

FAST MOVING TECHNOLOGY

**STÄUBLI**

## FORMACIÓN VAL3

ROBOT Gama RX / TX RS Controlador CS8 / CS8C



# ÍNDICE

- *Introducción*
- *Presentación y arranque del sistema.*
- *Error de Hardware.*
- *Primeros pasos.*
- *Calibración Robot RX 60-90-130-170.*
- *Control de una aplicación.*
- *Estructura de una aplicación*
- *Aprendizaje de la célula.*
- *Movimientos simples.*
- *Edición de programas.*
- *Aproximación a puntos cartesianos.*
- *Gestor de tareas y depurador.*
- *Gestión de entradas / salidas digitales.*
- *Programación estructurada.*
- *Ventana Operador y dialogo operador.*
- *Variables locales y paso de parámetros.*
- *Planos de referencia locales y palatización.*
- *Instrucciones del sistema y multitarea.*
- *Librerías.*
- *Stäubli Robotics Suite.*
- *Anexos.*

# FORMACIÓN VAL3



## *Introducción*



# STÄUBLI

Stäubli es un innovador proveedor de soluciones de mecatrónica con tres divisiones:



Textile

Maquinas de textil que incluyen: excéntricas, maquinas de lizos, maquinas Jacquard, monturas, preparación de tejeduría, sistemas de preparación de alfombras y soluciones automáticas para las maquinas de textil.

Conectors



Enchufes rápidos de: aire comprimido, hidráulica, y electricidad.  
Multi-conectores, sistemas de cambio rápido de moldes y cambiadores de herramienta de robots .



Robotics

Una amplia gama de robots para la automatización industrial, incluyendo robots SCARA y antropomórficos.  
Software de programación para cualquier robot y aplicación y software específico para un segmento industrial.

## HISTORIA DE LA DIVISIÓN DE ROBOTICS

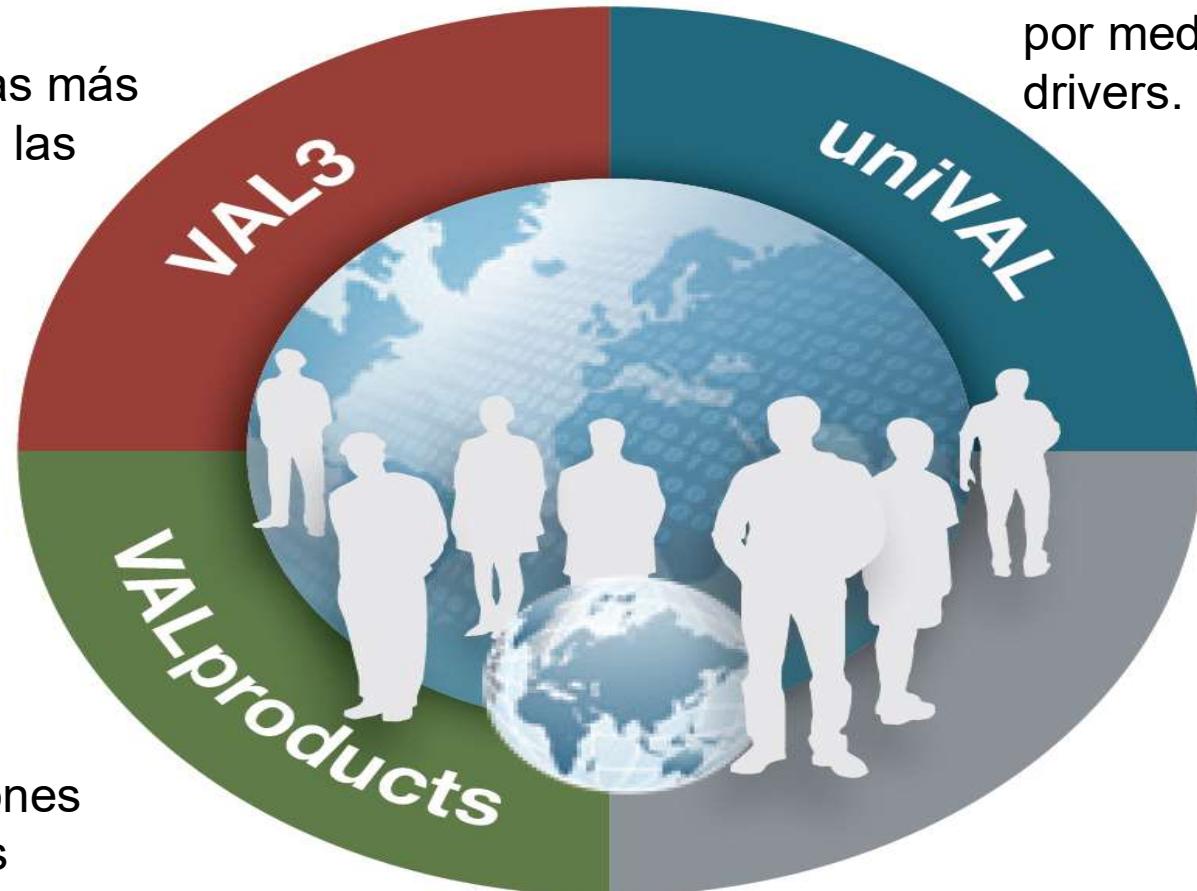
- 1982 : Creación de la división de Robotics
- 1982 – 1989 : Distribución de Unimation robots (USA)
- 1989 : Compra de Unimation, pionera en la industria robótica
- 1992 : Lanzamiento de la serie RX de robots
- 2000 : La producción alcanza 1000 robots / año
- 2002 : Lanzamiento de controlador CS8
- 2004 : Lanzamiento de la serie TX de robots
- 2005 : Compra de la división de robótica de Rexroth
- 2008 : RX170 HSM (Robot de mecanizado) / TX200 / Stericlean
- 2009 : TS80 – TS60 SCARAs / TX200L
- 2010 : uniVAL drive
- 2011 : TS40 SCARA
- 2013 : TP80



# DIVISIÓN DE ROBÓTICA, SOFTWARE

VAL3: Lenguaje de programación desde las más simples aplicaciones a las más complejas.

UniVal drive: Control externo por medio de control de drivers.



VALproductos: Soluciones orientadas a mercados específicos.



Online



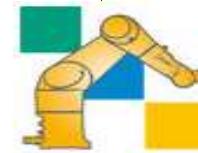
Offline



VAL3 Studio



Emulador



3D Studio



Transfer Manager



Remote Acces

## VAL3

Es un mismo lenguaje de programación para toda nuestra gama de robots, desde un RS20 hasta un RX260L.



Combina las características básicas de un lenguaje informático de alto nivel en tiempo real, multitarea y las funciones específicas del control de una célula robotizada.

### VALtending



Carga/descarga de la máquina y operaciones de recogida y colocación

### VALplast



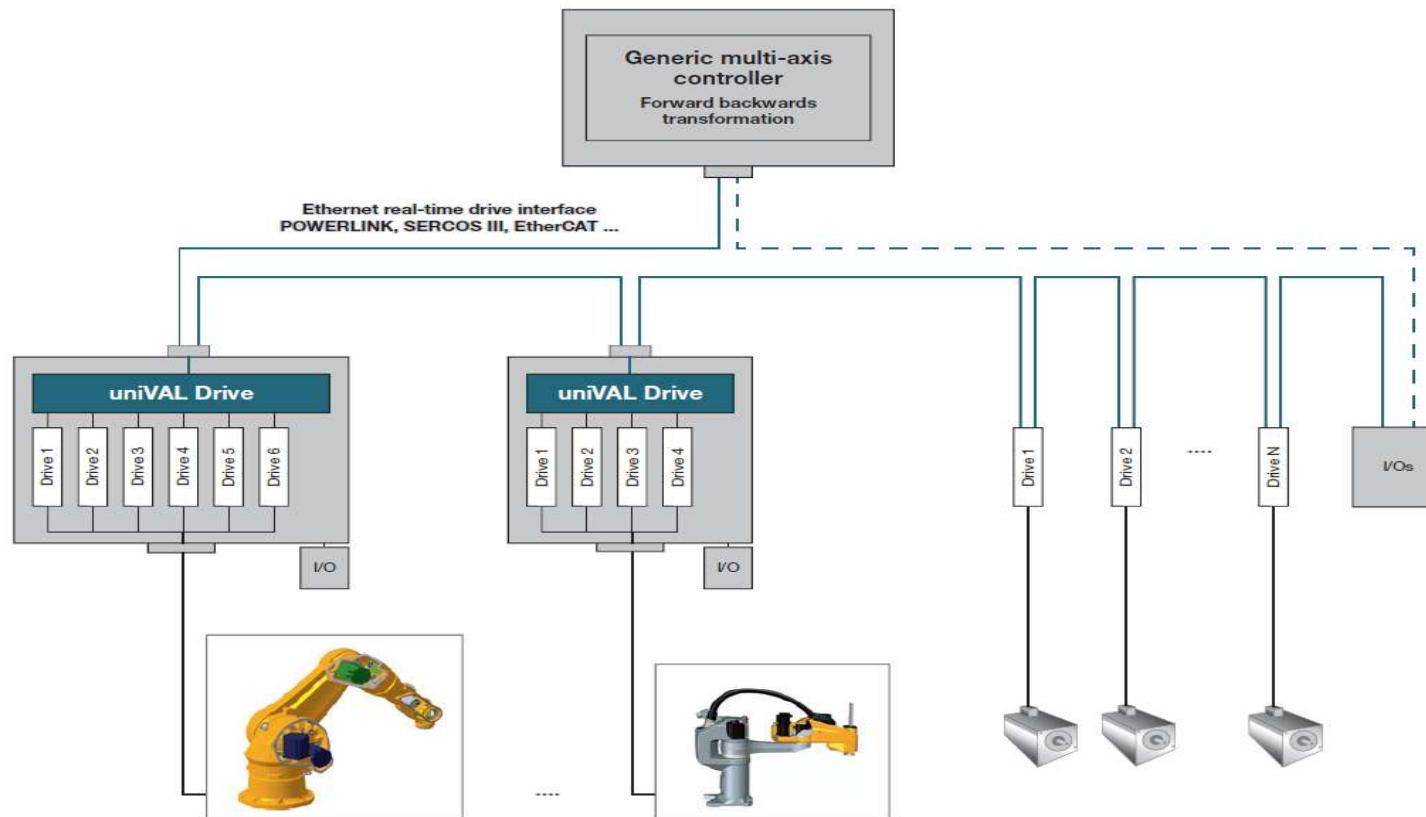
Aplicaciones de carga/descarga de máquina inyección y operaciones a pie de la prensa.

### PaintiXen



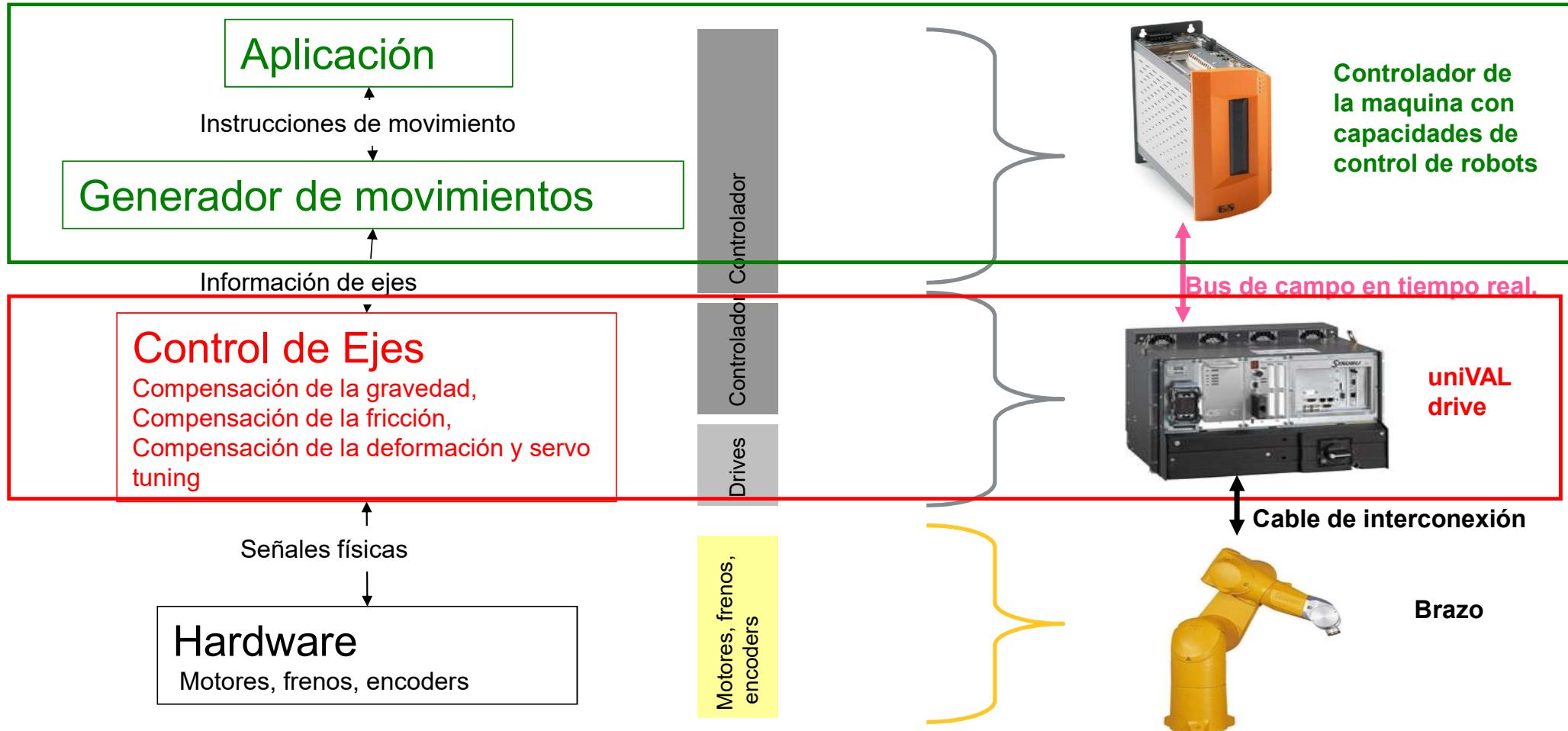
Software de pintura diseñado para simplificar el uso y la programación de la aplicación de pinturas.





Es la solución de Stäubli para controlar cualquiera de nuestros robots con un controlador industrial de ejes.

# CAPAS DE CONTROL CON uniVAL



## FORMACIÓN VAL3

# *Presentación y arranque del sistema*



# SISTEMA ROBOT RX/TX CS8/CS8C

STÄUBLI



*Controlador*

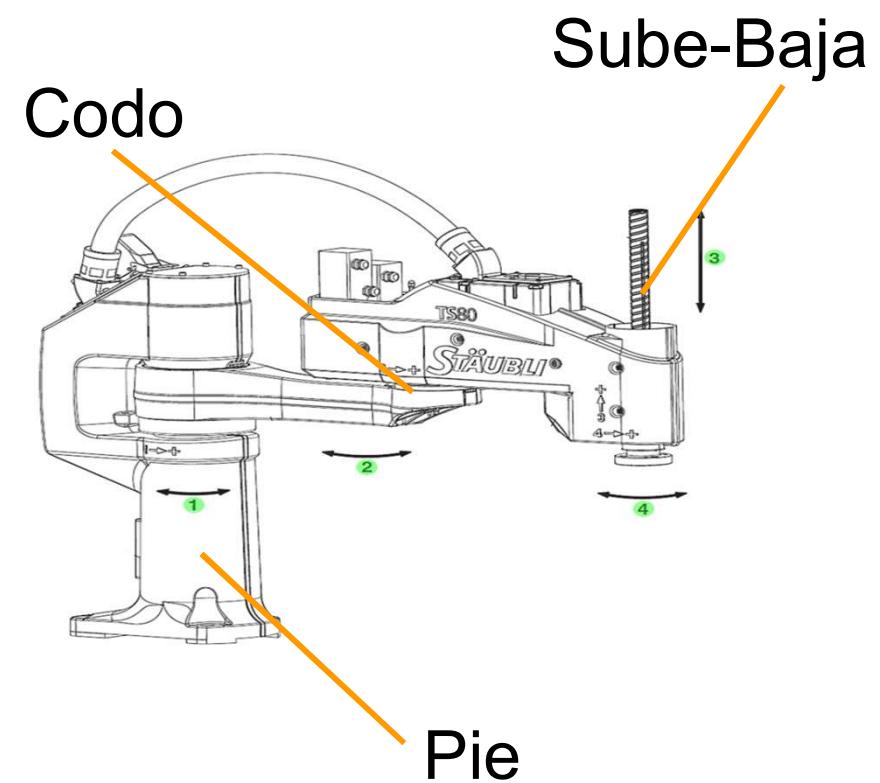
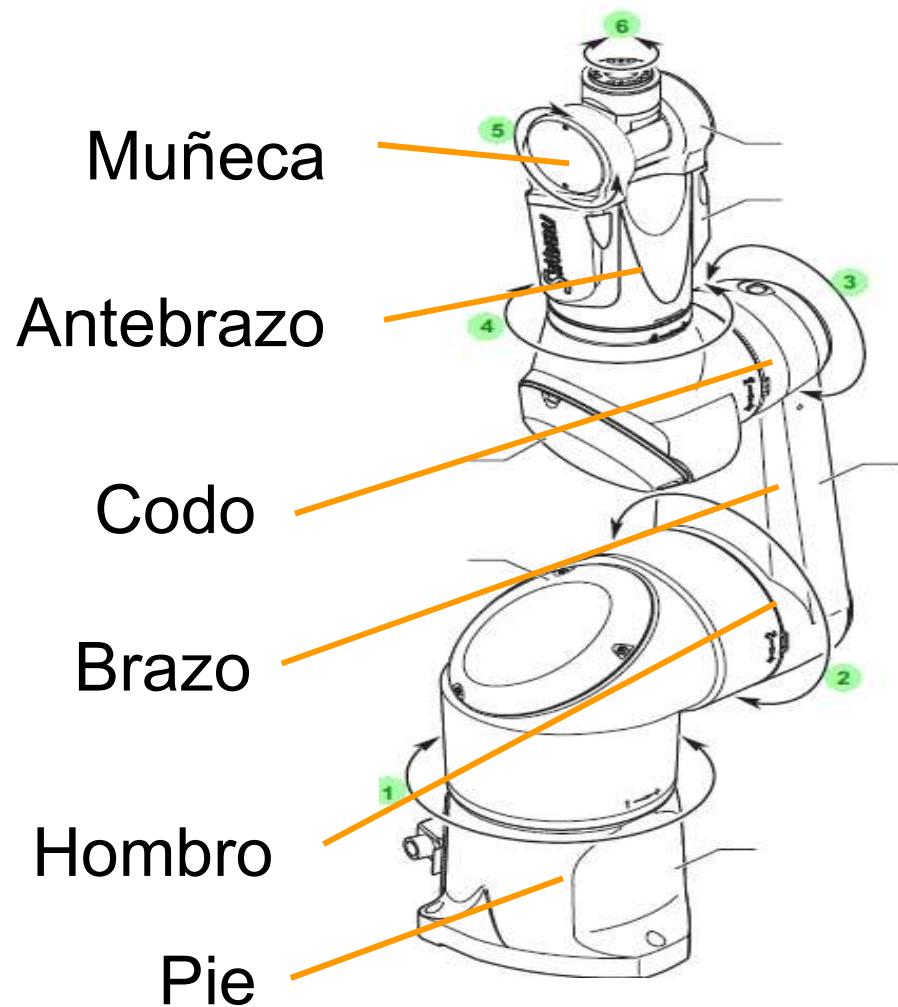
*SP1 (Mando de control)*



*Brazo*



## BRAZOS



## CONEXIONES RX / TS

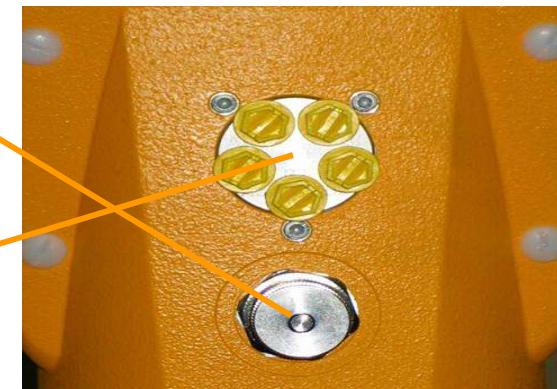
RX



Cables eléctricos para uso del cliente.

Conexiones neumáticas.

TX



Selector de frenos + botón liberación frenos

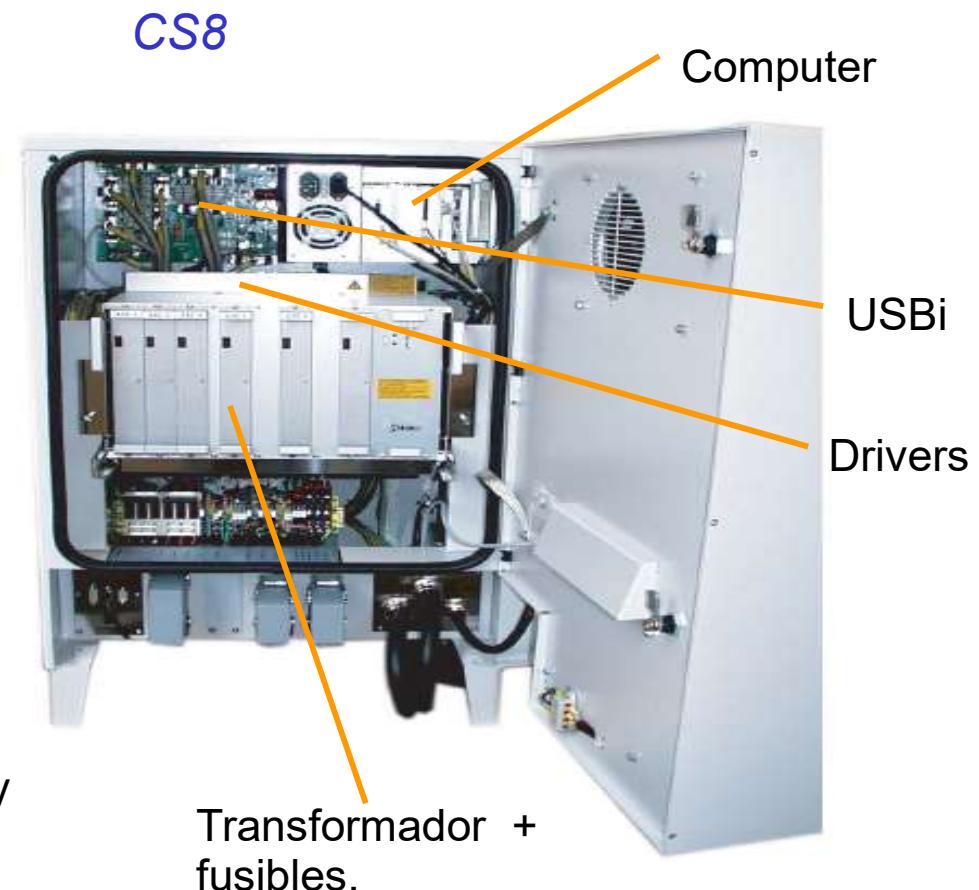
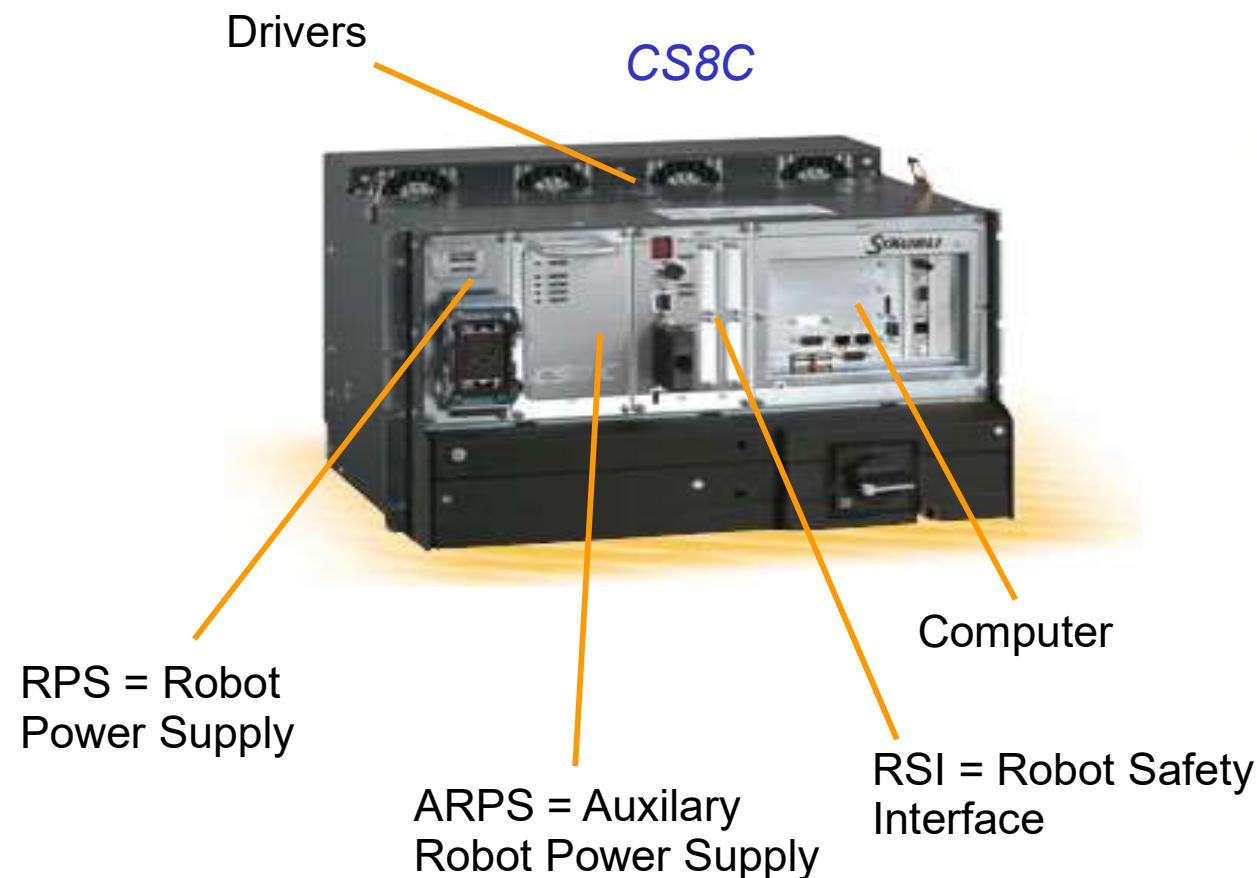
Conexiones neumáticas.

Cables eléctricos para uso del cliente.



# CONTROLADORES

STÄUBLI



## CONEXIONES CONTROLADOR

CS8C



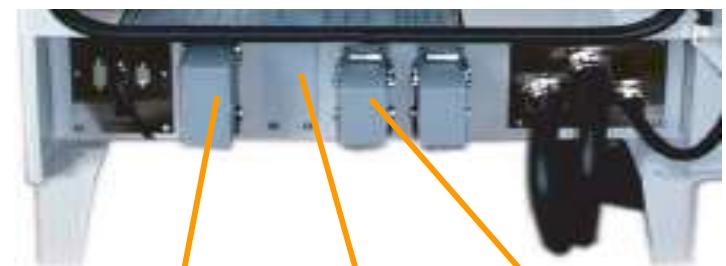
Entradas y  
salidas rápidas

Entradas y  
salidas

Can Bus

Bus de campo

CS8

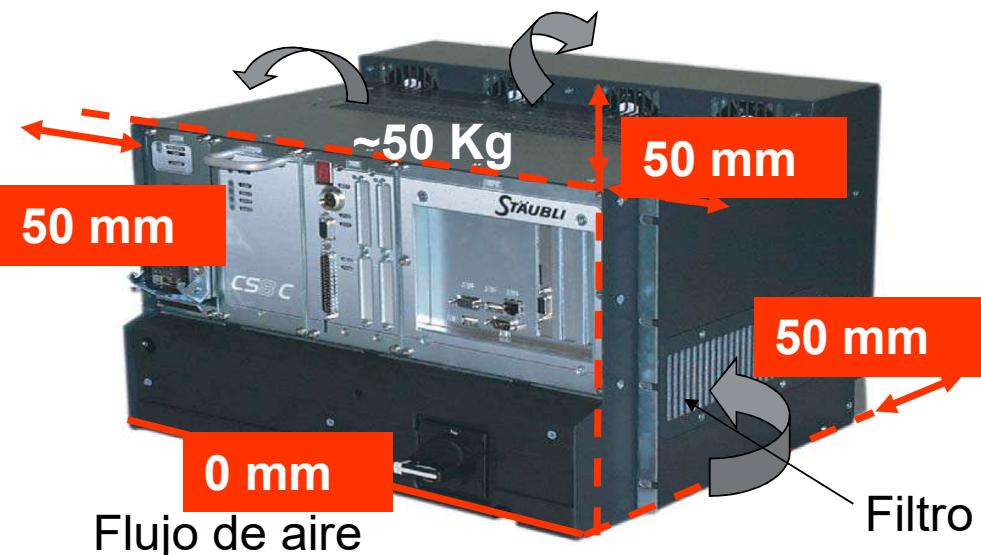
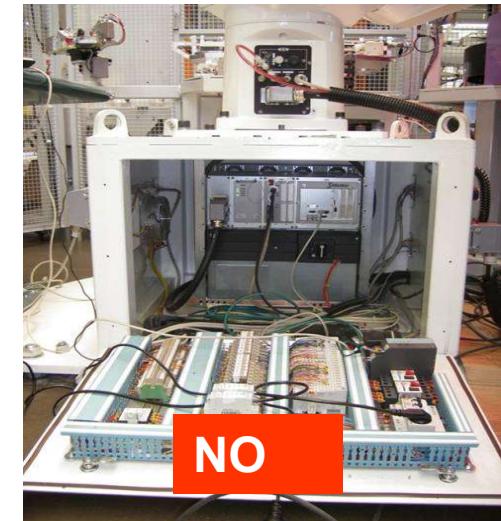


Entradas y  
salidas

Entradas y  
salidas

Bus de campo

## INTEGRACIÓN CS8C (IP20)





# **! PELIGRO !**

**STÄUBLI**

- **Un robot puede moverse a gran velocidad.**
- **Un robot puede moverse de forma inesperada.**
- **Un robot puede causar:**
  - Daños materiales (mecánica del brazo o equipos cercanos).**
  - Daños personales, que pueden ser graves .**

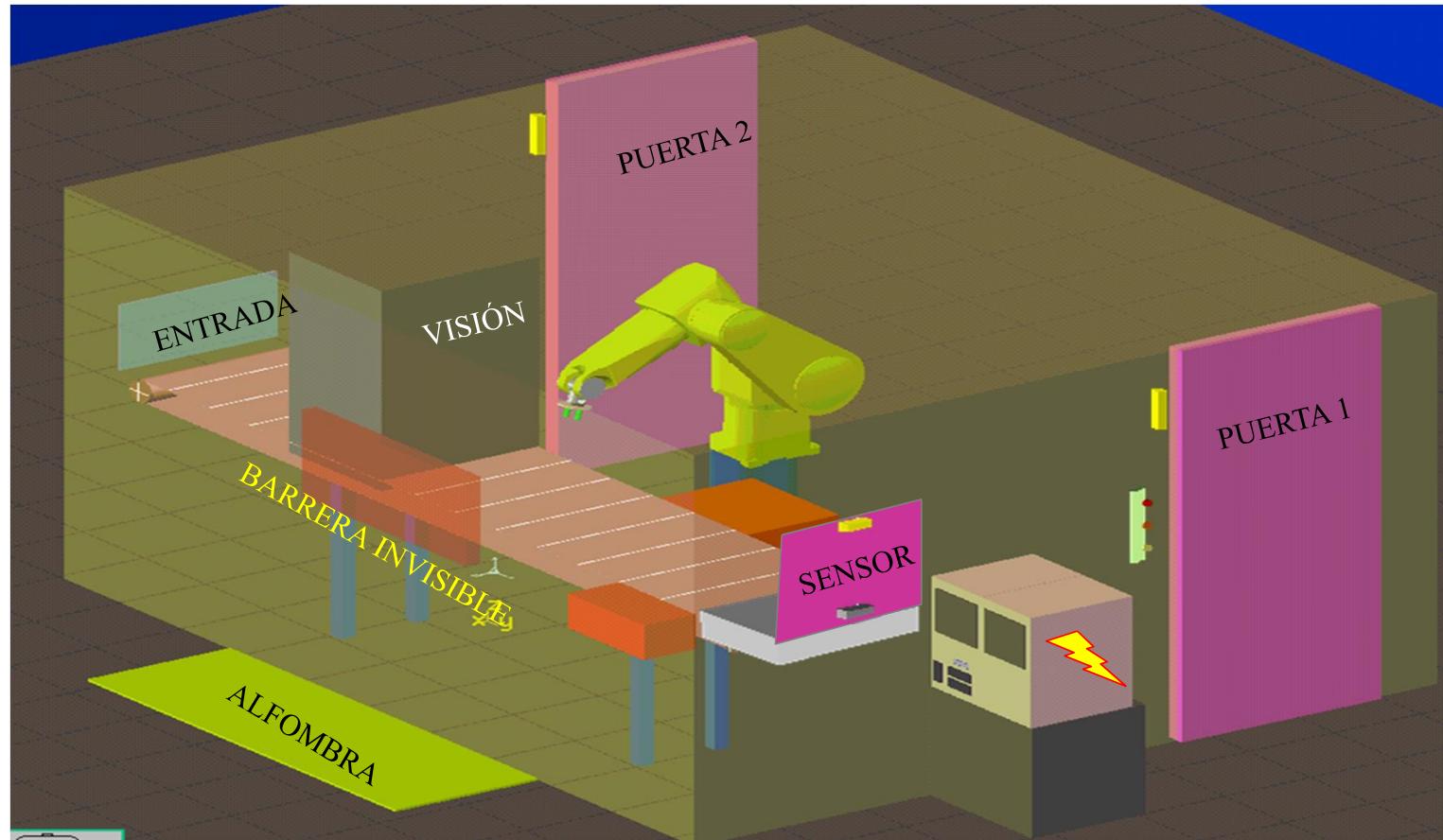


## CÉLULA ROBOTIZADA

- El sistema robotizado siempre es instalado dentro de una célula que cumple con las normas de seguridad.
- El integrador debe crear la célula con el objeto de mantener la seguridad de la instalación:
  - Área cerrada
  - Paros de emergencia
  - Indicadores
- Creada para cumplir con las especificaciones y tareas de producción.
- En una zona de trabajo donde existen otras máquinas.

# LOS COMPONENTES DE UNA CÉLULA

STÄUBLI



# LA SEGURIDAD DE LOS OPERADORES

**No** entrar en la célula si la potencia del brazo está habilitada.



**No** realizar ninguna intervención dentro del controlador si este está encendido.

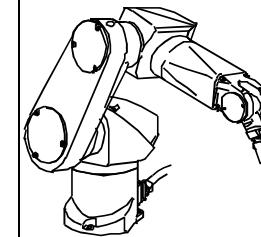
# ¡CONSEJOS INTELIGENTES!

- Parar el robot después de la producción.
- Verificar la posición del brazo antes de habilitar la potencia del brazo.
- Lanzar el primer ciclo a velocidad reducida (10%).
- Estar listos a detener el movimiento del brazo.

- **Parada de emergencia → solo en caso de emergencia**
- Desactivación potencia brazo.
- Move/Hold → Parada inmediata por software, no desactivación de potencia.



Una sola persona dentro de la célula robotizada

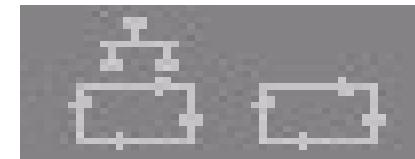


- Seguir las reglas de seguridad
  - Barreras
  - Estar lejos del alcance del brazo.
  - Etc.

## MODOS DE TRABAJO

### AUTOMÁTICO (Modo producción)

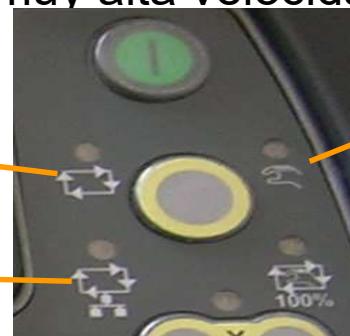
- La celda de robot está cerrada, nadie dentro.
- El robot está bajo el control de programas.
- Los movimientos se pueden hacer a muy alta velocidad.



**AUTOMÁTICO**

Local

Remoto



**MANUAL**

Manual (Lento)  
Modo: Joint, Frame, Tool y Point.  
Test + Máxima velocidad.

### MANUAL (enseñanza de trayectorias, movimiento manual a la posición de inicio )

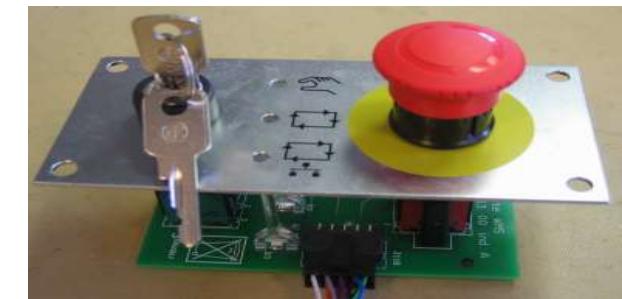
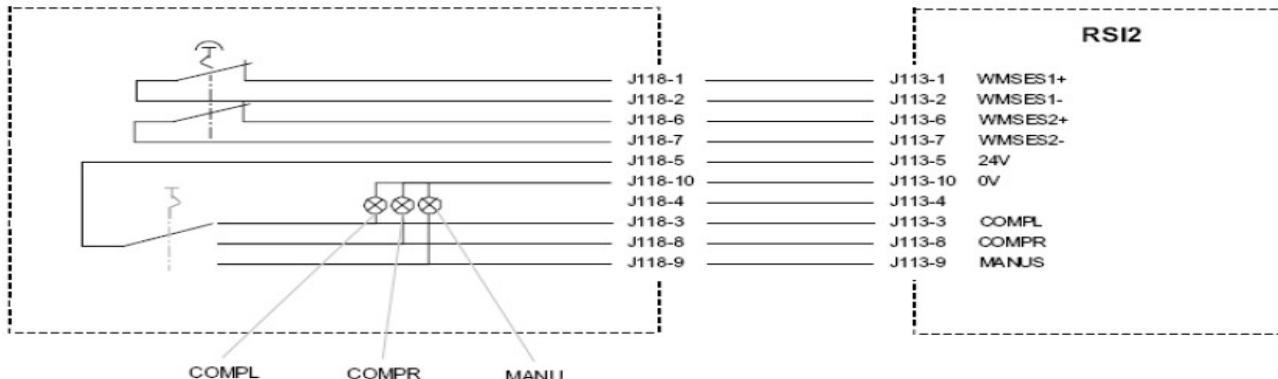
- Este robot está controlado por el operario con el mando (SP1) en las manos.
- La velocidad está limitada a un máximo de 250 mm/s.
- El operador puede estar cerca del robot.



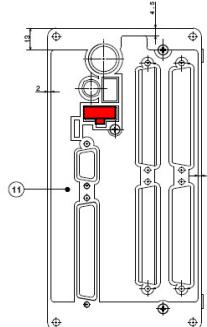
## La Norma ISO NF EN ISO 10218-7

**La selección de modo de trabajo no puede estar en el MCP (SP1). Ha de estar fuera de la celda robotizada y con una llave.**

- CS8C Solución: WMS (Working mode system) integrado con una tarjeta RSI2.
  - Cable 5m o 10m.
  - El integrador puede hacer su propio dispositivo.

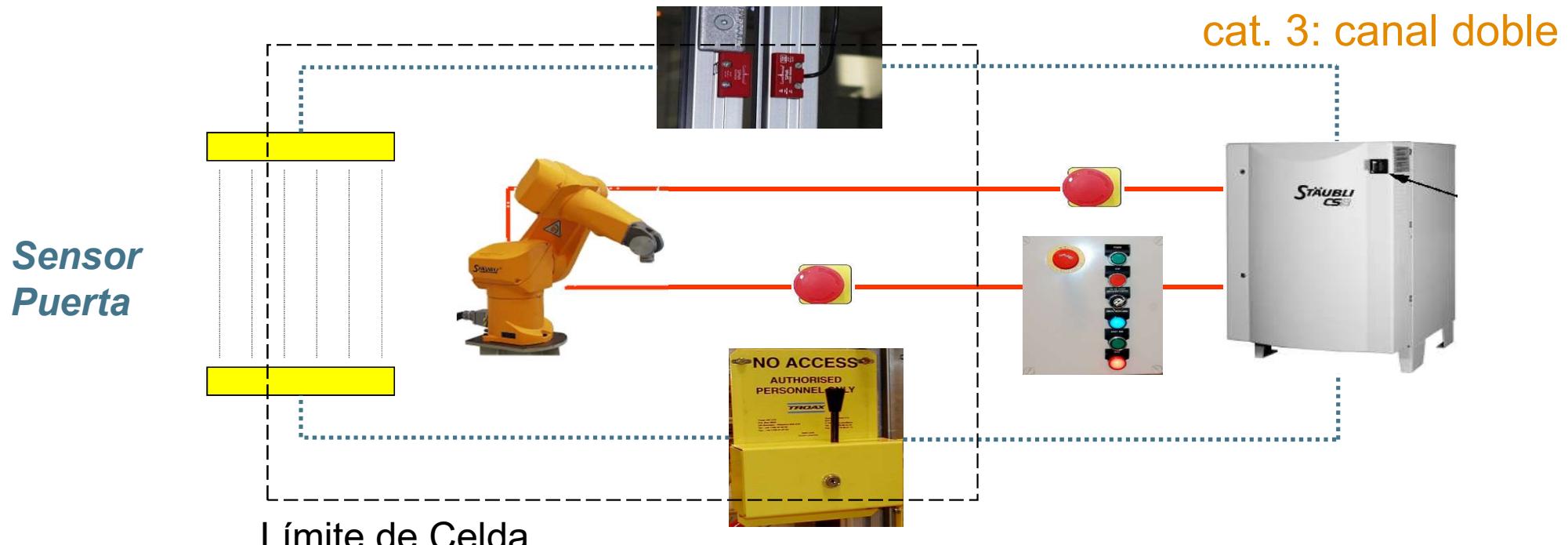


1	2	3	4	5	-
6	7	8	9	1	n



# CANALES DE EMERGENCIA

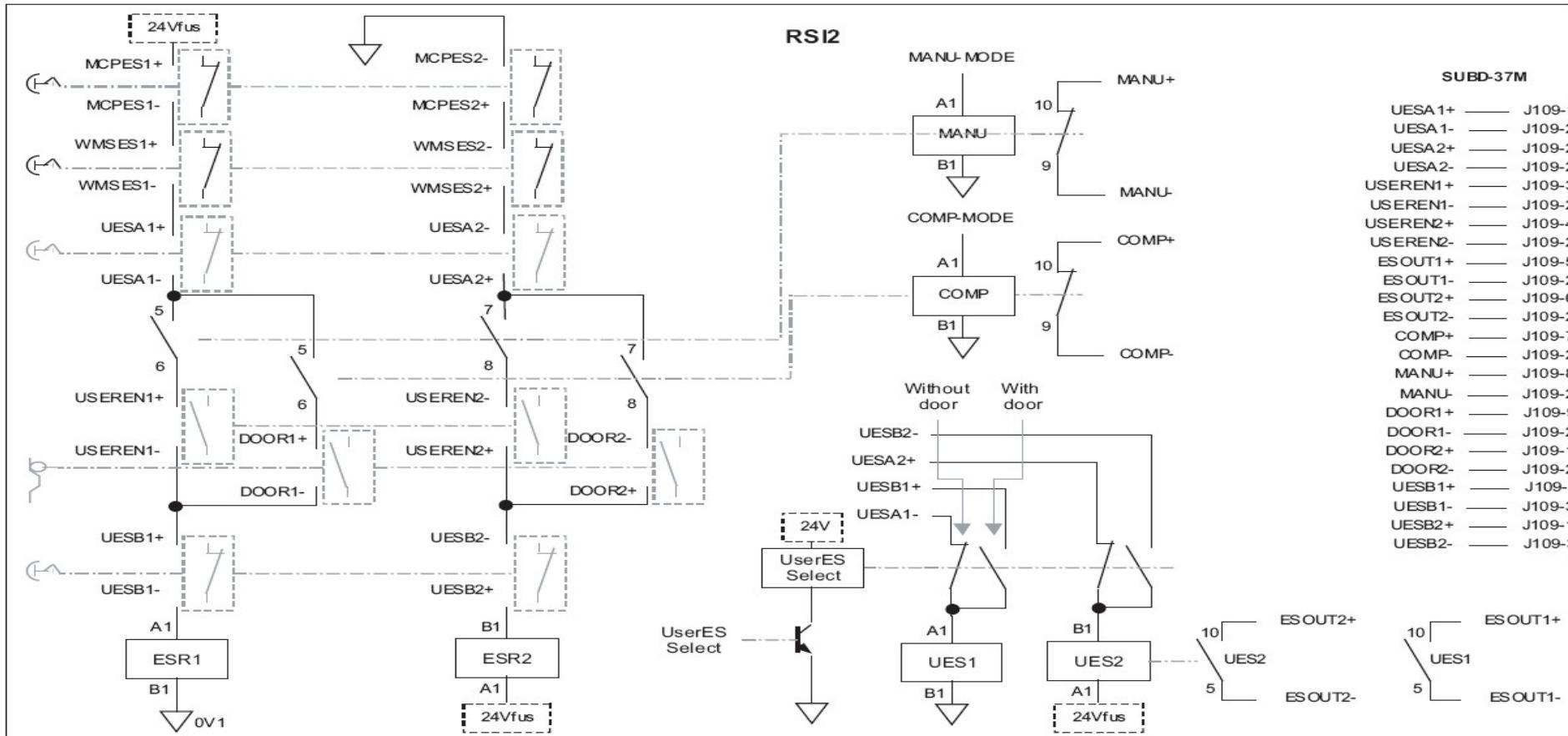
## Puerta 1



## Puerta 2

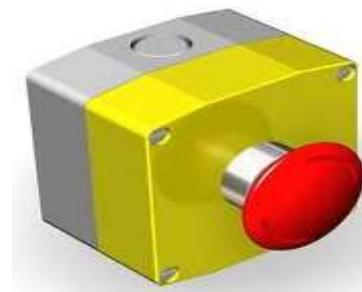
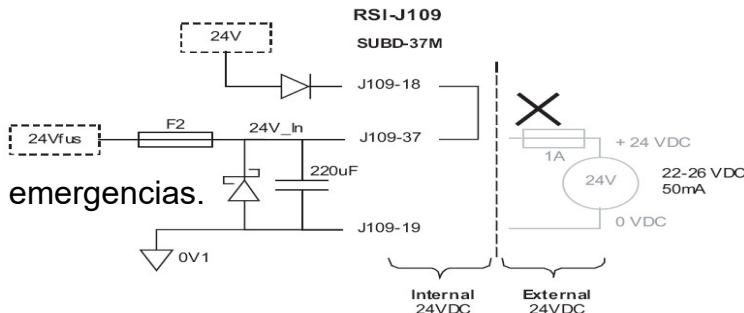
- Permanente (UESA / UESB)
- Solo actúa en modo automático (DOOR)

# CONFIGURACIÓN DE LAS EMERGENCIAS



## Contactos de la cadena de emergencia:

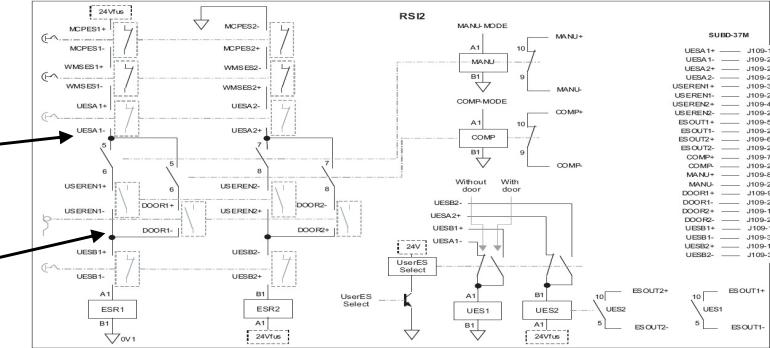
- MCPES Parada emergencia del MCP (mando).
- WMS Parada emergencia del Working Mode Selection Panel.
- UESA Primera parada emergencia de usuario conectada a la célula.
- UESB Segunda parada emergencia de usuario conectada a la célula.
- DOOR Contacto de las puertas, solo operativo en modo automático.
- USEREN Contacto „User Enable“, solo operativo en modo manual.
  
