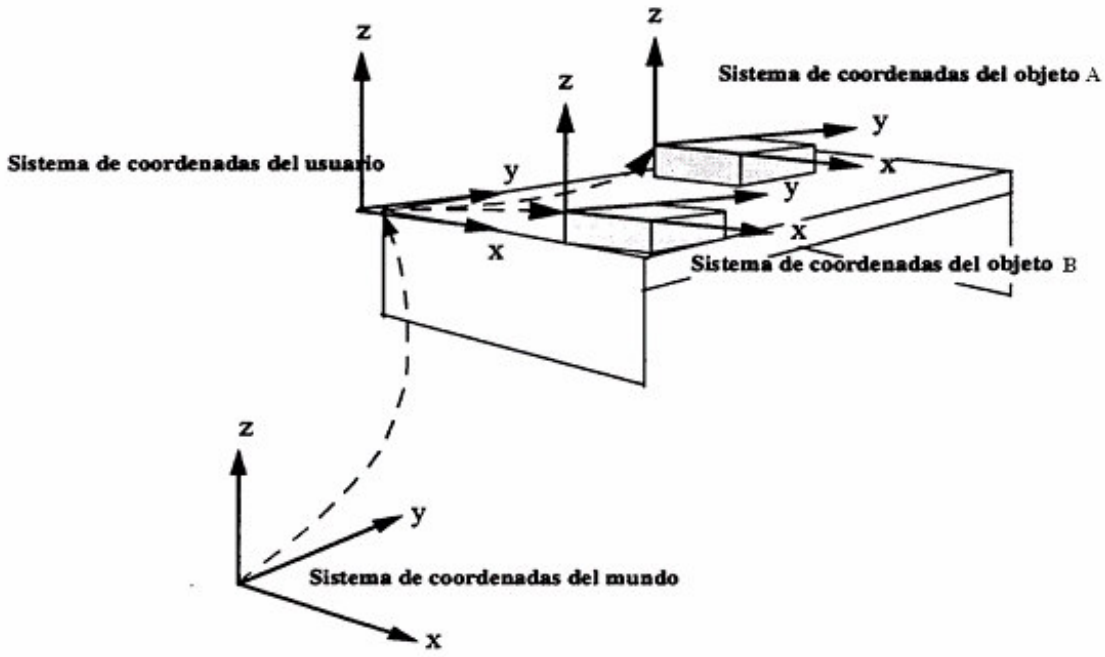
```
VAR string text;
text:= “Arranque del sistema”;
```

Lenguaje RAPID



- **Tipos de Datos: Registros**
  - **pos:** Representar posiciones sólo X, Y y Z en milímetros.

x es de tipo num.
y es de tipo num.
z es de tipo num.

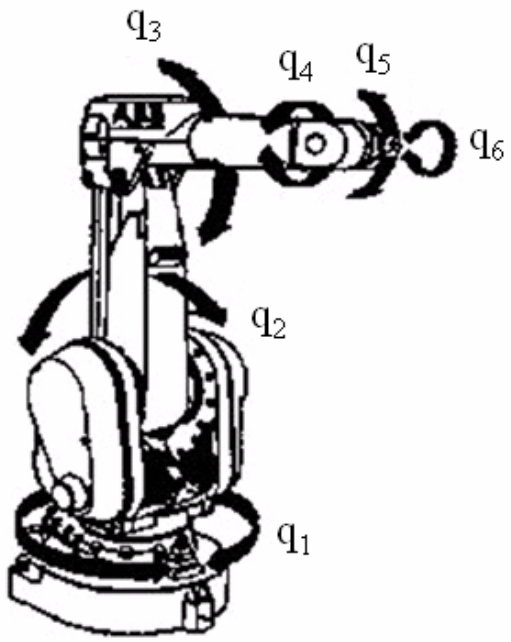


Ejemplo:

```
VAR pos posicion1;
posicion1 := [500, 0, 940];
posicion1.x := posicion1.x + 50;
```

Lenguaje RAPID

Tipos de Datos: Registros



- **orient:** Es un dato de tipo registro que guarda la orientación de algún elemento, como por ejemplo la de la herramienta. ABB usa cuaternios para almacenar la orientación.

  (q1, q2, q3, q4,q5,q6) de tipo num.

- **pose:** Se usa para cambiar de un sistema de coordenadas a otro. Está compuesto por un dato pos y otro de tipo orient.

trans pos
rot orient

Ejemplo:

```
VAR pose pos1;
pos1 := [[500, 100, 800],[1,0,0,0]];
pos1.trans := [650, -230, 1230];
pos1.trans.y := -23.54;
```

Lenguaje RAPID

- **Tipos de Datos: Registros**
  - **confdata:** Permite definir las configuraciones de los ejes del robot.

    cf1: Cuadrante utilizado del eje 1.

    cf4: Cuadrante utilizado del eje 4.

    cf6: Cuadrante utilizado del eje 6.

    Ejemplo:

    ```
    VAR confdata conf10:=[1,-1,0]
    ```

  - **loaddata:** Sirve para describir la carga colocada en la muñeca del robot.

    mass: peso de la carga en kilogramos.

    cog: centro de gravedad de la carga.

    aom: orientación de los ejes de inercia en el centro de gravedad.

    ***ix,iy,iz*:** momentos de inercia de la carga alrededor de los ejes x, y, z en kgm^2.

    Ejemplo:

    ```
    VAR loaddata pieza:=[5,[50,0,50],[1,0,0,0],0,0,0];
    ```

Lenguaje RAPID

- **Tipos de Datos: Registros**
  - **tooldata:** Describe las características de una herramienta.

robhold : Tipo bool que define si el robot sujeta la herramienta o no.
tframe : Sistema de coordenadas de la herramienta
Posición del TCP (x,y,z)
Orinetación. (q1,q2,q3,q4)
tload : Carga de la herramienta
Peso
Centro de gravedad (x,y,z)
Ejes de momento de la herramienta (q1,q2,q3,q4)
Momento de inercia de los ejes (x,y,z).