- MANU Sistema en modo manual.
- COMP Sistema en modo AUTOMATICO.
  
- ESOUT Contactos, libres de potencial, del estado de la cadena de emergencia para o hacia la célula.



El estado de la cadena de emergencia está disponible para la aplicación en ESOUT1 y ESOUT 2.

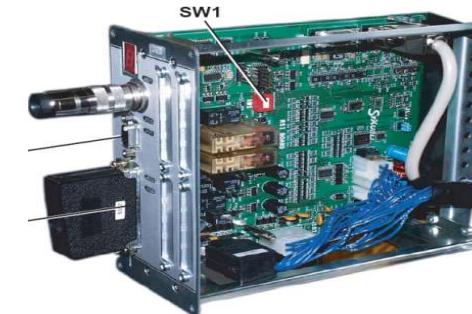
Dos configuraciones a considerar:

- MCPES, WMSES y UESA.
- MCPES, WMSES, UESA y USEREN o Door.



Configuración:

- Tarjeta RSI1: Selector SW1
- Tarjeta RSI2: Configuración por software en Panel de control del MCP del CS8C.



## FORMACIÓN VAL3

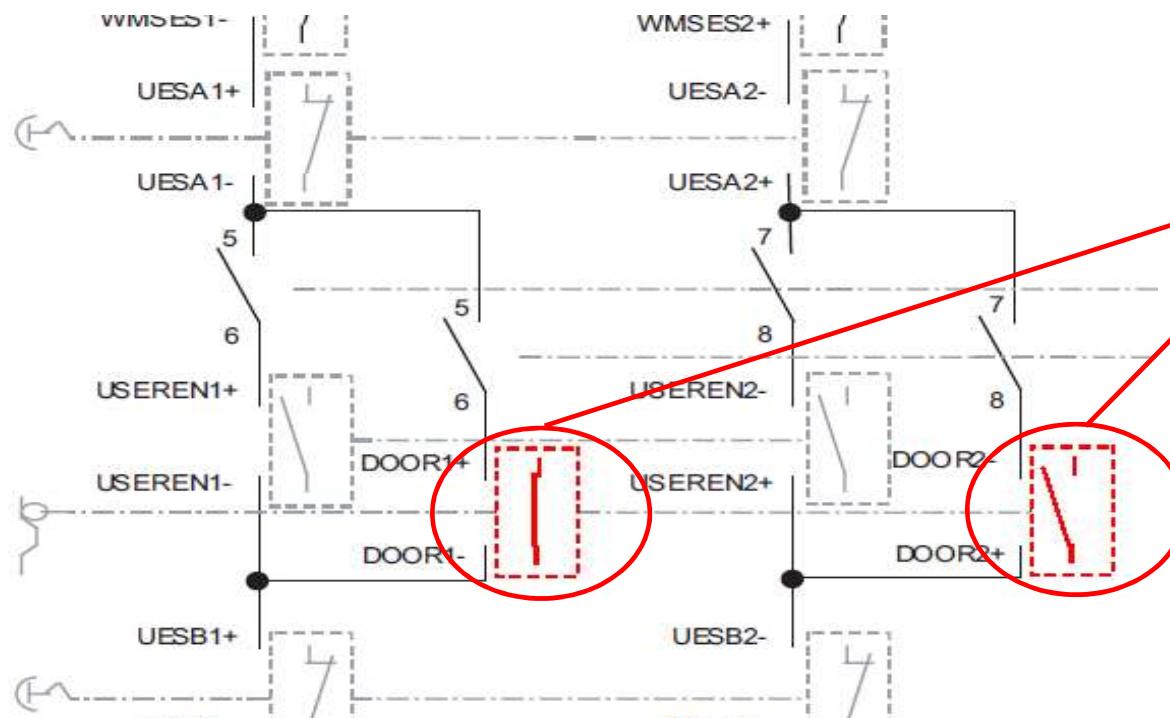


# *Error de Hardware*



# ERROR DE HARDWARE

Error detectado



Señal de error

Error: Solo uno de los dos contactos redundantes se ha abierto.

Control panel  
 +I/O  
 -Controller status  
**Hardware Safety Fault: DOOR2**  
 +I/O  
 Message  
**Emergency stop chain miss wired  
 for signal DOOR2**  
 +Drives Information  
 +Sensors status  
 RSI 7 Segment Display Unit : U  
 Arm time 0:00:00  
 Cabinet Temperature : 34°C 93.2°F  
 0k

VAL3 >= 6.8

## Paso 1: Localizar el fallo

```

■ Control panel 10%
+I/O
-Controller status
  Hardware Safety Fault: DOOR2
  Message
    Cannot settle arm power: fault on
    signal DOOR2 needs to be corrected
    and tested.

+Drives Information
+Sensors status
  RSI 7 Segment Display Unit : U
  Arm time 0:00:00
  Cabinet Temperature : 34°C 93.2°F
    Ok

```

Error:  
Señal de  
DOOR2

Tecla menú → Panel de Control →  
Estado del controlador

```

1 Control panel 10%
+I/O
-Controller status
  Hardware Safety Fault: DOOR2
  +I/O
  +Joint position
  +Joint velocity
  +Max manual joint velocity
  +Cartesian position
  +Drives Information
  +Sensors status
    RSI 7 Segment Display Unit : U
    Arm time 0:00:00
    Cabinet Temperature : 34°C 93.2°F
      Ok

```

Parpadeando.

VAL3 >= 6.8

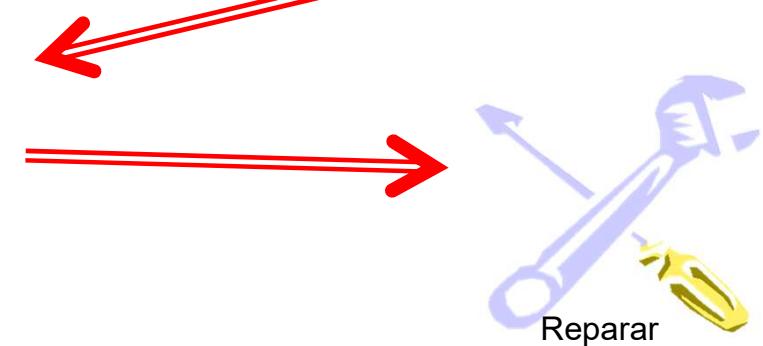
```

■ Control panel 10%
+I/O

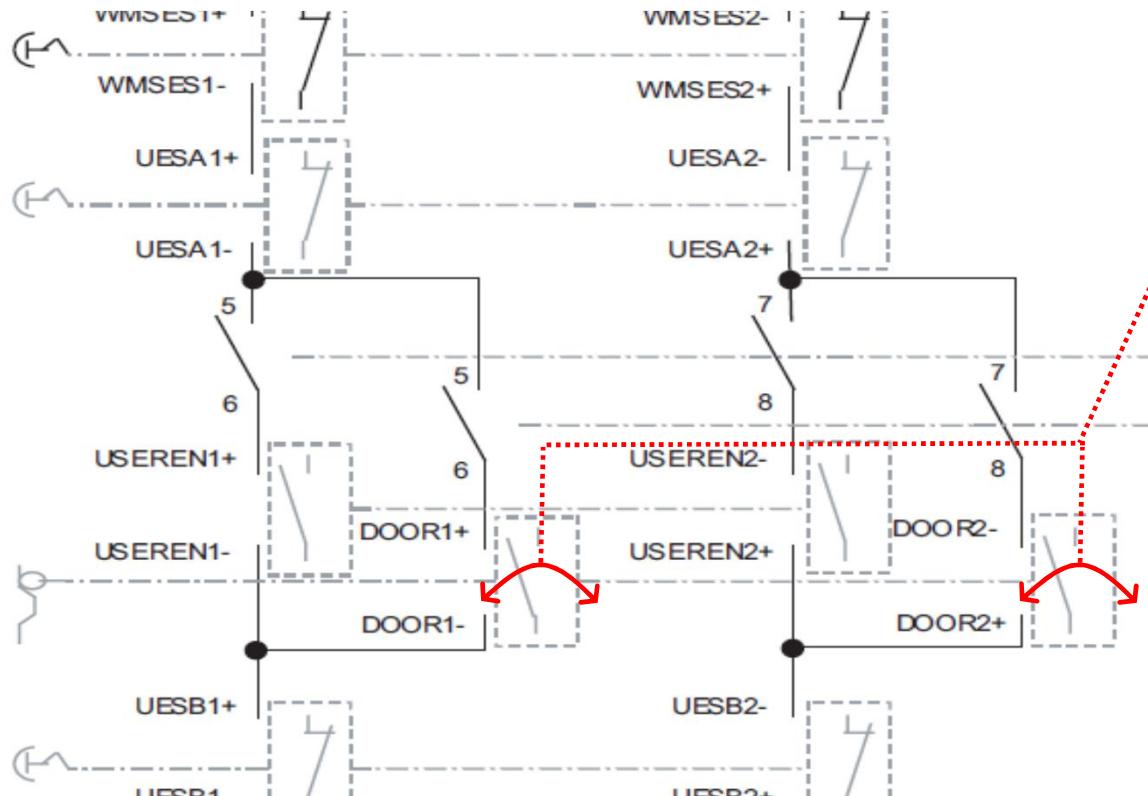
  A hardware fault was detected by
  the controller: robot safety level
  is reduced. Repair the faulty
  signal and active it to show that
  the hardware fault is solved. Then
  acknowledge the fault to resume
  robot operation

  RSI 7 Segment Display Unit : U
  Arm time 0:00:00
  Cabinet Temperature : 34°C 93.2°F
    Ok

```



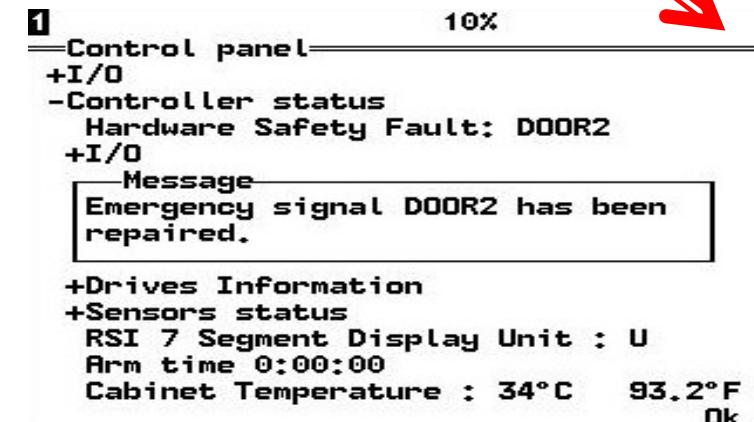
## Paso 2: Validar el error



$VAL3 \geq 6.8$

**Cuidado:** El error de Door se ha de reproducir en modo automático y el User en modo manual para que el sistema pueda verificar que se ha reparado.

Reproducir la situación para que el sistema verifique que se ha resuelto el problema. Los dos contactos redundantes se abren simultáneamente.



## Paso 3: Confirmar el fallo solucionado

```

1                                     10%
=Control panel
+I/O
-Controller status
  Hardware Safety Fault: DOOR2
+I/O
+Joint position
+Joint velocity
+Max manual joint velocity
+Cartesian position
+Drives Information
+Sensors status
  RSI 7 Segment Display Unit : U
  Arm time 0:00:00
  Cabinet Temperature : 34°C   93.2°F
    Ack.

```

Nota: Al quitar el MCP y colocar el puente del MCP en modo automático o remoto no genera un error de Hardware.



```

1                                     10%
=Control panel
+I/O
-Controller status
  Hardware Safety Fault: No fault
+I/O
+Joint position
+Joint velocity
+Max manual joint velocity
+Cartesian position
+Drives Information
+Sensors status
  RSI 7 Segment Display Unit : U
  Arm time 0:00:00
  Cabinet Temperature : 34°C   93.2°F

```

También es posible validar el fallo:

- Presione simultáneamente las teclas A, C, y K en la consola de programación (sin necesidad de navegar por el panel de control)
- Entrada digital asignada en iomap.cf

VAL3 >= 6.8

## FORMACIÓN VAL3

# Primeros pasos



# PUESTA EN MARCHA DEL ROBOT

STÄUBLI

1

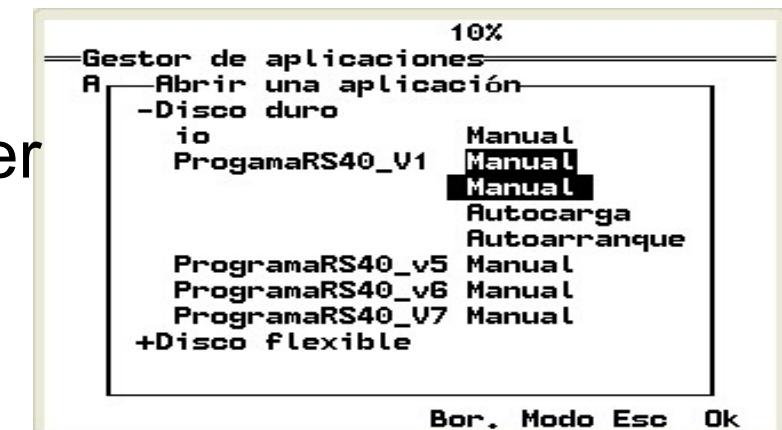
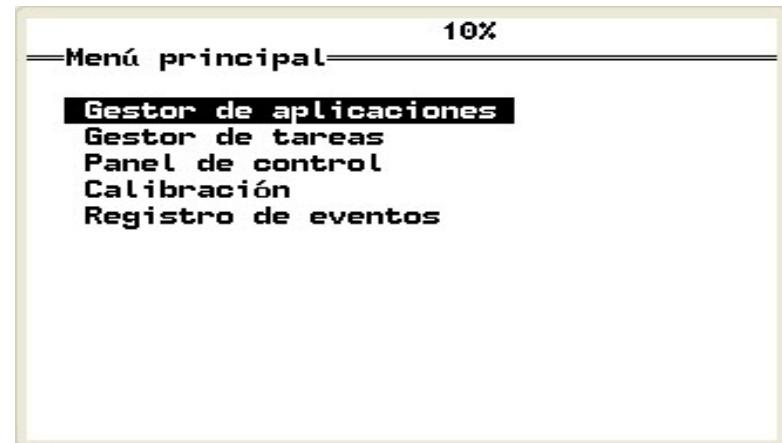


2

Esperar al menú principal.

3

Una aplicación puede ser cargada o ejecutada automáticamente.



## PARO DEL CONTROLADOR

1

Desactivar la potencia  
del brazo.

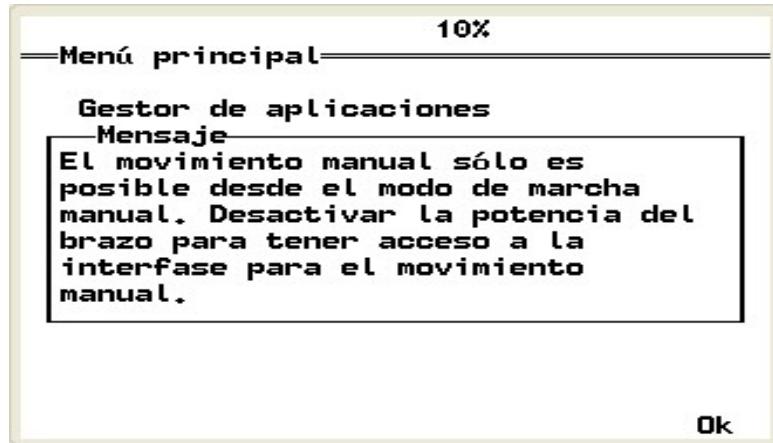


2

Desconectar el interruptor principal



# VALIDAR ERRORES



La validación es necesaria para eliminar el error de la pantalla.

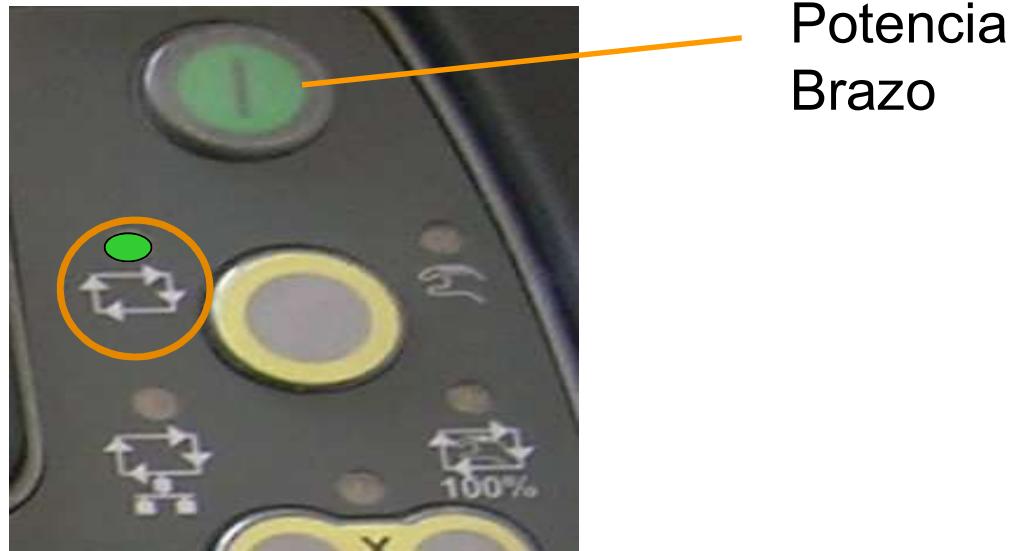
**El sistema no espera la validación del mensaje:** Si el error no está relacionado con la ejecución de la aplicación, el programa prosigue sin esperar la validación del mismo.

Esto puede ocasionar que un error que detenga la ejecución quede enmascarado por otro error.

El error queda en pantalla hasta que se valida y los posibles errores que ocurran posteriormente, no aparecerán una vez validado el error, tendremos que consultar el log para poderlos visualizar.

## ACTIVACIÓN DE LA POTENCIA EN LOCAL

- 1 Cerrar la puertas de la célula (Cadena DOOR).
- 2 Pulsar Power ON (Potencia Brazo).



## ACTIVACIÓN DE LA POTENCIA EN MANUAL

### Célula abierta

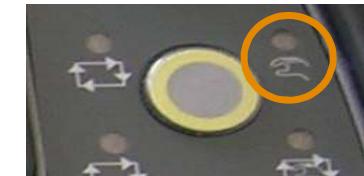
1 MCP en el soporte



2 Sacar y colocar el MCP en el soporte en menos de 15 segundos.

MCP en las manos

Soltar y apretar el hombre muerto en menos de 15 segundos.



Único modo con Val 3:  
= 6.9  
=> 7.3

3 Pulsar el Power ON (Potencia Brazo).

# PROCEDIMIENTO DE PARADA DE EMERGENCIA

STÄUBLI

## Situaciones de parada de emergencia



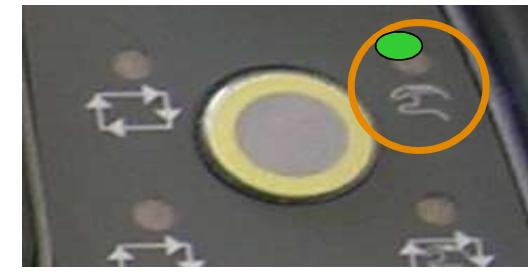
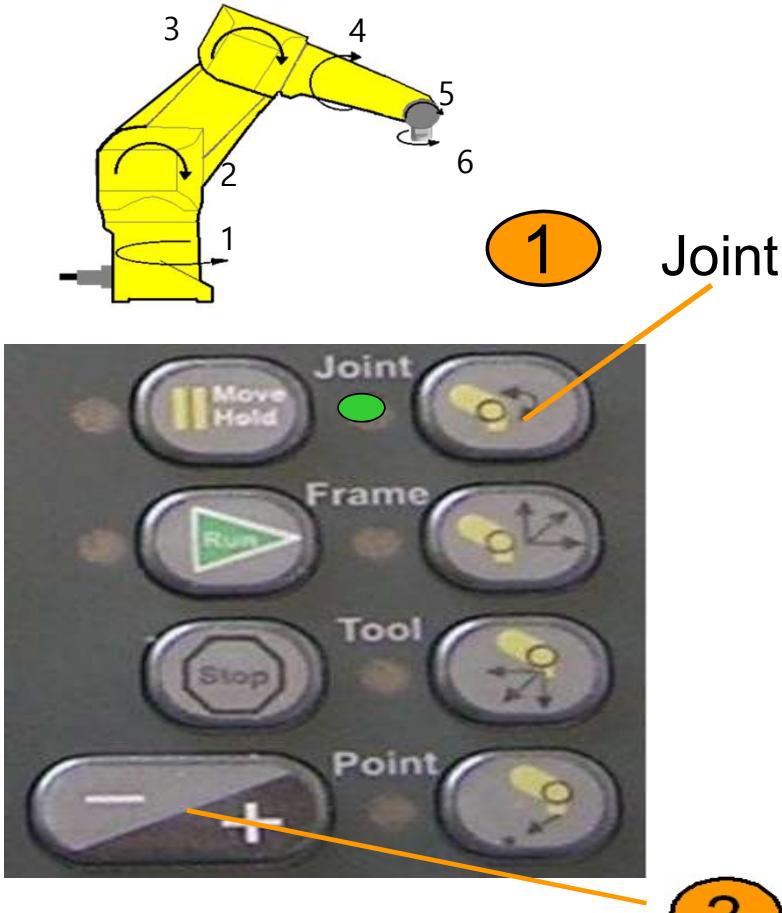
- Emergencia pulsador en el MCP.
- Emergencia pulsador en el WMS.
- Emergencia de la Célula.
- Selector de frenos fuera de 0.
- Contacto de final de límite de un eje.



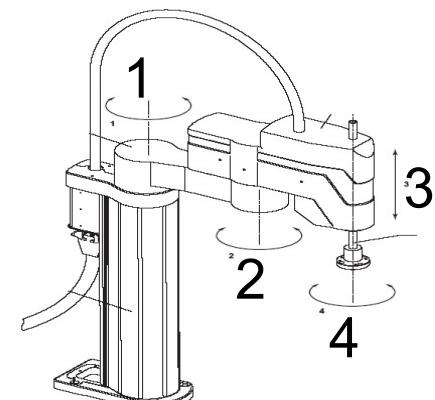
*En modo manual y después de una parada de emergencia , es necesario de colocar el MCP en su base para poder activar la potencia del brazo.*

*El soporte (Base del MCP) ha de estar colocado siempre fuera de la célula.*

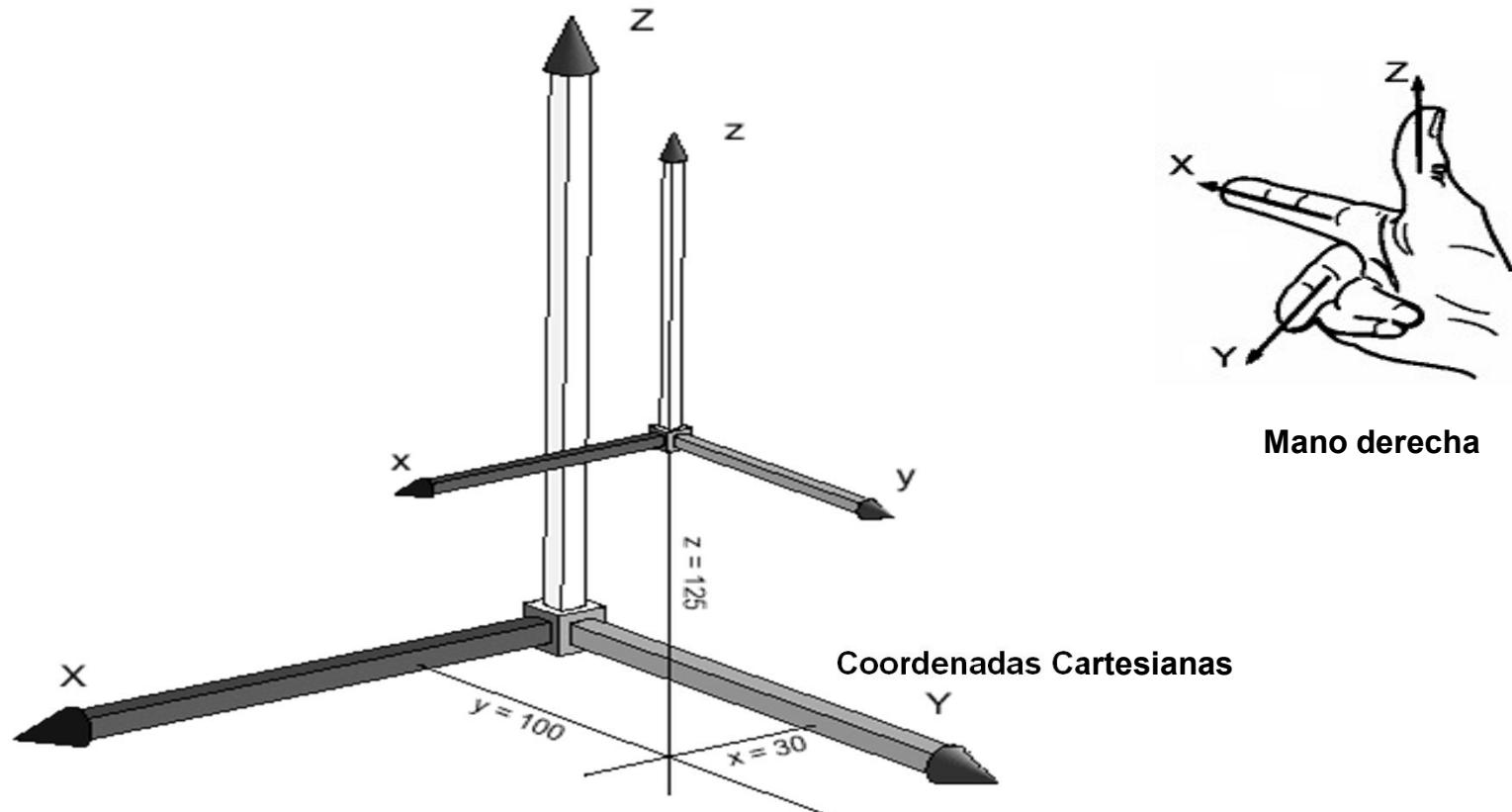
## MOVIMIENTO MANUAL JOINT



¡ Necesario!



# SISTEMA DE COORDENADAS



**Tres ejes perpendiculares entre ellos**

## MOVIMIENTO MANUAL CARTESIANO



1

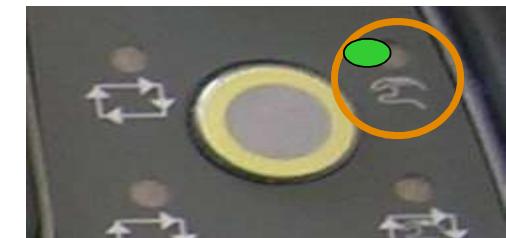
- Seleccionar el modo :
- Frame
  - Tool

2

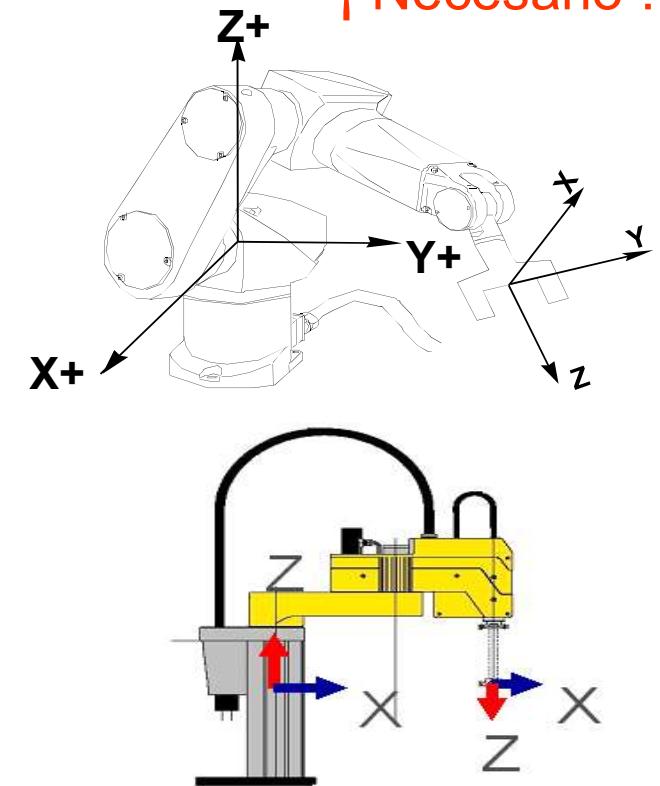
Ajustar la velocidad,



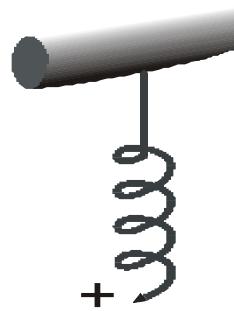
3



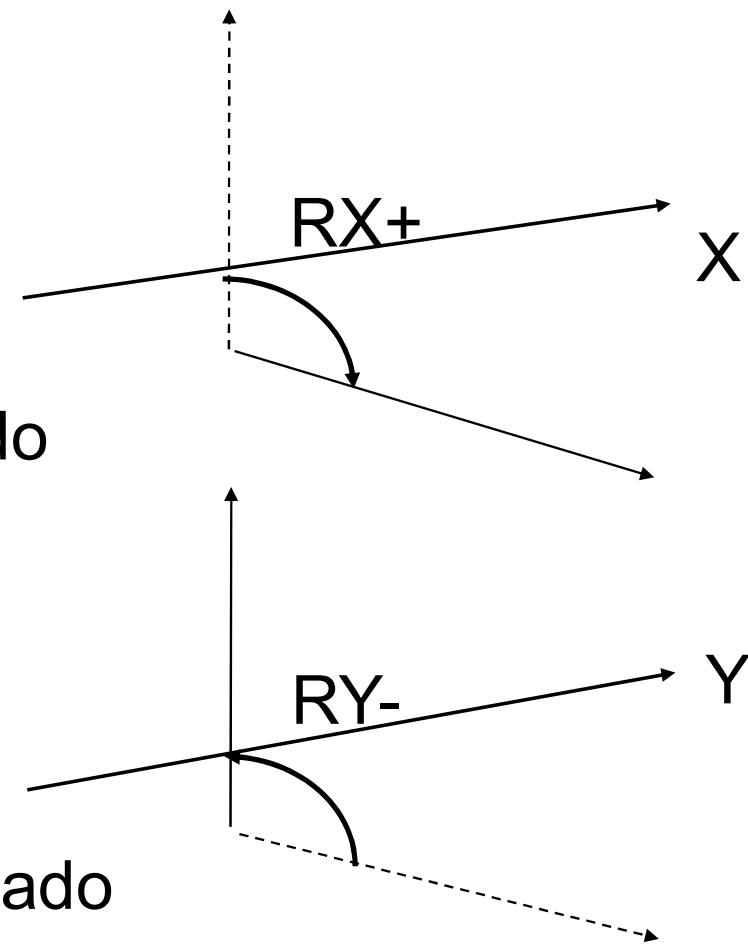
¡ Necesario !



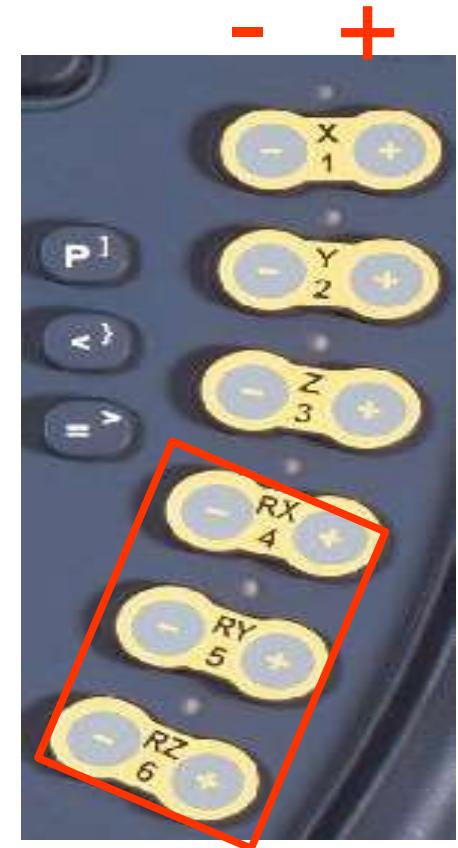
## Rotaciones



Atornillado

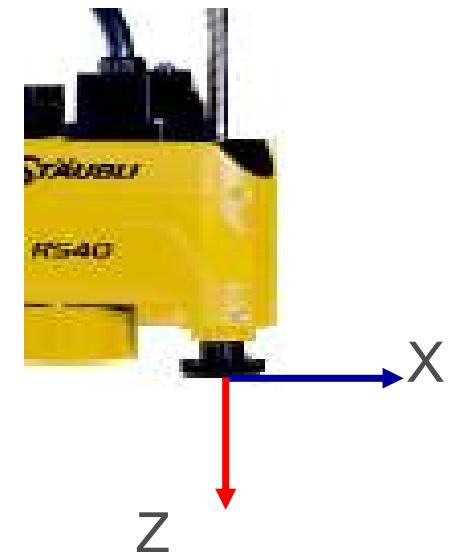
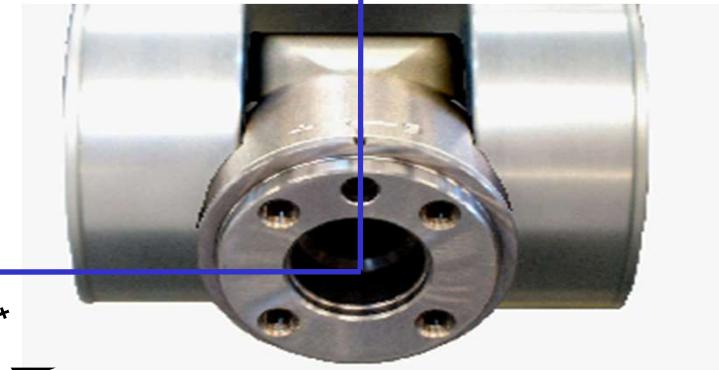
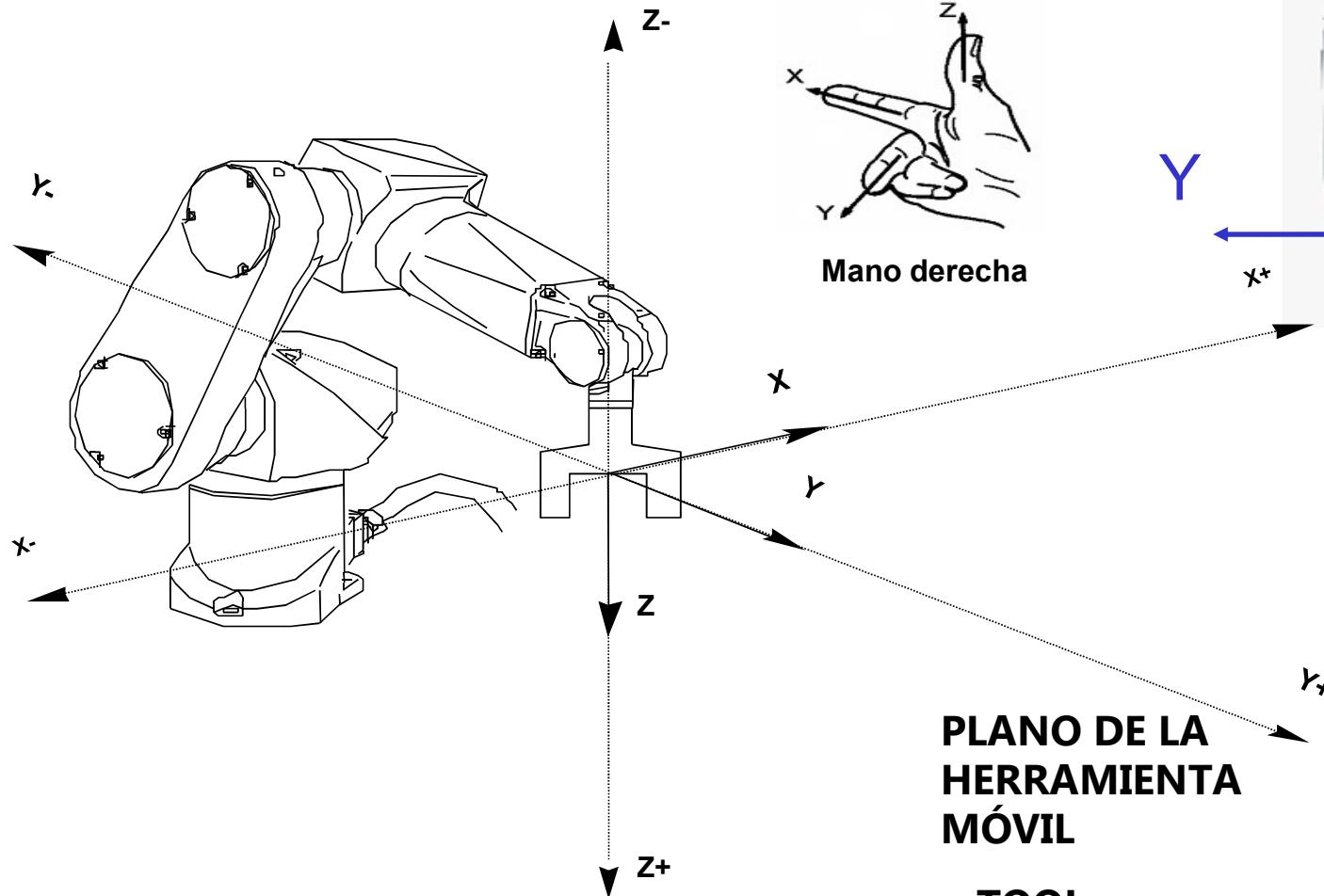


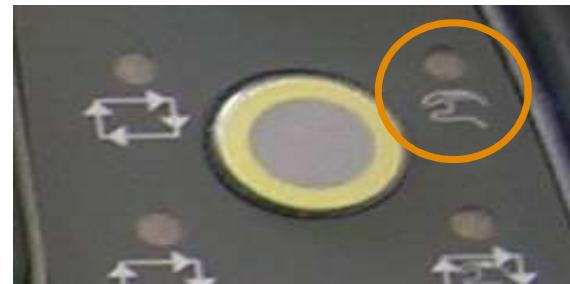
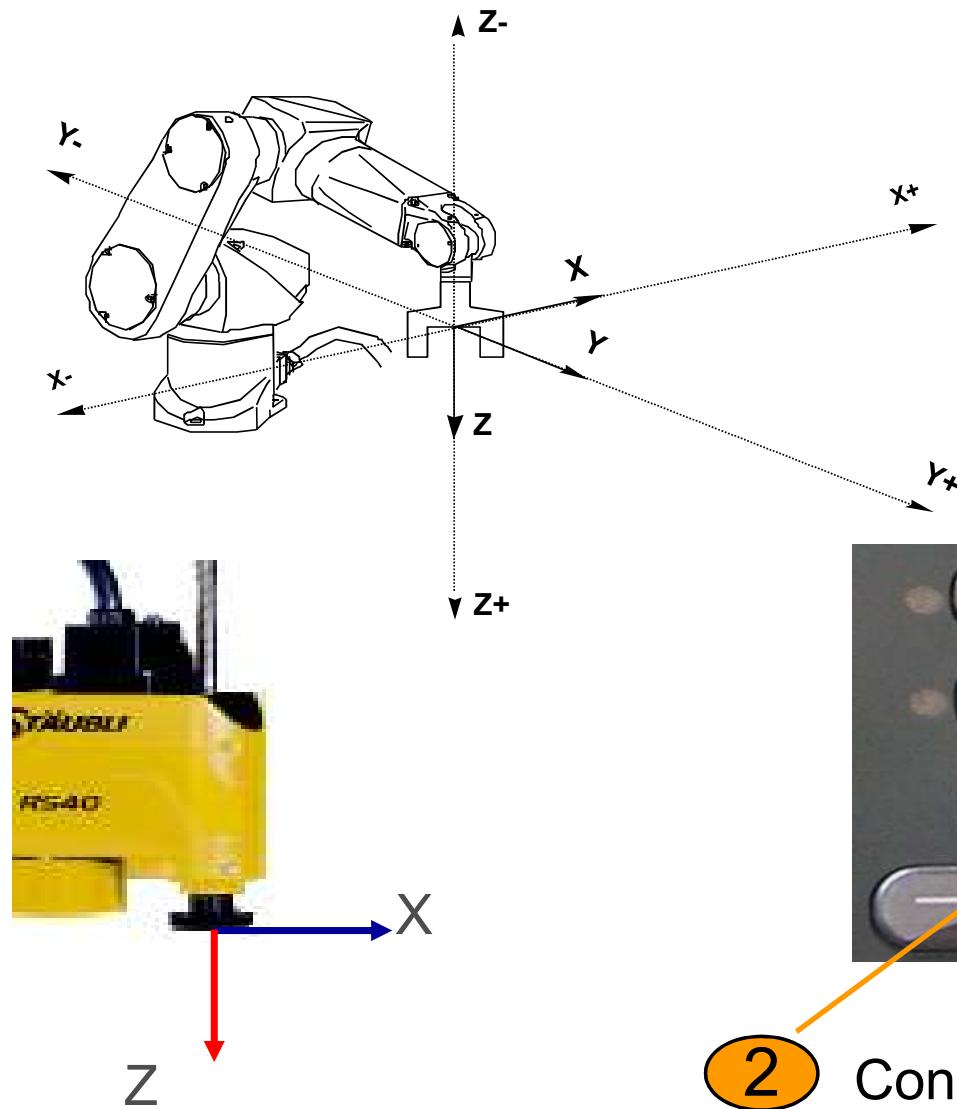
Desatornillado



Rotación

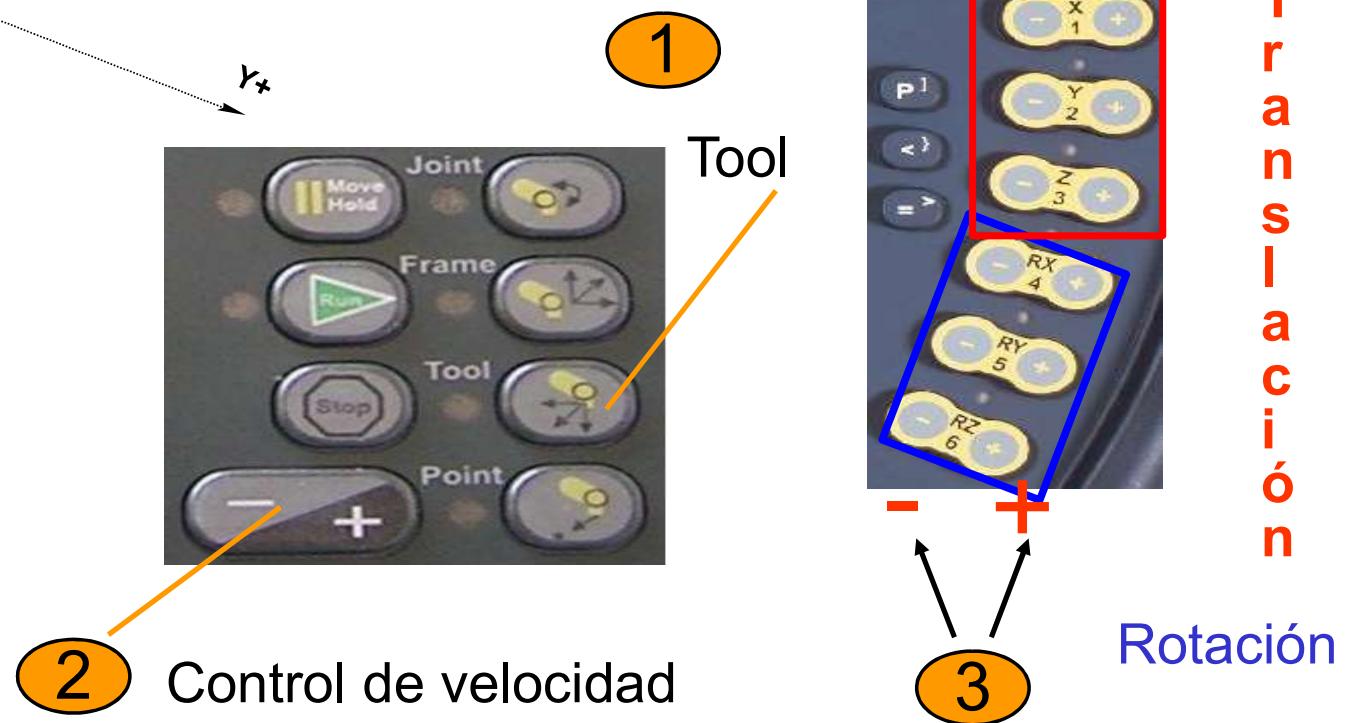
## MOVIMIENTO MANUAL TOOL





¡ Obligatorio !