Ejemplo:

```
PERS tooldata pinza:=[TRUE,[[97,0,220],
[0.924,0,0.383,0]],5,[-23,0,75],[1,0,0,0],0,0,0]]
```

Lenguaje RAPID

- **Tipos de Datos: Registros**
  - **robtarget:** Sirve para definir la posición del robot y de sus ejes externos.

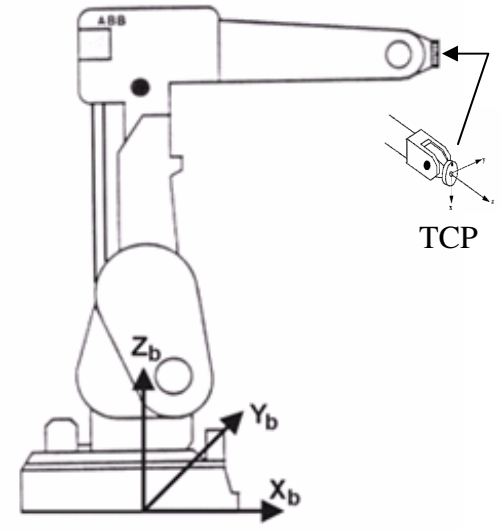
trans	**:** Posiciones (x, y, z)
rot	**:** Orientación de la herramienta.
robconf	**:** Configuración de los ejes.
extax	**:** posición de los ejes externos

Ejemplo:

```
VAR robtarget punto1;
punto1 := [[500, 100, 800],[1,0,0,0]];
punto1.trans := [650, -230, 1230];
punto1.trans.y := -23.54;
```

Lenguaje RAPID

- **Tipos de Datos: Registros**
  - **speeddata:** Para especificar la velocidad a la que se moverán los ejes externos del robot.



v_tcp	**:** Velocidad del TCP mm/s.
v_ori	**:** Velocidad de reorientación del TCP grados/s.
v_leax	**:** Velocidad de los ejes externos lineales en mm/s.
v_reax	**:** Velocidad de los ejes externos rotativos grados/s.

Ejemplo:

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

Datos predefinidos:

v5 [5,5,50,5]
v10,v20,……..,v80
v100, v150, v200, v300,……,v800
v1000, v1500, v2000, v2500, v3000
vmax [5000, 500, 5000, 500]

Lenguaje RAPID

Tipos de Datos: Registros

- **zonedata:** Para especificar como debe terminarse una posición.

finep	**:** Punto de paro o de paso bool
pzone_tcp	**:** Radio de zona del TCP en mm.
pzone_ori	**:** Tamaño de zona de reorientación mm.
pzone_eax	**:** Zona de los ejes externos mm.
zone_ori	**:** Tamaño de la zona de reorientación grados.
zone_leax	**:** Tamaño de la zona ejes externos mm.
zone_reax	**:** Tamaño zona ejes rotativos externos grados.

Ejemplo:

```
VAR zonedata trayec:=[FALSE, 25, 40, 50, 5, 35 10];
```

Datos predefinidos:
z1 [1, 1, 1, 0.1, 1, 0.1]
z5, z10, z15, z20, ……..,z100
z150
z200[200, 300, 300, 30, 300, 30]

Lenguaje RAPID

- **Módulos**
  - **Encabezado de archivo:**

    Puede estar formado de diferentes datos y rutinas.

    Uno de los módulos contiene el procedimiento de entrada, un procedimiento global de entrada llamado `main`.

  - **Módulos del sistema:**

    Sirven para definir datos y rutinas normales del sistema, como por ejemplo las herramientas.

Lenguaje RAPID

- **Módulos**
  - **Declaración:**

```
MODULE      <nombre_módulo> [<Lista de atributos>]
            <Lista declaración de datos>
            <Lista declaración rutina>
ENDMODULE
```

```
[<Lista de atributos>]     :
    SYSMODULE        :  Módulo del sistema.
    NOSTEPIN         :  No se podrá entrar durante ejecución paso a paso.
    VIEWONLY         :  No podrá ser modificado.
    READONLY         :  No podrá ser modificado pero sí sus atributos.
```

Lenguaje RAPID

- **Rutinas**
  - **Tres tipos:**

Procedimientos:

```
PROC <nombre procedimiento> ( Lista de parámetros )
     <Lista de declaraciones de datos>;
     <Lista de instrucciones>;
     ERROR <lista instrucciones>;
ENDPROC
```

Funciones:

```
FUNC <tipo valor dato> ( Lista de parámetros)
     <Lista de declaraciones de datos>;
     <Lista de instrucciones>;
     RETURN dato;
     ERROR <lista instrucciones>;
ENDFUNC
```

Interrupciones:

```
TRAP <nombre trap>
     <Lista de declaraciones de datos>;
     <Lista de instrucciones>;
     ERROR <lista instrucciones>;
ENDTRAP
```

Lenguaje RAPID

Expresiones del lenguaje

Las expresiones se utilizan para evaluar un valor y así poder asignarlo a una variable o utilizarlo como argumento de una instrucción o de una rutina. Según el tipo de valor que devuelve la expresión se distinguen dos tipos:

- **Aritméticas:** Devuelven una valor numérico si operan con variables de tipo num y una cadena si operan con cadenas de caracteres.

Utilizan los operadores aritméticos: *, +, -, /, DIV (división entera), MOD (resto)

Ejemplo:

```
perimetro = 2 * 3.14 * radio
"IN" + "PUT"
```

Lenguaje RAPID

- **Expresiones del lenguaje**

  - **Lógicas:** Devuelven un valor de tipo bool

  Utilizan los operadores lógicos: <, >, <>, =, <=, >=, AND, OR, NOT, XOR

Ejemplo:

```
DInput(di1) = 1 Doutput(do3) = 0
num1 < num2;
nombre1 = nombre2;
Doutput(do1) = 0 AND pos1.x > 100
```

Lenguaje RAPID

- **Instrucciones: Movimiento**

Para mover el robot hay tres instrucciones:

```
MoveJ Punto, Velocidad, Zona, Herramienta
```

Se mueve el robot hacia un punto usando coordenadas articulares.

Cuando no tiene que seguir ninguna trayectoria determinada.

```
MoveL Punto, Velocidad, Zona, Herramienta
```

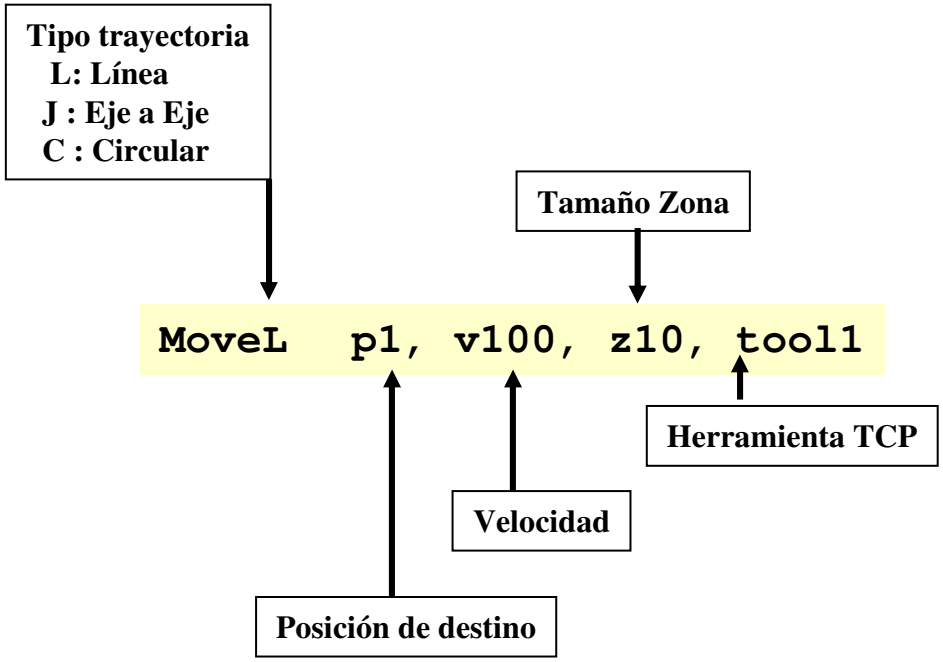
Se mueve el robot hacia un punto usando la línea recta.

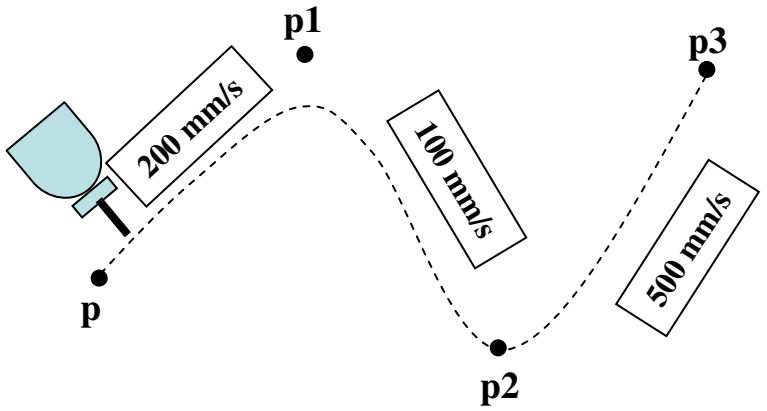
```
MoveC Punto_Circulo, Punto_Destino, Velocidad, Zona, Herramienta;
```

Se mueve el extremo del robot hacia el punto de destino pasando por el punto del circulo trazando un arco de circunferencia.

Lenguaje RAPID

- **Instrucciones: Movimiento**

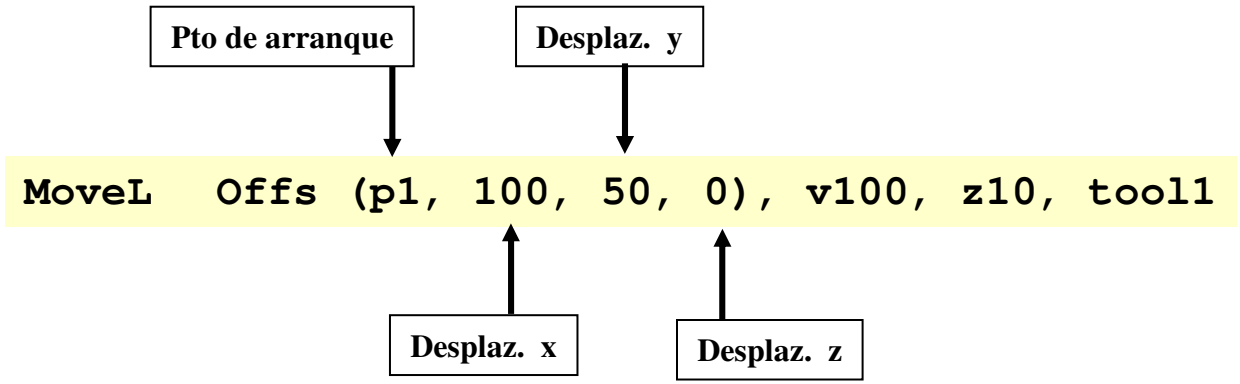


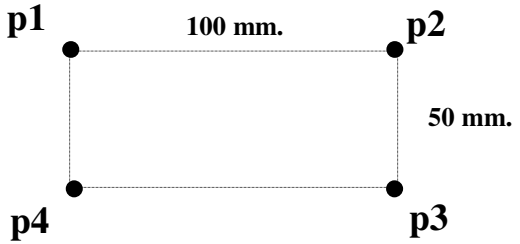



```
MoveL  p1, v200, z10, herram1
MoveL  p2, v100, fine, herram1
MoveJ  p3, v500, fine, herram1
```

Lenguaje RAPID

- **Instrucciones: Posicionamiento (Programación con desplazamiento)**






```
MoveL  p1, v200, fine, herram1
MoveL  Offs (p1, 100, 0, 0 ), v100, fine, herram1
MoveL  Offs (p1, 100, 50, 0 ), v100, fine, herram1
MoveL  Offs (p1, 0, 50, 0 ), v100, fine, herram1
MoveL  p1, v100, fine, herram1
```

La función Offs sirve para desplazar una posición del robot. Devuelve el dato de la posición desplazada de tipo robtarget.