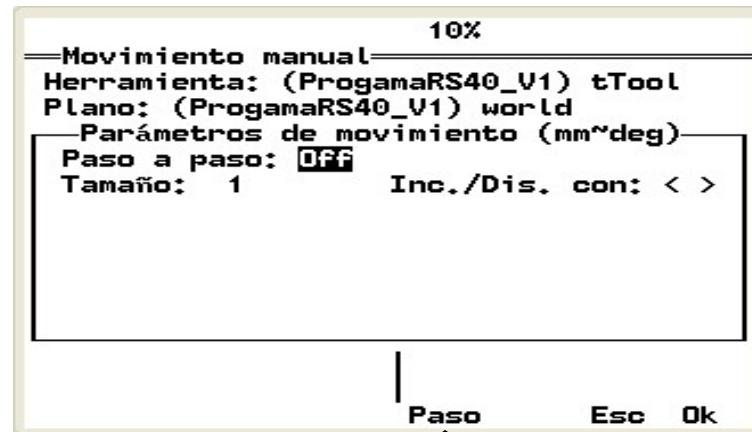
STÄUBLI



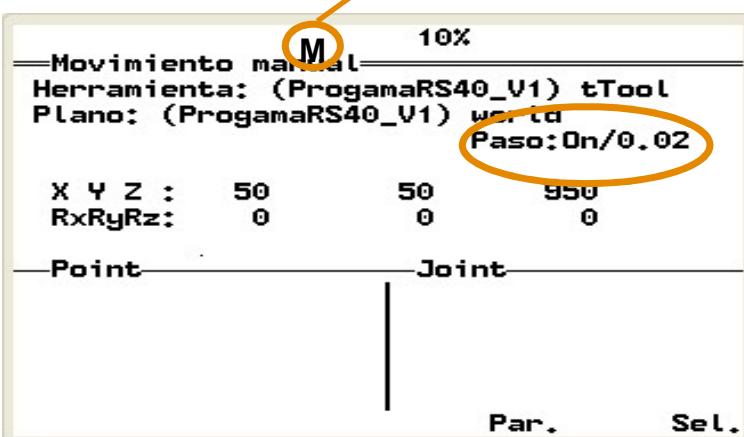
# MOVIMIENTO A PASOS

STÄUBLI

VAL3>=6.5



Durante el desplazamiento



El modo paso, permite mover el robot en incrementos (0.01 a 50 mm/deg).



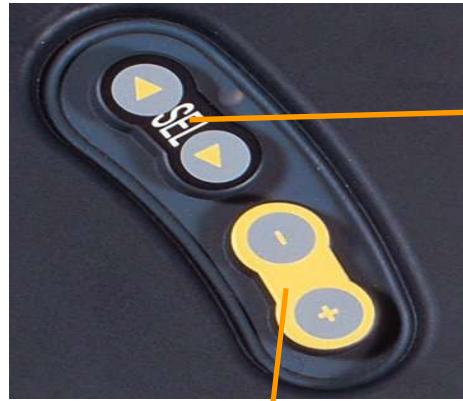
1 pulsación + hold mueve el robot el valor indicado.

X pulsaciones +hold mueve el robot X veces valor indicado

< >: Para incrementar o decrementar el paso.



## MINI JOG

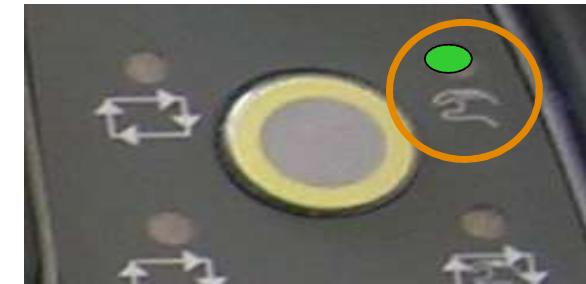


Control de movimiento



Ajuste de velocidad.

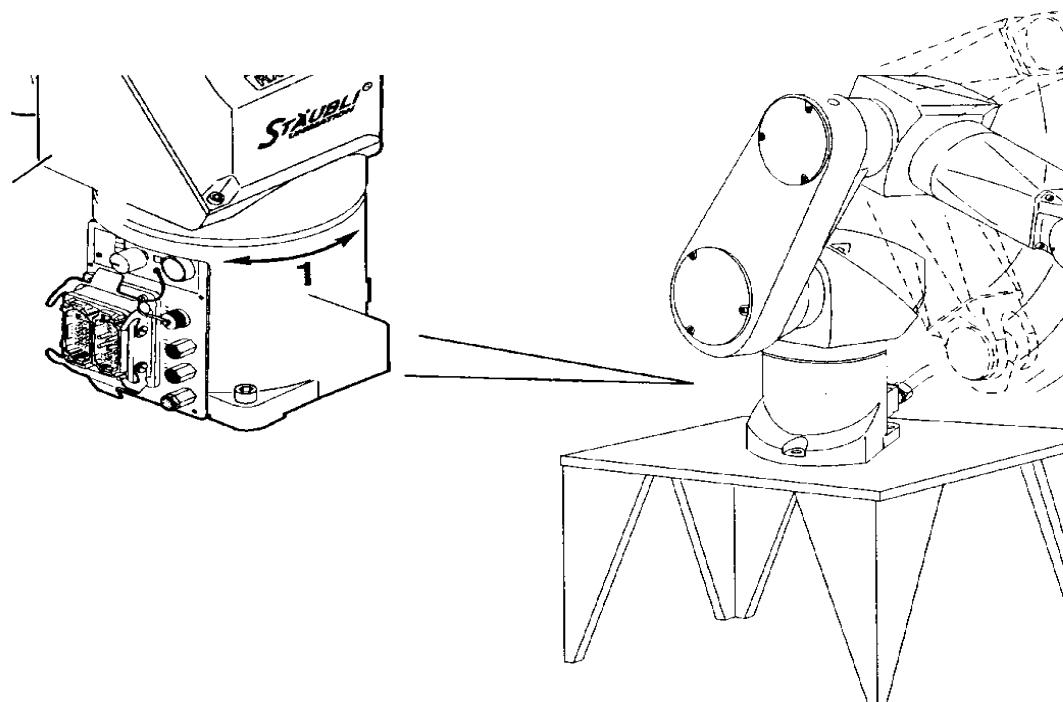
Selección del eje uno a uno o directamente el eje.



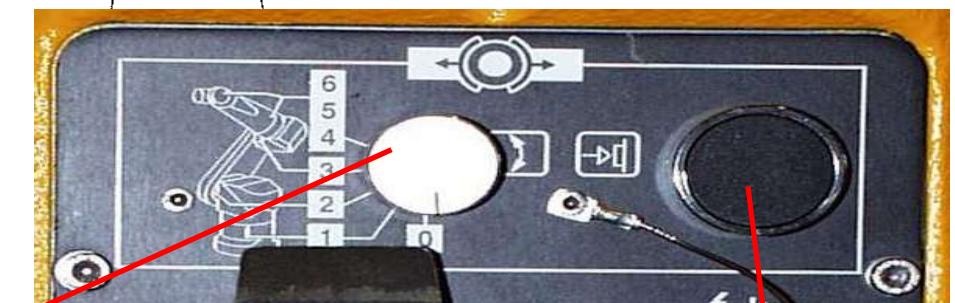
! Necesario!



## LIBERACIÓN DE FRENOS



Selector de frenos

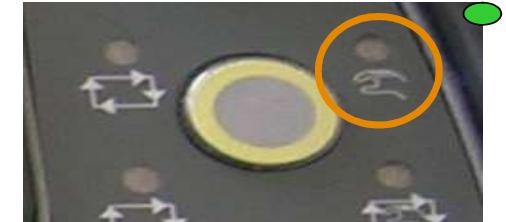


Liberador

*Si el selector  $\neq 0 \Rightarrow$  no es posible dar potencia al brazo.*

# CONTROL MANUAL DE PINZAS

STÄUBLI



¡ Obligatorio !



Libre

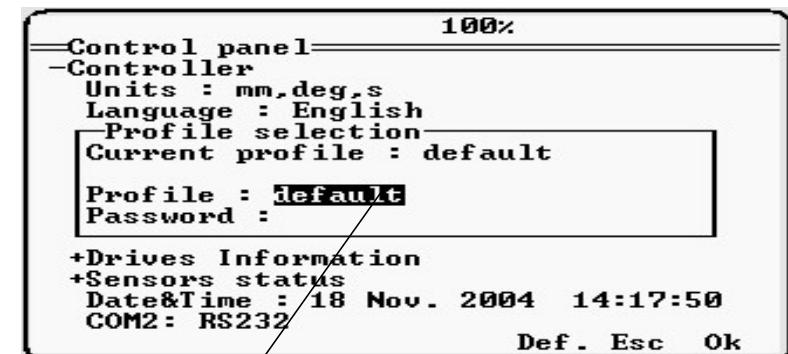
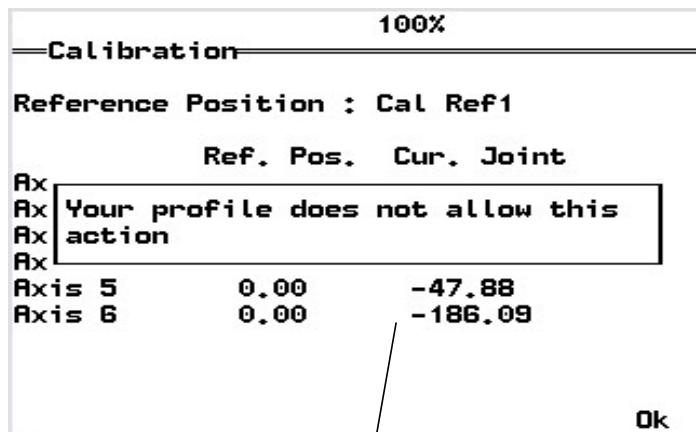
Pinza 2 (Electroválvula 2)

Pinza 1 (Electroválvula 1)

Las tres señales pueden ser mapeadas a otras salidas del controlador.

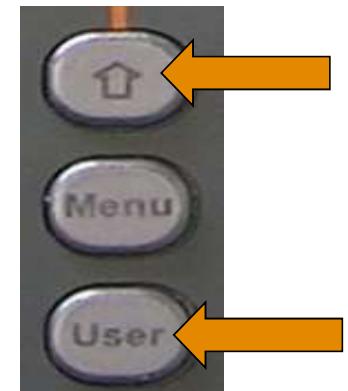
El perfil default no permite utilizar los botones 1, 2 y 3.

# SELECTOR DE USUARIO / PERFIL

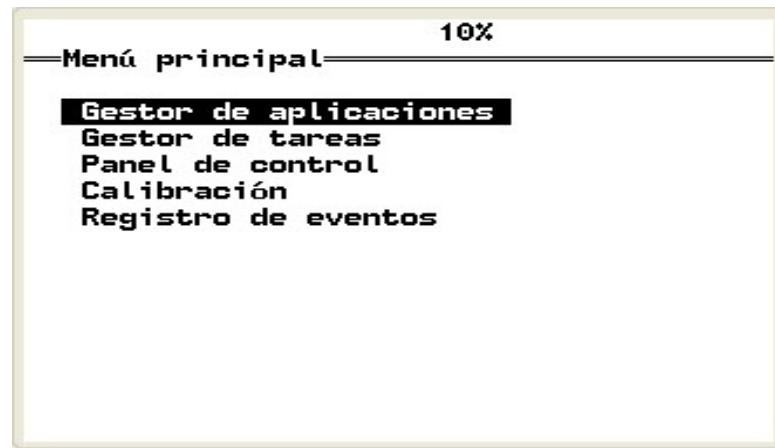


Si el perfil actual no le permite hacer una acción:

- Pulsar « SHIFT » + « User » para cambiar de perfil:
- Introducir perfil (usuario) y password , el perfil default por defecto no tiene password.



# MENU PRINCIPAL



- Gestor de aplicaciones : Ambiente de programación.
- Gestor de tareas : Depuración de aplicaciones en ejecución.
- Panel de control : Información de sistema, prueba de E/S.
- Calibración: Calibración y recuperación de la calibración del brazo.
- Registro de eventos : Historial de los últimos 250 eventos después del arranque del controlador.

# ÁRBOL DE NAVEGACIÓN

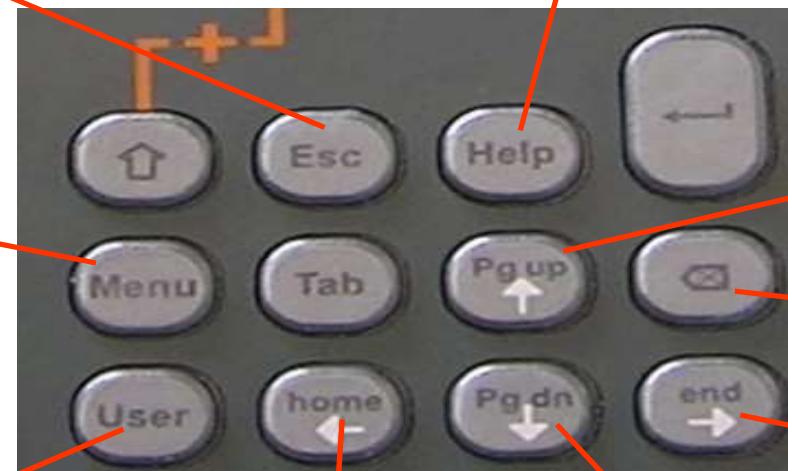
STÄUBLI

Anular la operación actual

Ayuda en línea

Menú Principal

Pantalla de usuario



Arriba

Retroceder

Expansión : Línea con +

Abajo

Contracción : Línea con -

## PANEL DE CONTROL

- Tecla



- Tecla



para navegar.

- Seleccionar « Panel de control » con



- Tecla



para expandir « + ».

- Tecla



para contraer « - ».

- Utilizar los menús en función de la opciones.

## PANEL DE CONTROL - IO -

- Tecla



- Seleccionar «Panel de control » con



- Tecla



en la línea E/S y controlador.

- Desplazarse hasta BasicIO o ModuleIO con



- Tecla



en la línea deseada.

- Visualizar los datos.

- Modificar en valor de las salidas con la tecla F6 (ON / OFF).

## REGISTRO DE EVENTOS

Cuando « Info » parpadea :

Algún mensaje de información no ha sido visualizado debido a que otro mensaje anterior se estaba mostrando.

Ir a Menú → Registro de eventos para visualizar los últimos 250 mensajes después del arranque del controlador.

Botón **Exp** permite exportar una copia del **ERRORS.LOG** en una llave USB (máxima capacidad del log 1Mb).

El fichero exportado se puede editar en un PC (Wordpad / TextPad).

58 Mensajes con fecha y hora.



---System-Versions-Begin-----

```
CPU board: EM07N 400Mhz; 64Mb RAM; 112Mb disk; bios: 1.22; fpga: D5; atalde: Rev:1.5d
Cycle time set to 0.004s
RSI Board rev: 2 - Driver ver: 2
Dirección Mac = 00-c0-3a-60-3b-9b
BoardId:0x0531='STARC2' - BoardRev:1 - FpgaRev:0x0205='2.5'
120210172
Power On Board counter=97
STARC_Version Pack: 1.19.2 - Date:Jan 26 2012
STARC_Version Board:DVC:1329 - BOARD:1 - FPGA:2.5 - DSP1:3.15 - DSP2:3.15
STARC-Version Dsi: DVC:756 - MOTH:0 - DAUG:0 - FPGAR:3.0 - FPAGAF:3.1 - DSPr:4.0 - DSPF:4.6
STARC-Version Encoder1: IdNb=524536-04 ; SerialNb= 36248892B
STARC-Version Encoder2: IdNb=524536-04 ; SerialNb= 36099961B
STARC-Version Encoder3: IdNb=524536-04 ; SerialNb= 36099979B
STARC-Version Encoder4: IdNb=524536-04 ; SerialNb= 36249453B
STARC-Version Drv0: DP:32 - DF:151 - FP:1 - FF:10 - HD:1 - HP0:2 - HP1:2 - LIB:22
STARC-Version Drv1: DP:32 - DF:151 - FP:1 - FF:10 - HD:1 - HP0:2 - HP1:2 - LIB:22
MCP firmware version: D24275105A
VAL3: s6.9 Dec 2 2011
Option testMode: enabled
Option VAL 3: enabled
Configuración: D24266417A
N/S del controlador: : F12_XXXXXX_C_01
Brazo: ts60-S1-D25-L400-floor-R1 suelo
N/S del brazo:: F12_XXXXXX_A_01
Starc: 1.19.2 Jan 26 2012
EP: s6.9 Nov 29 2011
SCA: s6.9 Nov 29 2011
PLC: s6.9 Nov 29 2011
---System-Versions-End-----
```

[AUG 30 2006 20:59:54]:

```
--System-Diagnostic-Begin-----
ARPS = OK
RSI board = OK
Starc board = OK
Bus between Starc board and DSI board = OK
Bus between Starc board and drives = OK
DSI board = OK
Encoder1 = OK
Encoder2 = OK
Encoder3 = OK
Encoder4 = OK
Drive1 = disabled
Drive2 = disabled
Drive3 = disabled
Drive4 = disabled
Mcp = connected
RSI display = i
---System-Diagnostic-End-----
```

Dirección Mac

Firmware

Versión Val 3

Número de serie

Tipo de robot

Diagnóstico

# BACKUP CON EL SP1

- Tecla



- Seleccionar «Panel de control» con

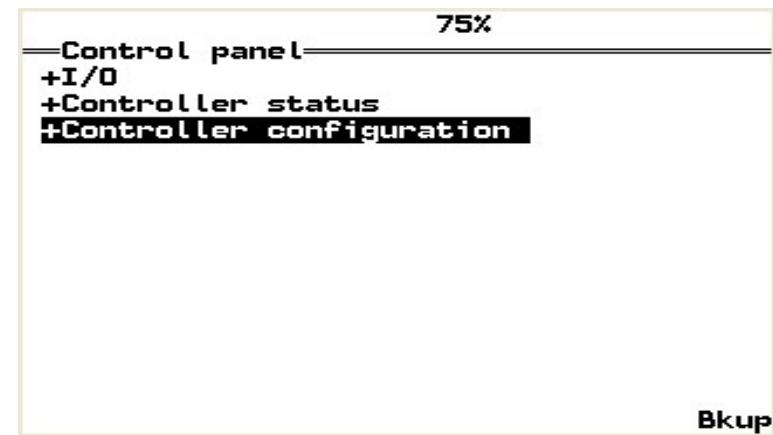


- Tecla en “Configuración del controlador”



- Tecla F8 : Respaldo

Verificar fecha y hora: Tecla menú → panel de control → configuración del controlador → Fecha y hora



FORMACIÓN VAL3

# *Calibracion Robot*

**RX 60-90-130-170**



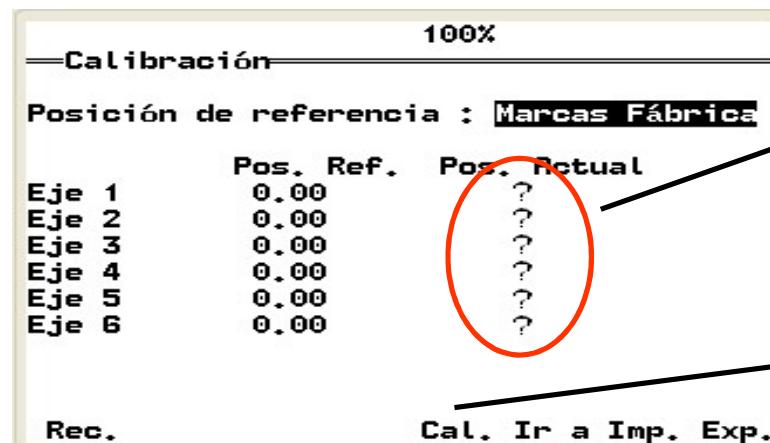
## CALIBRACIÓN

En un CS8 después de :

Desconexión de la batería del sistema de calibrado (DAPS).

Desconexión cable brazo controlador.

Procedimiento de recuperación de la calibración manualmente:



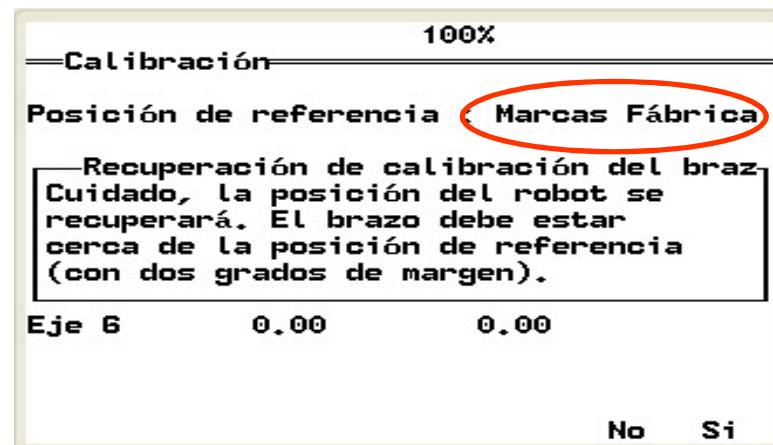
Posición  
indeterminada. ?

Recuperar  
Calibración

Mover el brazo cerca de la referencia escogida  
Ej. Marcas de fábrica

Pulsar Cal

Después de la calibración, la posición actual puede estar muy cercana a la posición de referencia (depende de la precisión de la alineación de los ejes).



1%		
Calibración		
Posición de referencia : Marcas Fábrica		
Eje 1	Pos. Ref.	Pos. Actual
0.00	0.00	0.06
0.00	0.00	0.12
0.00	0.00	0.04
0.00	0.00	0.51
0.00	0.00	0.42
0.00	0.00	0.13
<b>Rec.</b>		<b>Cal. Ir a Imp. Exp.</b>

## REFERENCIAS DEL BRAZO Y LA CÉLULA

Marcas Fábrica : Alineación de marcas STÄUBLI

Cero : Todas las articulaciones a 0

Marcas Usuario : Otras marcas en el brazo

Célula Ref1 : 1<sup>a</sup> Referencia en la célula

Célula Ref2 : 2<sup>a</sup> Referencia en la célula



A la salida de fabricación :

RX90B & RX130B    Marcas Fábrica = Cero (marcas ajustables)

RX60B y RX170B    Marcas Fábrica < > Cero (medias lunas))

# VERIFICACIÓN DE LA CALIBRACIÓN

STÄUBLI

Después de desmontar el brazo, colisión, .....

Si existe una duda de la geometría del brazo, es posible llevar el brazo a una de las posiciones de referencia para verificar si las marcas están alineadas.

Calibración		
Posición de referencia : Marcas Fábrica		
	Pos. Ref.	Pos. Actual
Eje 1	0.00	0.06
Eje 2	0.00	0.12
Eje 3	0.00	0.04
Eje 4	0.00	0.51
Eje 5	0.00	0.42
Eje 6	0.00	0.13

↑      ↑

Rec.                            Cal. Ir a Imp. Exp.

- Menú Calibración
- Modo Manual/ con potencia
- Seleccionar el tipo de referencia
- Botón **Ir a**
- Botón Move/Hold

+ Es posible ajustar la velocidad del movimiento

Posibilidad de Importar/ Exportar a un disco flexible o a una llave USB los datos de calibración del brazo

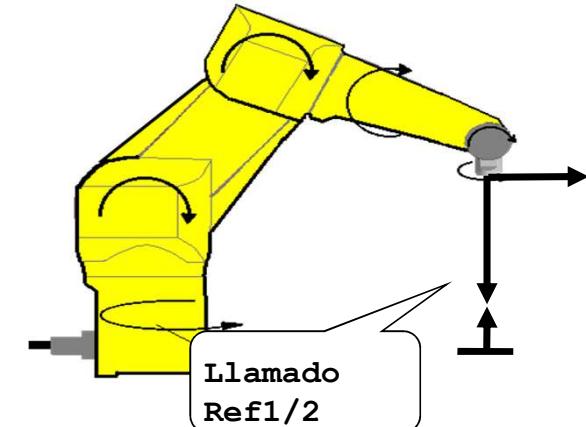
# COMO DETECTAR UN PROBLEMA DE COLISIÓN

1

- Tomando joint de referencia en la célula:  
 Puntero contrapunta  
 → Identificar problema de geometría del brazo.

Calibration		
Reference Position : Cal Ref1		
	Ref. Pos.	Cur. Joint
Axis 1	0.00	0.00
Axis 2	0.00	0.00
Axis 3	0.00	0.00
Axis 4	0.00	0.00
Axis 5	0.00	0.00
Axis 6	0.00	0.00

Adj.      Here Edit Rec. Move Imp. Exp.



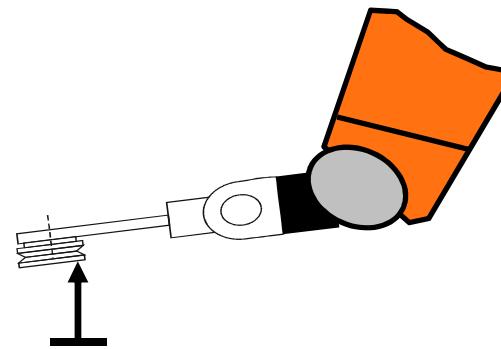
2

- Punto cartesiano tomado con la pinza:  
 (En la aplicación)  
 Identificar problema de geometría del brazo.

Después de

1

2



Si problema de posición, los puntos de la aplicación se han movido.

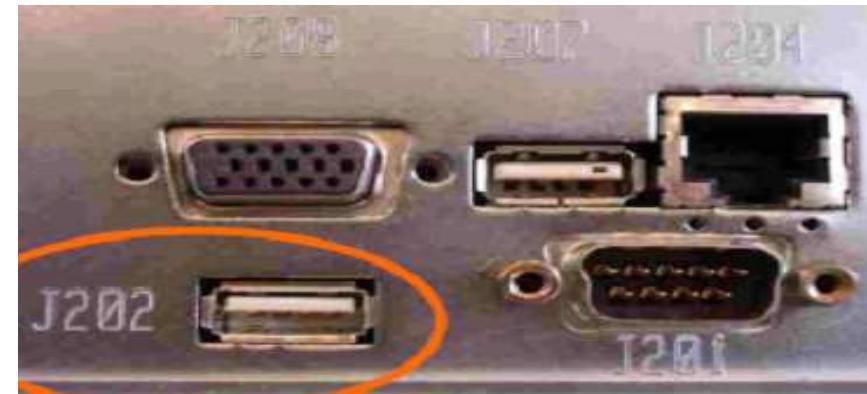
FORMACIÓN VAL3

# *Control de una Aplicación*



## CARGAR UNA APLICACIÓN

STÄUBLI



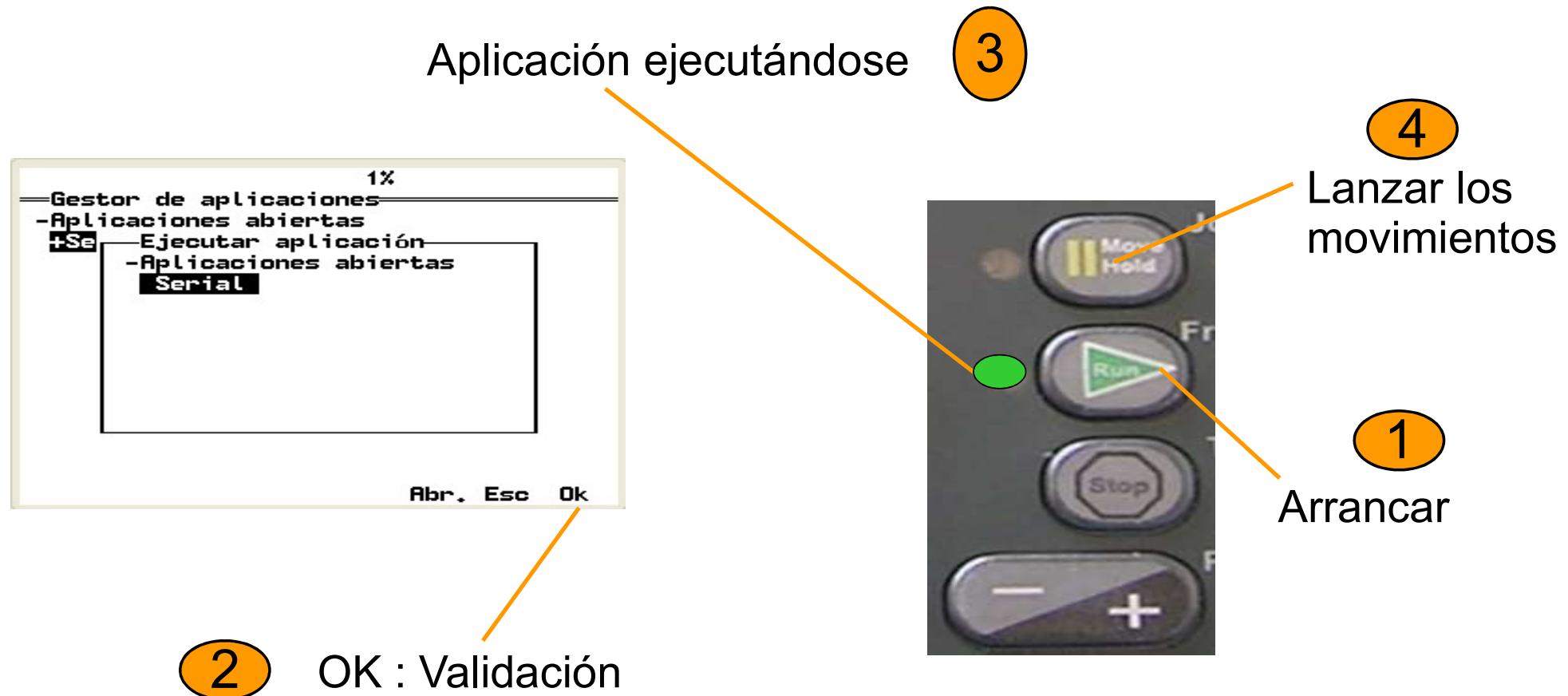
- Botón Menú
- Seleccionar el «Gestor de aplicaciones» con
- Botón « Abr. »
- Botón en la línea Disco duro o USB
- Ir a la línea de la aplicación con o
- Validar con « OK »

## TIPOS DE ARRANQUE PARA LAS APLICACIONES

- Una aplicación puede ser abierta de los siguientes modos:
  - Manual
  - AutoCarga: La aplicación es cargada en memoria ram.
  - AutoArranque : La aplicación es cargada en memoria ram y ejecutada.
- Cambio de modo con la tecla Mode
- AutoArranque : Sin MCP la aplicación tiene que estar en este modo.



## ARRANCAR UNA APLICACIÓN

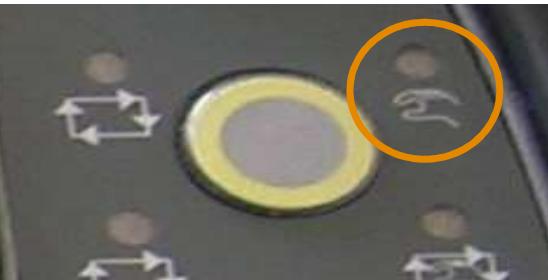


## MOVIMIENTOS PROGRAMADOS (LENTO)

2

Movimiento mientras se mantenga presionado

Control de velocidad



Célula abierta

1

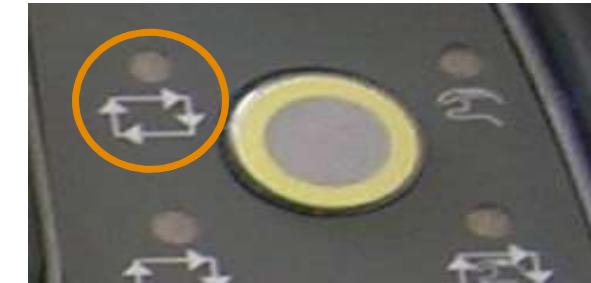
Suprimir el modo de desplazamiento activo presionando en el botón

## MOVIMIENTOS PROGRAMADOS (LOCAL)

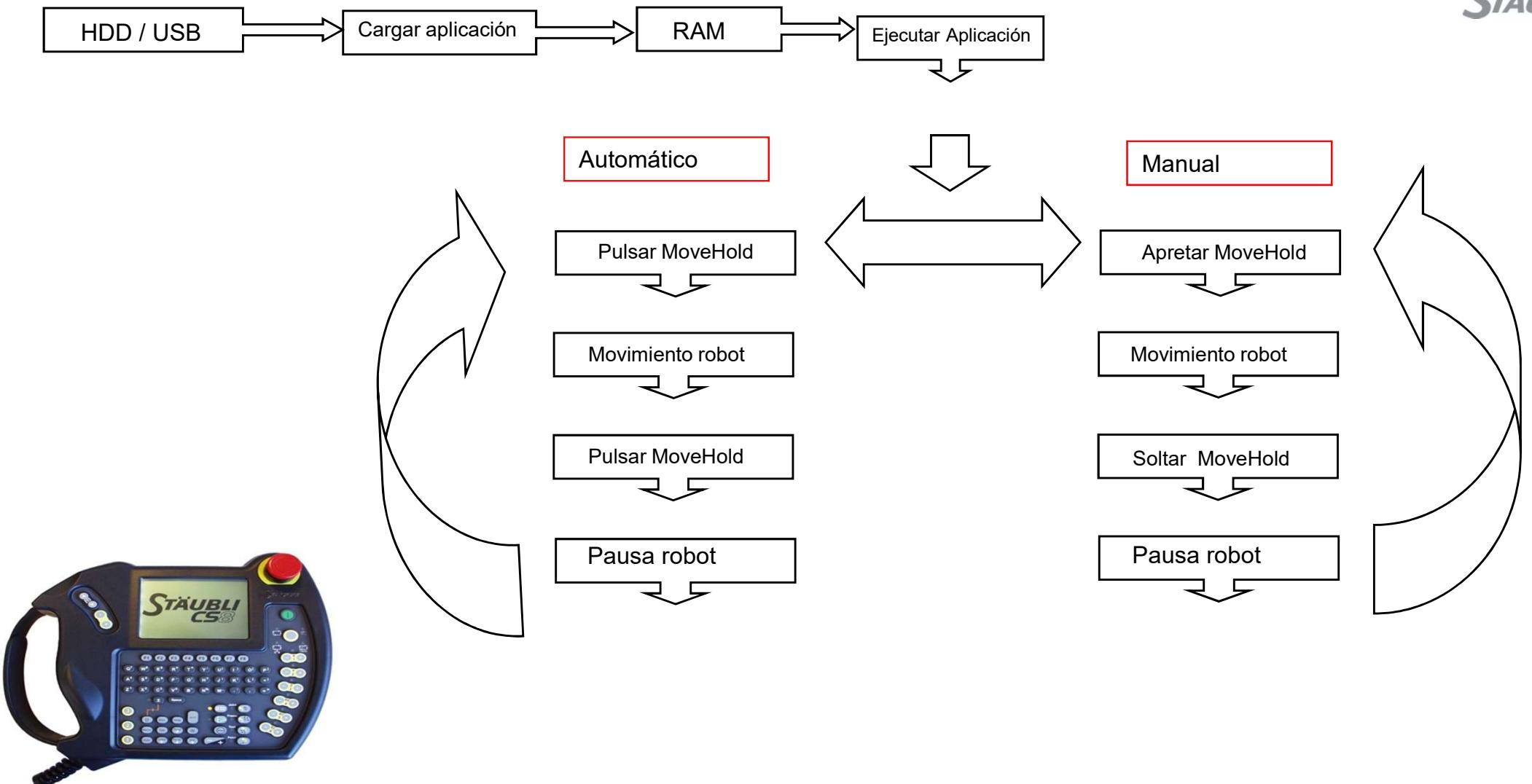
- Presionar 1 vez = Movimiento continuo (LED encendido)
- Presionar 1 vez más = Paro inmediato (LED parpadeante).
- 1 vez = Run
- 2 vez = Pausa



Control de velocidad  
100% = velocidad máx..



¡¡¡ Célula cerrada !!!

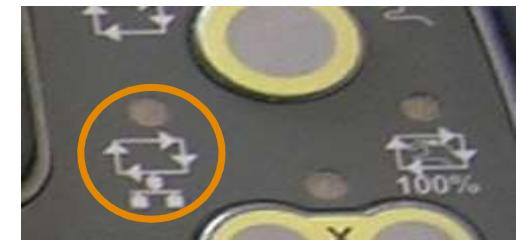


# MOVIMIENTOS PROGRAMADOS (REMOTO)

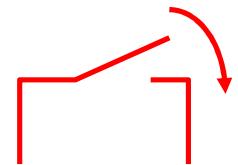
STÄUBLI

*Imposible habilitar la potencia a partir de mando de control*

Movimientos del robot controlados por medio de un señal externa (Flanco positivo en la entrada `enablePower` del archivo IOMAP.CF) o por orden de programa.

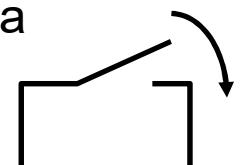


Habilitación de potencia :



Cuando la potencia del brazo esta habilitada, los movimientos son habilitados con el botón MOVE/HOLD (salvo si la función `AutoconnectMove(true)` es utilizada o la señal MOVE/HOLD es mapeada en el IOMAP.CF)

Desactivación de potencia :



El control de velocidad sigue siendo posible a partir del mando.

El uso del mando de control no es obligatorio en este modo de marcha.

El fichero a editar se encuentra en: \Usr\configs\templates\iomap\

Es altamente aconsejable de editar con el programa TextPad.

El fichero una vez modificado se ha de dejar en: \Usr\configs\

### Ejemplo VAI3 >= 7

- <!-- popup - serial output - transmit popup messages -->
- <String name="popup" value="" />
- <!-- power - digital output - status of arm power -->
- <String name="power" value="BasicIO-1%Q0" />
- <!-- fastSpeed - digital output - status of the fast speed mode in test working mode -->
- <String name="fastSpeed" value="" />
- <!-- dummyPlug - digital output - presence of the dummy plug in replacement of the MCP -->
- <String name="dummyPlug" value="" />

### Ejemplo VAI3 < 7

- //popup = serial output - transmit popup messages
- **power = fOut0**
- //fastSpeed = digital output - status of the fast speed mode in test working mode
- //dummyPlug = digital output - presence of the dummy plug in replacement of the MCP
- //temperature = analog output - temperature of the cabinet in °C (measured on safety board)

## MOVIMIENTOS PROGRAMADOS (PRUEBA)

*Habilitación de potencia desde el mando de control.*

Los movimientos se hacen como en el modo lento :

- Si MOVE/HOLD se mantiene presionado

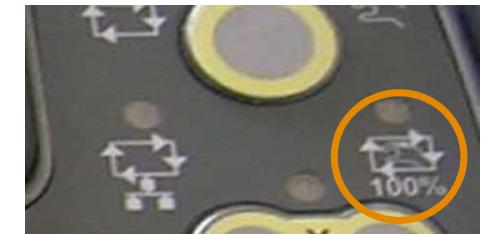
Para eliminar la limitación de velocidad:

- Pasar a modo de prueba.
- Habilitar la potencia del brazo con el mando de control => 250 mm/s
- Activar la entrada validationkey del archivo IOMAP.CF (flanco positivo)

*La limitación de velocidad está desactivada hasta que la potencia del brazo sea deshabilitada o un nuevo flanco positivo en la entrada.*

*Aumentar la velocidad a mas de 250 mm/s con el botón +*

iiii Modo reservado a personas formadas en la utilización de robots y de la célula robotizada. El contacto de la entrada digital debe estar situado al exterior de la célula y no debe estar al alcance de personas no habilitadas. (interruptor de llave) !!!!



célula abierta

# INTERRUPCIÓN MOVIMIENTOS PROGRAMADOS

En todo momento es posible interrumpir los movimientos programados:

- Botón MOVE/HOLD.
- Desactivar la potencia.
- Cambiar de modo de marcha para continuar en otro modo.
- Desplazar el robot con el control de mando en modo manual.
- Desplazar el robot liberando los frenos.

*Sin interrumpir la ejecución de movimientos programados*

## MOVIMIENTO DE CONEXIÓN

Para continuar el ciclo, si la potencia fue desactivada o el brazo fue desplazado

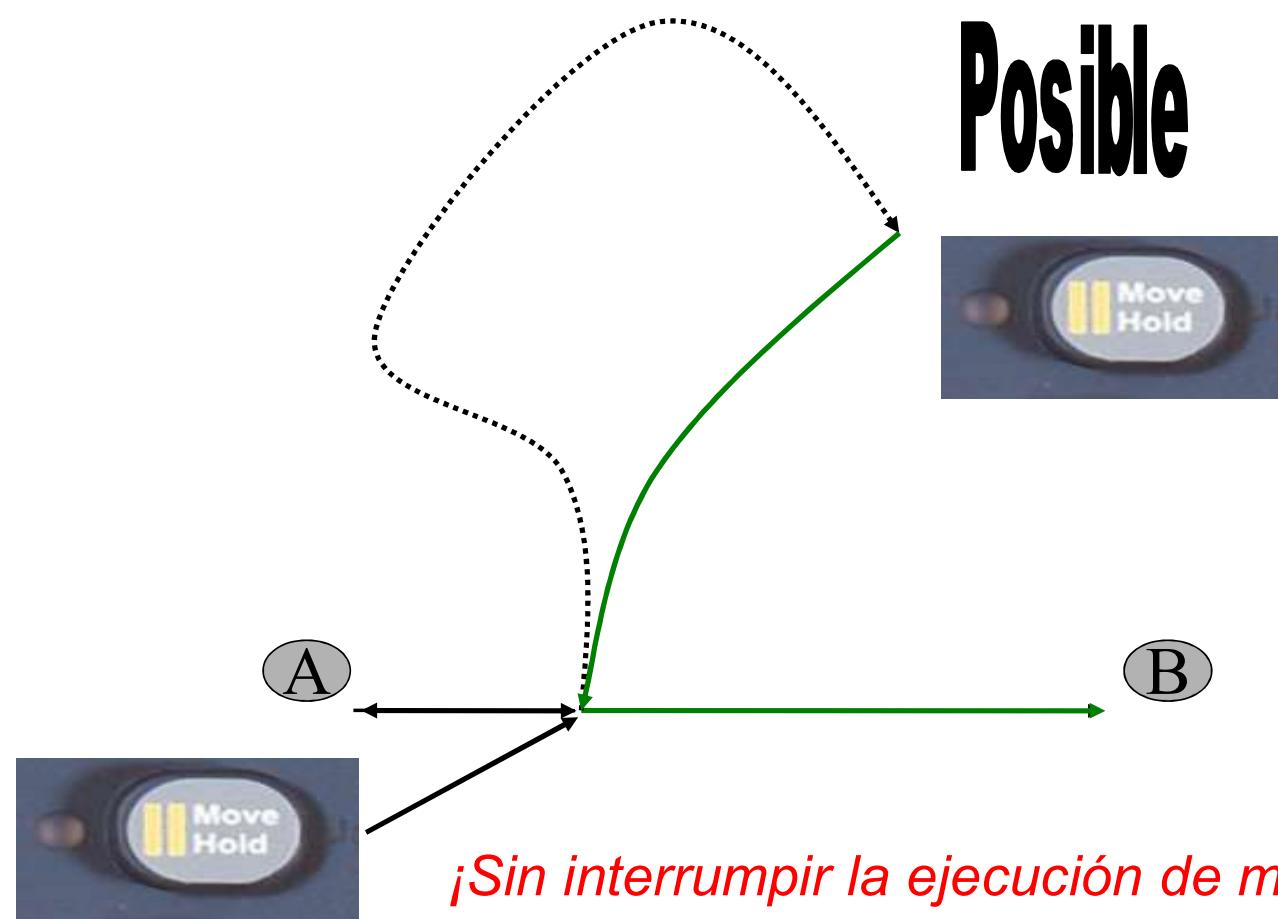
=> *Movimiento de conexión al punto de paro*

- 1 El robot se deslaza del punto actual al punto de paro manteniendo presionado MOVE/HOLD (modo LENTO)
- 2 En el punto de paro, el robot se detiene y retoma los movimientos programados dependiendo del modo de marcha seleccionado, al presionar MOVE/HOLD

*Nota : En modo remoto, después de deshabilitar la potencia del brazo y si este no fue desplazado fuera de su trayectoria, al momento de una nueva habilitación de potencia los movimientos comenzaran automáticamente (a partir de V5.1.1)*

# MOVIMIENTO DE CONEXIÓN

STÄUBLI

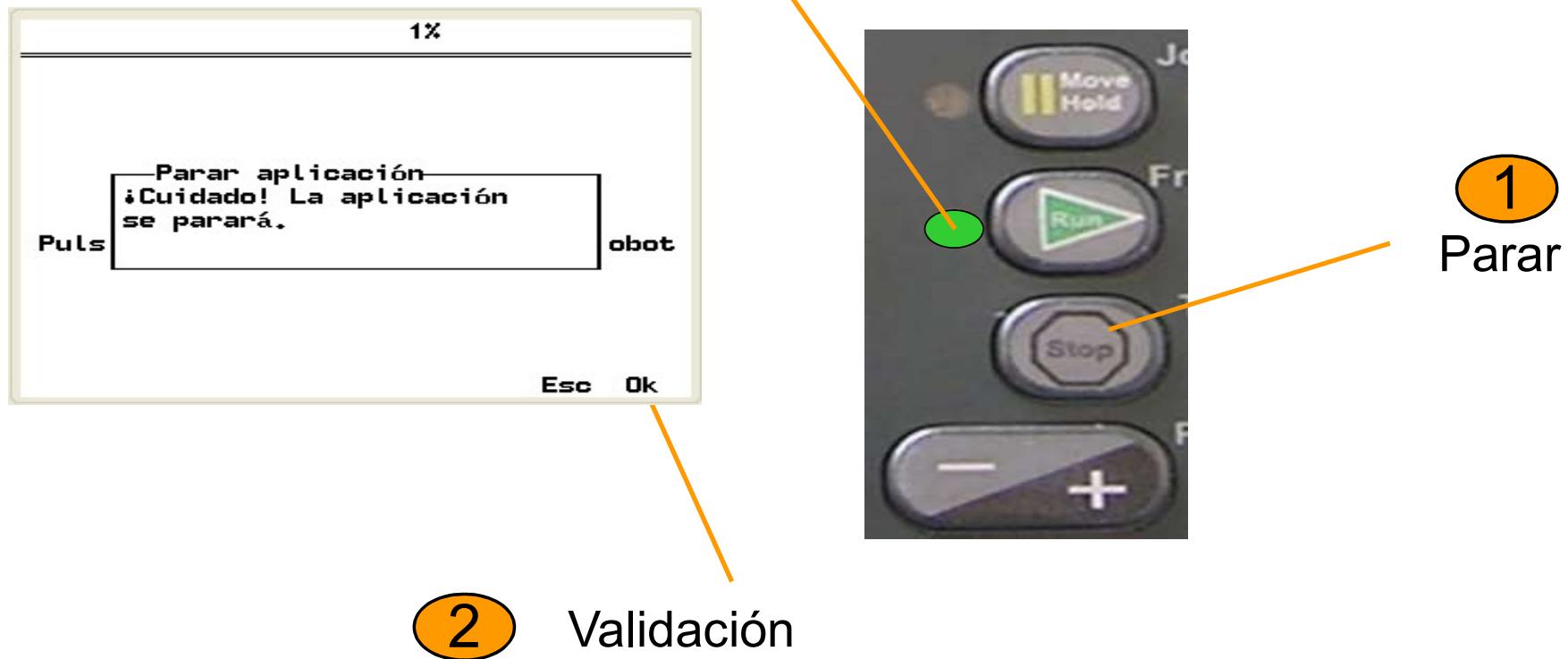


Movimiento del brazo  
Desactivación de potencia  
Cambio de modo  
Presionar 1 vez  
(Mantener presionado) :  
Movimiento de conexión a  
velocidad lenta hasta el punto  
de paro  
Presionar otra vez :  
Los movimientos  
programados continúan a la  
velocidad inicial

*¡Sin interrumpir la ejecución de movimientos programados!*

# PARAR UNA APLICACIÓN

Si aplicación esta en ejecución



## RESUMEN : CONTROL DE UNA APLICACIÓN

OFF : potencia no habilitada.

Parpadea : Detenido.

ON : Movimiento

Aplicación en ejecución

Arrancar / Parar los movimientos del robot



Arrancar la aplicación

Parar la aplicación

Velocidad

Al encender el controlador : Se conservan la velocidad monitor y el modo de marcha de cuando fue apagado.