Offs(punto, OffestX, OffestY, OffestZ)

Lenguaje RAPID

- **Instrucciones: entrada/salida**

```
Set señal;
```

Sirve para colocar el valor de la señal de la salida digital a uno.

```
Reset señal;
```

Sirve para poner una señal de salida digital a cero

```
SetDO señal, valor;
```

Sirve para cambiar el valor de una señal de salida digital

```
DInput( di1)     /    DOutput(do2)
```

Lectura de Entradas / Salidas digitales

Ejemplo:

```
SetDO do1, 1          ! Activación =1  Desactivación = 0
```

Lenguaje RAPID

- **Instrucciones: Condición de espera**

```
WaitDI          di, 1    !Esperar hasta que se active una señal digital

WaitTime        0.5      !Esperar cierto tiempo

WhileUntil               !Esperar hasta que se cumpla cierta condición
```

Lenguaje RAPID

- **Control de Flujo: Compact IF**

  Ejecutar una instrucción sólo si se cumple una condición.

```
IF <condición> Instrucción;
```

- **Control de Flujo: IF**

  Diferentes instrucciones se ejecutan si se cumple la condición.

```
IF <condición>          THEN
                Instrucciones;
ELSE
                Instrucciones;
ENDIF
```

- **Control de Flujo: FOR**

```
FOR <contador> FROM VI TO VF [ STEP Incremento ] DO
    Instrucciones;
ENDFOR
```

Lenguaje RAPID

- **Control de Flujo: WHILE**

```
WHILE  <condición>          DO
       Instrucciones;
ENDWHILE
```

- **Control de Flujo: TEST**

```
TEST <dato>
       CASE valor1, valor2,.., valor(n-1):
                    rutina1;
       CASE valor n:
                    rutinax;
       DEFAULT
                    instrucciones;
ENDTEST
```

- **Control de Flujo: GOTO**

```
GOTO   Etiqueta
```

Lenguaje RAPID

Juego de instrucciones del RAPID

:=	Asignar un valor
Abs()	Obtener el valor absoluto
AInput()	Leer el valor de una señal de entrada analógica
AccSet	Reducir la aceleración
Add	Sumar un valor numérico
Clear	Borrar un valor
ClkStart	Iniciar un reloj para la toma de tiempos
ClkStop	Parar un reloj para la toma de tiempos
comment	Comentario
CompactIF	Si se cumple una condición, entonces... (una instrucción)
ConfJ	Controlar la configuración durante movimiento articular
ConfL	Monitoriza la configuración del robot durante movimiento en línea recta
Decr	Decrementar en 1
EXIT	Terminar la ejecución del programa
FOR	Repetir un número de veces
GetTime()	Leer el valor de la hora actual como valor numérico
GOTO	Ir a una nueva instrucción
GripLoad	Definir la carga del robot
HoldMove	Interrumpir el movimiento del robot
IF	Si se cumple una ocndición, entonces...; de otra manera...
Incr	Incrementar en 1
InvertDO	Invertir el valor de una salida digital
label	Nombre de una línea
LimConfL	Definir la desviación permitida en la configuración del robot
MoveC	Mover el robot en movimiento circular
MoveJ	Movimiento articular del robot
MoveL	Movimiento del robot en línea recta
Offs()	Desplazamiento de la posición del robot
Open	Apertura de un fichero o de un canal serie

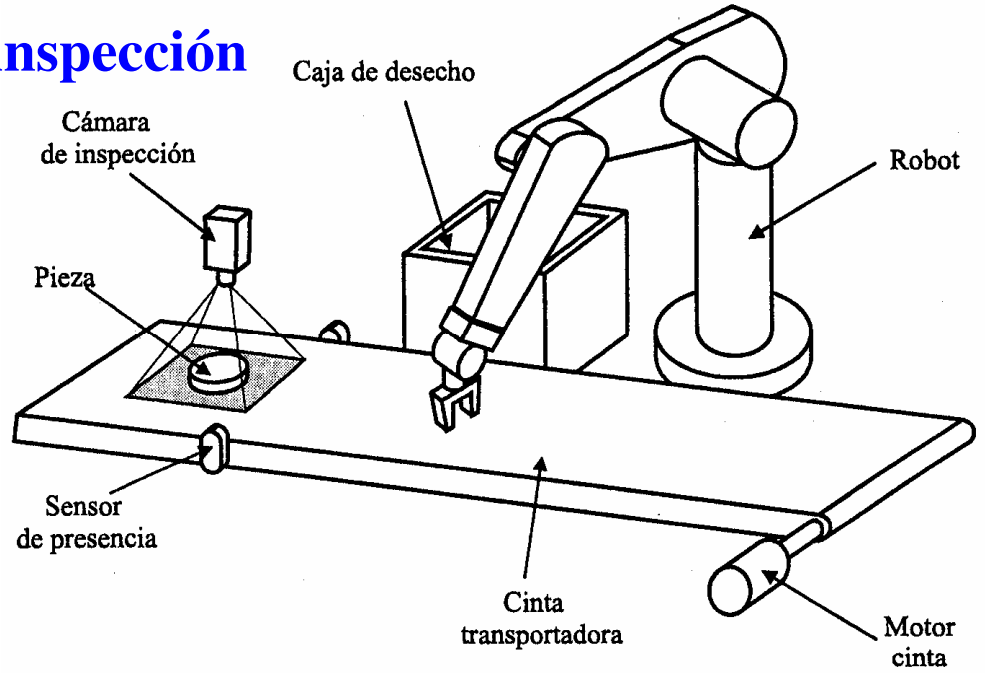
Lenguaje RAPID

Juego de instrucciones del RAPID

Present()	Comprobar que se utiliza un parámetro opcional
ProcCall	Llamada a un nuevo procedimiento
PulseDO	Generar un pulso en una señal digital de salida
RAISE	Llamada a un manejador de errores
RelMove	Continuar con el movimiento del robot
Reset	Reset de una salida digital
RETRY	Recomenzar tras un error
RETURN	Terminar la ejecución de una rutina
Set	Set de una salida digital
SetAO	Cambiar el valor de una salida analógica
SetDO	Cambiar el valor de una salida digital
SetGO	Cambiar el valor de un grupo de salidas digitales
SingArea	Definición de la interpolación alrededor de puntos singulares
Stop	Parar la ejecución de un programa
TEST	Dependiendo del valor de la expresión...
TPErase	Borrar el texto de la paleta de programación
TPReadFK()	Leer las teclas de función de la paleta de programación
TPWrite	Escribir en la paleta de programación
VelSet	Cambiar la velocidad programada
WaitDI	Esperar hasta el set de una entrada digital
WaitTime	Esperar un tiempo determinado
WaitUntil	Esperar hasta que se cumpla una condición
WHILE	Repetir mientras ...
Write	Escribir en un fichero de caracteres o en un canal serie
WriteBin	Escribir en un canal serie binario

Lenguaje RAPID

- **Ejemplo: Célula robotizada de inspección**
  - El robot retira de una cinta transportadora aquellas piezas identificadas como defectuosas.



- La operación se desarrolla como sigue:

1. El robot se encuentra en espera hasta la llegada de una señal indicando la existencia de una pieza defectuosa sobre la cinta transportadora.
2. El robot procede entonces a parar la cinta y a coger la pieza y a depositarla en un almacén de piezas defectuosas.
3. El propio robot se encarga de activar de nuevo el movimiento de la cinta una vez la pieza ha sido cogida.
4. Tras la operación, el robot vuelve a su posición inicial y se repite de nuevo el ciclo.

Lenguaje RAPID

Célula de inspección: Definición de variables

herramienta: una variable de tipo *tooldata* que representa una pinza en el extremo del robot para la manipulación de piezas.

carga: una variable de tipo *loaddata* para definir la carga a transportar por la pinza.

```
PERS tooldata herramienta:=[FALSE,[[97,0,223],
                [0.924,0,0,0.383,0]],[5,[-23,0,75],[1,0,0,0],0,0,0]]
PERS loaddata carga:=[5,[50,0,50],[1,0,0,0],0,0,0];
```

tooldata

```
robhold   : Tipo bool que define si el robot sujeta la herramienta o no.
tframe    : Sistema de coordenadas de la herramienta
            Posición del TCP (muñeca del robot) (x,y,z)
            Orientación         (q1,q2,q3,q4)
tload     : Carga de la herramienta
            Peso
            Centro de gravedad (x,y,z)
            Ejes de momento de la herramienta (q1,q2,q3,q4)
            Momento de inercia de los ejes (x,y,z).
```

Lenguaje RAPID

Célula de inspección: Definición de variables

```
VAR signaldo pinza              !señal de activación de pinza
VAR signaldo activar_cinta      !señal de activación de cinta
VAR signaldi pienza_defectuosa  !señal de pieza defectuosa
VAR signaldi terminar           !señal de terminar programa
```

Es necesario definir una configuración inicial en la que el robot espera la señal que le indica que puede recoger la pieza defectuosa.

```
VAR robtarget conf_espera:=[[600,500,225],[1,0,0,0],[1,0,0,0],
                           [9E9,9E9,9E9,9E9,9E9,9E9]];
```

robtarget

```
trans      : Posiciones ( x, y, z )
rot        : Orientación de la herramienta.
robconf    : Configuración de los ejes.
extax      : posición de los ejes externos
```

Lenguaje RAPID

- **Célula de inspección: Rutinas de control de la pinza**

```
PROC Coger()
   Set pinza     !Cerrar la pinza activando la señal digital pinza
   WaitTime 0.3          !Esperar 0,3 segundos
   GripLoad carga        !Señalar que la pieza está cogida
ENDPROC

PROC Dejar()
   Reset pinza           !Abrir la pinza
   WaitTime 0.3          !Esperar 0,3 segundos
   GripLoad LOAD0        !Señalar que no hay pieza cogida
ENDPROC
```

Lenguaje RAPID

- **Célula de inspección: Rutina de coger la pieza de la cinta**

```
PROC Coger_pieza()
  MOVEJ *,VMAX,z60,herramienta   !Mov. en articulares con poca precisión
  MOVEL *,V500,z20,herramienta   !Mov. Línea recta con precisión
  MOVEL *,V150,FINE,herramienta  !Bajar con precisión máxima
  Coger                          !Coger la pieza
 MOVEL *,V200,z20,herramienta    !Subir con la pieza cogida
ENDPROC
```

- **Célula de inspección: Rutina de dejar la pieza**

```
PROC Dejar_pieza()
  MOVEJ *,VMAX,z30,herramienta   !Mover hacia almacén piezas dañadas
  MOVEJ *,V300,z30,herramienta
  Dejar                           !Dejar la pieza
ENDPROC
```

Lenguaje RAPID

- **Célula de inspección: Rutina de ir a la posición de espera**

```
PROC Ir_posicion_espera()
  MOVEJ conf_espera,VMAX,z30,herramienta  !Mover a posición inicial
ENDPROC
```