FORMACIÓN VAL3

# *Estructura*

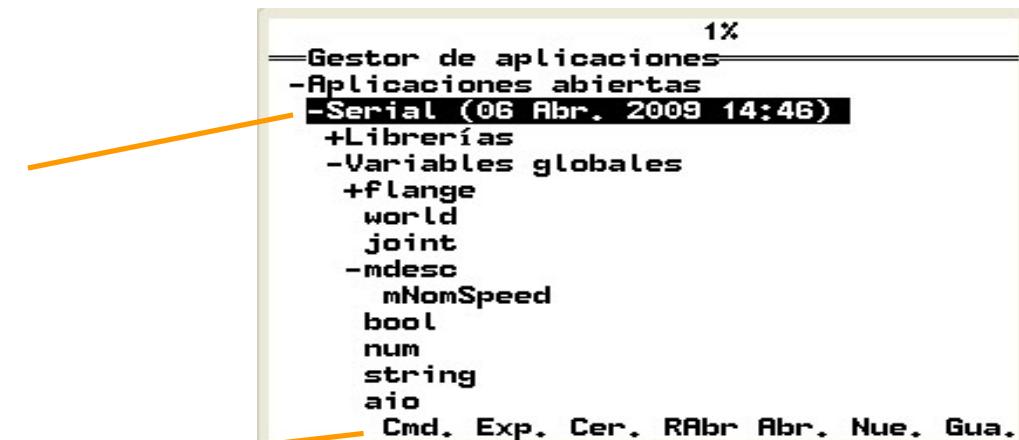
*de una*

# *Aplicación*



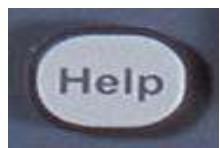
## VENTANA APLICACIÓN

Nombre de la aplicación



Comandos

*Menú  
Contextual*



Ayuda sobre menús

Exportar :  
Guardar  
como

Cerrar  
Aplicación

Reabrir la  
aplicación

Guardar  
aplicación  
Abrir una  
aplicación

# ESTRUCTURA DE UNA APLICACIÓN

- **VARIABLES GLOBALES** : variables de diferentes tipos definidas por el programador : Herramientas, planos, puntos cartesianos, posiciones angulares, numéricos, cadenas de caracteres, etc ...
- **PROGRAMAS** : Ejecutados como tareas o subprogramas.

START( )



STOP( )



- **PARÁMETROS** : Unidades, valores por defecto.
- **LIBRERIAS**: Gestión de datos de otras aplicaciones.

## VARIABLES GLOBALES

- Son variables que pueden ser utilizadas en cualquier punto del programa.
- Puede ser de una dimensión (único valor) o tridimensional (un array de tres dimensiones)
- Se puede insertar o suprimir elementos.
- Operaciones clásicas con las variables numéricas: +, -, /, \*, ...
- Prioridad de los operadores => utilización de paréntesis  $nX=(nZ+nY)*nT$
- Operaciones

nContador=nPiezas

= Operador de asignación

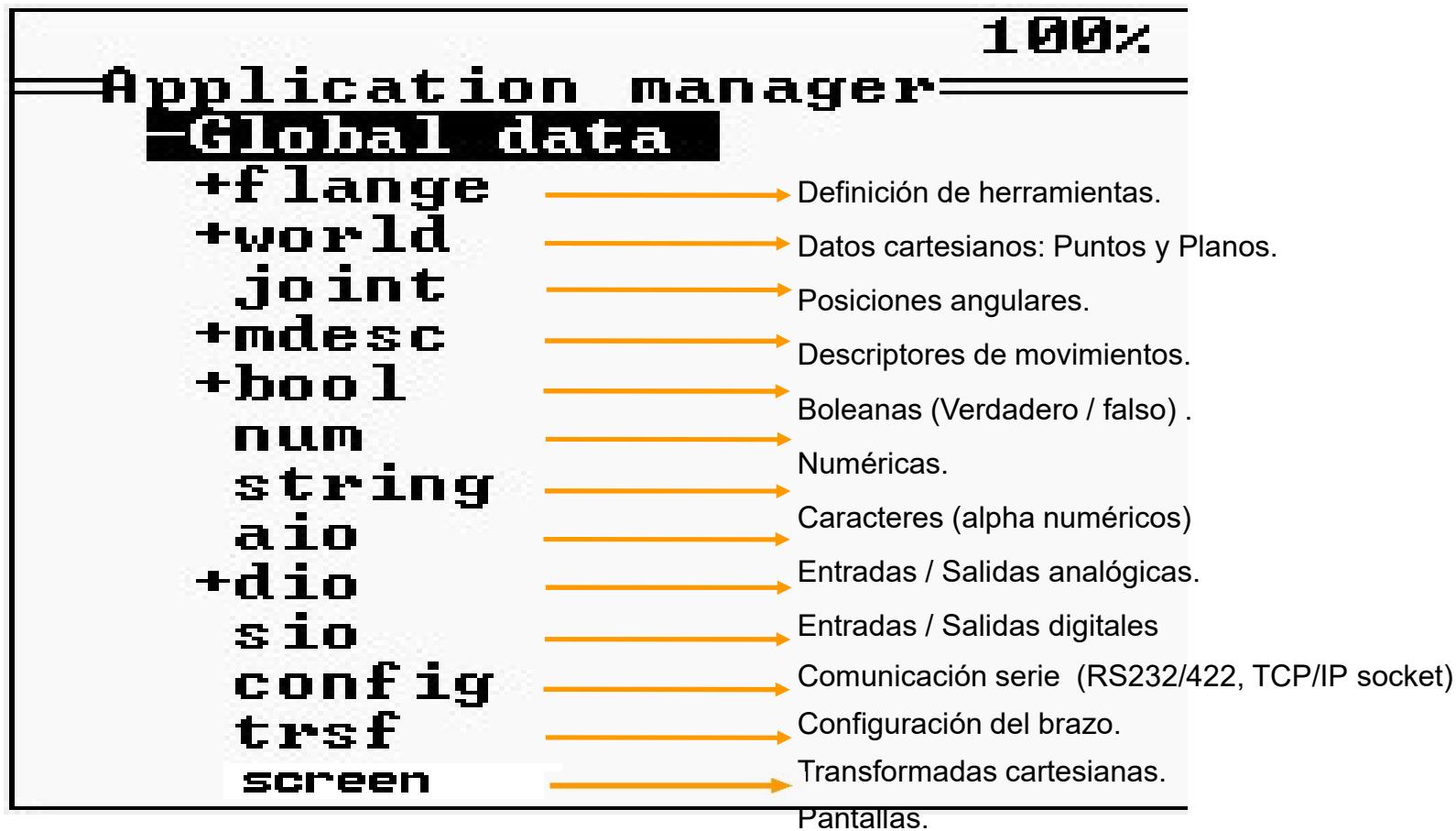
bRun=!bStart

! Operador de NEGACIÓN (NOT)

bRun==bStart

Operador de interrogación

## ÁRBOL DE VARIABLES GOBALES



# NOMBRE DE LAS VARIABLES

STÄUBLI

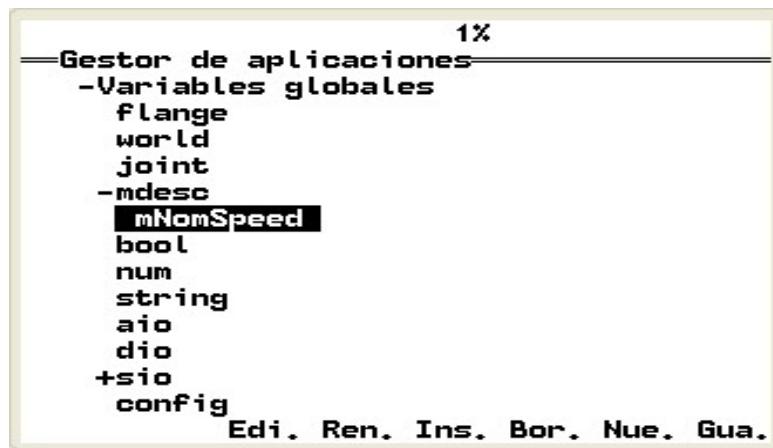
- Libertad para escoger los nombres de las variables:
- 15 caracteres máx. : letras (a...z A.....Z) , dígitos (0....9) y barra baja \_
- Ni espacios ni : ; , + - \* / . ? « ! Etc.....
- El primer carácter es una letra o \_

Convención interna de STÄUBLI :

- 1º carácter en minúscula para determina el tipo variable
- La 1<sup>a</sup> letra de cada palabra comienza con mayúsculas
- Ejemplo :      bool bArranqueProduccion      num nContador  
                  string sMensaje      point pCoger   joint jCasa  
                  tool tPinza      trsf trAltura      etc.....

# EDICIÓN DE VARIABLES

STÄUBLI



Anular Edición



Para editar un elemento



Escribir el valor o seleccionar de la lista

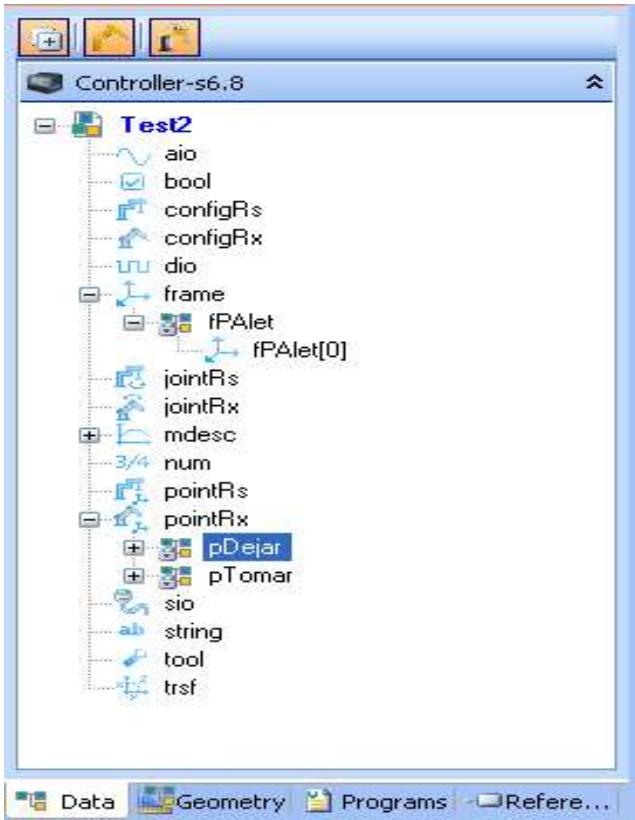


Para validar

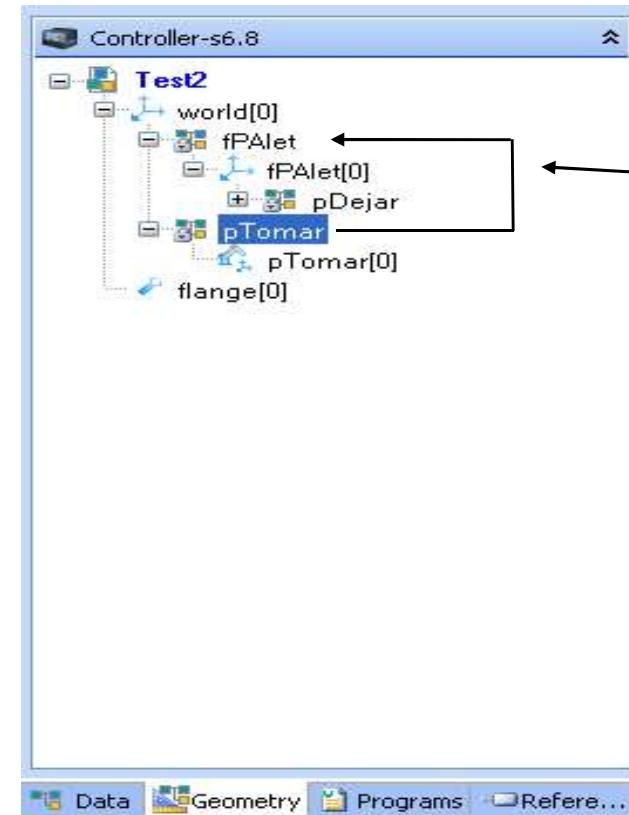


Para las variables de múltiples valores :  
validar con el botón OK

## Arbol de variables en el SRS:



No hay relación puntos / planos con un plano



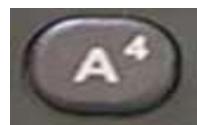
Relación planos y punto

## LETRAS / NUMEROS EN EL SP1

STÄUBLI



+

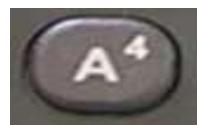


=

a



+



=

4



+



=

4

A

# TABLAS (ARRAYS)

- 1 dimensión con un elemento:

**nPosicion=10**

- 1 dimensión varios elementos:

**nPosicion [3]=10**

- Hasta 3 dimensiones (V7+):

**nPosicion [3,36,2]=10**

**! Max 10000 elements**

Si  $[n_1, n_2, n_3] \rightarrow n_1 * n_2 * n_3 < 10001$

Redimensionar

```
10%
--Gestor de aplicaciones
+world
+jo Variable nueva
+md Nombre : nPosicion
+bo Tipo : num
-nu Contenedor :Arreglo
n No. elementos dim1 : 1
n No. elementos dim2 : 0
n No. elementos dim3 : 0
n Atributo :Privado
n
nLimitMinimumZ=0
nOffsetLanguage=0
Esc Ok
```

```
10%
--Gestor de aplicaciones
+world
+jo Variable nueva
+md Nombre : nPosicion
+bo Tipo : num
-nu Contenedor :Arreglo
n No. elementos dim1 : 12
n No. elementos dim2 : 0
n No. elementos dim3 : 0
n Atributo :Privado
n
nLimitMinimumZ=0
nOffsetLanguage=0
Esc Ok
```

```
10%
--Gestor de aplicaciones
+world
+jo Variable nueva
+md Nombre : nPosicion
+bo Tipo : num
-nu Contenedor :Arreglo
n No. elementos dim1 : 12
n No. elementos dim2 : 3
n No. elementos dim3 : 48
n Atributo :Privado
n
nLimitMinimumZ=0
nOffsetLanguage=0
Esc Ok
```

```
10%
--Gestor de aplicaciones
-nPosicion [12]
nPosicion [0]=0
nPosicion [1]=0
nPosicion [2]=0
nPosicion [3]=0
nPosicion [4]=0
nPosicion [5]=0
nPosicion [6]=0
nPosicion [7]=0
nPosicion [8]=0
nPosicion [9]=0
nPosicion [10]=0
nPosicion [11]=0
Edi. Ren. Ins. Bor. Nue. Gua.
```

```
10%
--Gestor de aplicaciones
nPosicion [1,1,0]=0
nPosicion [1,1,1]=0
nPosicion [1,1,2]=0
nPosicion [1,1,3]=0
nPosicion [1,1,4]=0
nPosicion [1,1,5]=0
nPosicion [1,1,6]=0
nPosicion [1,1,7]=0
nPosicion [1,1,8]=0
nPosicion [1,1,9]=0
nPosicion [1,1,10]=0
nPosicion [1,1,11]=0
nPosicion [1,1,12]=0
Edi. Ren.
Nue. Gua.
```

# COLECCIONES (V7+)

Las colecciones es un array el identificador del cual es una clave y no una numérica. (string) :

```
move(pControl["tipoA"],tPinza, mLento)
```

La clave puede ser un String.

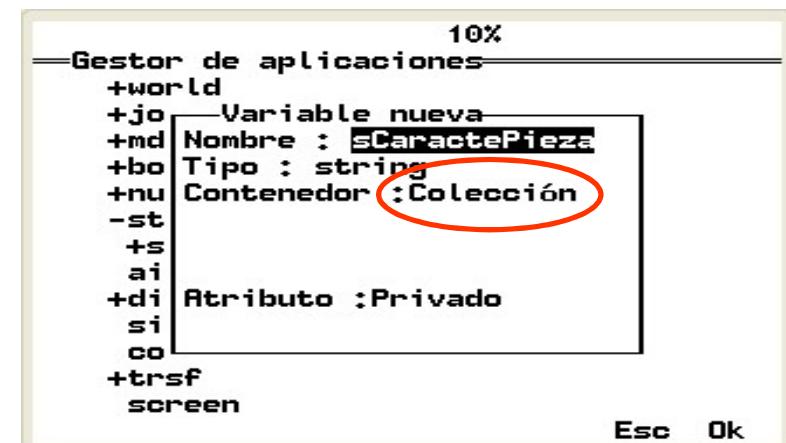
```
sRef="tipoA"
```

```
move(pControl[sRef], tPinza, mLento)
```

La clave es de tipo alfanumérica.

Instrucciones posibles de navegación por las colecciones: first , next, last, prev

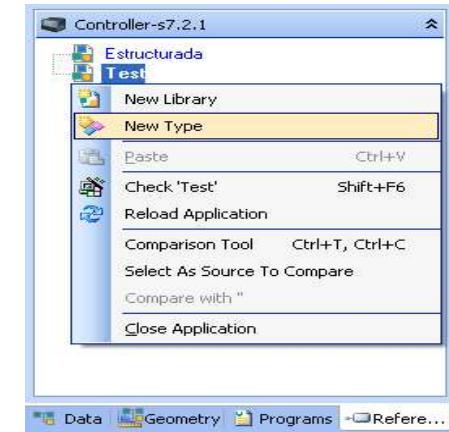
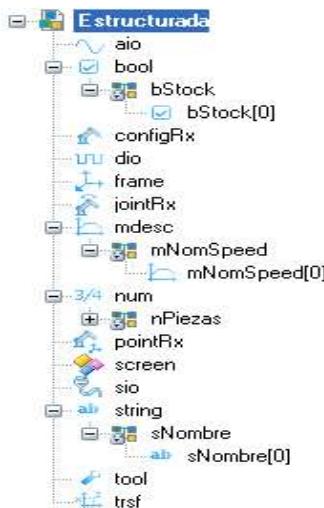
Práctico para manejar varias referencias de un producto en la misma aplicación



# VARIABLES ESTRUCTURADAS (V7+)

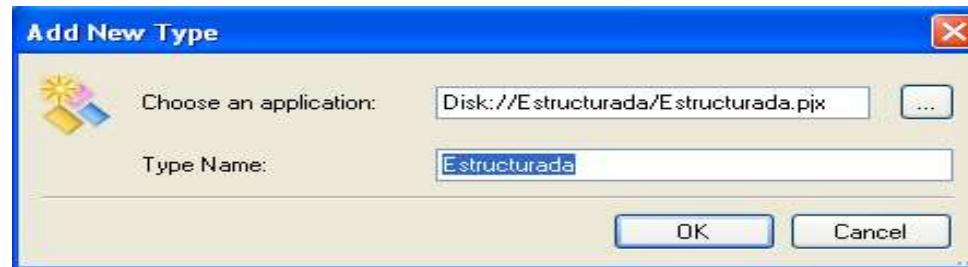
STÄUBLI

Las variables estructuradas son un tipo de variable en la que el programador puede combinar diferentes tipos de variables.



Es necesario crear una nueva aplicación en la que definiremos todas las variables de nuestra nueva variable estructurada.

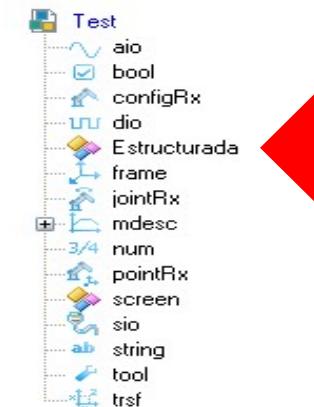
Crear la nueva variable en nuestro programa



## Definir la ruta y el alias



Inicialmente



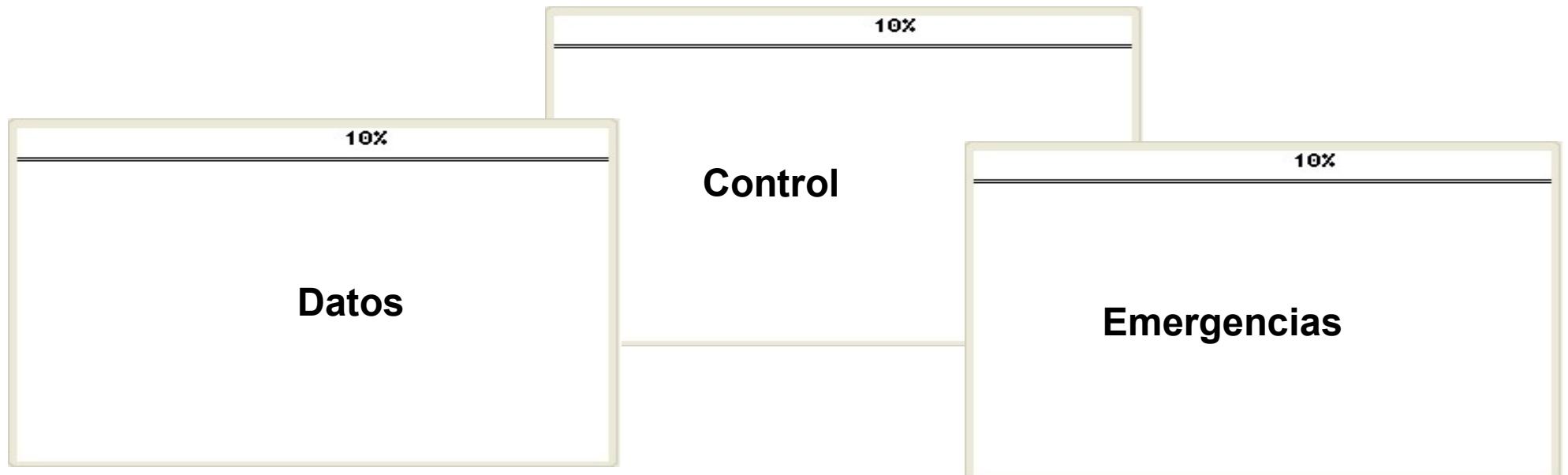
Con la nueva variable

# PANTALLA (V7+)

STÄUBLI

El sistema operativo Val3 >= V7 es multipantalla.

Las variables de tipo Pantalla son variables que nos permitirán identificar las diferentes pantallas que tenemos e interrelacionar con ellas.



# DATOS GEOMÉTRICOS

STÄUBLI

Puntos cartesianos: Rama **WORLD**

vinculado a un plano (World o auxiliar)

X, Y, Z, Rx, Ry, Rz

Posiciones articulares: Rama **JOINT**

relacionado con la posición «CERO » del brazo

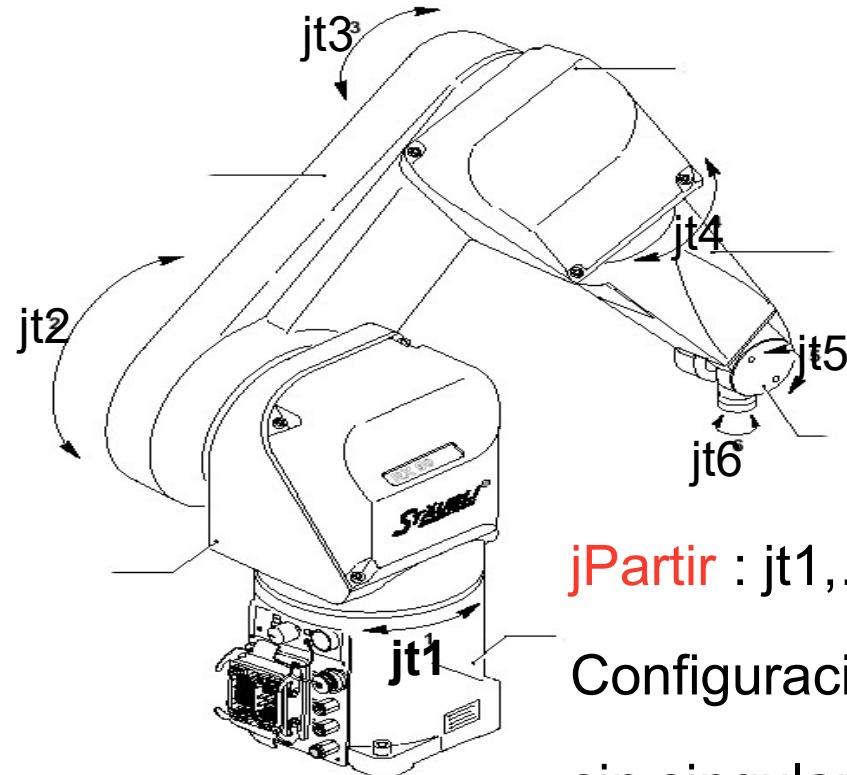
0, 0, 0, 0, 0, 0    ← Antropomorfico    Scara → 0,0,0,0

Planos de Referencia: Rama **WORLD**

El plano se aprende con 3 puntos tomados en la célula.

Seleccionar el plano deseado con el MCP moverse en modo FRAME

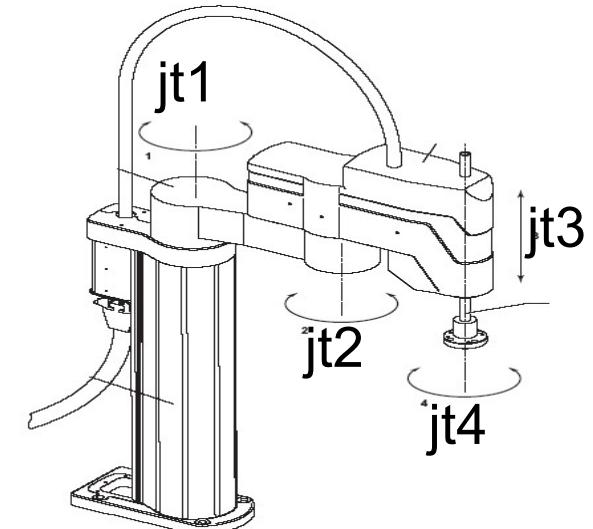
## POSICIONES ARTICULARES : JOINT



jPartir : jt1, ..., jt6

Configuración

sin singularidades

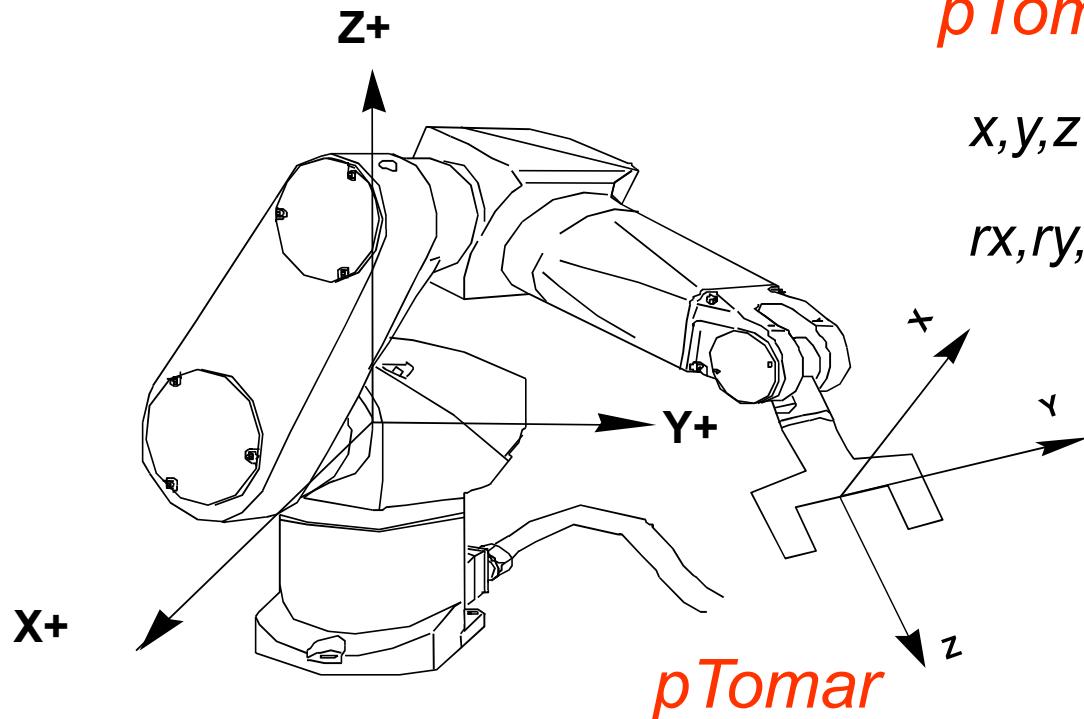


jStart : jt1, ..., jt4

Configuración

sin singularidades

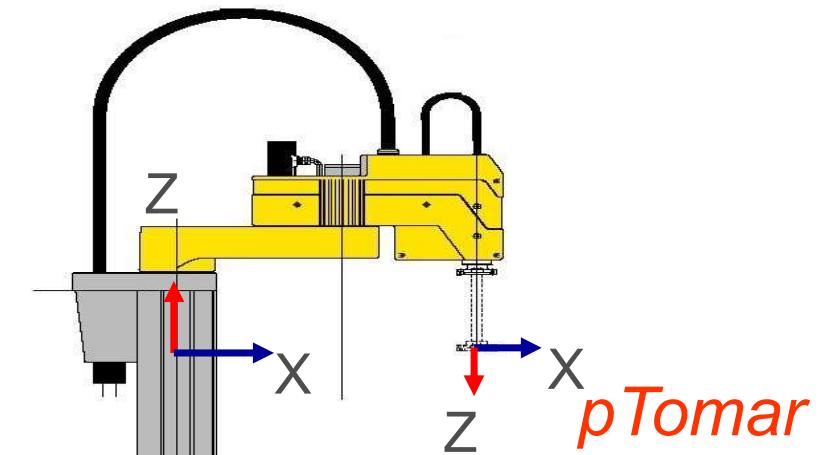
## PUNTOS CARTESIANOS : POINT



*pTomar* :  $x, y, z, rx, ry, rz$

$x, y, z$  valores en mm

$rx, ry, rz$  valores en grados



Puntos cartesianos de un RS alineados :

$X, Y, Z, RX, RY, RZ$

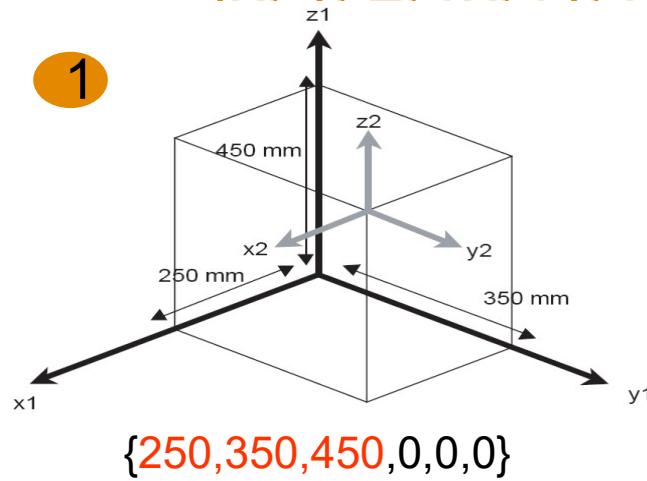
-180

0

## TRANSFORMACIÓN

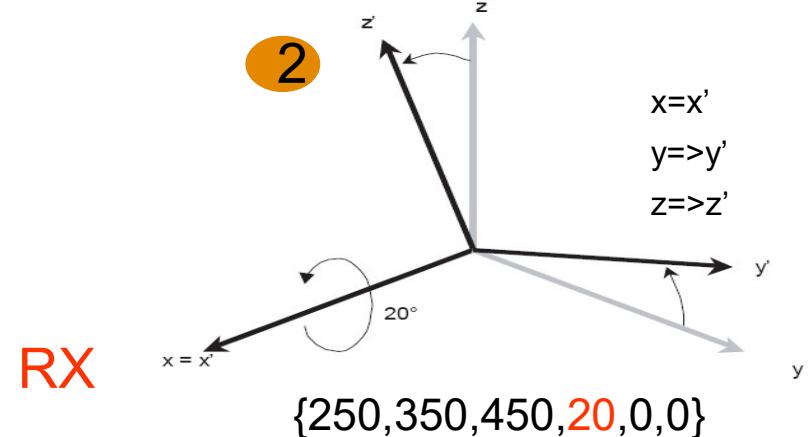
$$\{x, v, z, rx, rv, rz\} = \{250, 350, 450, 20, 10, 30\}$$

1



$$\{250, 350, 450, 0, 0, 0\}$$

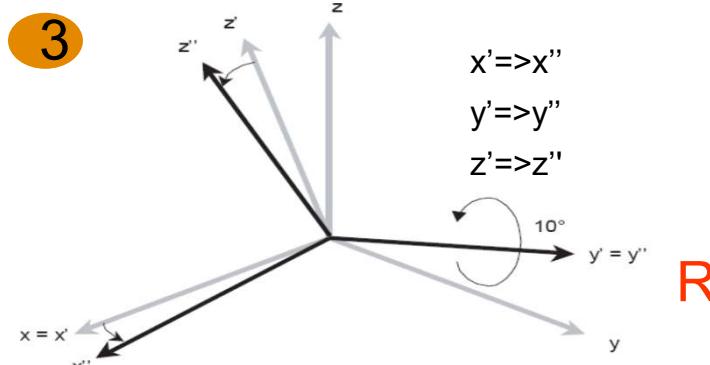
2



RX

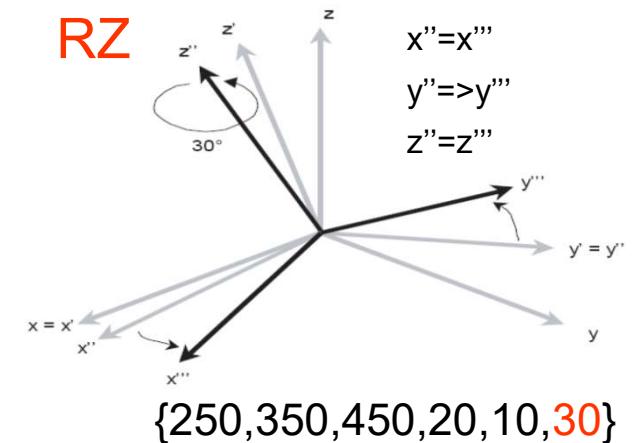
$$\{250, 350, 450, 20, 0, 0\}$$

3



$$\{250, 350, 450, 20, 10, 0\}$$

4



RZ

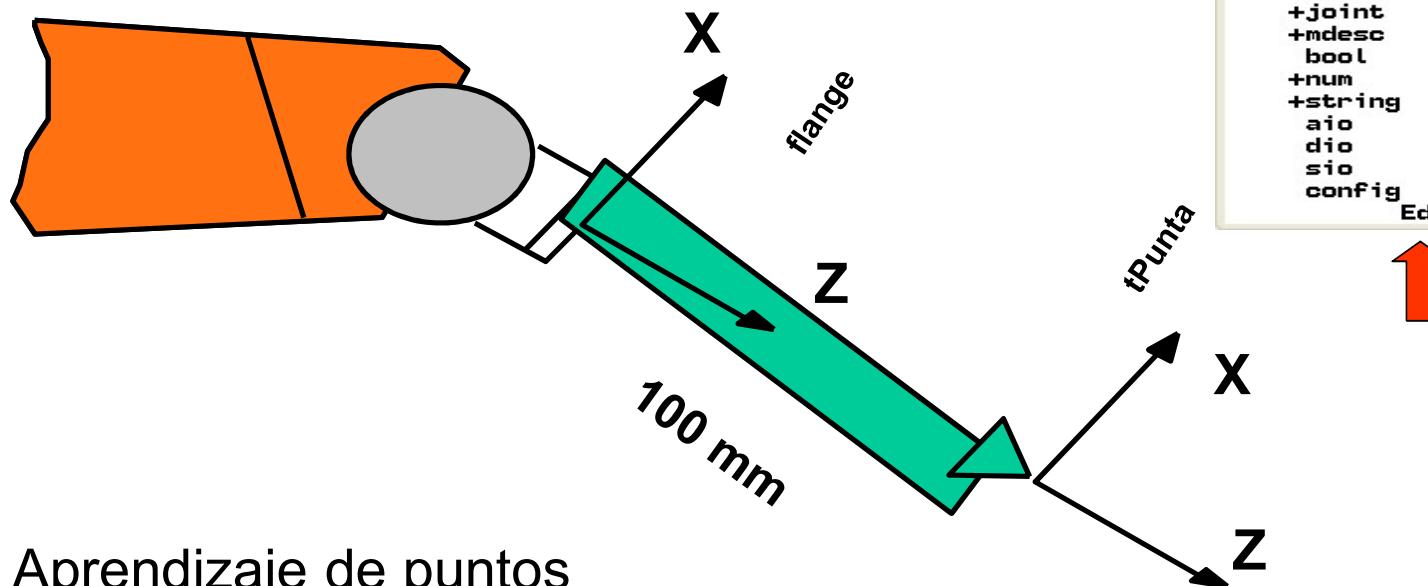
$$\{250, 350, 450, 20, 10, 30\}$$

FORMACIÓN VAL3

# *Aprendizaje de la célula*



# HERRAMIENTAS : flange

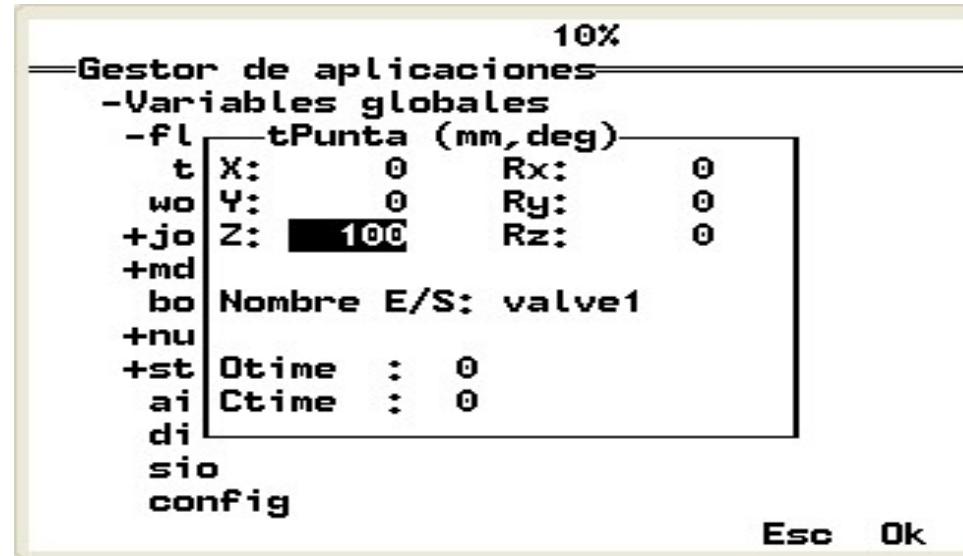


Aprendizaje de puntos  
control de velocidad y posición  
corrección geométrica

10%

```
Gestor de aplicaciones
-VARIABLES GLOBALES
-tPunta
world
+joint
+mdesc
bool
+num
+string
aio
dio
sio
config
Edi. Ren. Ins. Bor. Nue. Gua.
```

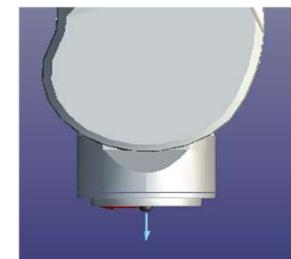
# DEFINICIÓN DE HERRAMIENTAS



- Definición geométrica (dirección tool)
- Salida digital asociada
- Tiempo de abertura y cerrado en segundos.

[www.staubli.com](http://www.staubli.com) → robotics → Login → Technical database → Download → VAL3 libraries and program examples → Motion Management → Calculate a tool transformation

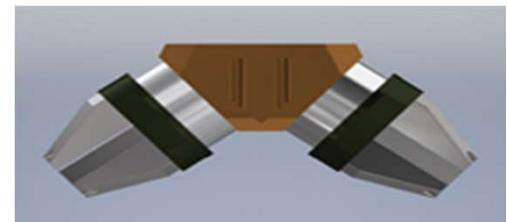
**STÄUBLI**



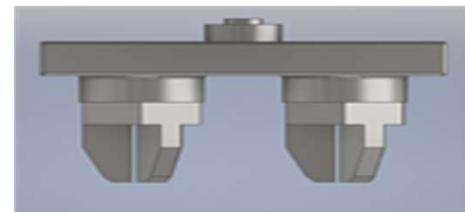
**3**



**2**



**6**



**4**



**1**



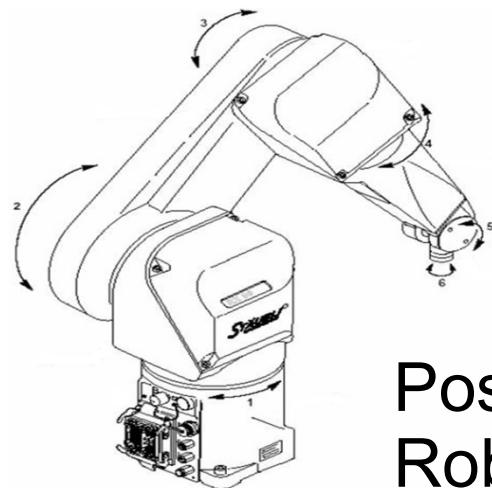
**5**

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>
X												
Y												
Z												
RX												
RY												
RZ												

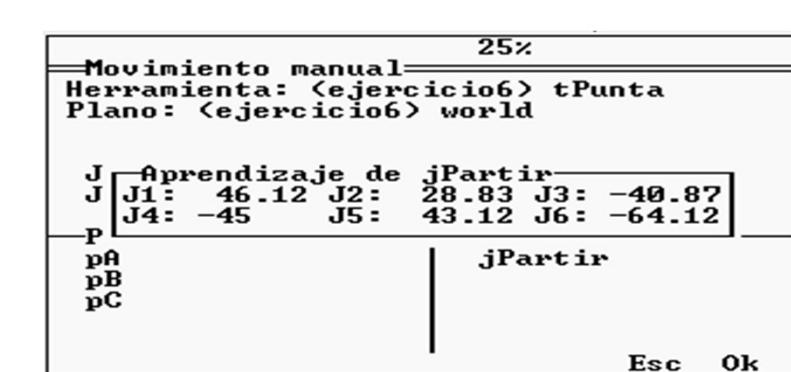
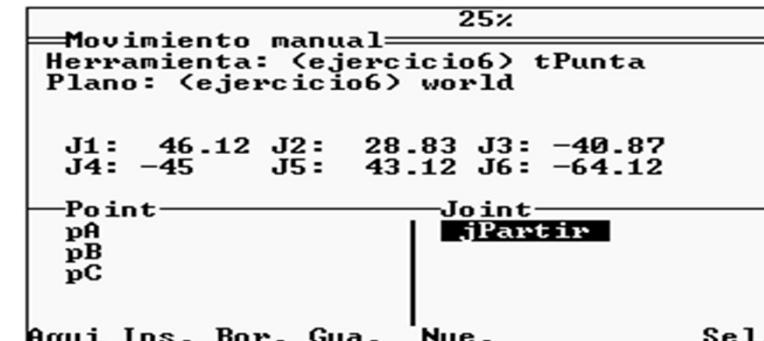
## APRENDIZAJE JOINT



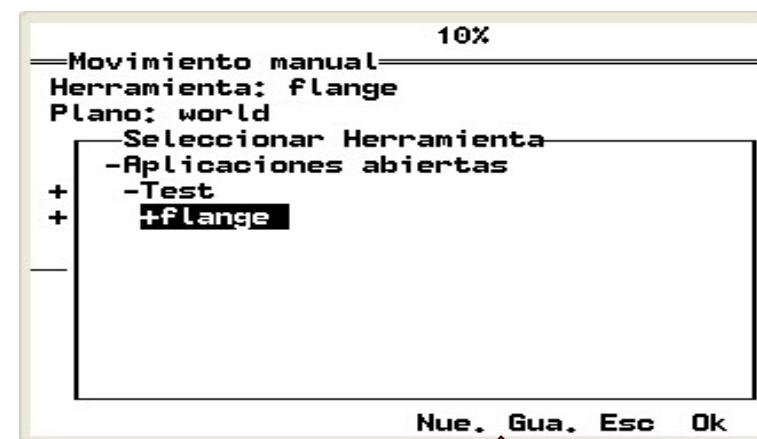
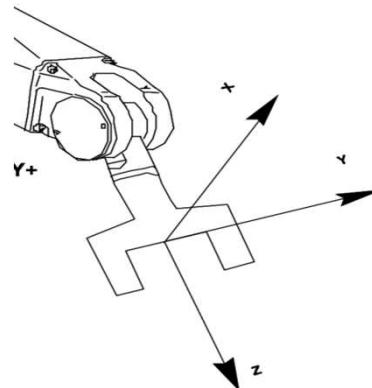
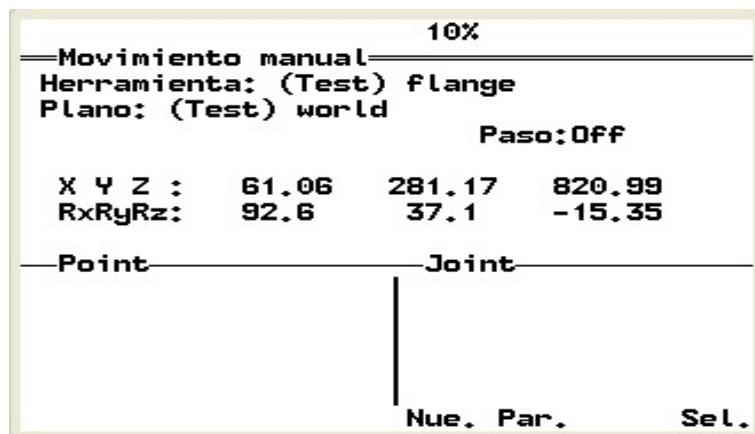
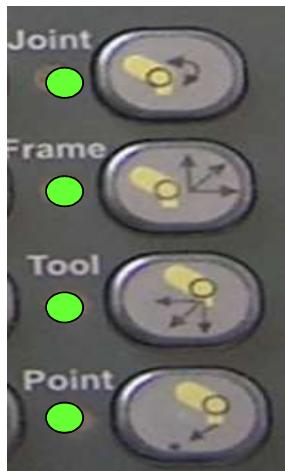
Utilizar los diferentes modos de desplazamiento



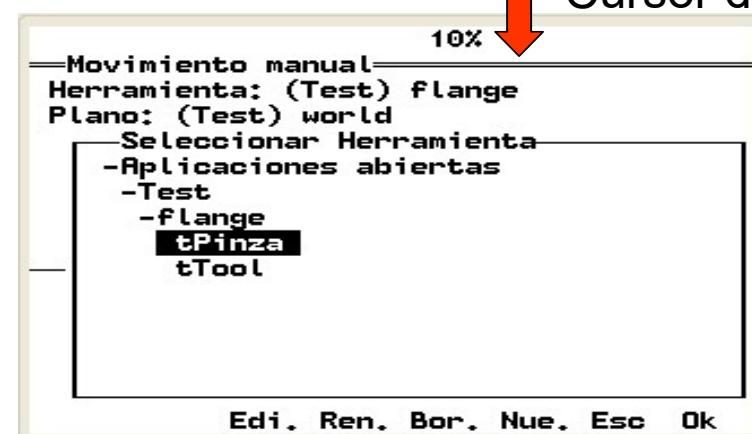
Posicionar el Robot



# SELECCIÓN DE HERRAMIENTA



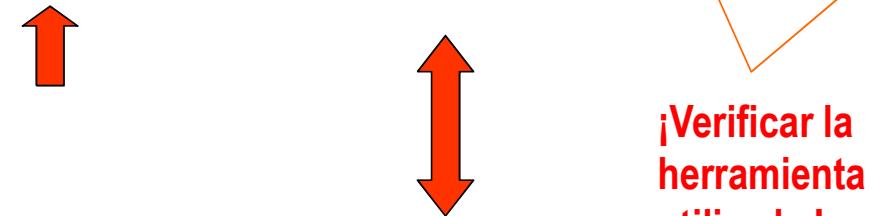
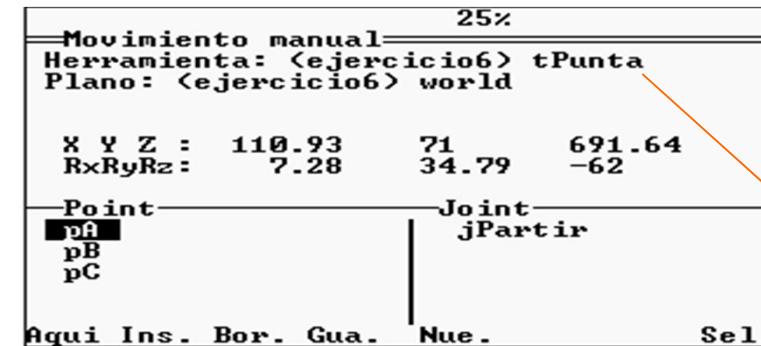
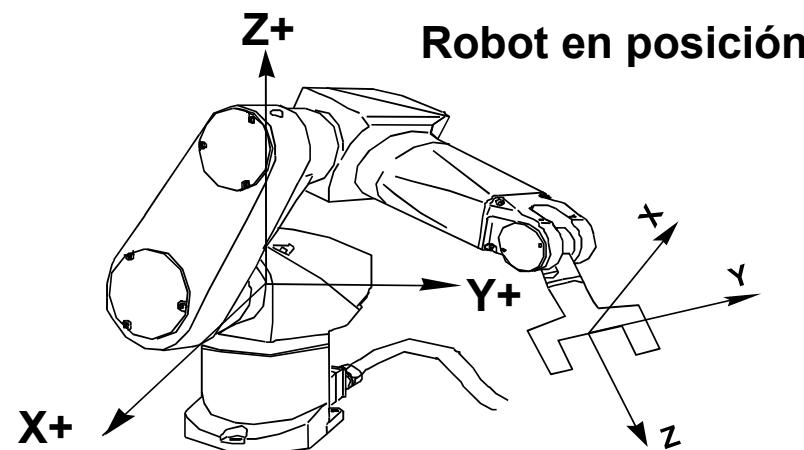
Cursor derecha



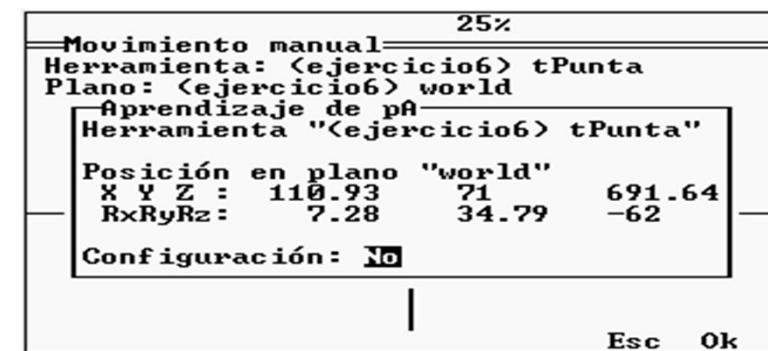
# APRENDIZAJE DE PUNTOS



Utilizar los diferentes modos de desplazamiento



¡Verificar la herramienta utilizada !