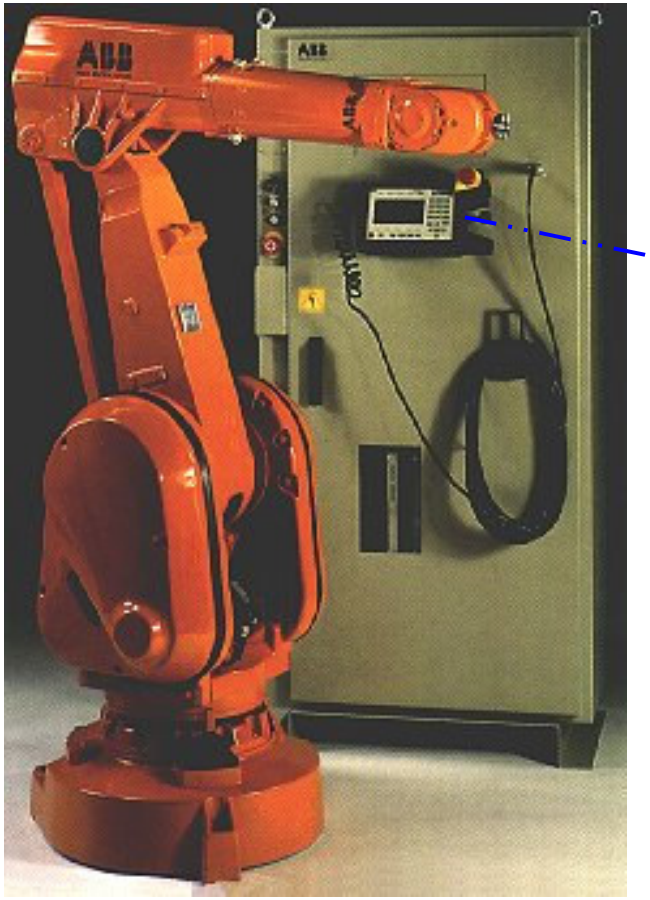
Lenguaje RAPID

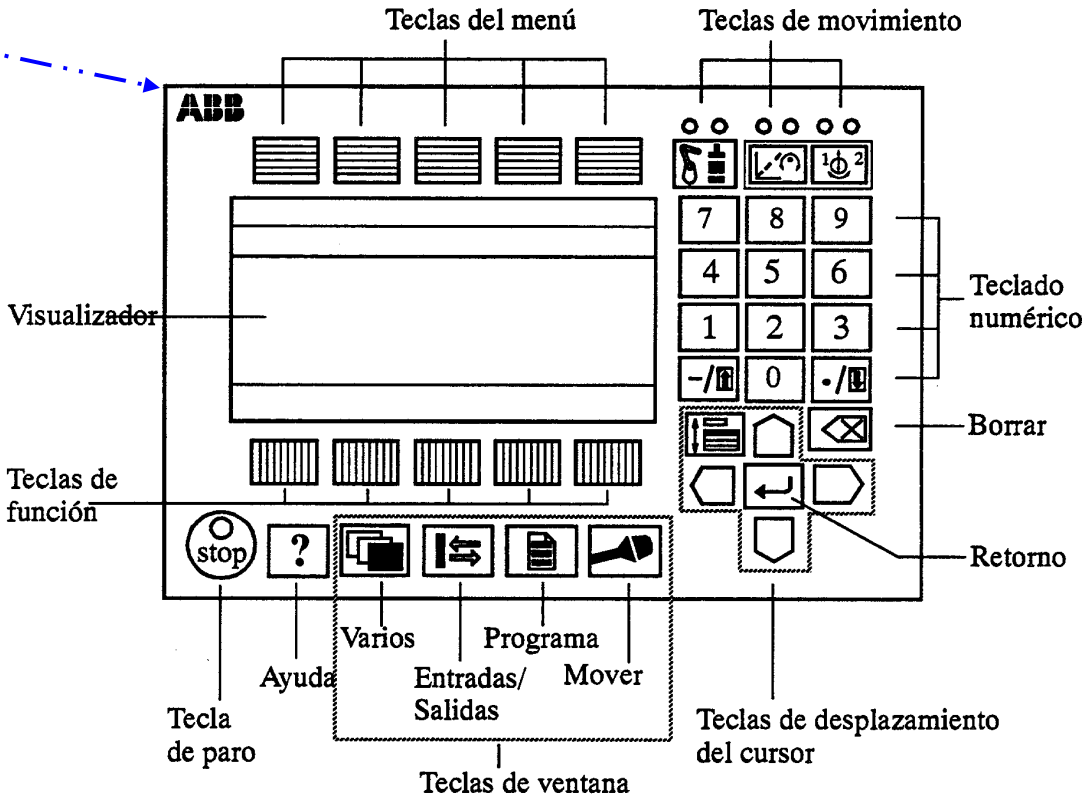
Célula de inspección: Programa principal

```
PROC main()
  Ir_posicion_espera;           !Mover a posición de espera
 WHILE Dinput(terminar)=0 Do    !Esperar la señal de terminar
  IF Dinput(pieza_defectuosa)=1 THEN !Esperar la señal de pieza
                                !defectuosa
  SetDO activar_cinta,0;        !Parar cinta
  Coger_pieza                   !Coger la pieza
  SetDO activar_cinta,1;        !Activar señal de cinta
  Ir_posicion_espera;           !Mover a posición de espera
  ENDIF
 ENDWHILE
ENDPROC
```