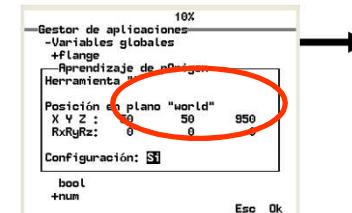
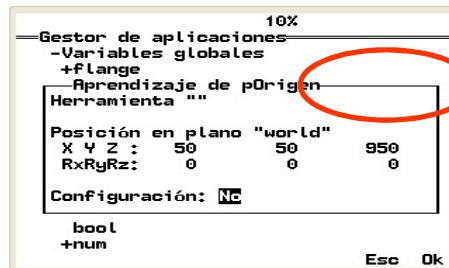


# PUNTOS CARTESIANOS

STÄUBLI

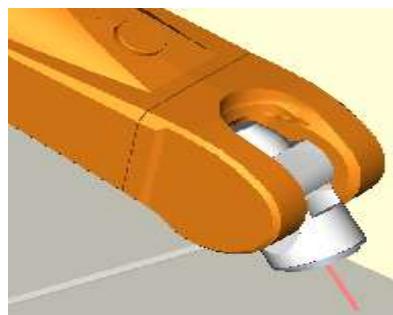
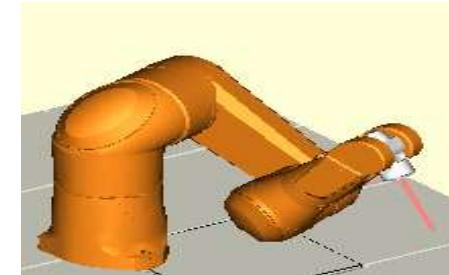
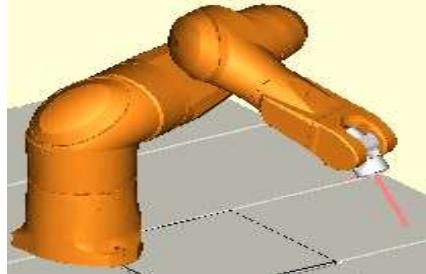
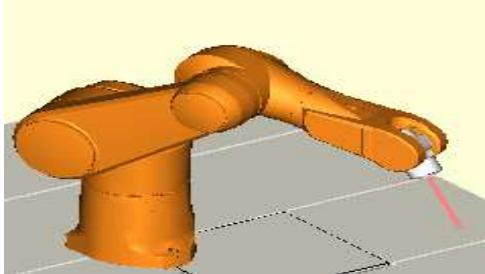
Cuando un punto es tomado:

- Configuración : No : El robot ira al punto con la configuración que mejor le funcione.
- Configuración : Si : Queda guardado el punto con la configuración que tiene el brazo actualmente.

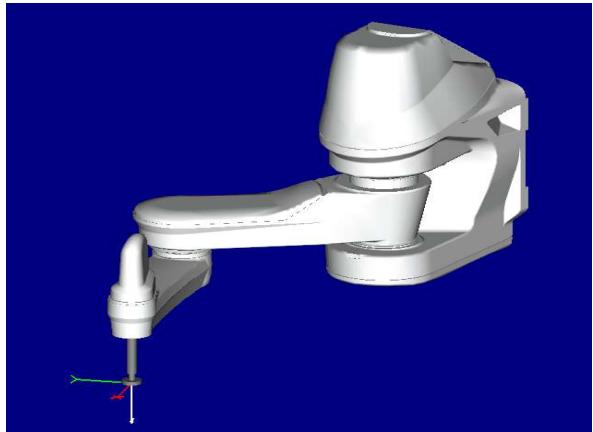


Una vez guardado el punto, podremos editarlo y modificar las configuraciones. Si el punto es sin conjugación las distintas posibilidades nos aparecerán en "FREE", sino, con la configuración del momento de tomar el punto.

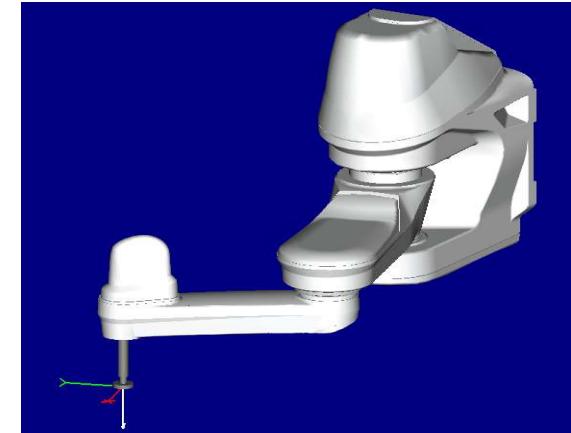
## CONFIGURACIÓN DEL BRAZO



- Shoulder (Hombro): Righty, Lefty
  - Elbow (Codo) : Positive, Negative      ( $Jt3 \geq 0$  o  $< 0$ )
  - Wrist (Muñeca): Positive, Negative      ( $Jt5 \geq 0$  o  $< 0$ )
- 
- Otras posibilidades :
  - FREE : El sistema escoge la mejor configuración para el.  
(permite que la configuración cambie en líneas rectas).
  - SAME : Mantiene la configuración del punto previo (= modo por defecto).



Right



Left

- Shoulder (Hombro): Righty, Lefty
- Otras posibilidades :
- FREE : El sistema escoge la mejor configuración para el. (permite que la configuración cambie en líneas rectas).
- SAME : Mantiene la configuración del punto previo (= modo por defecto).

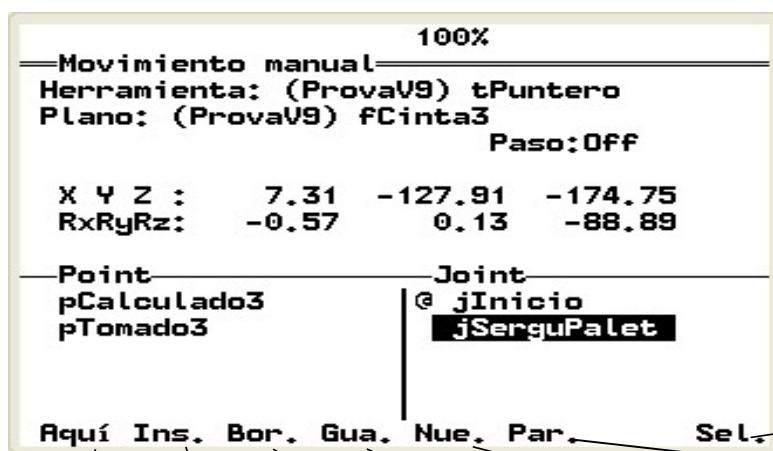
# INTERFACE DE TRABAJO

## Puntos de una aplicación

VAL3&gt;=6.5

### Símbolos enfrente de nombre :

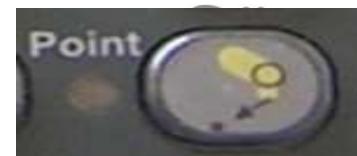
- 0 : Punto no tomado, coordenadas = 0
- @ : Robot en la posición con la pinza actual (aprox. 0.01 mm)
- ~ : Robot cerca de la posición (aprox. 1 mm)
- ! : Posición no accesible con la herramienta actual.



- Insertar punto en array
- Aprender punto actual
- Seleccionar la herramienta y el plano.
- Parámetros del modo paso.
- Nuevo punto
- Guardar la aplicación actual.
- Borrar un punto (chequear si no se usa) o elemento de un array.

## RE APRENDIZAJE DE PUNTOS

Verificar / reaprender puntos : En modo POINT



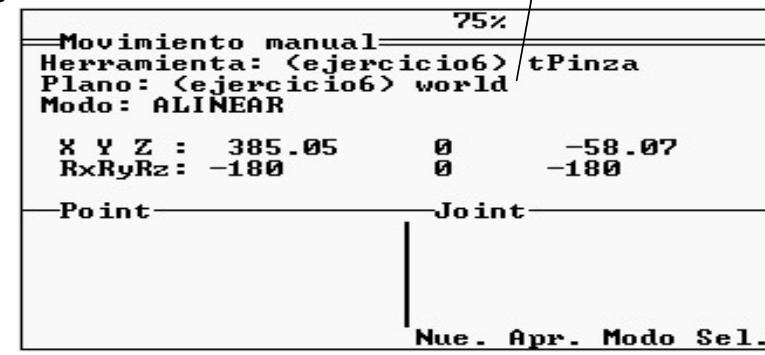
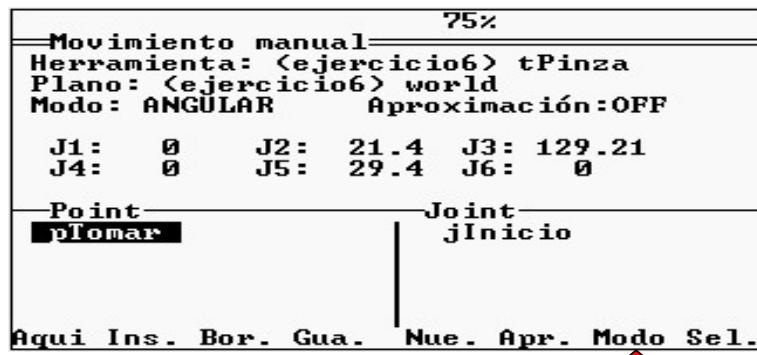
1 Tipo de movimiento = Modo :

- LINEAL: Línea recta (solo a puntos)

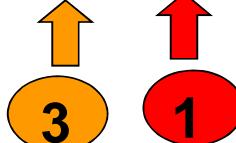
2 •ANGULAR: Libre( a puntos y joints)

3 •ALINEAR: La herramienta del robot se alinea con el plano mas cercano.

Con o sin APRO : selección de dirección y distancia



Control de movimiento



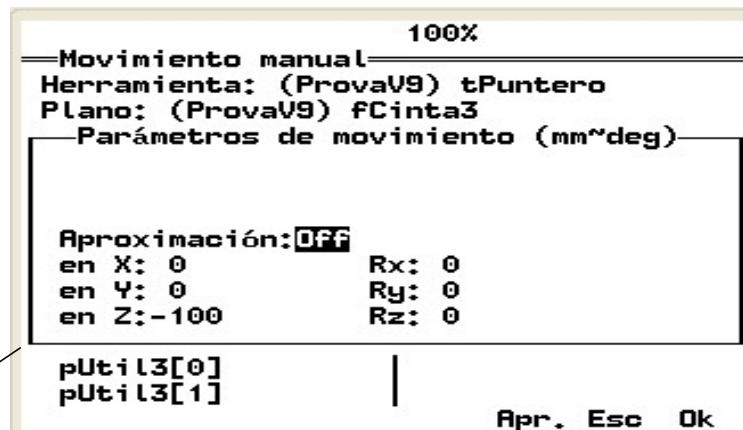
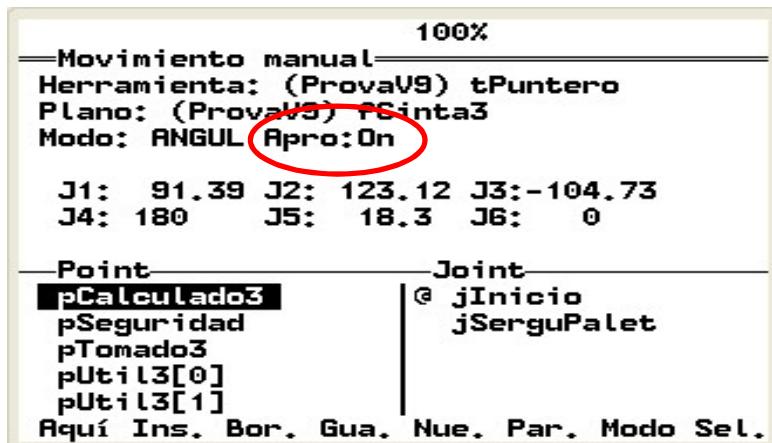
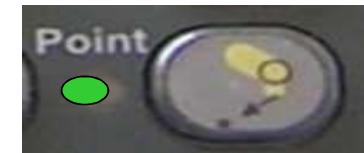
Selección de herramienta

# PARAMETROS EN MODO POINT

STÄUBLI

Aproximación:

VAL3>=6.5



Valores de los ajustes de la aproximación.

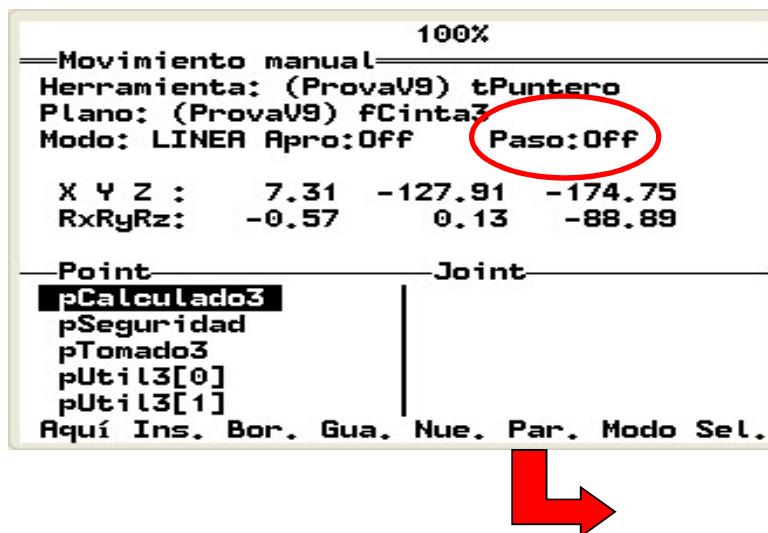
En modo Tool.

Aproximación ON / OFF

Acceso directo:

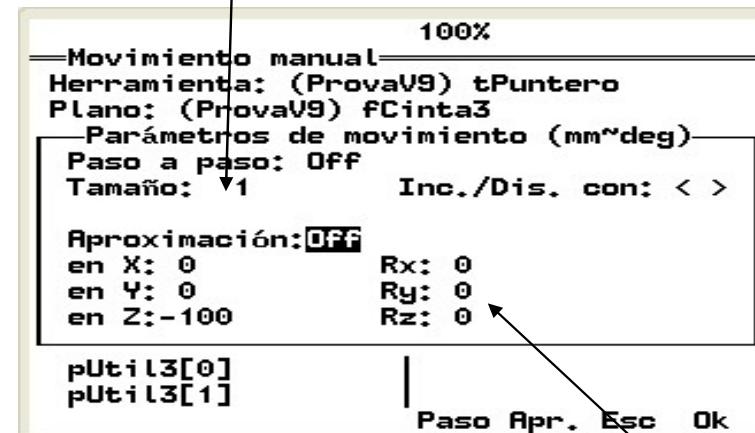
- Aproximación: F6 + F6

Paso a paso:



Modo paso a paso ON / OFF

Valores de los ajustes del paso a paso.



Aproximación ON / OFF

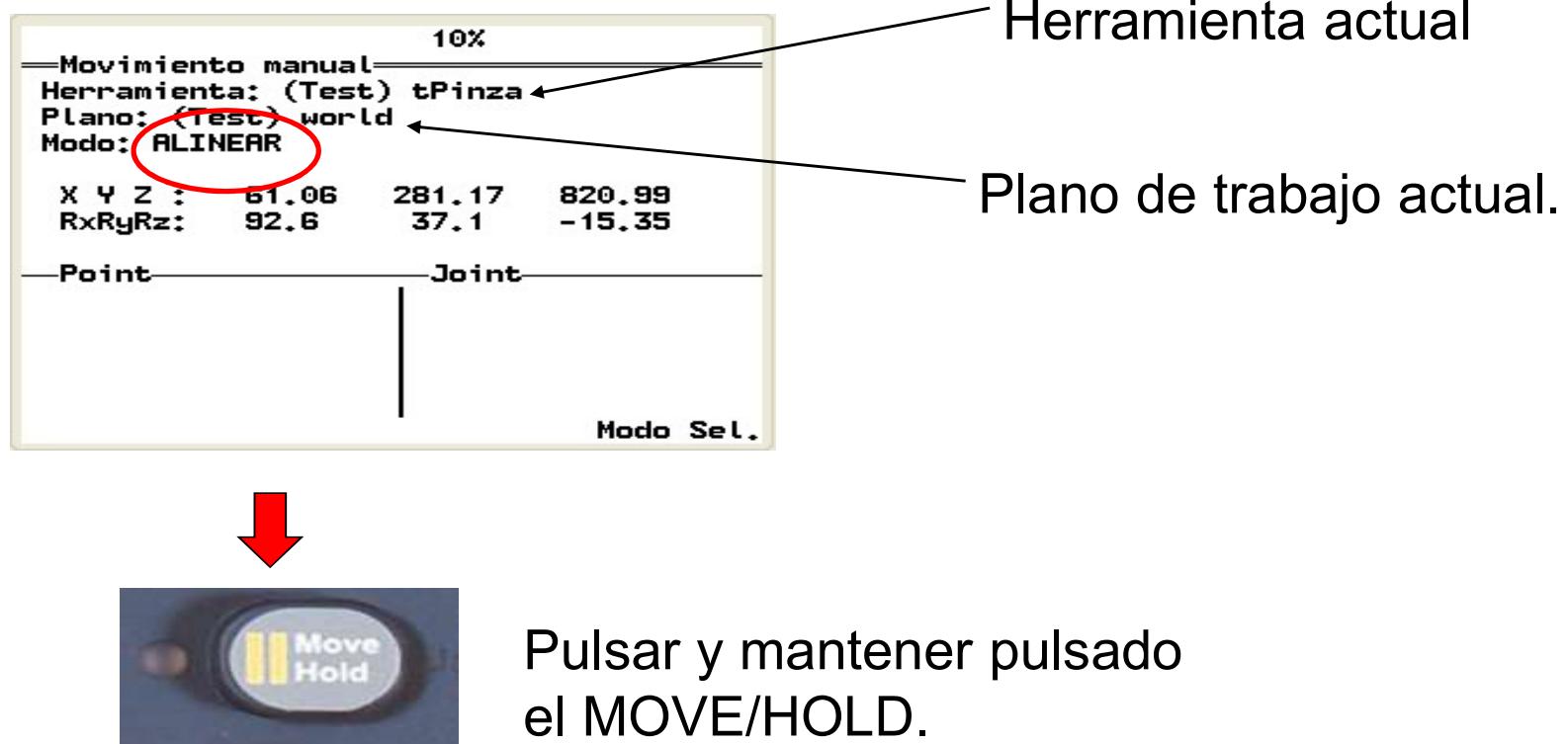
Acceso directos:

- Aproximación: F6 + F6
- Paso a paso: F5 + F5

Valores de los ajustes de la aproximación.

## Alinear:

El robot alineara la herramienta con el plano mas cercano (XY, XZ, YZ) del plano de trabajo actual.



## FORMACIÓN VAL3

# Movimientos *simples*

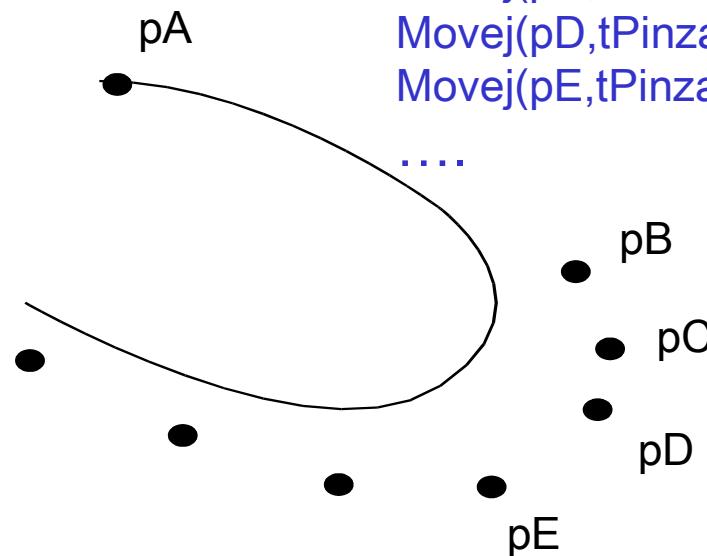


# MOVIMIENTO : MOVEJ

Movej(point,tool,mdesc)

O

Movej(joint,tool,mdesc)



Movej(pA,tPinza,mRapido)  
Movej(pB,tPinza,mRapido)  
Movej(pC,tPinza,mRapido)  
Movej(pD,tPinza,mRapido)  
Movej(pE,tPinza,mRapido)

- Interpolación articular : movimiento circular
- Velocidad y aceleración descritas por el descriptor de movimientos
- Ningún problema de singularidad
- Movimiento posible si no hay interferencias: Obstáculos, . . .

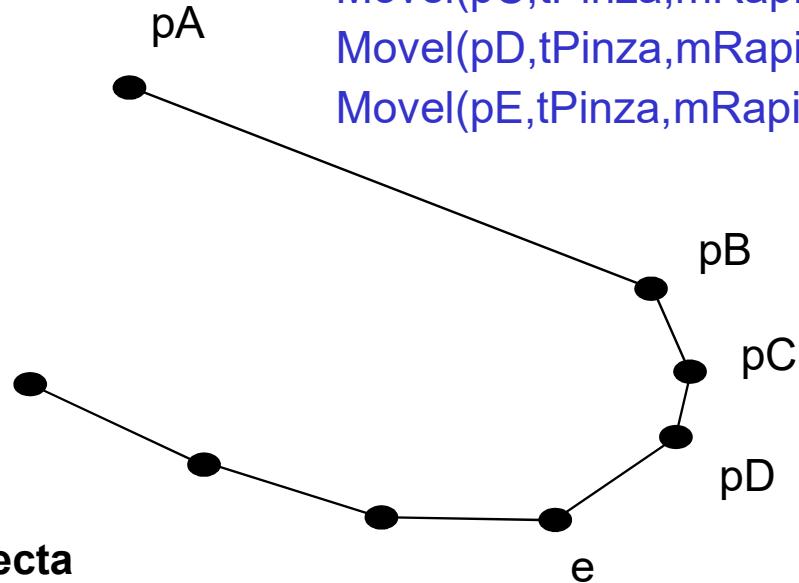
# MOVIMIENTO : MOVEL

STÄUBLI

Movel(point,tool,mdesc)

No disponible con JOINT

Movel(pA,tPinza,mRapido)  
Movel(pB,tPinza,mRapido)  
Movel(pC,tPinza,mRapido)  
Movel(pD,tPinza,mRapido)  
Movel(pE,tPinza,mRapido)



- Interpolación cartesiana : movimiento en línea recta
- Velocidad y aceleración descritas por el descriptor de movimientos
- Problema de singularidad => reducir la velocidad (Si se produce)
- Movimiento posible si no hay interferencias: Obstáculo,...

# SINGULARIDADES (BRAZOS RX/TX)

STÄUBLI

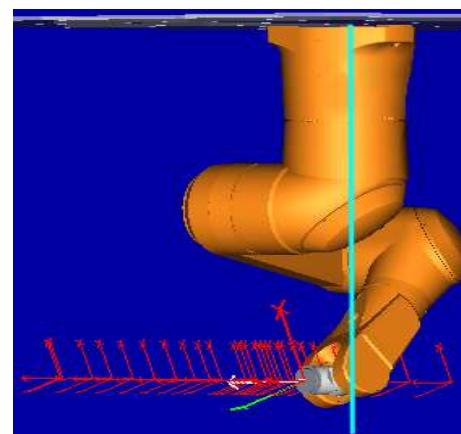
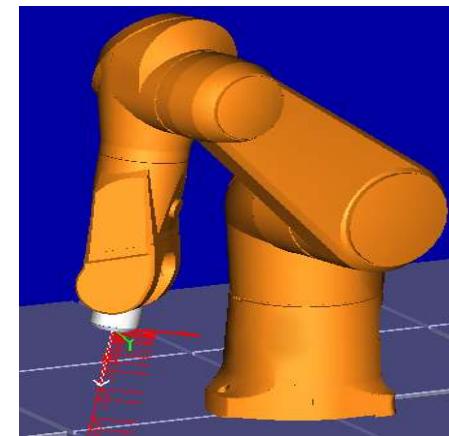
Las singularidades son una característica inherente a todos los robots de 6 ejes.

Se puede definir como los puntos en los cuales el robot cambia de configuración. Entonces, ciertos ejes se encuentran alineados, comportándose como un único eje.

Los movimientos libres (moveJ) son siempre posibles pero en cambio los movimientos en los que se impone la geometría (movel y movec) pueden ser imposibles.

Soluciones:

- Usar movimiento moveJ (si fuese posible).
- Modificar los puntos.
- Modificar el layout de la célula.
- Modificar el diseño de la pinza.



# MOVIMIENTO CIRCULAR: MOVEC

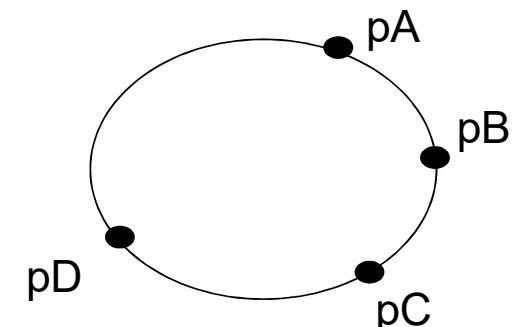
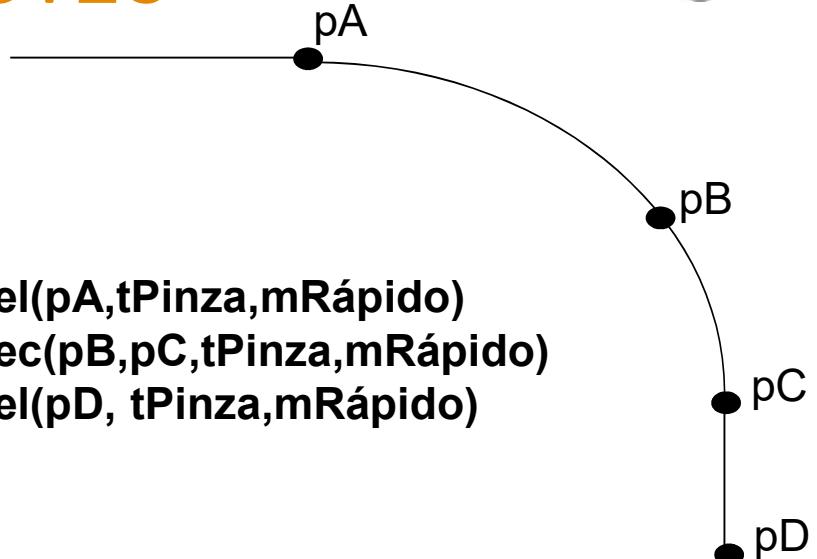
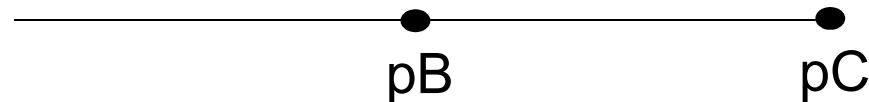
`movec (point,point,tool,mdesc)`

- Interpolación circular:
- El punto de paso nos definirá el arco de circunferencia.
- Mdesc (Blending) especifica el redondeo final del arco
- Un círculo se puede hacer con 4 puntos

`movec(pB,pC,tPinza,mRápido)`

`movec(pD,pA,tPinza,mRápido)`

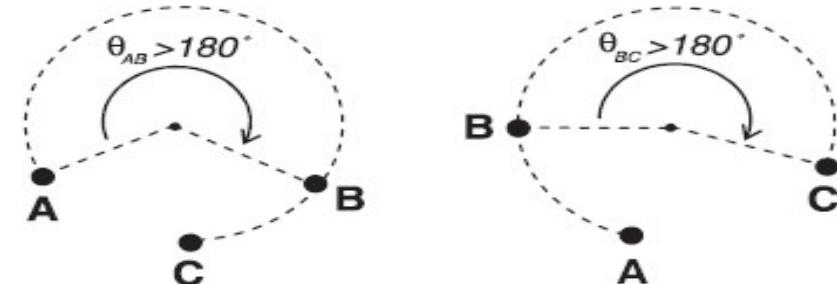
- Línea recta (si puntos alineados): `movec(pB,pC,tPinza,mRápido)`



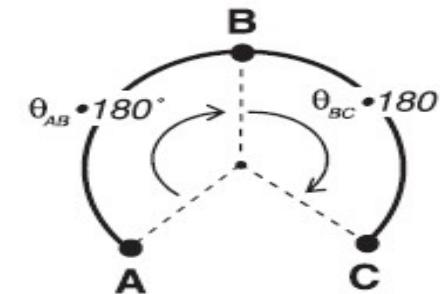
# MOVEC: PARTICULARIDADES

STÄUBLI

Movimientos imposibles  
(más de una solución)



- El punto intermedio ha de tener un ángulo inferior a  $180^\circ$  entre el punto de inicio y fin

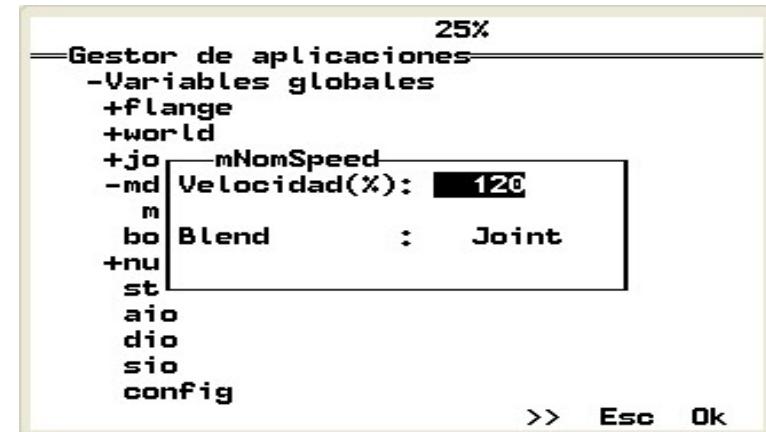


Ver capítulo «Control de los Movimientos → Control de Trayectoria » en el Manual de Referencia del VAL3

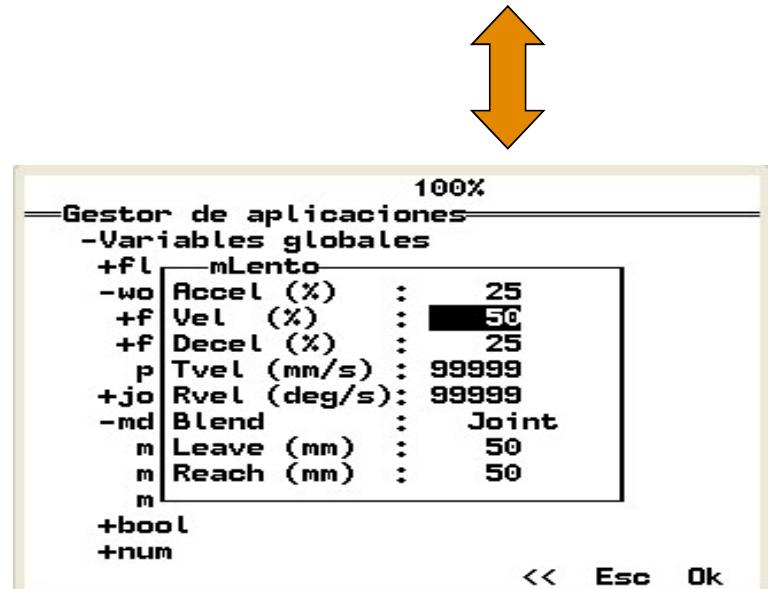
# DESCRIPTORES DE MOVIMIENTOS

VAL3>=6.5

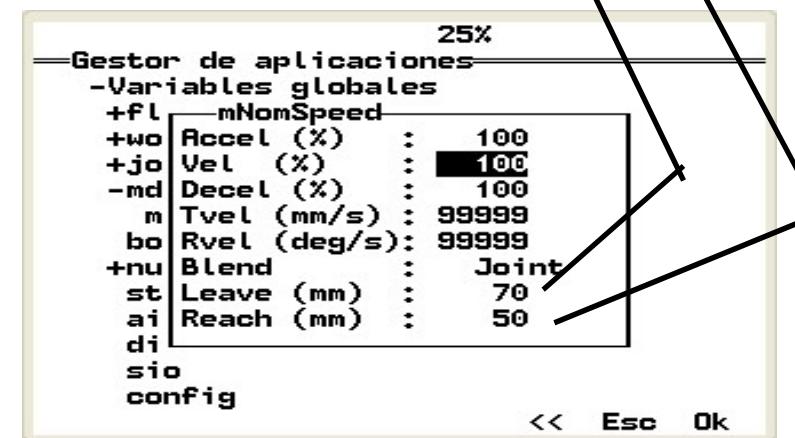
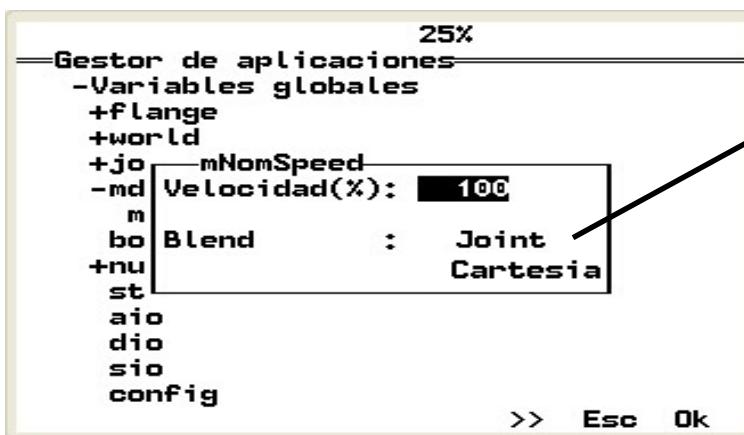
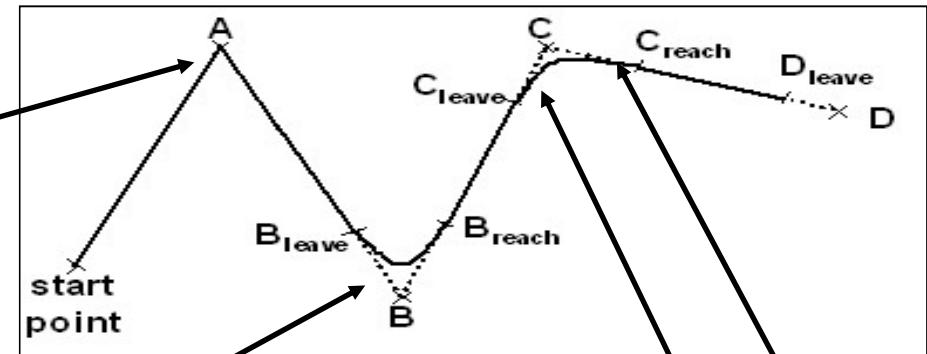
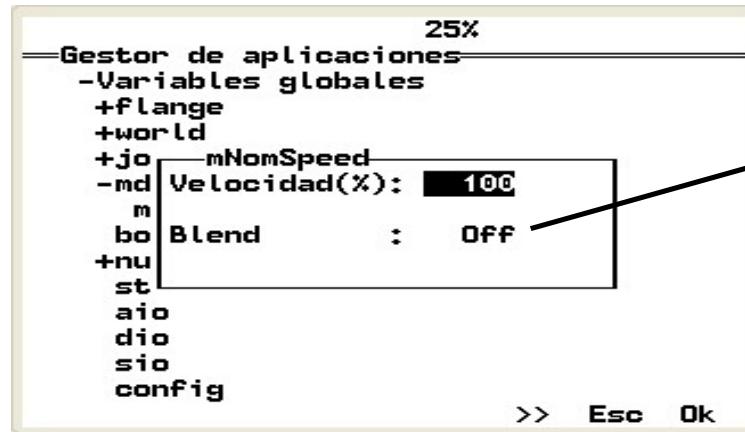
- Página de edición simplificada (<<) :
- Ajuste automático de la acel/decel
- Optimizados los valores de vel/acel
- $\text{acel} = \text{vel}^2$
  
- Ejemplo : velocidad=120%  $(1,2)^2 = 1,44$
  
- Resultado : aceleración=deceleración= 144%
  
- Para ajustes personalizados, ir a pagina avanzada (>>) con F6



- **Vel:** Velocidad máxima permitida en % del valor nominal del robot (ejes).
- **Accel:** Aceleración máxima permitida en % del valor nominal del robot (ejes).
- **Decel:** Deceleración máxima permitida en % del valor nominal del robot (ejes).
- **Tvel:** Máxima velocidad de translación en mm/s en el punto central de la herramienta.
- **Rvel:** Máxima velocidad de rotación en grados /s en el punto central de la pinza.
- **General:** Los valores más restrictivos son los utilizados.



## Blend: Redondeo

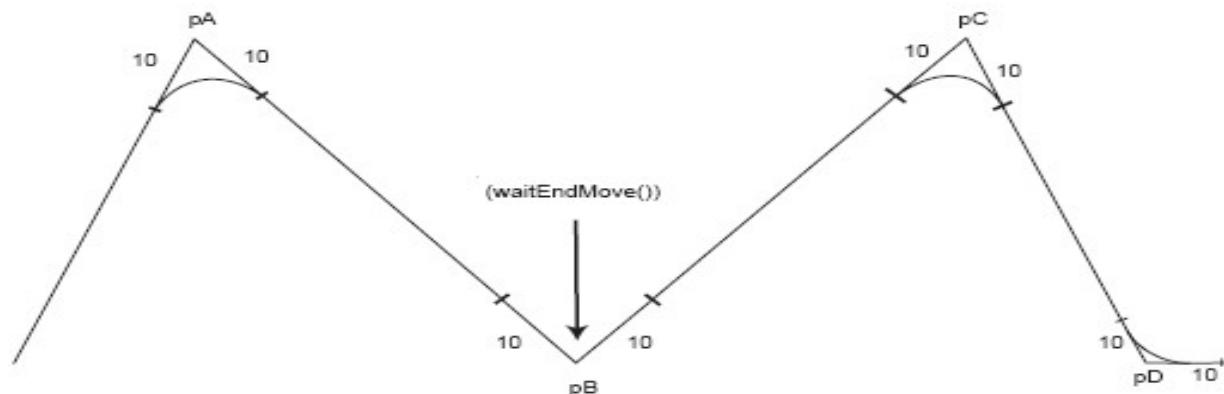


Tipos de Blend en VAL3 => 7:

- Joint
- Cartesiano

	Joint	Cartesiano
Tipo de movimiento	Libre	Tipo circular
Aceleración / deceleración	Más rápido que el Cartesiano	Precisa
Velocidad	Más rápido que el Cartesiano	Precisa

La instrucción waitEndMove (), open() y close () anulan el efecto del blend.



# OPTIMIZACIÓN: REGISTRO DE TRAJECTORIAS

STÄUBLI

Es posible registrar la trayectoria real del robot: Posiciones de los motores + el par de los motores. Todo esto cada 4ms y durante aproximadamente 60 segundos.

VAL3 6+

Conectar una USB de Stäubli.  
(configuración: records.cfx)

1



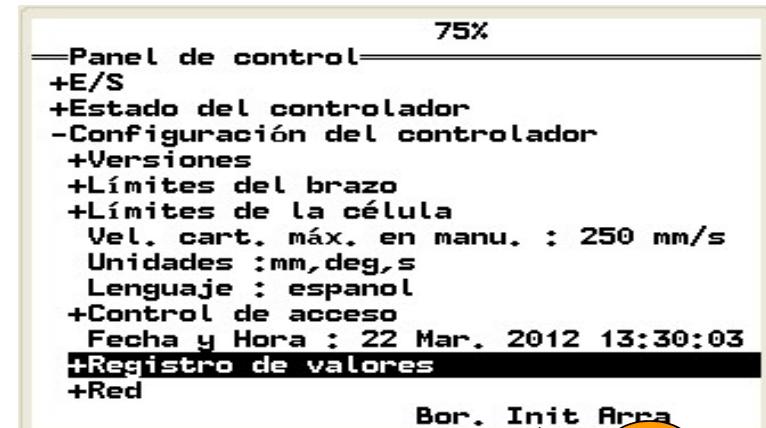
Iniciar el registro



No parar el registro,  
mejor esperar a que  
finalice.

Repetir el paso 3, 4, 5 si fuera necesario → records1, records2, .....

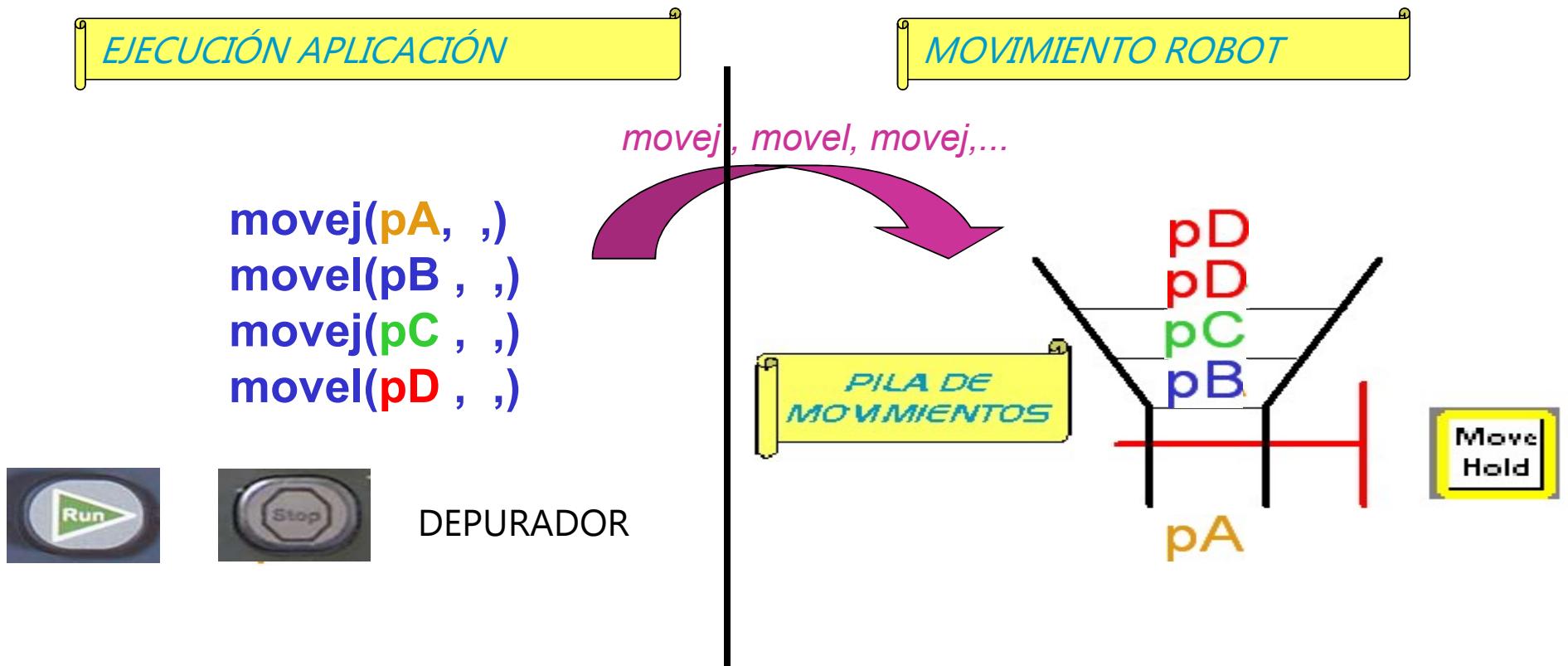
El servicio técnico puede analizar los resultados y detectar un riesgo de degradación prematura.