Lenguaje RAPID

- ## Entorno de programación

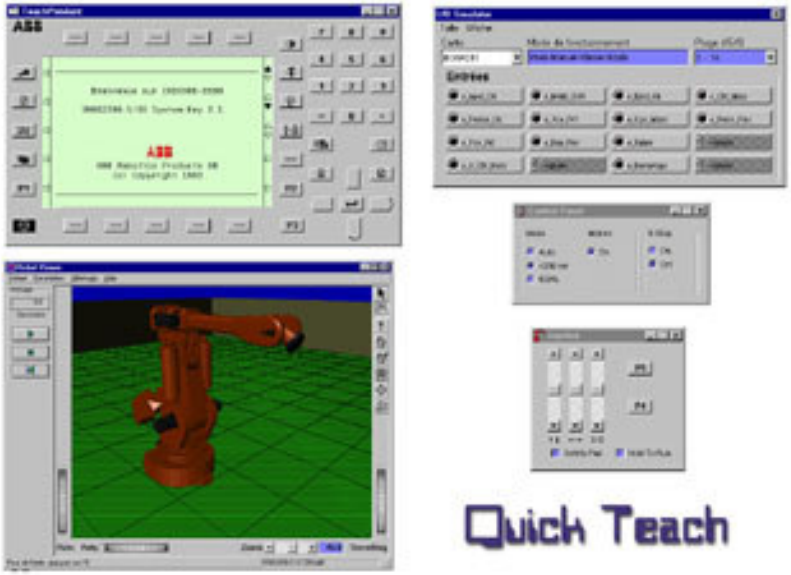




Lenguaje RAPID

- **Entorno de programación: ABB**

RAPID SyntaxChecker
(Analizador sintáctico fuera de línea)



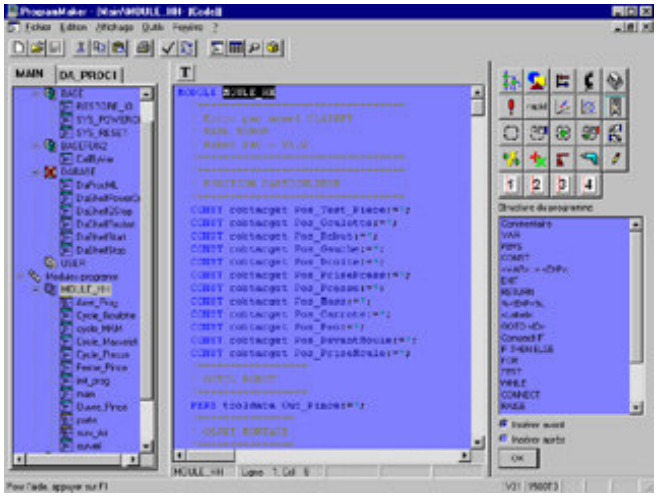
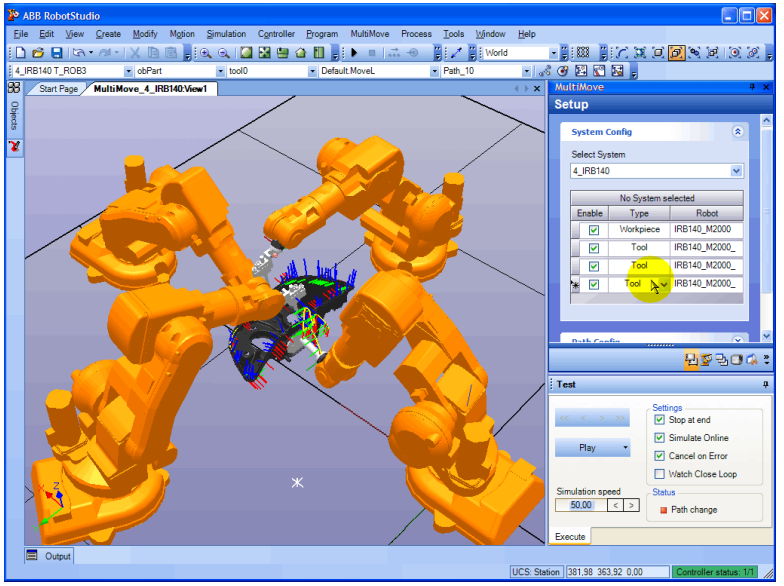