Guardar el resultado en la  
USB (records.rec)

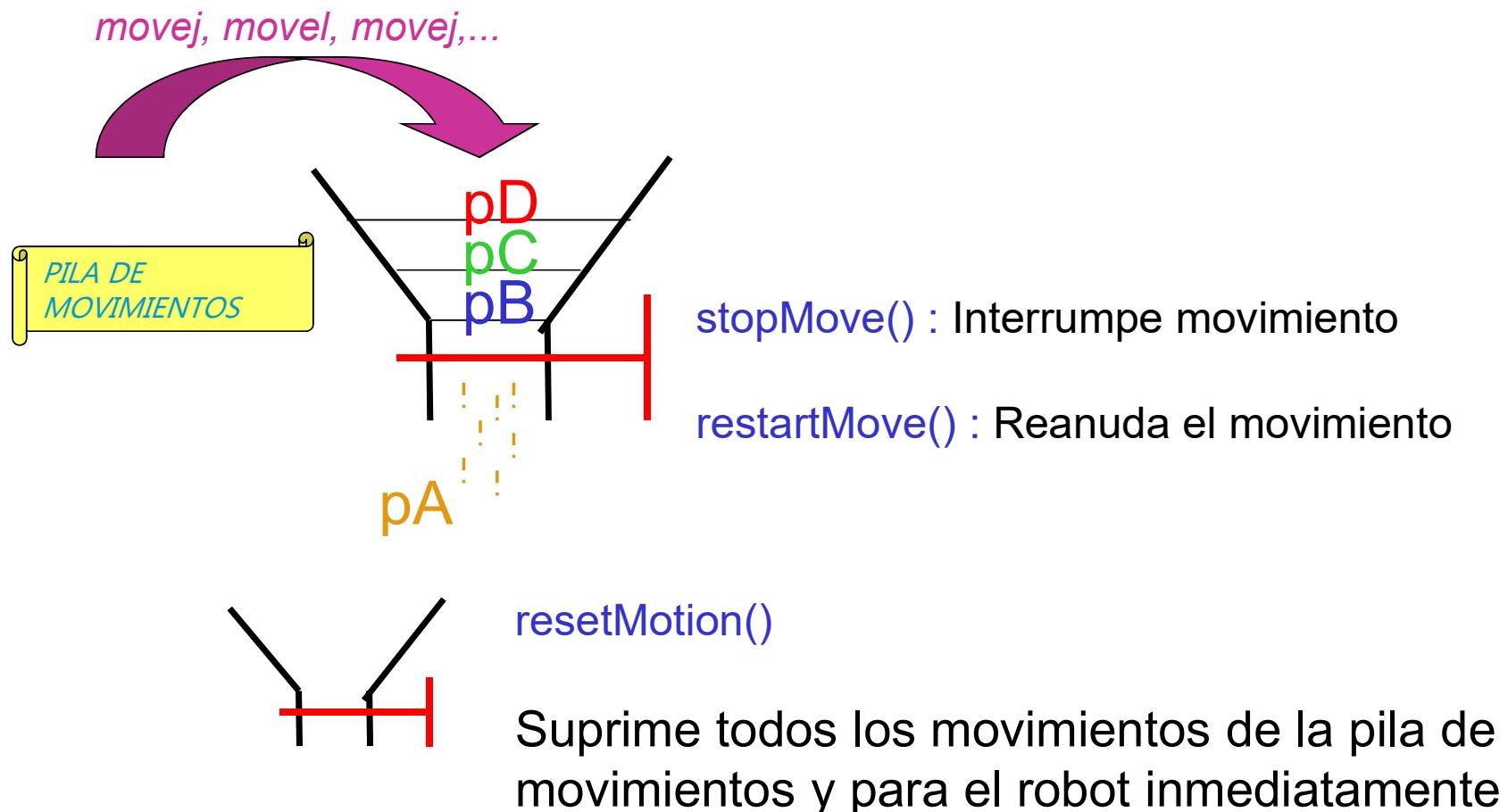
# APLICACIÓN Y MOVIMIENTO

STÄUBLI



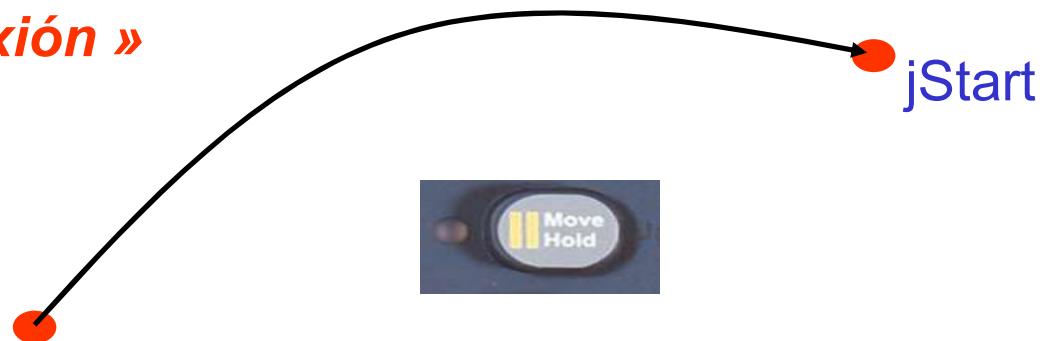
# CONTROL DE MOVIMIENTO

STÄUBLI



## «Movimiento de Conexión »

Posición Actual



- `resetMotion( jStart)` donde `jStart` es un punto en joint vacía la pila de movimientos y genera un movimiento de conexión hacia `jStart`.  
Por defecto, el movimiento de conexión en modo remoto se ejecuta manteniendo pulsada la tecla MOVE/HOLD.

`autoConnectMove(true)` los movimientos de conexión son ejecutados automáticamente a baja velocidad (útil si se trabaja sin MCP).

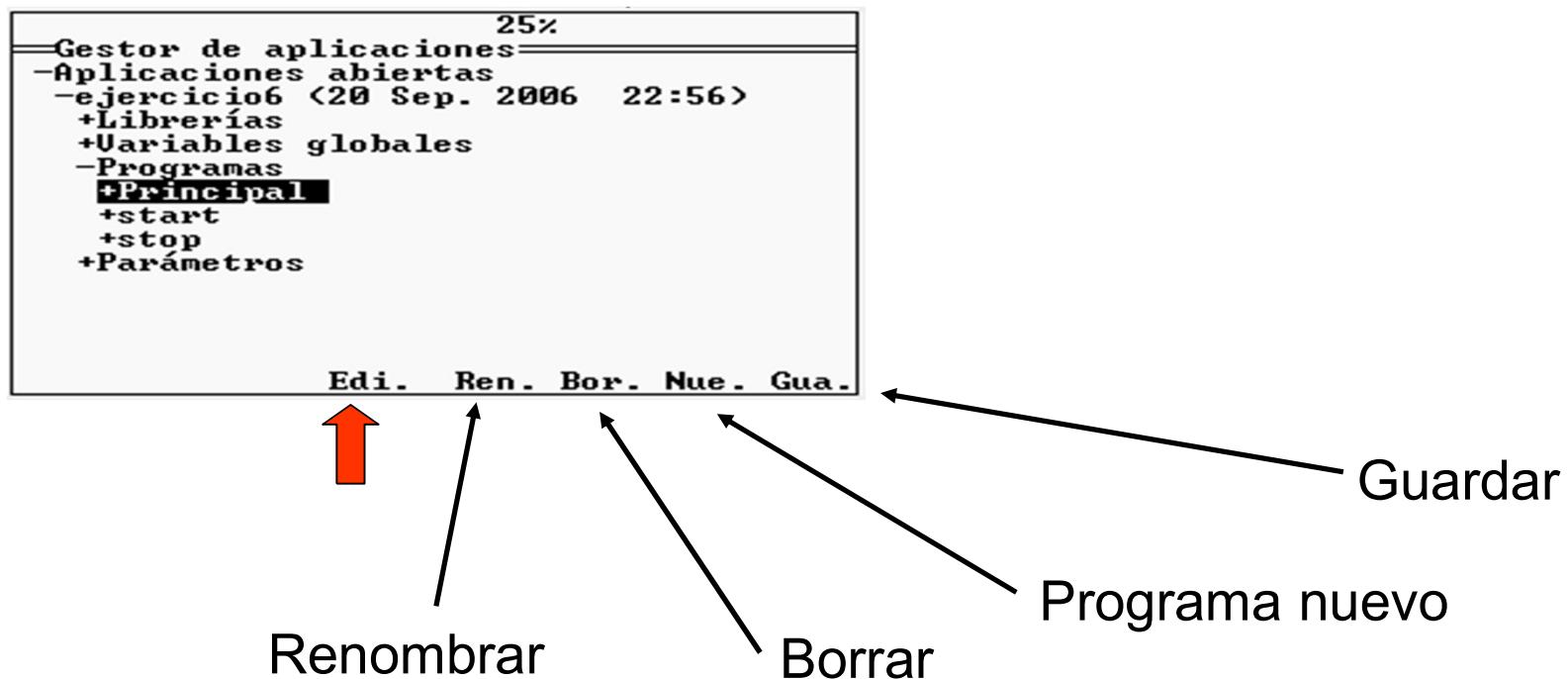
FORMACIÓN VAL3

*Edición  
de  
Programas*



## EDITOR

- Desde el «Gestor de aplicaciones», seleccionar el nombre del programa y presionar RETURN o el botón Edi.



## REGLAS DE EDICIÓN

- La sintaxis de cada línea debe ser correcta ==> todas las variables deben estar definidas (no necesariamente inicializadas)
- Una sola instrucción por línea
- El tipo de instrucción dada no puede ser modificada.
- (ej.: reemplazar una instrucción de movimiento por un cálculo)

```

25%
==LuerLockMov()
//Iniciar contadores para paleti
l_nContadorZ=0
l_nContadorY=0
l_nContadorX=0
-do
+do
    l_nContadorZ=0
    until (bFinalCiclo==true)
    bMovifFinalizado=true
-if ((l_nNumPiezasX==0) or (l_nNumPie
    //doutCajaLlena=true
    //wait (dinCajaNueva==true)
    //doutCajaLlena=false

```

Paro      Marc    Peg.   Cop.   Bor.   Ins.   Gua.

Marcar líneas para copiar / borrar / comentar

Pegar el contenido del portapapeles

Insertar una nueva línea después de la selección

# EDICIÓN DE UNA LÍNEA

- Escritura semiautomática de palabras : escriba las primeras letras de la palabra y presione uno de los botones para seleccionar de la lista.
- Presione RETURN para validar.

```
25%
=====LuerLockMovi()=====
//Iniciar contadores para paleti
l_nContadorZ=0
l_nContadorY=0
l_nContadorX=0
-do
+do
    l_nContadorZ=0
until (bFinalCiclo==true)
bMoviFinalizado=true
-if ((l_nNumPiezasX==0) or (l_nNumPie
//doutCajaLlena=true
//wait (dinCajaNueva==true)
move
```

Variables locales

Entradas  
Salidas

Subprogramas

Instrucciones Val3

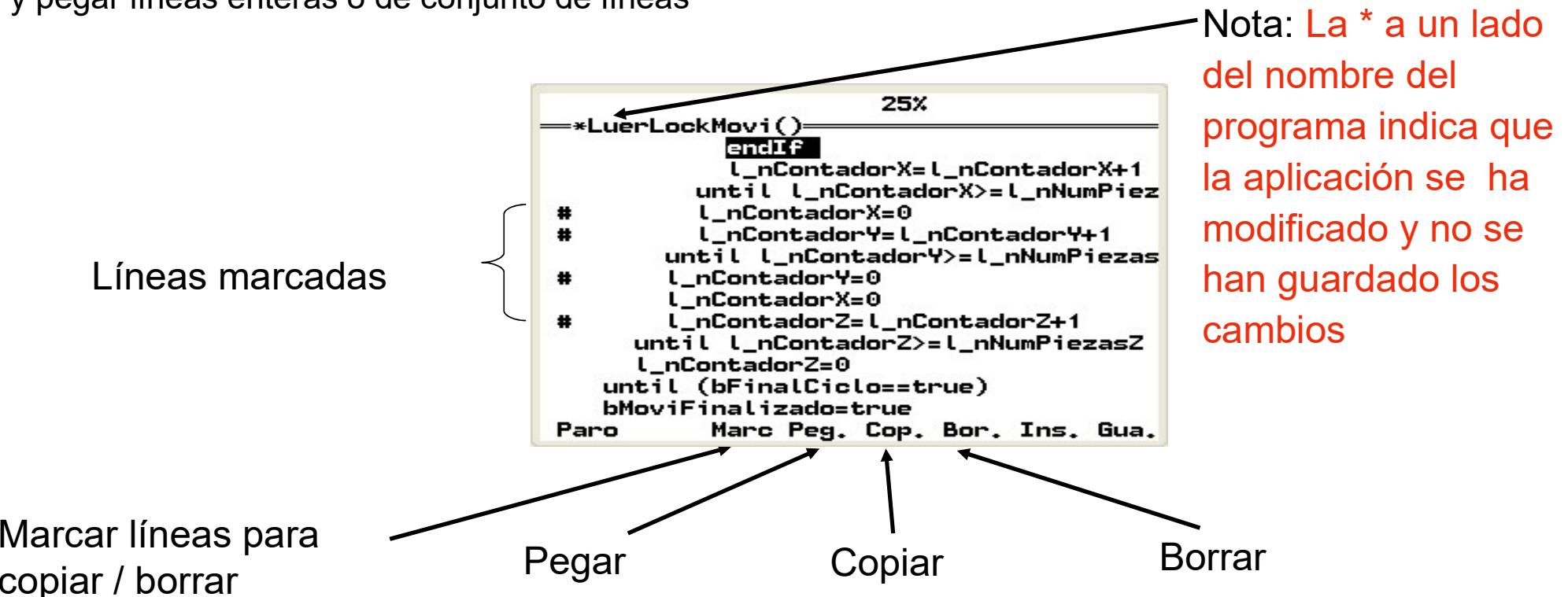
Variables  
Globales

His. Loc. E/S Prg. Glo. VAL3

# OPERACIONES DE EDICIÓN

STÄUBLI

Copiar y pegar líneas enteras o de conjunto de líneas



Marcar líneas para copiar / borrar

*Para desplazar un conjunto : Marcar + Copiar + Borrar + Pegar*

## RESUMEN DE EDICIÓN

- Utilizar el botón « RETURN » para reeditar una línea existente.
- Utilizar el botón « ESC » para anular las modificaciones a una línea que este editando.
- Escribir una línea carácter por carácter.
- Escribir una línea sin escribir un solo carácter únicamente seleccionando elementos de la lista.
- Método más rápido : escoger el menú deseado y escribir las primeras letras, cuando la palabra deseada sea seleccionada, validar « RETURN »
  - ejemplo : añadir la instrucción : movej(joint, tool,mdesc)
    - botón « Ins. » para insertar un nueva línea.
    - botón « VAL3 » para escoger de la lista de instrucciones.
    - escribir « m » + « o » + botón « Ok » para insertar la línea.
    - completar los 3 parámetros con el botón « Glo. ».

FORMACIÓN VAL3

# *Aproximación a Puntos Cartesianos*



## VARIABLE DE TIPO : TRSF

Variable transformada TRSF permite hacer operaciones matemáticas con los puntos cartesianos.

Ex : Aproximación a un punto, paso entre las posiciones de una paleta , ....

6 campos numéricos : x, y, z, rx, ry, rz

Si transformada trPaso está declarada como variable

Uso dentro de un programa:

trPaso ={0,0,-100,0,0,0}

trPaso.x=0 trPaso.y=0 trPaso.z=-100 trPaso.rx=0 ...

Imposible utilizarlo para movimientos

!! Reservado ÚNICAMENTE para cálculos !!

# APROXIMACIÓN A UN PUNTO

POINT <= appro(POINT,TRSF)

*APPRO calcula un punto Cartesiano relativo a un punto sobre el cual es aplicada la transformada*

POINT pt   POINT pCoger   TRSF trAprox   NUM nDistancia=100

1 - Forma:

```
trAprox={0,0,-nDistancia,0,0,0}
pt=appro(pCoger,trAprox)
movej(pt,tPinza,mRápido)
```

2 - Forma:

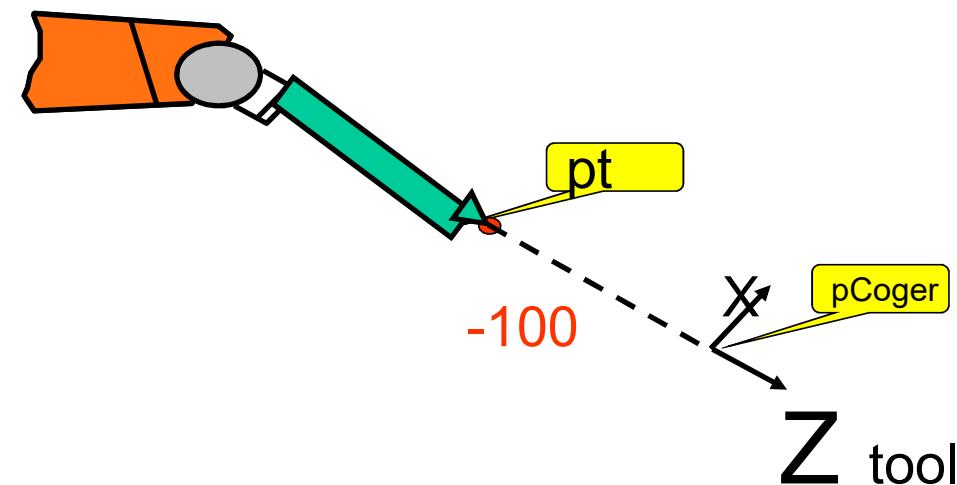
```
trAprox={0,0,-nDistancia,0,0,0}
movej(appro(pCoger,trAprox),tPinza,mRápido)
```

3 - Forma:

```
trAprox={0,0, -100,0,0,0}
movej(appro(pCoger,trAprox),tPinza,mRápido)
```

4 - Forma:

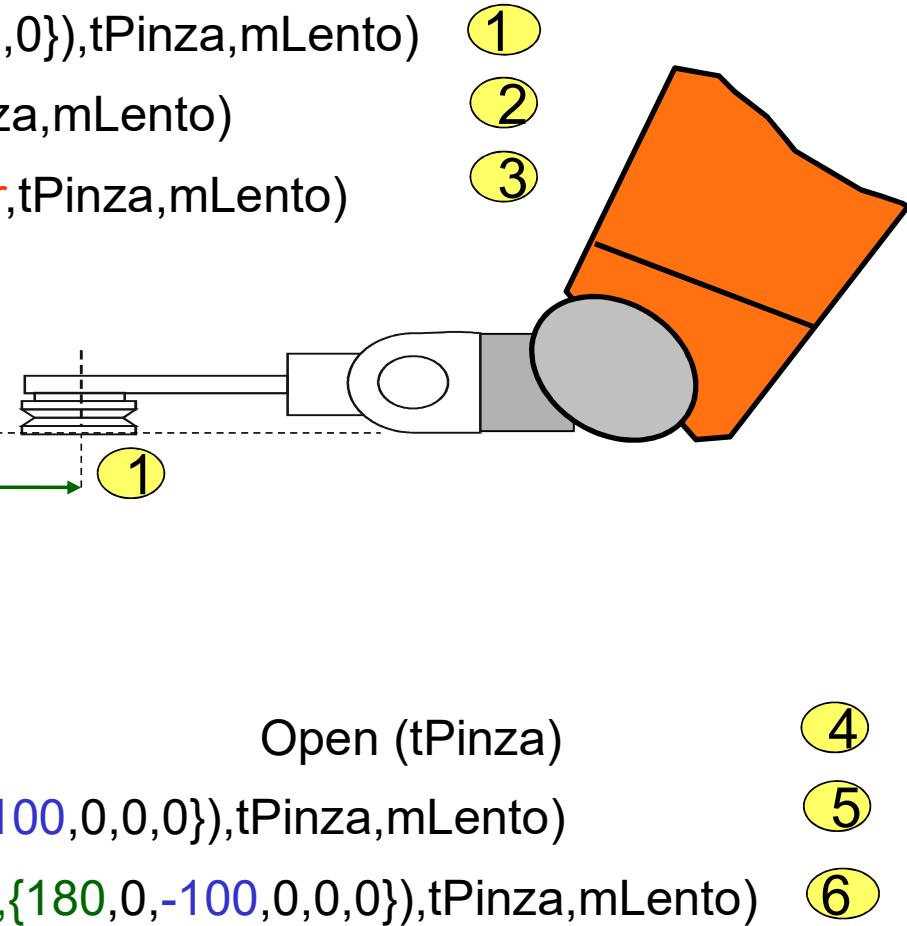
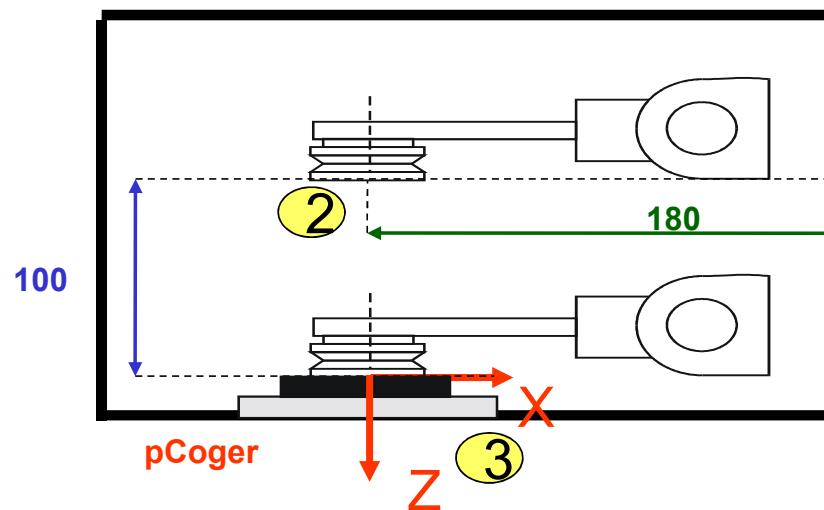
```
movej(appro(pCoger,{0,0,-100,0,0,0}),tPinza,mRápido)
```



# APROXIMACIÓN COMPLEJA - 1 -

STÄUBLI

```
movej(appro(pCoger,{180,0,-100,0,0,0}),tPinza,mLento)    1  
movel(appro(pCoger,{0,0,-100,0,0,0}),tPinza,mLento)  
    movel(pCoger,tPinza,mLento)    2  
    3
```

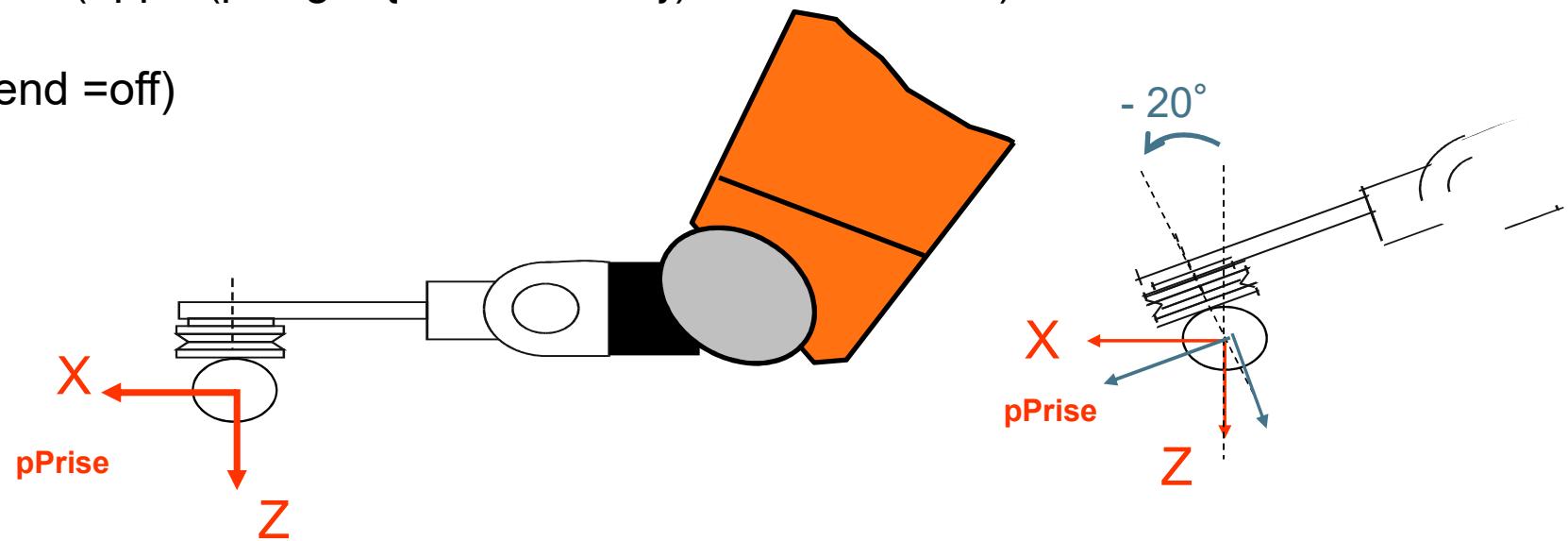


## APROXIMACIÓN COMPLEJA - 2 -

STÄUBLI

```
move(pCoger,tPinza,mLento)  
move(appro(pCoger,{0,0,0,0,-20,0}),tPinza,mLento)
```

(Blend =off)



```
move(appro(pCoger,{0,-100,0,0,-340,0}),tPinza,mLento)  
move(pCoger,tPinza,mLento)
```

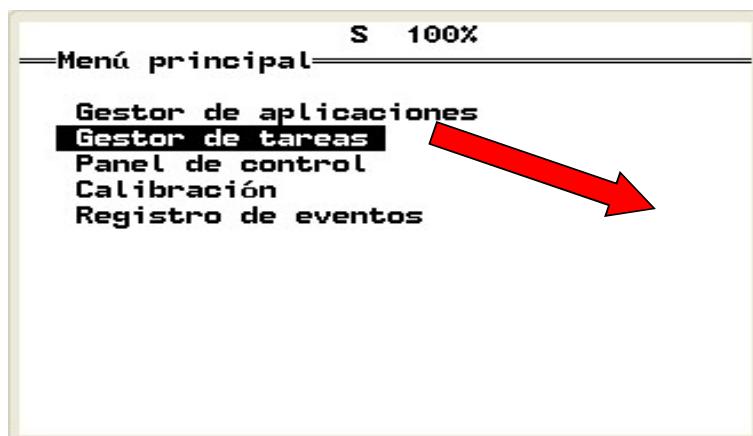
FORMACIÓN VAL3

# Gestor de tareas y *Depurador*



# GESTOR DE TAREAS

Cuando una aplicación se está ejecutando, el estado de sus áreas es mostrado en el « Gestor de tareas »



Gestor de tareas					
Nombre	Estado	Apli	Pri	Err	
Movimientos	RUNNING	Nub3dV2	30	0	
Finalizar	RUNNING	Nub3dV2	10	0	

Info Dep. Cer. Rea. Sus.

Entrar al depurador y para la tarea

Terminar la tarea

Reanudar una tarea parada

Suspender una tarea que se esté ejecutando

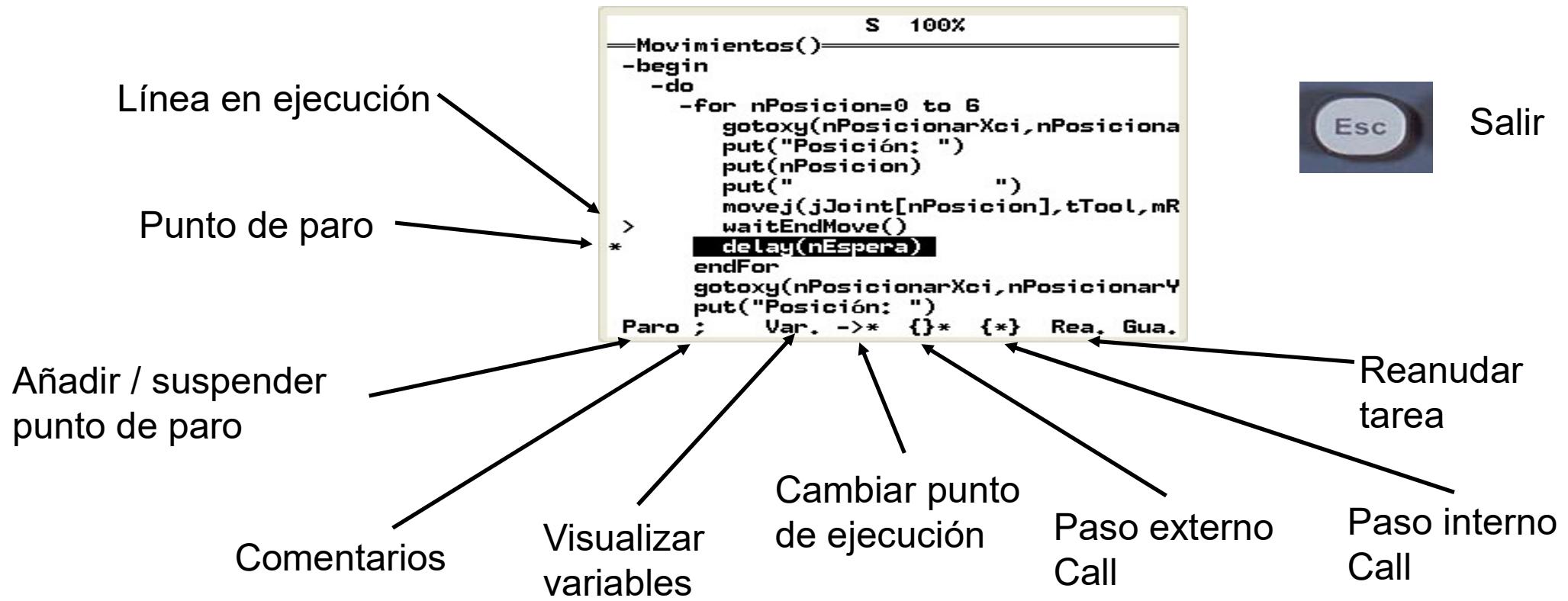
Código de error. 0 sin error

Mostrar el mensaje de error

Prioridad de la tarea

## DEPURADOR

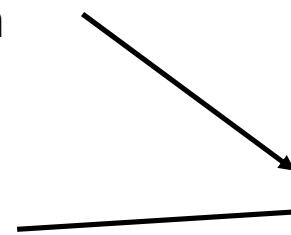
- Autoriza la ejecución del programa paso a paso y la visualización o modificación de variables



# VISUALIZACIÓN DE VARIABLES

Visualización/modificación de variables durante el paso a paso

Línea en ejecución



Punto de paro



The screenshot shows a software interface with a code editor and a variable viewer. The code editor contains pseudocode for a 'Movimientos()' function. The variable viewer on the right lists variables and their values, with 'Instrucciones' highlighted. Buttons at the bottom include 'Gua.', '{}\*', '{\*}', and 'Sali'.

```
=Movimientos()
-begin
-do
-for nPosiciones=
    gotoxy(nPosi
    put("Posició
    put(nPosicio
    put("
    movej(jJoint
    waitEndMove(
    delay(nEsper
    endFor
    gotoxy(nPosici
    put("Posición:
    Gua.  {}*  {*}  Sali
```

Variables  
-Instrucciones  
jJoint[0]  
mRapido  
nEspera=5  
nPosicion=0  
tTool  
Variables locale  
Parámetros

Paso externo Call

Paso interno Call

Visualización /  
edición de  
variables con la  
tecla Return

FORMACIÓN VAL3

# Gestión de entradas / salidas *digitales*



# E/S DIGITALES RX/CS8

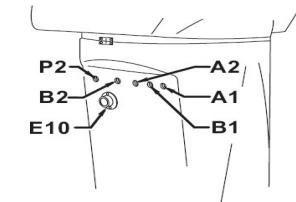
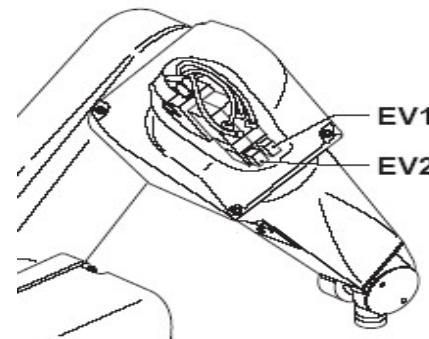
**De base :**

2 Salidas a válvulas en el brazo :

- UserIO: valve1, valve2
- 2 entradas digitales en HAN72
  - usrln0 & usrln1

**En opción :**

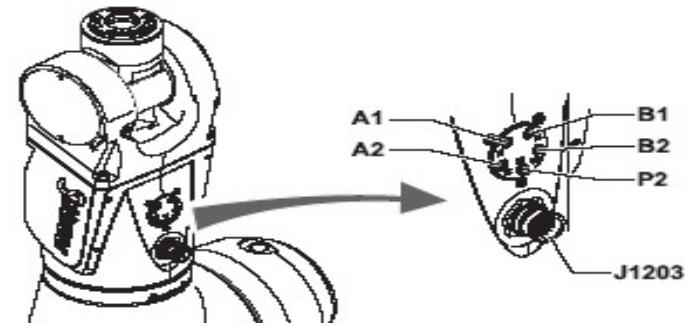
- tarjeta BasicIO : 16 entradas + 16 salidas
- tarjeta ModuleIO : 32 entradas + 32 salidas



# E/S DIGITALES TX/CS8C

## De base :

- 2 Salidas a válvulas en el brazo  
(excepto TX 40: 1 en opción)
  - UserIO: valve1, valve2
- 2 entradas digitales usrln0 & usrln1
- FastIO : 2 entradas + 1 salida rápidas
- Modbus TCP tipo Servidor.



## Opcional :

- BIO1 (tarjeta No.1) : 16 entradas + 16 salidas
- BIO2 (tarjeta No.2) : 16 entradas + 16 salidas
- Tarjetas de bus de campo: Profibus, DeviceNet, CanOpen, Modbus, Ethercat, SercosIII, PowerLink, Profinet, Ethernet/IP, ...
- EncoderIO : Tarjetas gestoras de encoders.



## BIO

Gama de tensión de funcionamiento	0 a 24 VDC
Gama de tensión del estado "OFF" (bajo)	0 a 3 VDC
Gama de tensión del estado "ON" (alto)	11 a 24 VDC
Tensión media de umbral	Vin = 8 VDC
Gama de intensidad de funcionamiento	0 a 6 mA
Gama de intensidad en el estado bloqueado	0 a 0.5 mA
Gama de intensidad en el estado de marcha	2 a 6 mA
Corriente media de umbral	2.5 mA
Impedancia (Vin / lin)	3.9 kΩ mínimo
Intensidad con Vin = 24 VDC	lin ≤ 6 mA
Tiempo de respuesta hardware y software	15 ms max
Tensión de aislamiento / linea de fuga	2,5 kV / 4 mm

### Salidas

Gama de tensión de alimentación (24 V Fast-Out)	12 a 28 VDC
Corriente consumida	5,5 mA
Gama de tensión de salida	12 a 28 VDC
Gama de utilización de corriente de salida	1 a 250mA
Caída de tensión en salida para I = 250 mA	1,2 V max
Resistencia de salida en estado activado	1 Ω
Corriente de fuga máxima en estado bloqueado	5 µA
Tiempo de respuesta hardware + software	50 µs
Limitación en corriente de salida (sobrecarga)	2 A

## USERS

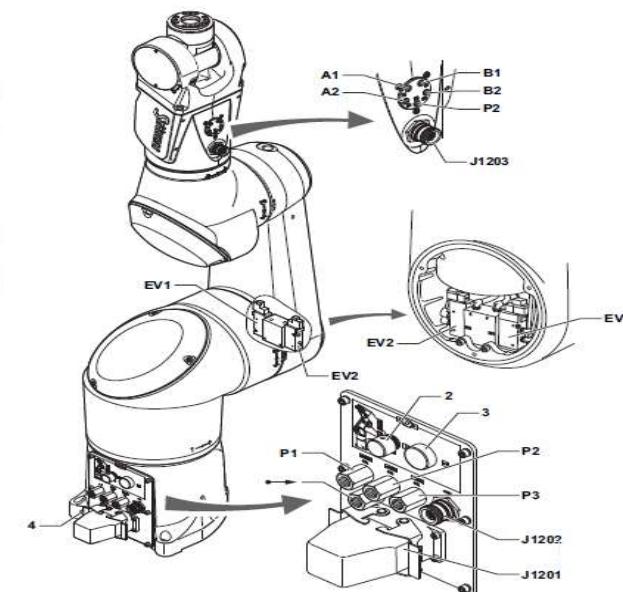
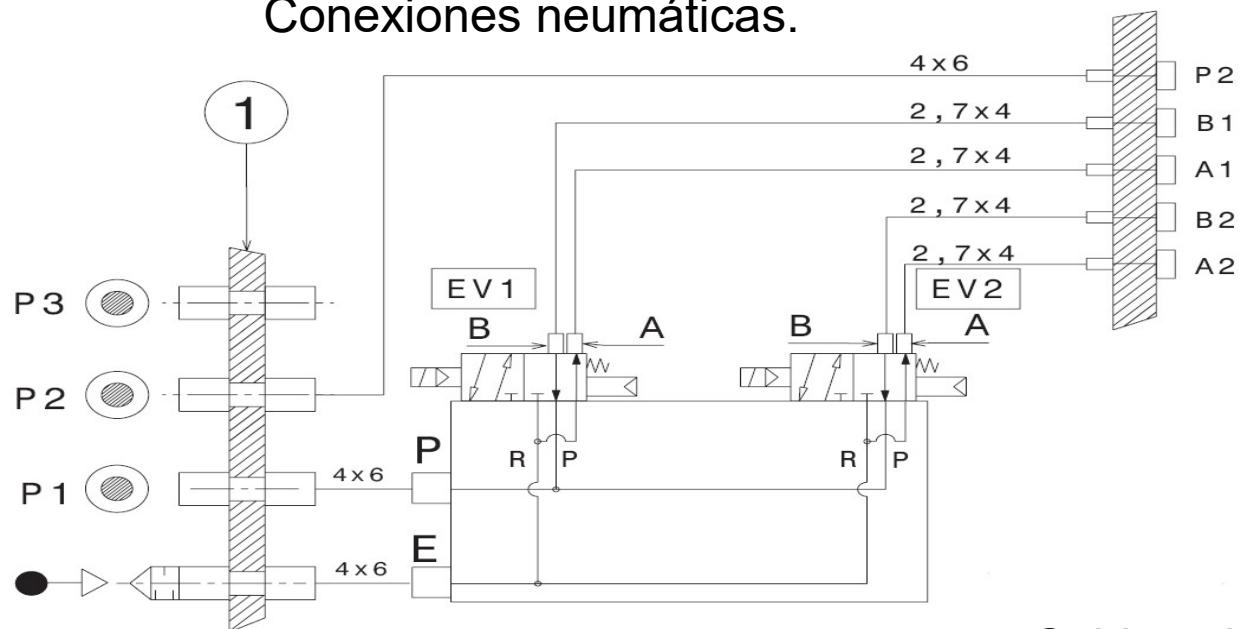
Gama de tensión de funcionamiento	0 a 30 VDC
Gama de tensión del estado "OFF" (bajo)	0 a 1 VDC
Gama de tensión del estado "ON" (alto)	4 a 30 VDC
Gama de intensidad de funcionamiento	0 a 240 µA
Gama de intensidad en el estado bloqueado	0 a 5 µA
Gama de intensidad en el estado de marcha	33 a 240 µA
Impedancia	100 kΩ
Tiempo de respuesta hardware + software	6,5 ms max

### Entradas

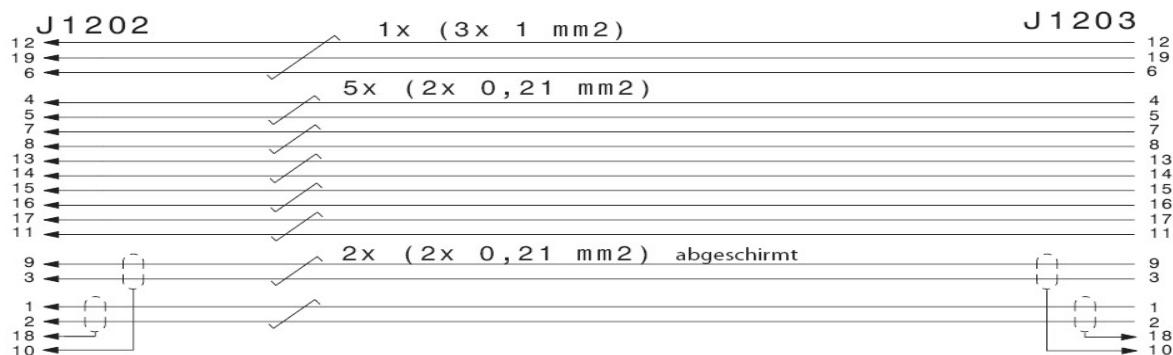
## FAST

Gama de tensión de funcionamiento	0 a 30 VDC
Gama de tensión del estado "OFF" (bajo)	0 a 2 VDC
Gama de tensión del estado "ON" (alto)	6 a 30 VDC
Gama de intensidad de funcionamiento	0 a 9mA
Gama de intensidad en el estado bloqueado	0 a 0,5mA
Gama de intensidad en el estado de marcha	2 a 9mA
Impedancia	3,3 kΩ
Tiempo de respuesta hardware + software	50 µs

## Conexiones neumáticas.

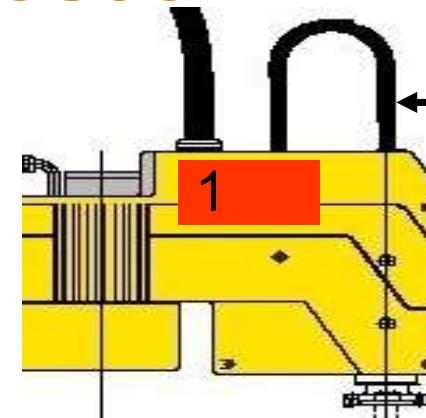


Cables eléctricos para uso del cliente.

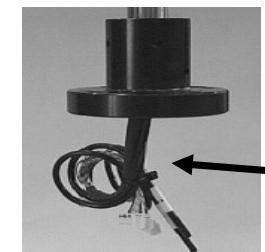


# E/S DIGITALES RS/CS8C

1. Tarjeta ARMIO en el brazo RS  
(Salvo RS40B es opción)
  - 8 entradas digitales (8DI)
  - 8 salidas digitales (8DO)
  - 4 entradas analógicas (4AI)
  - 4 salidas analógicas (4AO)



2. Opción : Cable que conecta la tarjeta ARMIO hasta la brida + 2 tubos de aire (por dentro del eje 3-4)



3. Opción : Conector de Herramienta TC
    - Permite montar 4 válvulas opcionales en la brida
    - Conexión para 5 DI + 8 DO de la tarjeta CANIO
- + las mismas tarjetas disponibles en el CS8C / TX



# E/S DIGITALES

```
=Panel de control=
-E/S
-Controlador
+UserIO
+BasicIO
+ModuleIO
+UsbIO
+CpuIO
```

RX



```
=Panel de control=
-E/S
-Controlador
+UserIO
+BasicIO-1
+BasicIO-2
+PtcFaultIO
+FastIO
+RsIO
+CpuIO
+EncoderIO
```

TX

```
=Panel de control=
-E/S
-Controlador
+UserIO
+BasicIO-1
+BasicIO-2
+PtcFaultIO
+FastIO
+RsIO
+CpuIO
+EncoderIO
-CanIO
+Can-Dio
+Can-Aio
```

RS

25%

```
=Panel de control=
-Salidas
0 Ini_Ciclo = On <inversa>
1 Fin_Ciclo = Off <directa>
2 bIn2 = Off <directa>
3 bIn3 = Off <directa>
4 bIn4 = Off <directa>
5 bIn5 = Off <directa>
6 bIn6 = Off <directa>
7 bIn7 = Off <directa>
8 bIn8 = Off <directa>
9 bIn9 = Off <directa>
10 bIn10 = Off <directa>
11 bIn11 = Off <directa>
Off Modo Ren.
```

Para asignar una salida digital a los botones 1, 2 o 3 :

Seleccione la salida en el panel de control y presione  
“Mayusculas + 1, 2 o 3”

## UTILIZACIÓN DE E/S

Modo lógico :

Directa : estado físico activo => valor **true**

(24Vdc = true 0Vdc = false)

Inversa : estado físico activo => valor **false**

(0Vdc = true 24Vdc = false)

Nota: El nombre de la entrada o salida en el panel de control es una etiqueta que no está relacionada con los programas.

```

10%
_____
Panel de control
-Dentrad
X10 = OFF (directa) (bIn0)
X11 = OFF (directa) (bIn0)
X12 = OFF (directa) (bIn2)
X13 = OFF (directa) (bIn3)
X14 = OFF (directa) (bIn4)
X15 = OFF (directa) (bIn5)
...
(bIn6)
(bIn7)
(bIn8)
(bIn9)
(bIn10)
(bIn11)
Modo Des.

10%
_____
Panel de control
-Dentrad
X10 = OFF (directa) (bIn0)
X11 = OFF (directa) (bIn0)
X12 = OFF (directa) (bIn2)
X13 = OFF (directa) (InSensor)
X14 = OFF (directa) (bIn4)
X15 = OFF (directa) (bIn5)
X16 = OFF (directa) (bIn6)
X17 = OFF (directa) (bIn7)
X18 = OFF (directa) (bIn8)
X19 = OFF (directa) (bIn9)
X110 = OFF (directa) (bIn10)
X111 = OFF (directa) (bIn11)
Blq. Modo Des.

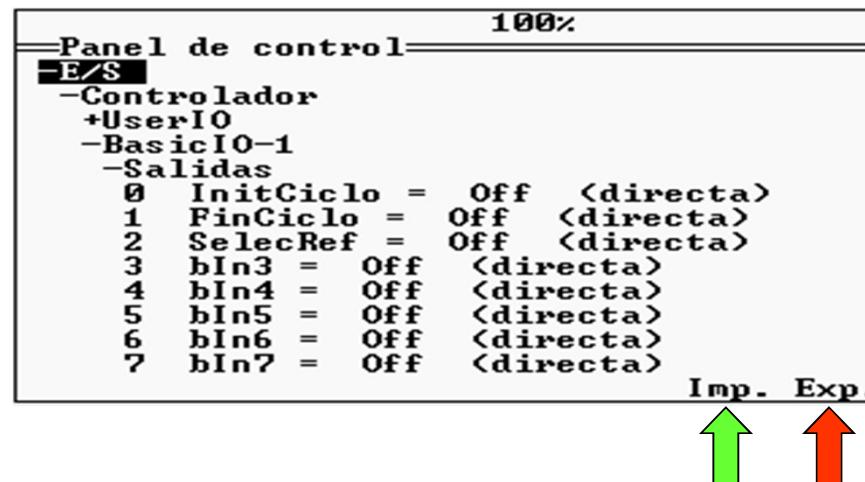
```

# IMPORTAR / EXPORTAR LAS E/S (VAL3 <7)

*Las E/S en el panel de control no son guardadas con la aplicación*

=>**Exportar/ Importar** la configuración de la célula:

- Trabajos fuera de línea en el emulador CS8
- Duplicar la configuración de la célula.



# CONTROL DE SALIDAS

Las DIO han de estar relacionadas con una entrada / salida física para poderse usar

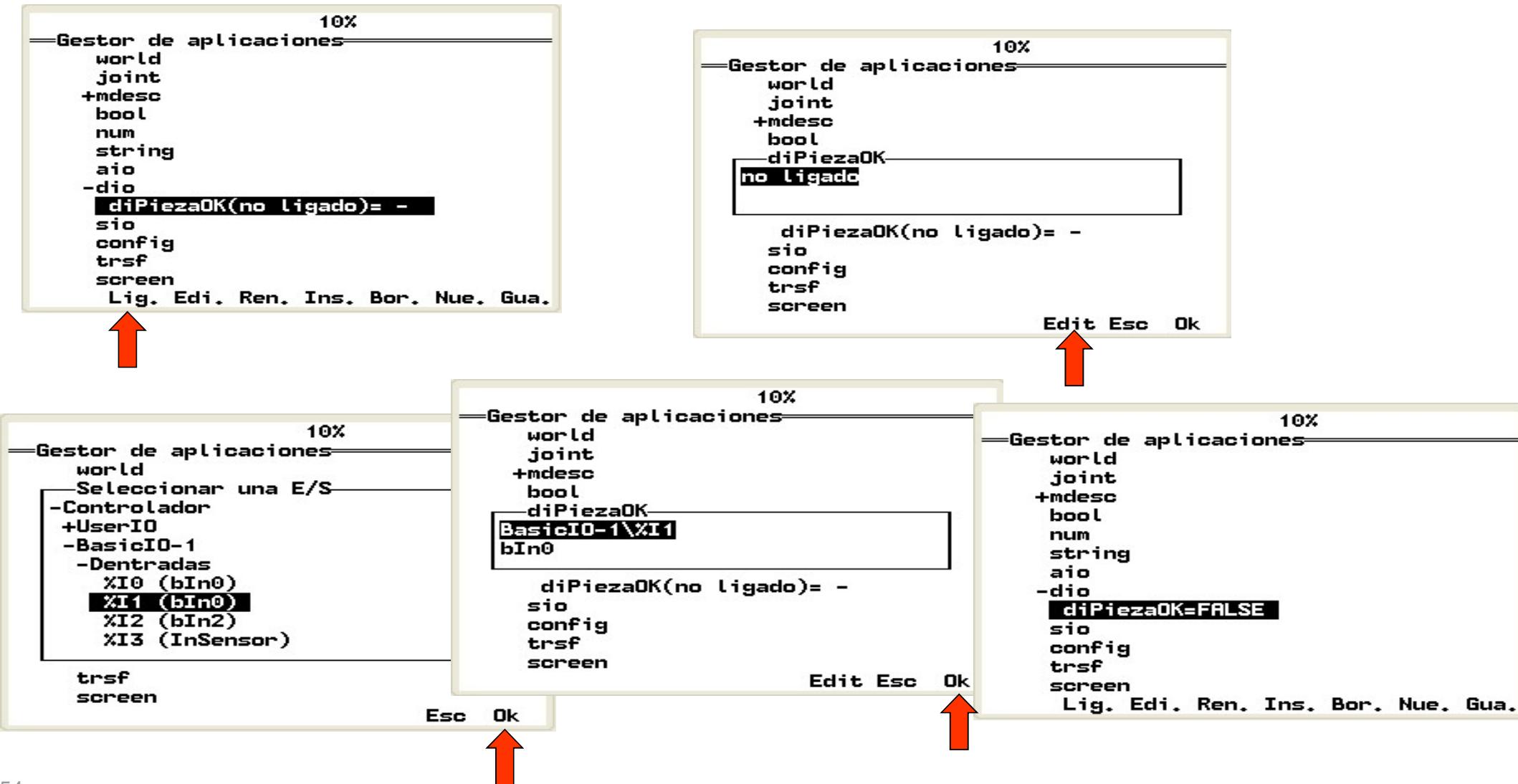
```
10%
==Gestor de aplicaciones
-Variables globales
  flange
  world
  +joint
  +mdesc
  bool
  +num
  string
  aio
-dio
  dinSensor(no ligado)= -
  sio
  config
    Lig. Edi. Ren. Ins. Bor. Nue. Gua.
```

Posterior <b>V7</b>	<p><b>Las DIO tienen que ser manualmente relacionadas desde el gestor de aplicaciones o el SRS</b></p> <p><b>Las DIO pueden ser relacionados con otras DIO</b></p> <p><code>dioLink(dio1,dio2)</code></p>
Anterior <b>V7</b>	<p><b>Las DIO pueden ser relacionadas con el nombre definido en el panel de control mediante el VAL3 :</b></p> <p><code>dioLink(diStart,io:bIn0)</code></p>

```
-dio
  diStart(bIn0)=FALSE
  sio
  config
```

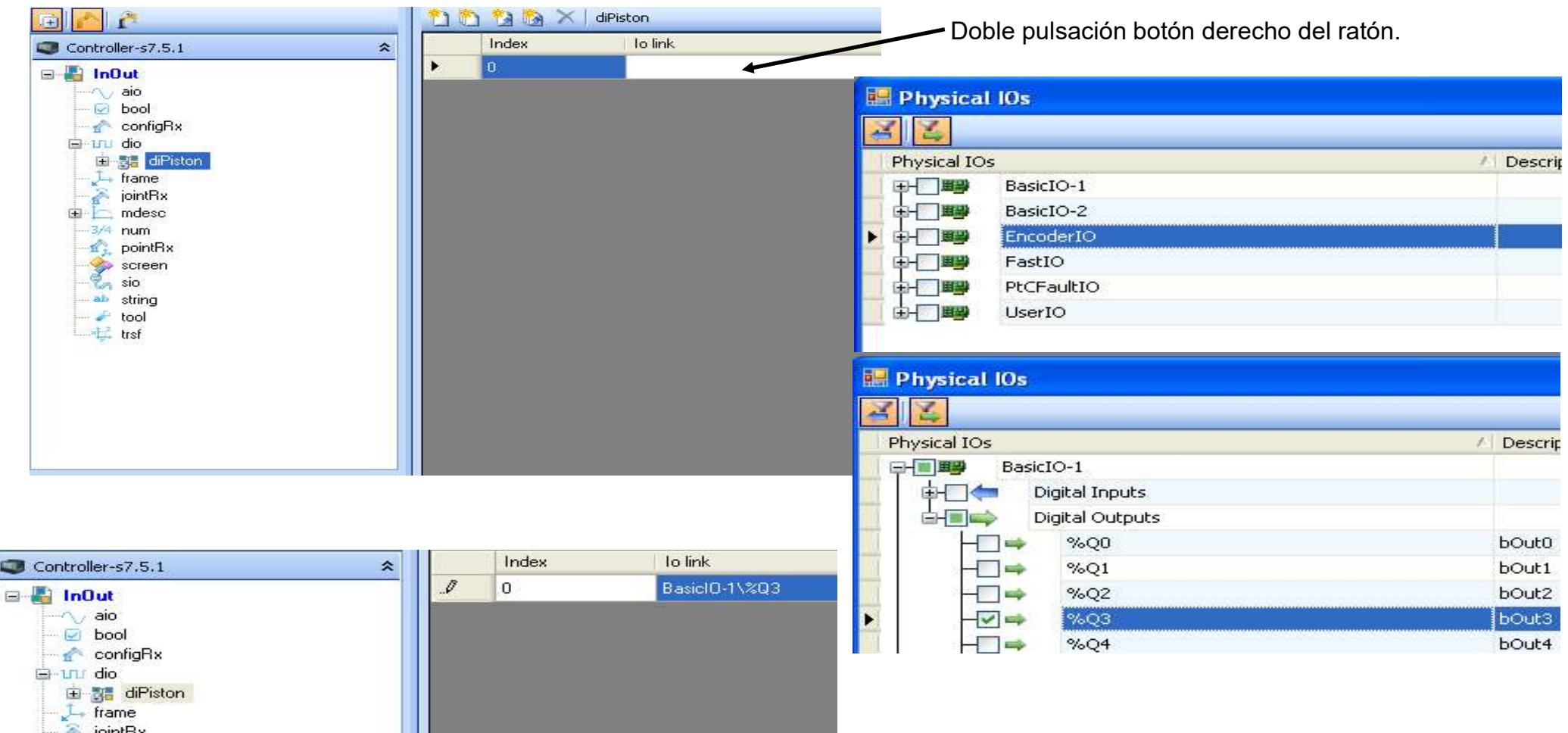
# VÍNCULO DIO EN SP1

STÄUBLI



# VÍNCULO DIO EN VAL3 STUDIO

STÄUBLI



# CONTROL DE SALIDAS

Definida la salida : doutPiston

En el programa :

- dOutPiston=true para activar la salida.
- dOutPiston=false para desactivar la salida.

```
10%
—Panel de control—
-Dentrad as
XI0 = OFF (directa) (bIn0)
XI1 = OFF (directa) (bIn1)
XI2 = OFF (directa) (bIn2)
XI3 = OFF (directa) (bIn3)
XI4 = OFF (directa) (bIn4)
XI5 = OFF (directa) (bIn5)
XI6 = OFF (directa) (bIn6)
XI7 = OFF (directa) (bIn7)
XI8 = OFF (directa) (bIn8)
XI9 = OFF (directa) (bIn9)
XI10 = OFF (directa) (bIn10)
XI11 = OFF (directa) (bIn11)
Blq. Modo Des.
```



**Blq** → No actualiza el estado de una I/O a excepción del on / off manual en panel de control.

Si deseamos cambiar el estado de una entrada, tendremos que bloquear previamente dicha entrada.

También es posible bloquear una salida para que el programa no pueda actualizarla.

```
10%
—Panel de control—
-Dentrad as
XI0 = OFF (directa) (bIn0)
XI1 = OFF (directa) (bIn1)
XI2 = OFF (directa) (bIn2)
XI3 = [OFF] (directa) (InSensor)
XI4 = OFF (directa) (bIn4)
XI5 = [OFF] (directa) (bIn5)
XI6 = OFF (directa) (bIn6)
XI7 = OFF (directa) (bIn7)
XI8 = OFF (directa) (bIn8)
XI9 = OFF (directa) (bIn9)
XI10 = OFF (directa) (bIn10)
XI11 = OFF (directa) (bIn11)
DBlq On Modo Des.
```

## SINCRONIZACIÓN MOVIMIENTOS & SALIDAS

VAL3 : Anticipación de la ejecución de programas con respecto al movimiento => Para cambiar el estado de una salida en una punto dado :

waitEndMove()

movej(pControl ,tPinza, mRapido)

waitEndMove()

dOutStartControl= true

....

Anula el redondeo del movimiento precedente.

Genera un movimiento durante la ejecución paso a paso del programa.

# OPEN / CLOSE

Se utiliza para activar /desactivar una herramientas.

`open(tPinza)` o `close(tPinza)`

Las características son:

- Espera fin de movimientos + anular el redondeo.
- Activa o desactiva la salida relacionada con la herramienta.
- Incluye tiempos abertura y cierre de la pinza.

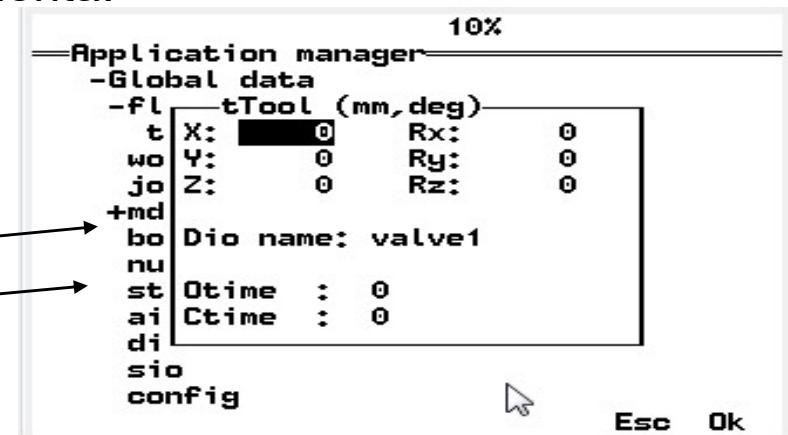
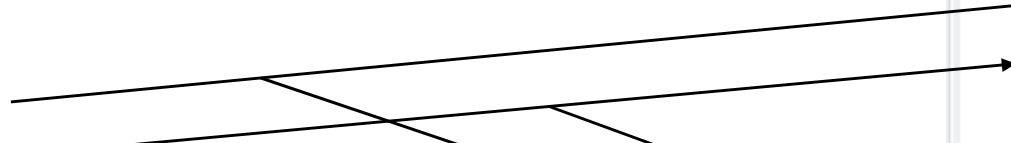
`open (tTool)`



`waitEndMove()`

`dotTool=true`

`delay(Otime)`



newProject1-start*		newProject1 Data-tTool* X									
Index	x	y	z	rx	ry	rz	IO Link	otime	ctime		
0	0	0	0	0	0	0	valve1	0	0		

## LECTURA DE ENTRADAS

Definida la entrada: dInCaptor

En el programa :

- La comparación `dInCaptor==true` regresa `true` si activa.
- La comparación `dInCaptor!=true` regresa `true` si inactiva.

Espera sobre una entrada : `wait(dInCaptor ==true)`

Espera con un tiempo máximo definido en segundos:

`bResultado=watch (dInCaptor ==true, 2)`

Regresa sobre la variable `bResultado` `true` si la condición se cumple antes de 2 segundos, de lo contrario regresa `false`

Copiar una entrada en una salida: `dOutPistor=(dInActivarPiston==true)`

## RETRASO: DELAY()

delay(nRetardo)      delay(seg)

Permite suspender la ejecución de un programa durante un tiempos dado en segundos, si no existe un waitEndMove() delante, el retardo se realizara durante el movimiento del robot.

movel(pControl, tPinza, mLento)

waitEndMove()      señal de medida en el punto  
doMedir= true      de control durante 2.5 seg.

delay(2.5)

doMedir=false

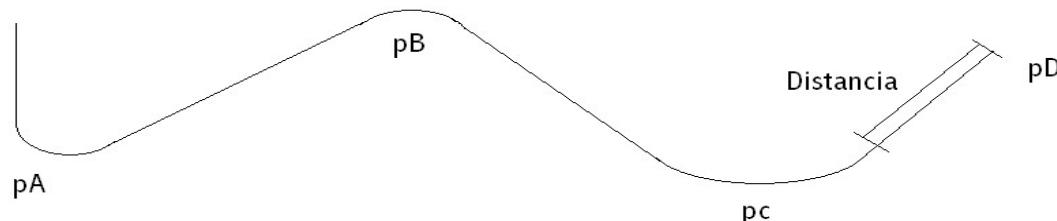
....

## GESTIÓN IN/OUT EN MOVIMIENTO - 1

La orden distance (punto1, punto2): devuelve la distancia lineal entre el punto 1 y el punto 2.

Ejemplo:

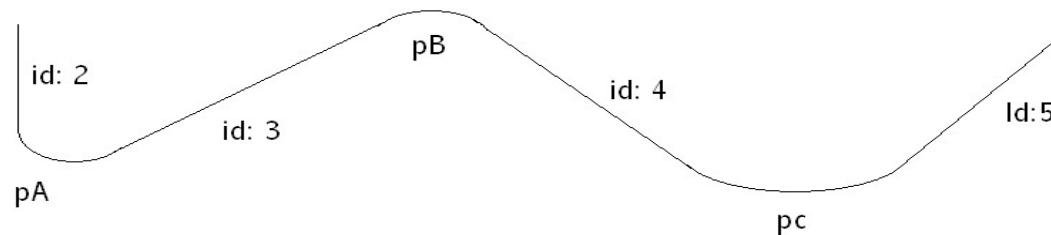
```
.....  
movej (pD, tTool, mRapido)  
//do  
// delay(0)  
// l_pActual=here(tTool,world)  
//until (distance(l_pActual,pD)<=20)  
wait((distance((here(tTool,world)), pD))<=20)  
doutVenturi=true  
waitEndMove()
```



## GESTIÓN IN/OUT EN MOVIMIENTO - 2

STÄUBLI

- Cada movimiento que es cargado en la pila de movimientos genera un id de movimiento.  
`nIdPrimero= movej (pA, tTool, mRapido)`
- El id de movimiento se incrementa con cada movimiento que introducimos en la pila de movimientos, para poder modificar o inicializar el id de movimientos podremos hacer:  
`resetMotion()` = Inicializa el id de movimientos a cero para la siguiente instrucción de movimiento.  
`setMovId()` = establece id de movimientos para la siguiente instrucción de movimientos. El nuevo valor id de movimientos nunca debe ser igual al id de movimiento de un movimiento pendiente a realizar.



Para conocer el id de movimiento que se esta realizando hay que utilizar la orden getMoveID(). La orden devuelve un valor del cual:

- Parte entera: Es el Id de movimiento que se esta realizando
- Parte decimal: Es el % por ciento de movimiento que se ha realizado.

Ejemplo:

- 15,8: Id movimiento 15 y un 80 por ciento realizado
- 16,559: Id movimiento 16 y un 55,9 por ciento realizado

Nunca debe esperarse a que el id de movimiento tenga un valor exacto (==), sino que esperar a que sea mayor o igual (>=).

Ejemplo:

```
.....  
movej(pA, tTool, mRapido)  
nId = movej(pB, tTool, mRapido)  
movej(pC, tTool, mRapido)  
wait(getMoveID()>=(nId+0.8))  
doSeñal=true  
open(tTool)
```

## GESTIÓN IN/OUT ANALOGICA

La gestión de señales analógicas no es directa y hay que consultar o asignar un valor.

Aioget

Esta instrucción devuelve el valor numérico de ailInput.

Numérico = Aioget (ailInput)

Aioset

Esta instrucción asigna nValue a aiOutput

Aioset (aoOutput, nValue)

nTemperatura = aioGet(aiTemperatura)

aioSet(aoVelocidad, nVelocidad)

# DATOS CODIFICADOS EN BITS

STÄUBLI

- Crear un array de variables dio out[8] en la aplicación
- Relacionar cada dio con una dio física del panel de control

`dioLink(out[0], io:bOut0)`

.....

`dioLink(out[7], io:bOut7)`

- Escribir los valores en las dio

`dioSet(out, nVal)`       $nVal = 2^0 + 2^1 + \dots + 2^7$   
                                      out[0]    out[1]    ...    out[7]

`dioSet(OUT, 13)`       $13 = 2^0 + 2^2 + 2^3$   
                                      ==>            out[0]=out[2]=out[3]=true  
                                      out[1]=out[4]=out[5]= out[6]= out[7]= false

Leer los valores de las dio

FORMACIÓN VAL3

# *Programación Estructurada*



## OPERADORES - CONDICIONES

Operadores: <, >, ==, !=, >=, <=, and, or, xor, !

Condiciones :

nY == 1 Verdadero (true) si nY es igual a 1.

nY != 3 nY diferente de 3.

nY > 3 Verdadero (true) si nY mayor que 3

nY < 3 Verdadero (true) si nY menor que 3

nY >= 3 Verdadero (true) si nY mayor o igual que 3

nY <= 3 Verdadero (true) si nY menor o igual que 3

nY == 1 and !bBool Verdadero si nY es 1 y bBool

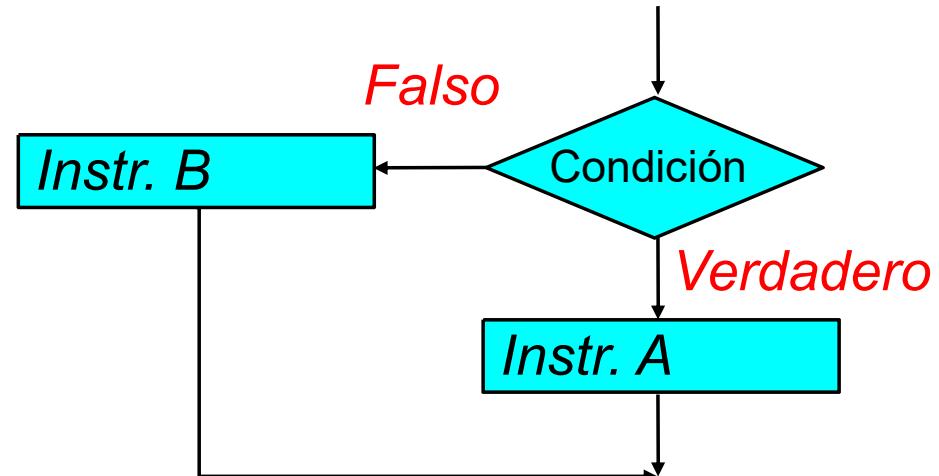
es falsa

# OPERADORES - CONDICIONES

```
if nZ == 3  
|  
| Instrucciones A  
else  
|  
| Instrucciones B
```

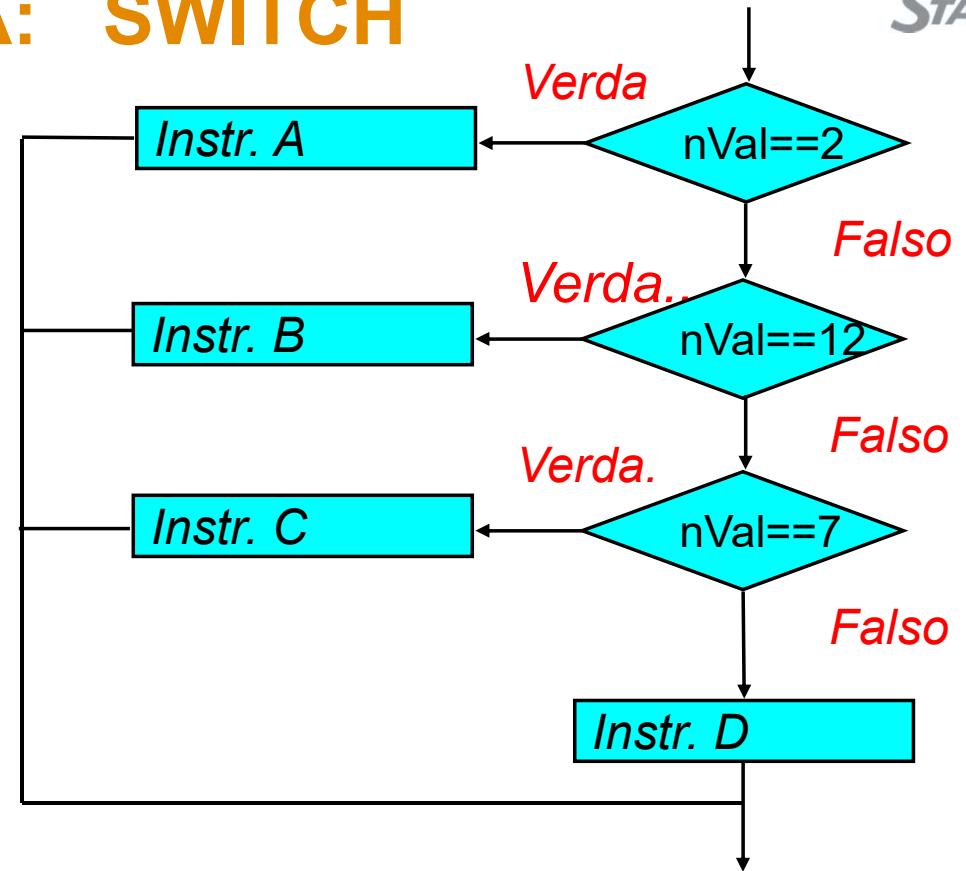
endif

VAL3 >=6.4 : Esta disponible la instrucción **elseif**, es mejor utilizar la instrucción **Switch** (según el caso).



# ESTRUCTURA: SWITCH

```
switch nVal
case 2
    | Instr. A
break
case 12
    | Instr. B
break
case 7
    | Instr. C
break
default
    | Instr. D
break
endSwitch
```



CASO 1,2,3 :  $nVal == 2 \text{ o } 12 \text{ o } 7$

*ii Únicamente con variables numéricas !! (VAL3>=6.4 : Todo tipo de datos)*

## CICLOS: NÚMERO DE CICLOS CONOCIDO

```
for nContador= 1 to 10  
    < Instrucciones >  
endFor
```

for nContador = 0 to 50

*moveI(pPunto[nContador],tPinza,mRapido)*

endFor

nInicio=100

nFin= 10

for nIndice = nInicio to nFin step -1                   paso diferente de 1

nPaso[nIndice]= nLado\*nIndice

endFor

# CICLOS CONDICIONALES

STÄUBLI

While nVal > 3

|

| Instrucciones

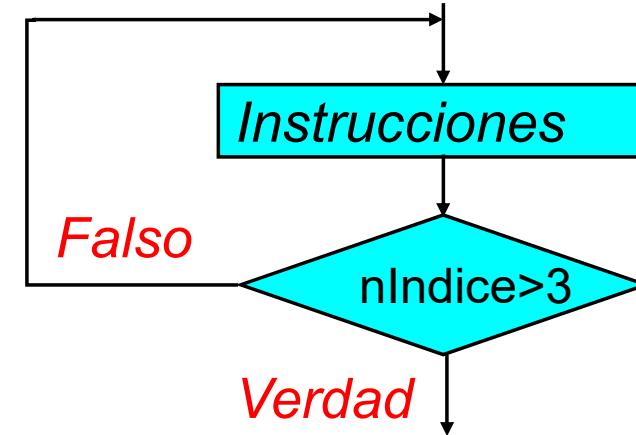
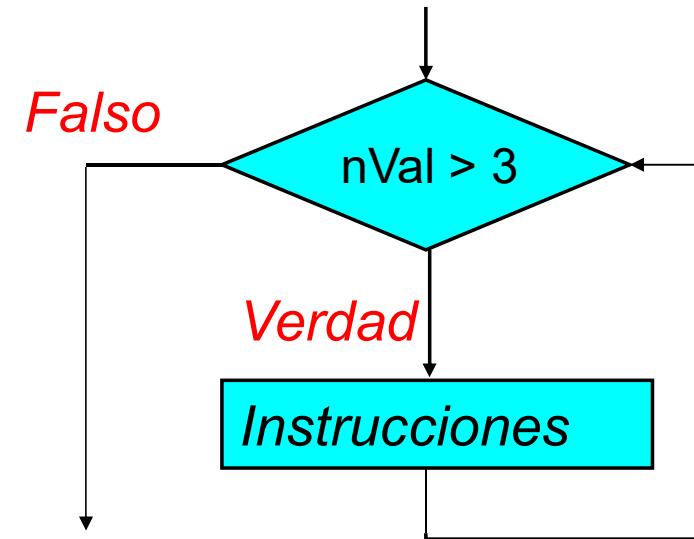
endWhile

do

|

| Instrucciones

until (nIndice>3)



# PROGRAMAS

STÄUBLI

Aplicación a.apl()

....

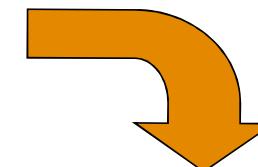
....

CALL Programa()

....

....

1



programa ()  
*<Instrucciones>*

2

3

.....

3

Vuelve a la aplicación a.apl al final del programa  
o con la instrucción return

## OPERACIONES SOBRE MDESC

MDESC	MDESC mRapido :
.vel (%)	mRapido.vel = mRapido.vel/2
.accel (%)	
.decel (%)	
.tvel (mm/s)	
.rvel (deg/s)	
.blending (off / joint)	mRapido.blending=joint
.leave (mm)	mRapido.reach= mRapido.leave= 20
.reach (mm)	
=	
==	
!=	

## OPERACIONES SOBRE PUNTOS

<b>POINT</b>  .trsfb .config link to a frame (invisible)  = != ==	POINT pta : POINT ptb :  <u>Copiar un punto sobre otro :</u>  pta=ptb <i>!!! pta toma el mismo plano de referencia que ptb !!!</i>  <u>Copiar las coordenadas de un punto sobre otro punto :</u>  pta.trsf=ptb.trsf  <u>Poner 100 en la coordenada X de pta:</u>  pta.trsf.x=100
--	--

## OPERACIONES SOBRE TOOL

<b>TOOL</b>	<b>TOOL tTool :</b>
.trsf	<u>La misma operación que un trsf en los points</u>
.gripper	
.otime	tTool.gripper= true false
.ctime	tTool.otime=0.2
link a una tool (invisible)	tTool.ctime= tTool.otime
=	tTool.trsf.z=100
!=	
==	<u>Copiar un tool :</u>
	tTool 1=tPinza

## OPERACIONES SOBRE FRAME / JOINT

<b>FRAME</b>	La misma operación que para los puntos
<b>.trs</b> <b>link to a frame (invisible)</b>	
<b>=   ==   !=</b>	
<b>JOINT</b>	<b>JOINT jStart</b>  <b>=</b> <b>jStart ={0,10,20,30,40 ,50}</b> o <b>==</b> <b>!=</b> <b>jStart.j1=0</b> <b>&gt;</b> <b>jStart.j2=10</b> <b>&lt;</b> <b>jStart.j3=20</b> <b>-</b> <b>jStart.j4= jStart.j2+ jStart.j3</b> <b>+</b> <b>jStart.j5=40</b> <b></b> <b>jStart.j6=50</b>

## OPERACIONES SOBRE CONFIGURACIÓN

<b>CONFIG</b>	CONFIG conf :
.shoulder	conf.shoulder = righty / lefty / ssame / sfree
.elbow	
.wrist	conf.elbow=epositive                  jt3 > 0 enegative                              jt3 < 0 esame / efree
=	
==	conf.wrist=wpositive                  jt5>=0 wnegative                              jt5 < 0 wsame / wfree
!=	conf={righty,epositive,wnegative}
	<b>POINT pta ptb :</b> <u>Salvar la configuración de un punto:</u> conf= pta.config
	<u>Copia la configuración de un punto sobre otro :</u> pta.config=ptb.config
	<u>Forzar la configuración del wrist del punto:</u> pta.config.wrist=wpositive

## OPERACIONES SOBRE TRANSFORMADAS

TRSF	trAproximaTomar={0,0,-100,0,0,0}
=	trAproximaTomar.x=0 trAproximaTomar.rz=45
==	<u>Multiplicación</u> : *
!=	t1=t2 * t3
	<u>Inversa</u> : !
	t1= ! t2

```

■ begin
■   l_pInicio=jointToPoint(flange,world,jInicio)
■   l_pActual=here(flange,world)
■   l_nDistacia=distance(l_pActual,l_pInicio)
■   close (tPinza)
■   if l_nDistacia>20
■     // Subir eje 3 a una altura máxima
■     l_pActual.trsf.z=nZsuperior
■     move(l_pActual, flange,mLento)
■     // Ir al jInicio con altura máxima
■     l_pInicio.trsf.z=nZsuperior
■     move(jlInicio, flange,mLento)
■     //Ir al jInicio
■     move(jlInicio, flange,mLento)
■     l_pInicio=jointToPoint(flange,world,jInicio)
■     //Realizar el movimiento mientras se pulsa el botón Home
■   do
■     l_pActual=here(flange,world)
■     l_nDistacia=distance(l_pActual,l_pInicio)
■     if (dinIrHome==false)
■       stopMove()
■     else
■       restartMove()
■     endif
■     delay(0)
■   until l_nDistacia<=10
■   else
■     move(jlInicio, flange,mLento)
■   endif
■   restartMove()
■   open (tPinza)
■ end

```

FORMACIÓN VAL3

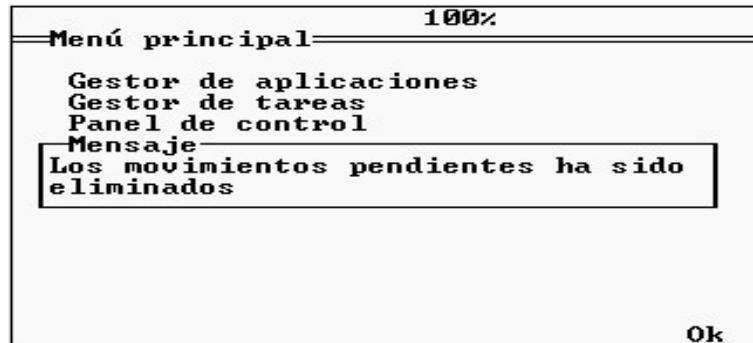
*Ventana Operador*

*y*

*Dialogo Operador*



# REGISTRO



2x	
<u>Registro de eventos</u>	
21:37	USR:Mensaje a escribir en el regis
21:37	Prova2 Aplicación arrancada
21:35	prova Aplicación parada
21:35	Evento de sistema INTERFACE-Stoppe
21:27	Mcp = TSimulated
21:27	Simulated MCP connected (10.140.80
21:27	Modo de trabajo:manual.
21:27	Evento de sistema INTERFACE-Stoppe
21:27	Evento de sistema INTERFACE-Stoppe
21:27	Nuevo perfil : default
	Exp.

**popUpMsg(string)** Muestra una pantalla con un mensaje que debe ser validado por el operador.

**logMsg(string)** Permite escribir mensajes desde la aplicación al “Registro de eventos”. Los mensajes son guardados con fecha y hora.

**logMsg(“Mensaje a escribir en el registro de eventos”)**

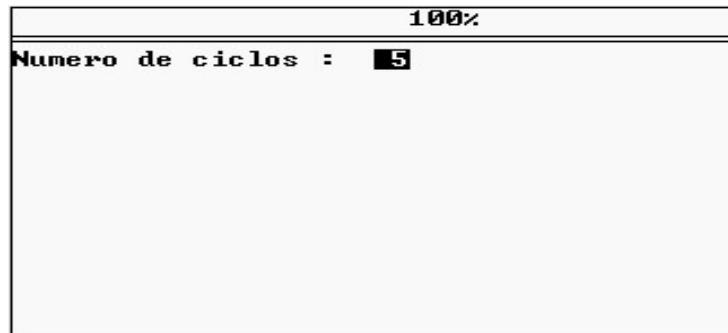
⇒Línea escrita en el archivo « errors.log » :

[01/01/06 00:00:01] USR:Mensaje a escribir en el registro de eventos

## PANTALLA DEL USUARIO

STÄUBLI

La página del usuario está disponible para mostrar mensajes y solicitar entradas de datos de usuario.



Manualmente se puede visualizar la pantalla de usuario con la tecla User

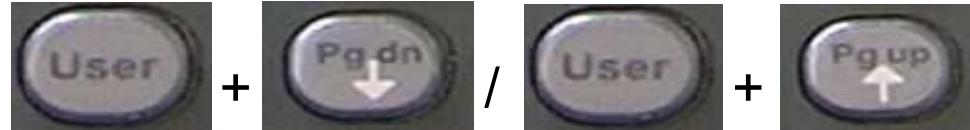
Val3 permite la gestión de múltiples pantallas.

En todas las ordenes relacionadas con pantalla se puede especificar la pantalla relacionada si no es específica será la pantalla actual.

Sólo la pantalla actual puede capturar un dato.

La gestión con el mando de las pantallas es:

Paso de una pantalla a otra :



Primera pantalla:



Ultima pantalla:



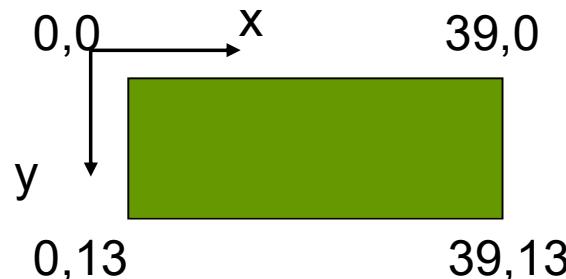
La gestión por programa de las pantallas es:

`userPage()` : Cambia a la pantalla del usuario

`cls()` : Limpia la pantalla del usuario

`title(string)` : Cambia el título de la pantalla del usuario

## PANTALLA DE USUARIO : VISUALIZACIÓN



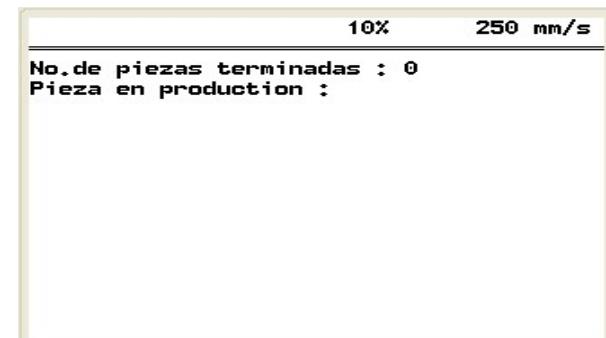
**gotoxy(numx,numy)** : sitúa el cursor en la pantalla

**put(num) / put(string)** : Muestra un mensaje o un numero, sin retorno de línea.

**putln(num) / putln(string)** : Igual pero con retorno (CR/LF)

**cls()**

```
put("No.de piezas terminadas : ")
putln(nPiezas)
sMensaje="Pieza en production : "
put(sMensaje)
```



## PANTALLA DE USUARIO : CAPTURA

get(num)    o    get(string)

Espera la entrada de datos hasta la validación con las teclas “RETURN“, “ESC“ o una tecla del menu.

nTecla = get() Espera y regresa el código de la primera tecla pulsada

Si se pulsa: 21 y enter

Al ejecutar nTecla=get(nNumero)

nTecla = 50 (ASCII del número 2)

nNumero = 21

nTecla = getKey() Sin espera => si no se pulsa ninguna tecla: nTecla = -1

Do

    delay (0)

    Until getKey()==-1

    nTecla=getKey()

## TEMPORIZADOR: CLOCK()

nVal =clock()

*Devuelve el valor actual del reloj interno en segundos.*

*Precisión = ms*

*Valor inicializado a “Cero” a la puesta en marcha  
del controlador*

Medida del tiempo de ciclo:

nCiclo=clock()

.....

waitEndMove()

nTiempo = clock()-nCiclo

*getDate() (VAL3 >=6.4): Devuelve el la fecha y hora del sistema.*

## EJEMPLO DE UTILIZACIÓN

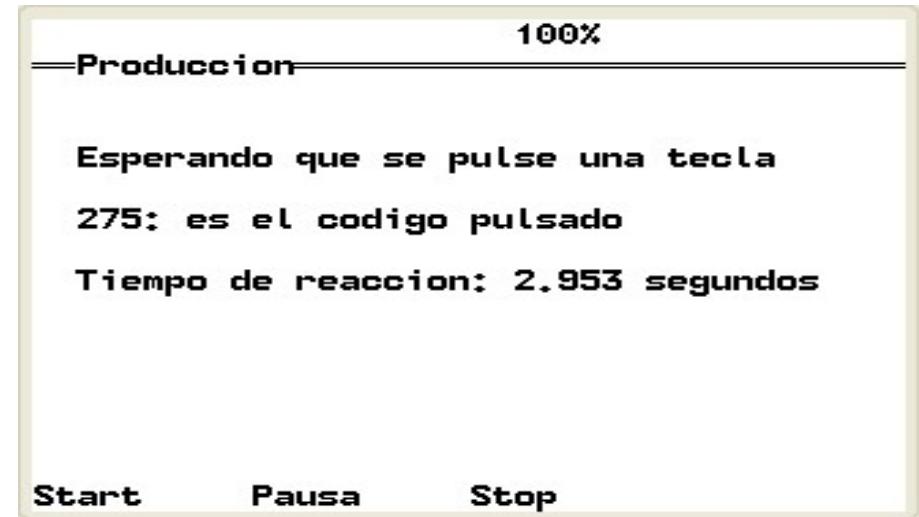
begin

```

userPage()
cls()
title("Produccion")
gotoxy(2,2)
putln("Esperando que se pulse una tecla")
gotoxy(0,13)
put("Start    Pausa    Stop")
nTime=clock()
nCode=get()
gotoxy(2,4)
put(nCode)
put(": es el codigo pulsado")
nTime=clock()-nTime
gotoxy(2,6)
put("Tiempo de reaccion: ")
put (nTime)
put(" segundos")

```

end



# CONVERTIR UN NUMERO EN STRING

sMensaje=toString("x.y", nNumero)

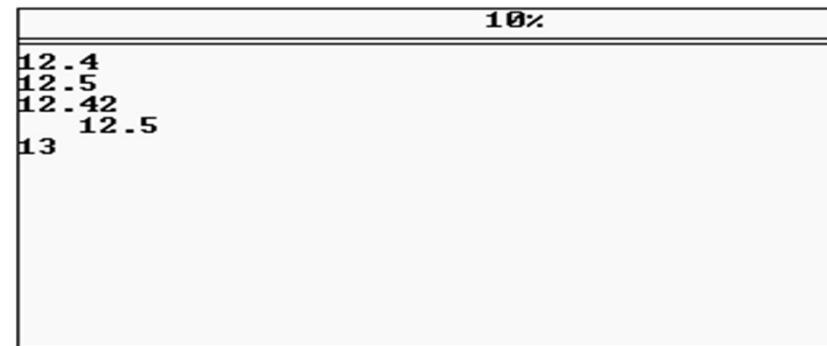
sMensaje: String de resultado

x: Caracteres parte entera.

y: Caracteres parte decimal.

nNumero: Número a transformar.

```
println(toString("2.1", 12.42))  
println(toString("2.1", 12.49))  
println(toString("3.3", 12.42))  
println(toString("7.1", 12.49))  
println(toString("0.0", 12.51))
```



## FORMACIÓN VAL3

# *Variables locales*

Y

# *paso de parámetros*



## *Variables Globales:*

- 💣 Comunes a todos los programas.
- Riesgo de corrupción de variables.

## *Variables Locales:*

- Independientes de un programa al otro.
- Necesitan el paso de parámetros para compartir datos.

# VARIABLES LOCALES

STÄUBLI



The screenshot shows a software interface titled "Gestor de aplicaciones" with a timestamp "ejercicio5 <06 Mar. 2007 11:41>". Under the "Variables locales" section, several variables are listed:

- num l\_nColumna
- num l\_nLinea
- num l\_nNoPieza
- point l\_pPoner
- point l\_pTomar
- trsf l\_trPtPaleta
- trsf l\_trPtTorre

The text "Nue . Gua ." is visible at the bottom right of the window.

Las variables locales deben ser inicializadas y definidas en el programa donde serán utilizadas.

Las variables locales, según la Convención interna de STÄUBLI, empiezan por **l\_** seguido del tipo de variable en minúscula y del nombre de la variable empezando en mayúscula.

***l\_nContador***

Variable local

Tipo numérica

Nombre de la variable

Las variables locales, como las globales, pueden ser de un único valor o un array.

`I_pLocal`      `I_pLocal[1..n]`

Las variables locales pueden tener el mismo nombre en diferentes programas, pero no existe ninguna vinculación entre ellas.

Las funciones principales de las mismas, son para cálculos o operaciones internas al programa:

`I_nTiempo = clock()`

`I_nTiempoCiclo= clock() - I_nTiempo`

`I_pActual=here (tPinza, world)`

`I_nContador=I_nPiezas`



# PASO DE PARÁMETROS

STÄUBLI

```
programa principal  
nVal =10  
call subprog(nVal)
```

```
? 100%  
Gestor de aplicaciones  
-ejercicio5 <06 Mar. 2007 11:41>  
+Librerías  
+Variables globales  
-Programas  
+paleta  
-poner  
+Variables locales  
-Parámetros  
  num p_nDistancia  
+start  
+stop  
+tomar  
+Parámetros  
Ref. Edi. Ren. Ins. Bor. Nue. Gua.
```

```
programa subprog(x_nDistancia)  
put(x_nDistancia)
```

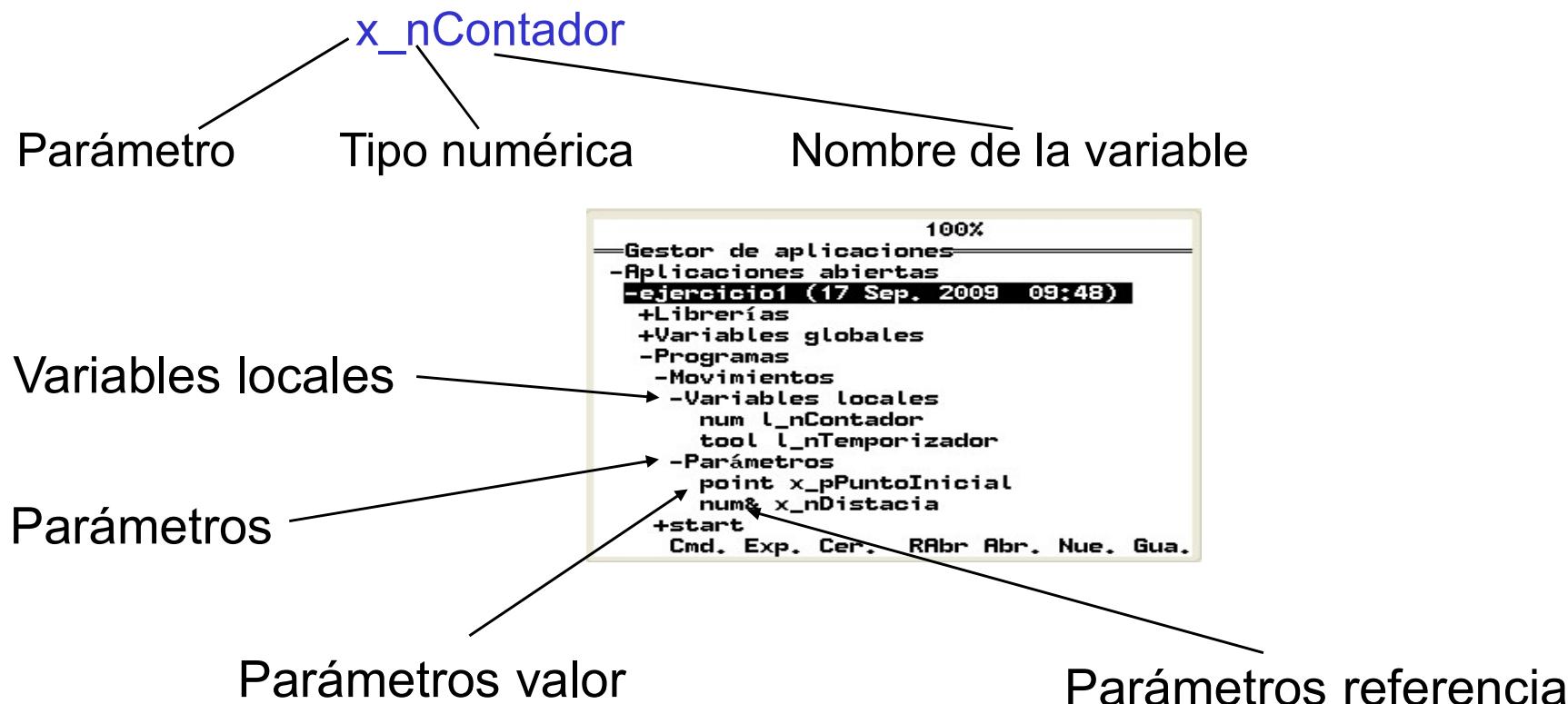
*==> Muestra 10*

Tantos parámetros como sean necesarios

*¡ Atención ! Al número y orden de los parámetros*

*¡ Atención ! Los parámetros deben ser del mismo tipo*

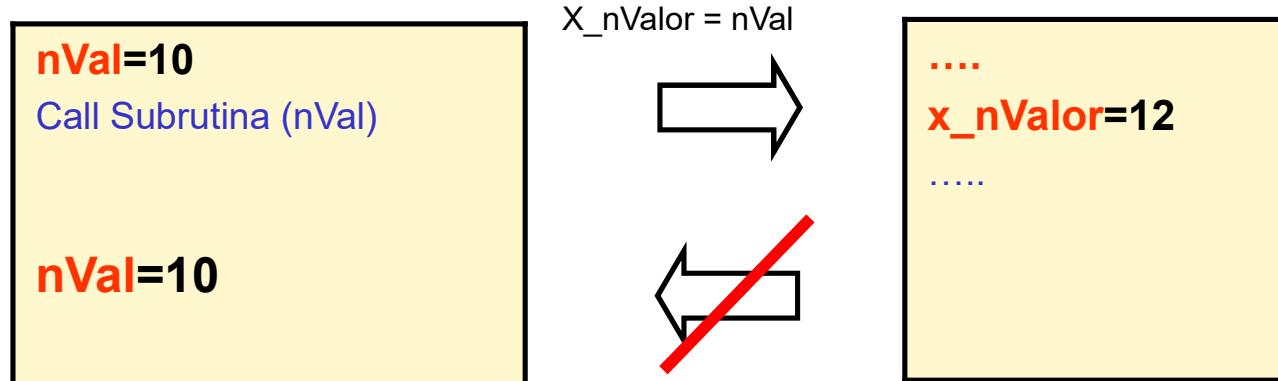
Los parámetros, según la Convención interna de STÄUBLI, empiezan por x\_ seguido del tipo del tipo de parámetro en minúscula y del nombre del parámetro empezando en mayúscula.