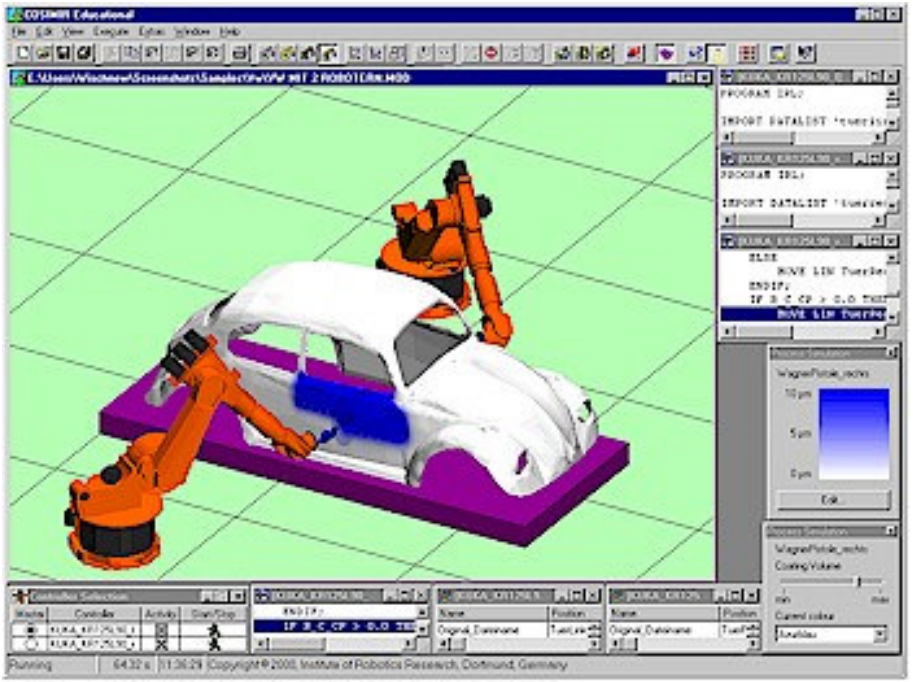
ABB Deskware

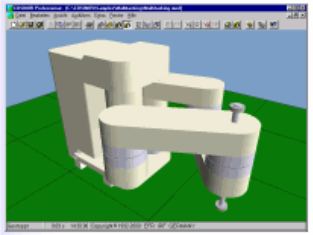


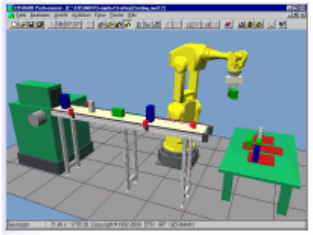
RobotStudio™ 5

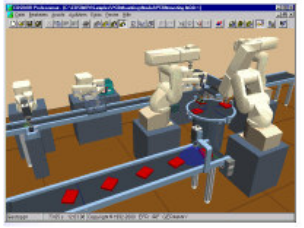
Lenguaje RAPID

- **Entorno de programación: FESTO COSIMIR ®**



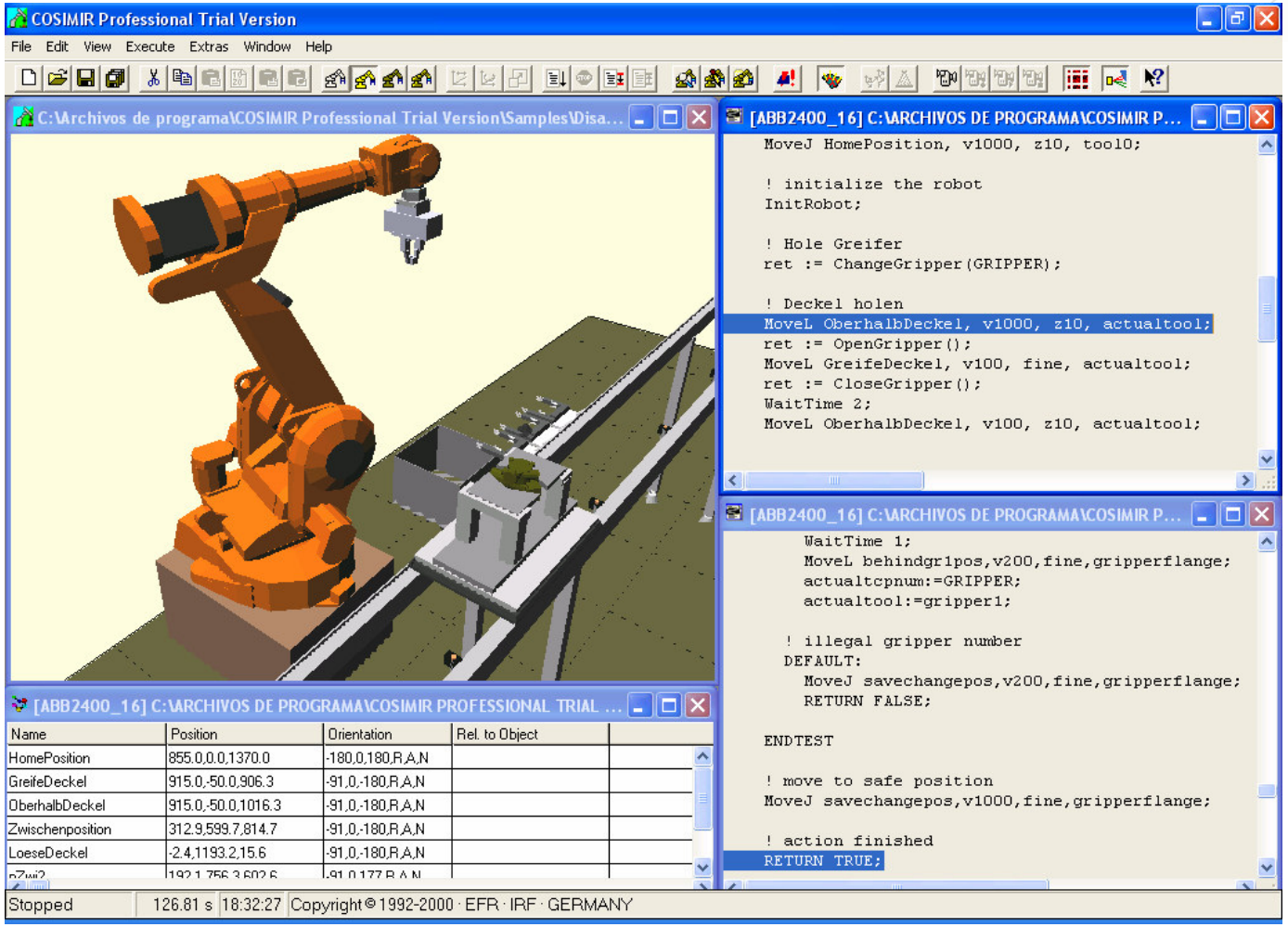






Lenguaje RAPID

- **FESTO COSIMIR ®** **Demo >>**



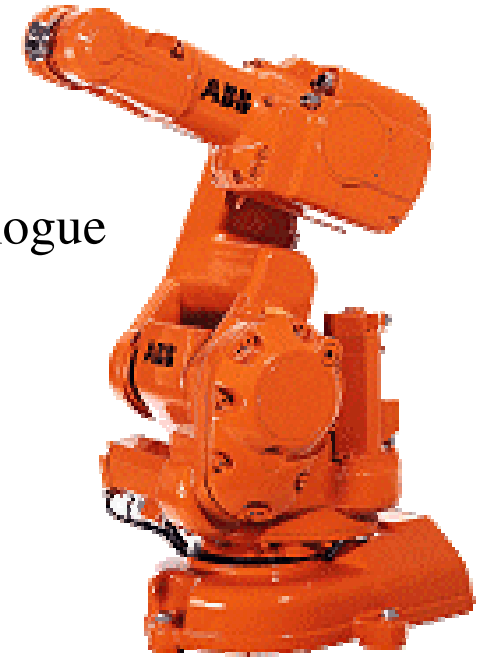
Automatización Industrial - II

56 – 10569 2º Cuatrimestre 2006

Práctica 3– Lunes y Miércoles 8 & 10 de Mayo 2006

Lenguaje RAPID

Robotics Application Programming Interactive Dialogue



ABB