## TIPOS DE PASO DE PARÁMETROS

### ➤ *Paso por valor:* num x\_nDistancia

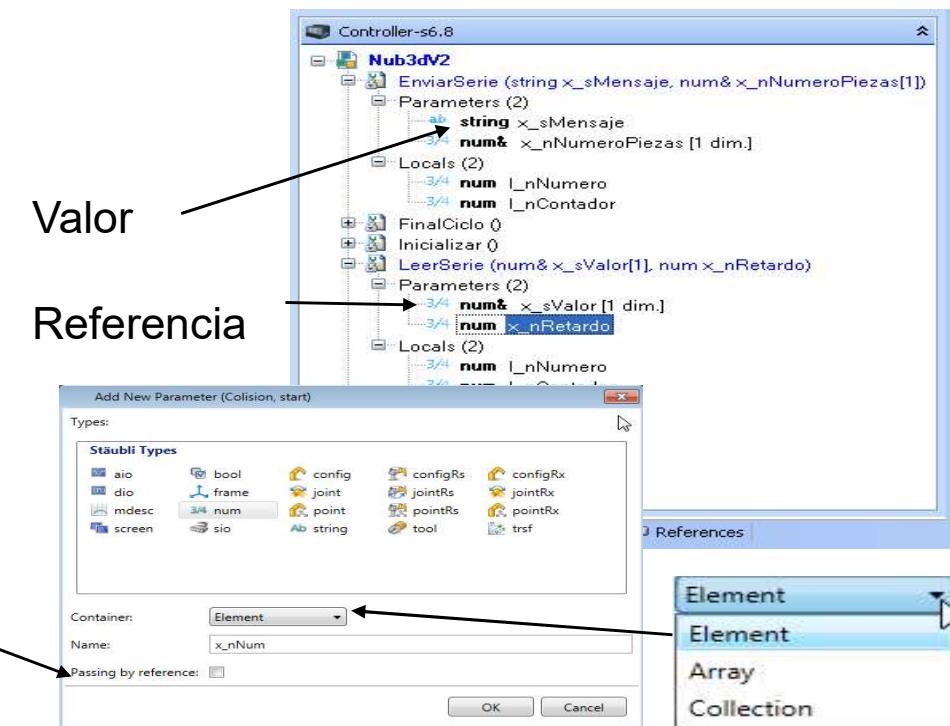
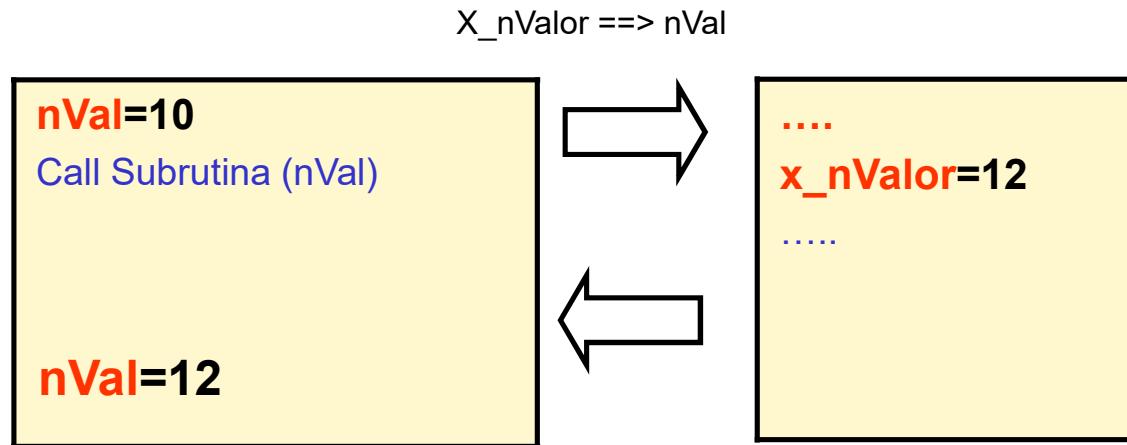
- El sistema crea una variable local en la subrutina y la inicializa con el valor de la variable o de la expresión suministrada por el programa o subprograma solicitante. Se hace una copia local de la variable
- *Ningún riesgo que la variable sea modificada en el programa que la llama aunque la subrutina la modifique.*



➤ *Paso por referencia:* num& p\_nDistancia

El programa deja de trabajar sobre una copia del dato pasado por el solicitante, haciéndolo sobre el propio dato, al que sencillamente se le cambia de nombre localmente.

*¡¡¡ Para modificar la variable del programa que le llama !!!!*



FORMACIÓN VAL3

*Planos de referencia*

*locales*

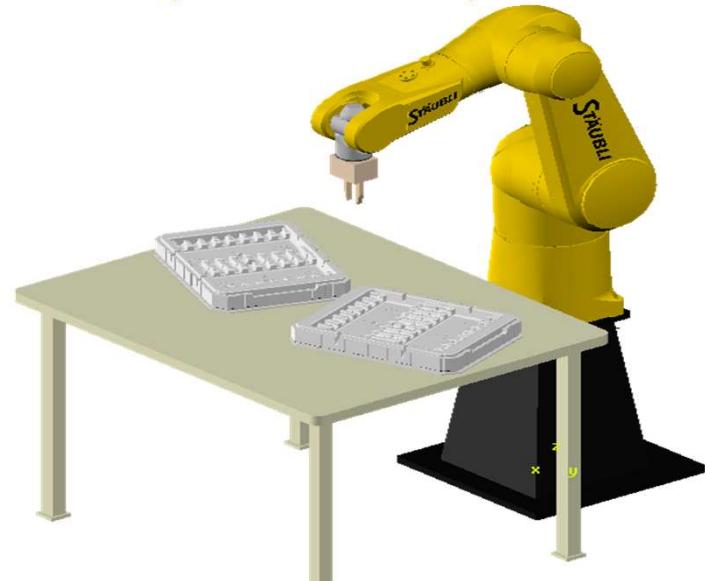
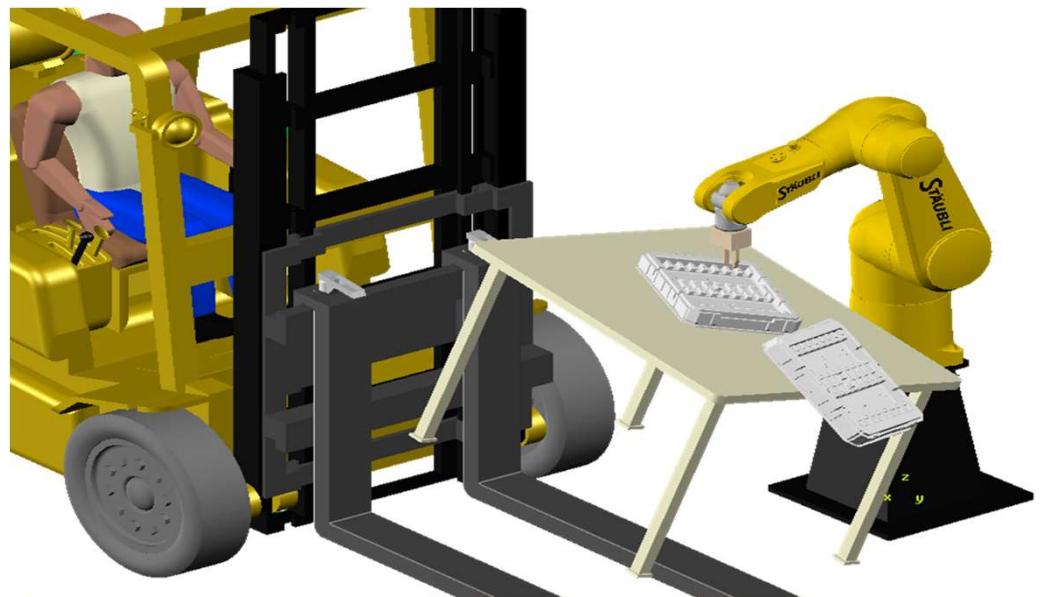
&

*Paletización*



## ¿PORQUÉ SE USA UN PLANO? ( FRAME )

El robot está en producción, la aplicación está trabajando al 100% de producción, pero....



Manolo conduce el montacargas y ...

*¡¡¡ DESASTRE !!!!*

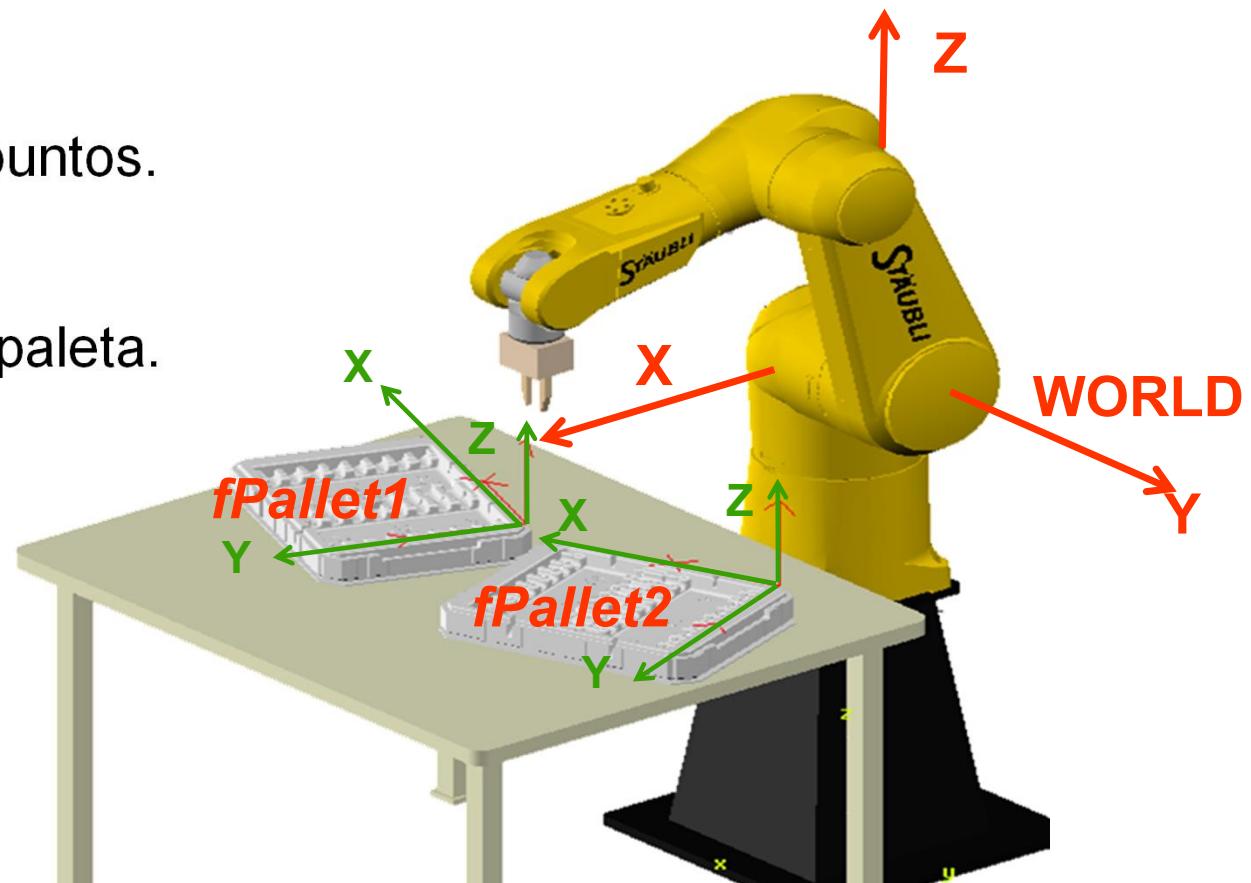
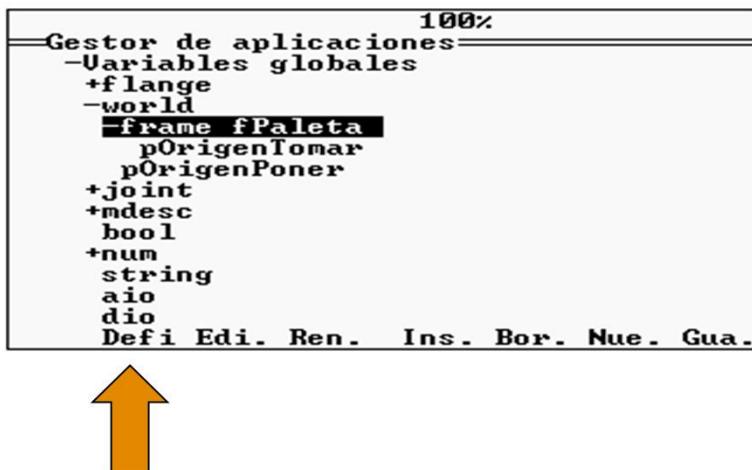
... un día para retomar los puntos...

Excepto si ...

## UTILIZACIÓN DE UN PLANO

*Sistema de referencia local:*

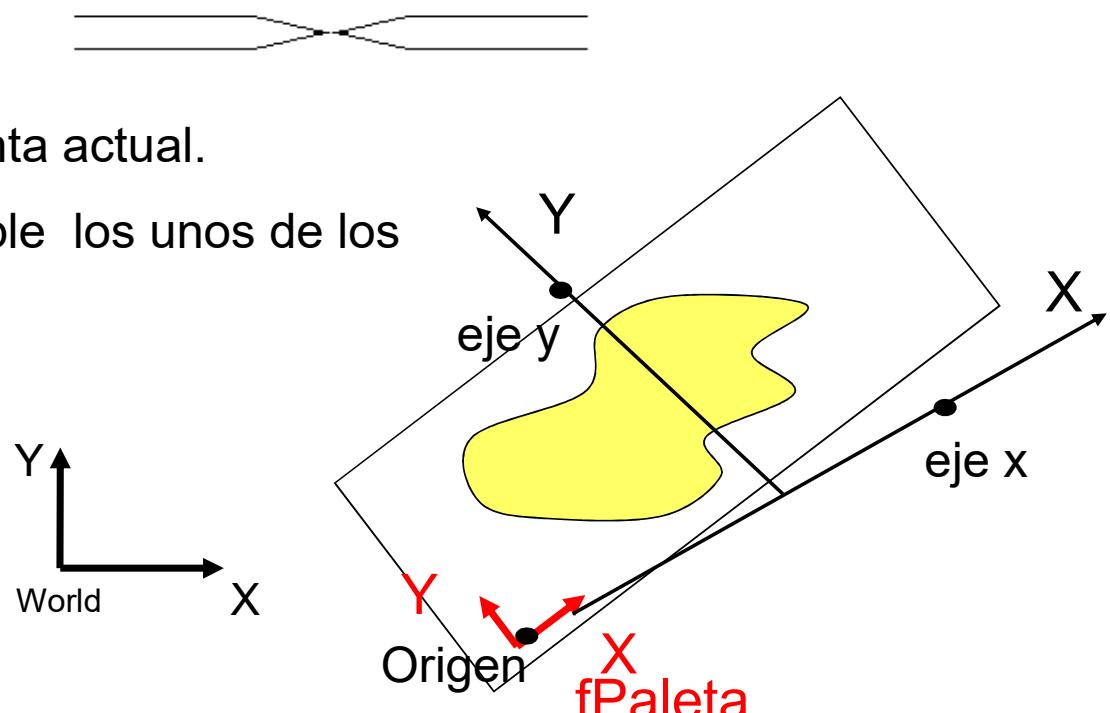
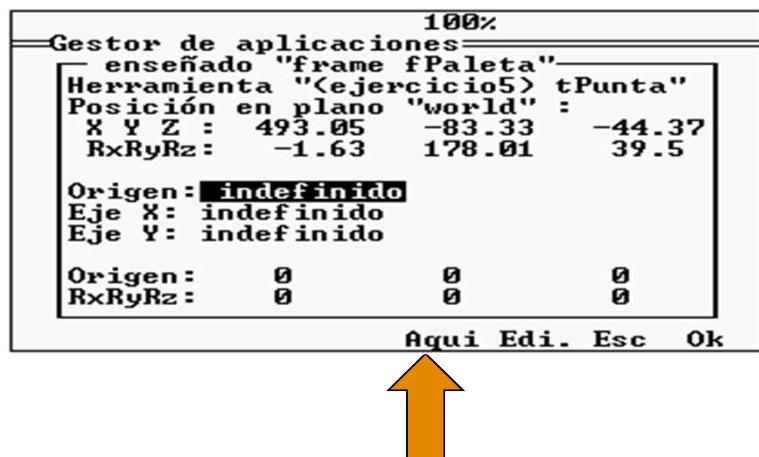
- Para facilitar aprendizaje de los puntos.
- Utilizados para duplicar puntos.
- Desplazamiento de puntos en la paleta.



## APRENDIZAJE DE UN PLANO ( FRAME )

*Definido con 3 puntos que deben aprenderse:*

- Usar una herramienta precisa: Punta.
- Definir esta herramienta como la herramienta actual.
- Aprender los puntos lo más separado posible los unos de los otros (Mayor precisión).



## PUNTOS EN UN PLANO ( FRAME )

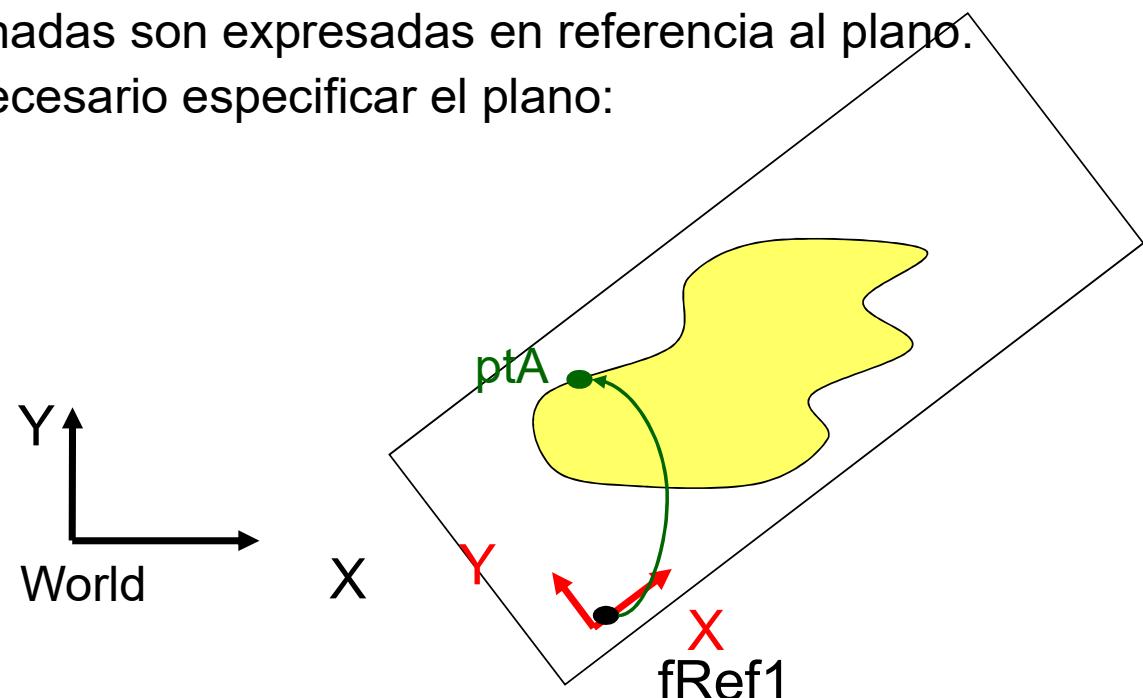
*Crear los puntos de la aplicación bajo la rama del plano:*

- Nunca definir los puntos antes que el plano.
- Durante la enseñanza del punto, las coordenadas son expresadas en referencia al plano.
- Para la instrucción del movimiento, no es necesario especificar el plano:

```
movej(ptA ,tPinza, mRapido)

? 100%
Gestor de aplicaciones
-Variables globales
+flange
-world
-frame fPaleta
  pOrigenTomar
  pOrigenPoner
+joint
+mdesc
bool
+num
string
aio
dio
Mov. Aqui Edi. Ren. Ins. Bor. Nue. Gua.

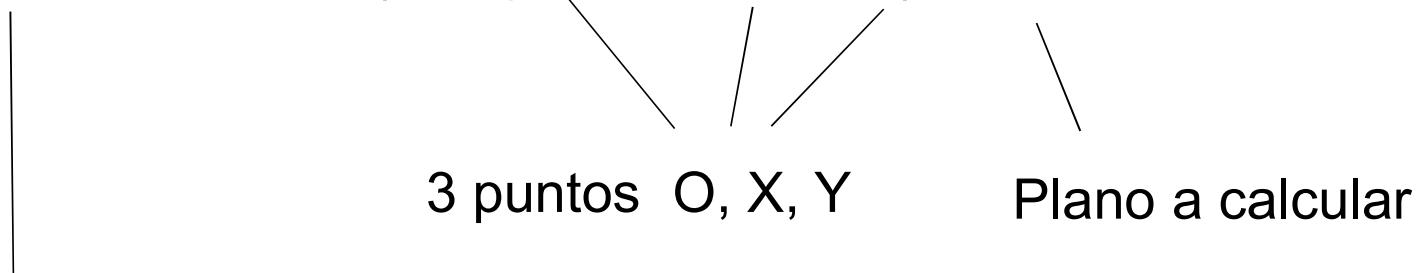
```

## CÁLCULO DEL PLANO ( FRAME ) POR PROGRAMA

*La función `setFrame` nos permite generar un plano por programa.*

`nError = setFrame(pOrigen, pX,pY, fRef)`



Código de error del cálculo:

- 0 : no hay error
- 1 : pX muy cerca del punto pOrigen
- 2 : Los 3 puntos están alineados

## PALETIZACIÓN EN UN PLANO ( FRAME )

*Compose(point,frame,trsf)* : calcula un punto desplazado de *trsf* en referencia al plano *frame*.

*Los incrementos en X, Y y Z se realizan respecto al punto indicado.*

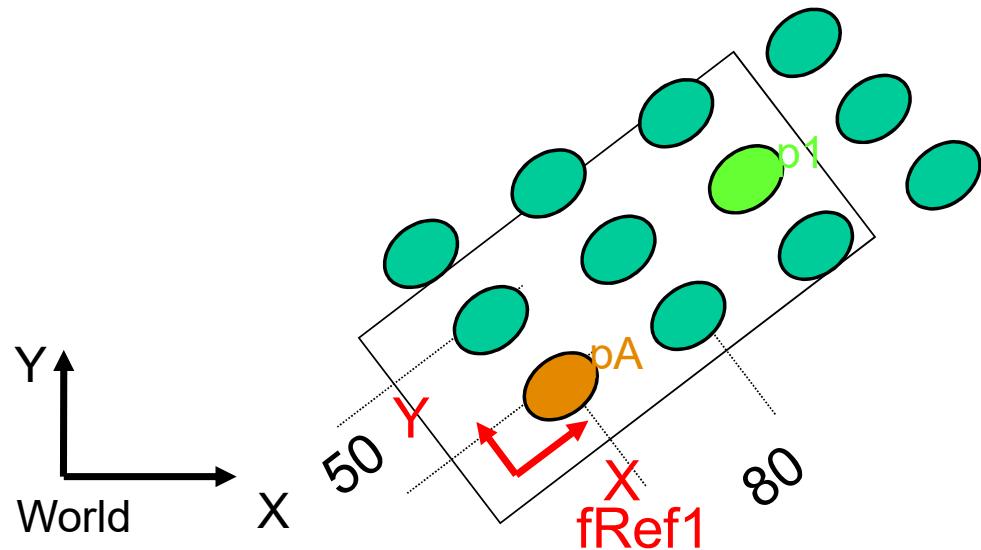
*Las rotaciones en RX, RY y Rz se realizan respecto al centro del frame (Paletizado circular).*

`p1=compose(pA,fRef1,{160,50,0,0,0,0})`

`movel(p1 ,tPinza,mLento)`



Ej. 5  
203

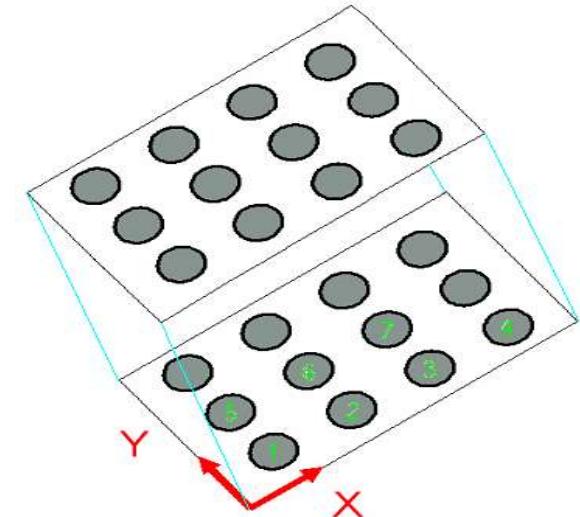


## Ejemplo de paletizado en X, Y y Z:

STÄURII

I\_nContadorX, I\_nContadorY, I\_nContadorZ: Numéricas.  
nPiezasX, nPiezasY, nPiezasZ: Número piezas en X, Y y Z.  
I\_trPalet: Transformada de aproximación a los punto en el palet.  
I\_pCalculado: Punto calculado.  
pInicial: Punto inicial tomado con el robot.  
fPalet: Frame que define el paletizado.  
tTool: Pinza

```
for I_nContadorZ=0 to nPiezasZ
    for I_nContadorY=0 to nPiezasY
        for I_nContadorX=0 to nPiezasX
            Procedimiento de tomar la pieza
            I_trPalet= {I_nContadorX*nIncrementoX,I_nContadorY*nIncrementoY,I_nContadorZ*nIncrementoZ, 0, 0, 0}
            I_pCalculado=compose(pInicial,fPalet,I_trPalet)
            movej(appro(I_pCalculado,trAproPalet),tTool,mLento)
            movel(I_pCalculado,tTool,mLento)
            open (tTool)
            movel (appro(I_pCalculado,trAproPalet),tTool,mRapido)
        endFor
    endFor
endFor
```



## Uso de dos planos idénticos

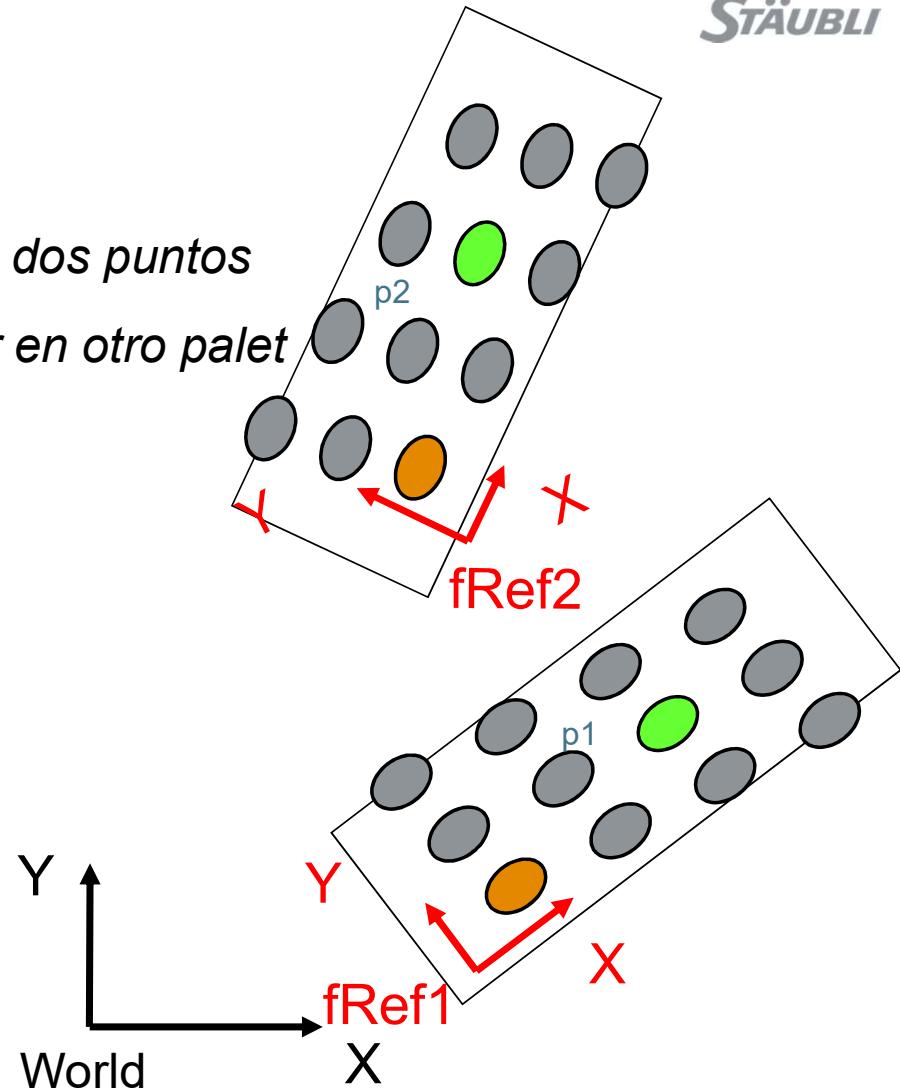
Para usar un punto con las mismas coordenadas en dos puntos

(Ejemplo: Despaletizar un palet, procesar y paletizar en otro palet idéntico) :

- Crear un punto en cada palet.
- Tomar un punto.
- Copiar la trsf de un punto sobre el otro.

$$\text{p2.trsf} = \text{p1.trsf}$$

p2 dista de fRef2 lo mismo que p1 de fRef1



FORMACIÓN VAL3

# *Instrucciones del sistema y Multitarea*



## INTRUCCIONES DE SISTEMA

STÄUBLI

`isPowered()` : Devuelve TRUE si el brazo tiene potencia.

`isCalibrated()` : Devuelve TRUE si el controlador esta calibrado (El robot reporta un mensaje en el arranque si no esta calibrado).

`isSettled()` : Devuelve TRUE si el robot esta parado al final de un movimiento.

`isEmpty()` : Devuelve TRUE la pila de movimientos esta vacía = todos los movimientos se han realizado.

`esStatus()` : Devuelve el estado del circuito de emergencias.

0: No hay emergencias.

1: Espera validación después de emergencia.

2: Emergencias activadas.

`workingMode()` : Devuelve el modo de trabajo actual.

Manual. 2 : Test.

3 : Automático.

4 : Remoto.

0 : En transición.

`disablePower()` : Desactiva la potencia del brazo por programa.

`enablePower()` : Activa la potencia del brazo por programa:

(Solo en modo Remoto)

`getMonitorSpeed()` : Devuelve el porcentaje de la velocidad seleccionada en el mando.

`setMonitorSpeed(num)`: (Val3> 6.4) Modifica la velocidad actual del mando. Solo funciona si el robot esta en modo Remoto y el operario no tiene acceso a modificar la velocidad.

# MULTITAREAS

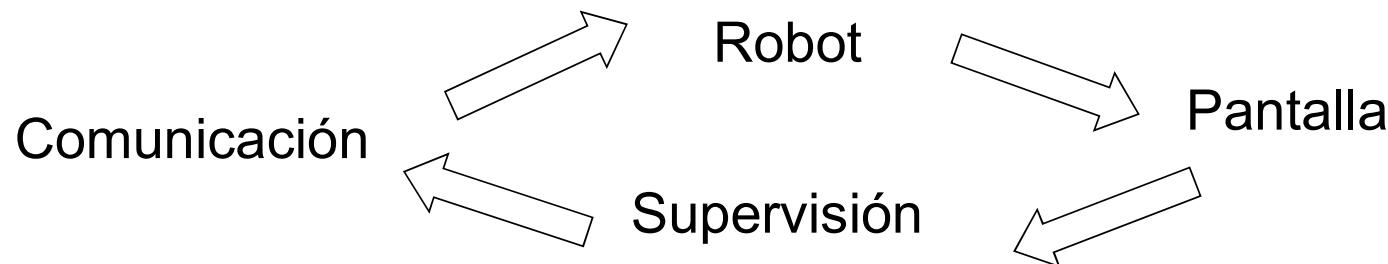
VAL 3 : Sistema multitareas = varios programas pueden ser ejecutados en paralelo.

Programa START utilizado para lanzar las tareas de la aplicación.

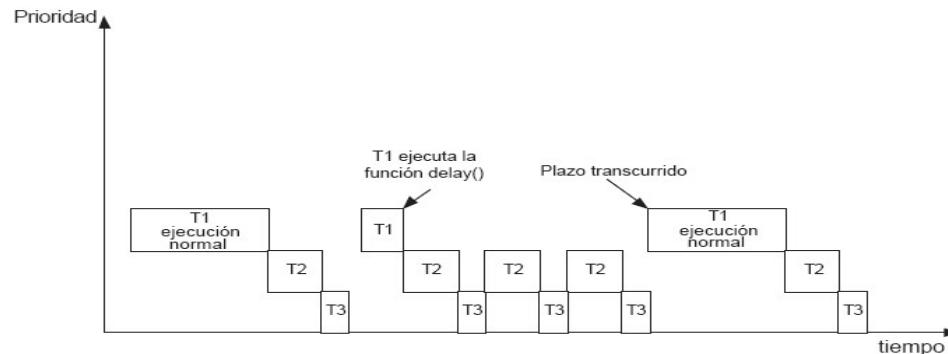
En toda aplicación hay por lo menos una tarea, aquella dedicada a la ejecución de movimientos.

Únicamente una tarea puede generar movimientos (dos tareas a la vez no pueden generar movimientos del robot)

Ejemplo: Robot (control de movimientos), Pantalla (Comunicación con usuario), Comunicación (Comunicación con la periferia) y Supervisión (Colisiones, posiciones, etc).



Cuando una aplicación tiene varias tareas en ejecución (multitarea), ésta parecen ejecutarse simultáneamente e independientes. Esto es cierto si se observa la aplicación globalmente en intervalos de tiempo suficientemente largos (del orden de segundos), pero no es exacto cuando uno se interesa en el comportamiento preciso en intervalos de tiempo reducidos (unidad de tiempo de microprocesadores).



Creación de una tarea paralela :

```
taskCreate "robot",100,ciclo( )
```

Nombre de la tarea

Prioridad de 1 a 100

Nombre del programa a ejecutar en la tarea

Prioridad de la tarea paralela:

- 1: Se ejecuta 1 línea cada vez.
- 100: Se ejecutan 100 líneas cada vez.

Para finalizar una tarea paralela utilizaremos la orden taskKill.

taskKill ("robot")  
                            ↑  
                            Nombre de la tarea

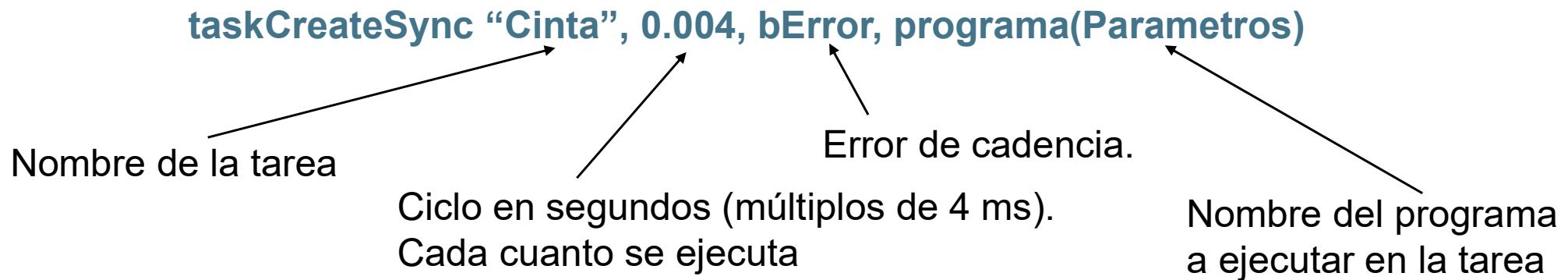
Normalmente, el código que existe en una tarea paralela esta englobado dentro de un bucle.

## MULTITAREAS SÍNCRONAS

A veces es necesario programar las tareas a intervalos regulares de tiempo (Adquisición de datos , seguimiento de cintas transportadoras, comunicaciones síncronas, etc), es cuando hablamos de tareas síncronas.

`taskCreate` crea y ejecuta una tarea síncrona, ejecución lo antes posible con control de planificación y con prioridad 3000.

Las tareas síncronas interrumpen las tareas asíncronas cada cierto tiempo para ejecutarse ellas.



## GESTOR DE TAREAS

`taskKill ("Nombre")` : Finaliza la tarea Nombre.

`taskSuspend ("Nombre")` : Suspende la ejecución de la tarea Nombre.

`taskResume ("Nombre", nLinea)` : Reanudada la ejecución de la tarea Nombre, dependiendo de nLinea la reanudación será:

- 0 : En la línea actual.
- >0: nLineas después de la linea actual de ejecución..
- <0: nLineas antes de la linea actual de ejecución.

`taskStatus ("Nombre")` : Devuelve el estado actual de la tarea Nombre:

- 1 : Ninguna tarea Nombre ha sido creada
- 0: La tarea Nombre esta parada.
- 1: La tarea Nombre esta ejecución.
- >1: Código de error.

## PROGRAMA START Y STOP



Escribir en la subrutina START todas las instrucciones necesarias para arrancar la aplicación. Se ejecutan al pulsar la tecla Start.



La subrutina STOP se llama cuando se pulsa la tecla stop o cuando se finalizan todas las subrutinas.

Cuando se pulsa la tecla Stop, todas las subrutinas son automáticamente finalizadas excepción de las multitareas.

Es posible adaptar el programa Stop para finalizar las tareas en un cierto orden y ejecutar otras tareas necesarias para finalizar el programa (posicionamiento final robot, retornos a origen, etc)

## FORMACIÓN VAL3



# *Librerías*



## OBJETIVO

Utilización de programas/datos en múltiples aplicaciones.

Ejemplo:

- Librerías de programas para reutilizar su código fuente.
- Librerías PIEZAS para utilizar una única aplicación con múltiples referencias de piezas.  
=> 1 aplicación única + 1 librería de puntos para cada referencia de piezas.

Una librería es una aplicación normal, creada como una aplicación simple: desde el control CS8 o emulador CS8

**PERO** se deben definir los datos/programas que serán exportados y utilizados por otras aplicaciones.

## EXPORTACIÓN DE DATOS DE LA LIBRERÍA

```
10%
Application manager
-Global data
+f1 New data
-wo Name : pPuntoTomar
+f
+jo Type :point
+md
+bo Elements nb. : 1
+nu
st Attribut :Private
ai
+di
sio
config
Esc Ok
```

Ejemplo : El punto **pPuntoTomar** es privado y solo puede ser usado internamente a la aplicación.

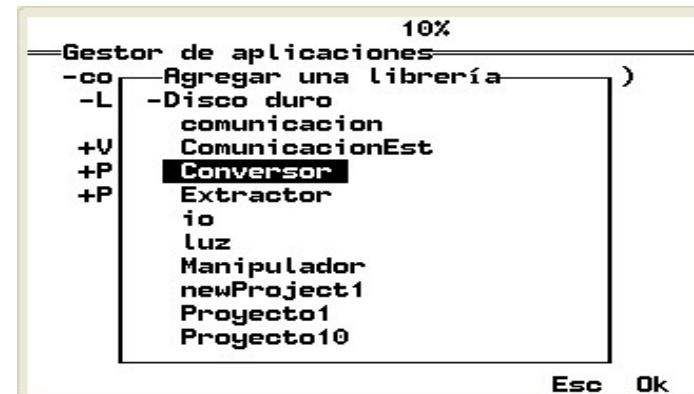
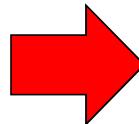
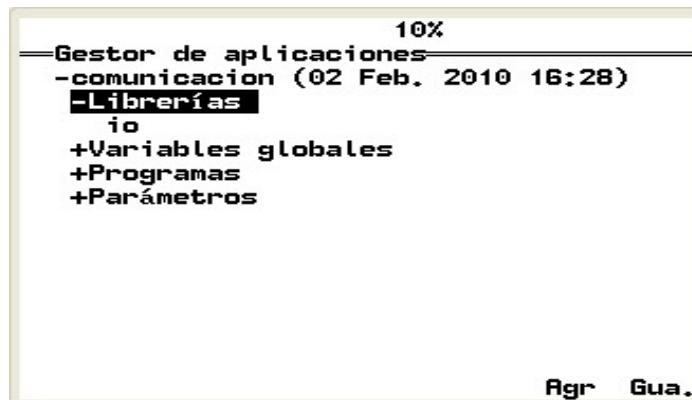
```
10%
Application manager
-Global data
+f1 New data
-wo Name : pPuntoDejar_
+f
+jo Type :point
+md
+bo Elements nb. : 1
+nu
st Attribut :Public
ai
+di
sio
config
Esc Ok
```

Cambia el atributo de Privado a **Público**

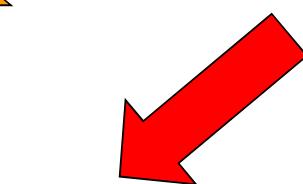
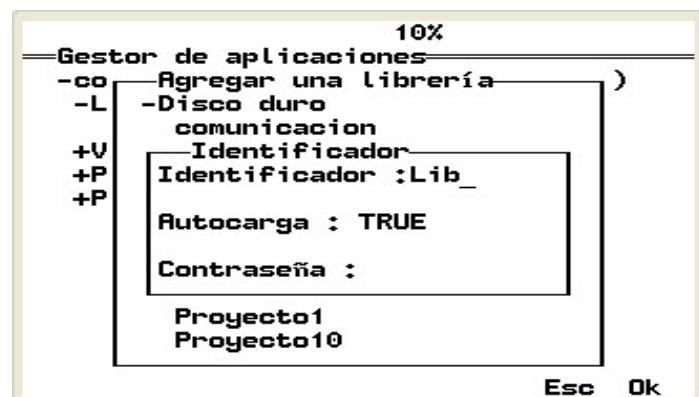
Ejemplo : El punto **pPuntoDejar** es público y por lo tanto accesible por otras aplicación.

# DECLARACIÓN DE LIBRERÍAS EN MCP

Las librerías utilizadas en una aplicación deben ser declaradas



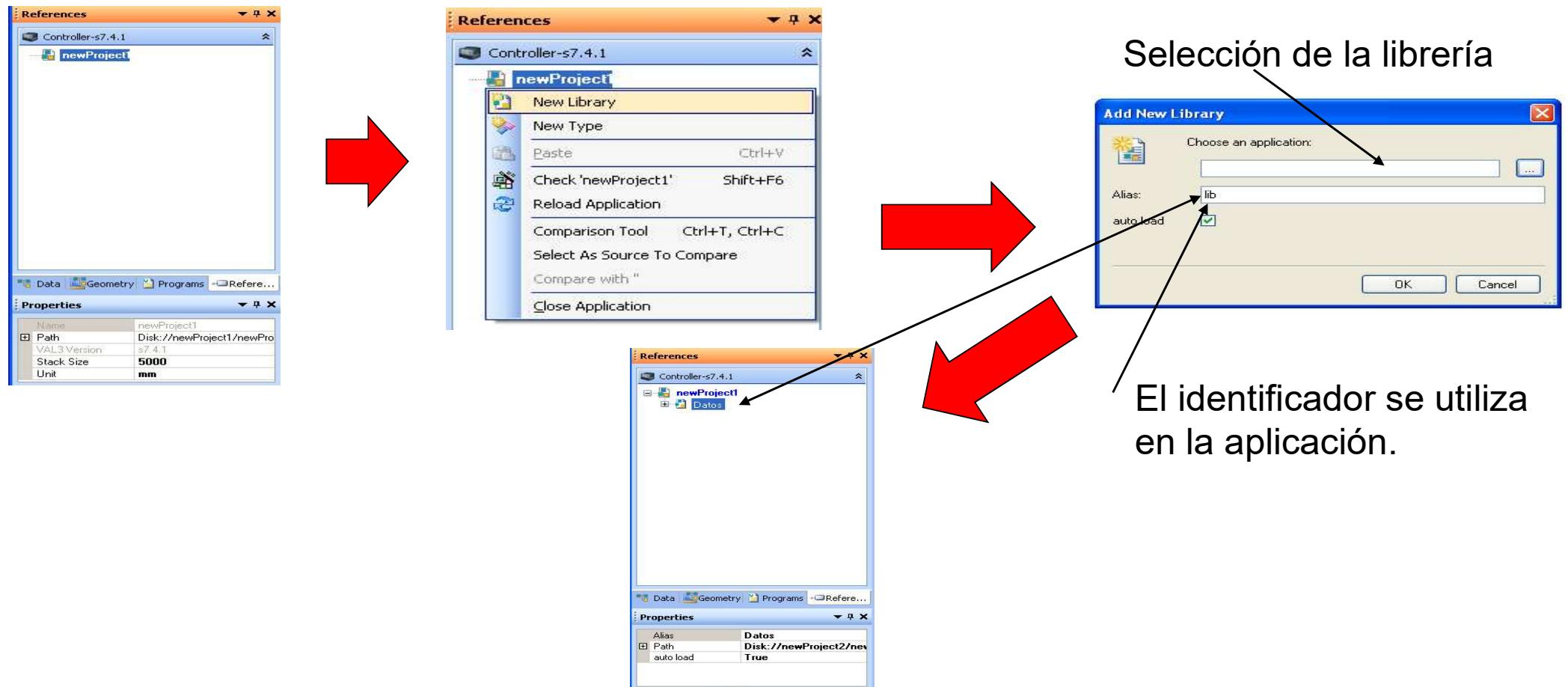
Selección de la librería



El identificador se utiliza en la aplicación que importa la librería

Ejemplo : La librería Conversor es utilizada en la aplicación “comunicacion” con el identificador “lib”

# DECLARACIÓN DE LIBRERÍAS EN SRS



# UTILIZACIÓN DE DATOS

Utilización del punto pPuntoDejar definido como público en la librería comunicación y con identificador lib:

movej(lib:pPuntoDejar, tPinza, mRapido)

- Asignación de la variable numérica nX definida como pública en la librería:

lib:nX=10

- Llamada al programa INIT definido como público en la librería:

call lib:init()

Durante la edición del programa, el sistema garantiza la coherencia de los datos y la sintaxis de los mismos.

## CARGAR LIBRERÍAS

Ejemplo: Una aplicación con múltiples librerías de puntos. ptA está definido como público en ref1 y en ref2.

```

nErr=lib:libload("ref2")           carga ref2 con el identificador Lib
//Resultado de la carga en nErr, nErr=0 si se carga bien
movej(lib:ptA,tPinza,mRapido)    utiliza ptA de ref2
nErr=lib:libload("ref1")           carga ref1 con el identificador Lib
//Resultado de la carga en nErr, nErr=0 si se carga bien
movej(lib:ptA,tPinza,mRapido)    utiliza ptA de ref1

```



1 sólo identificador de librería para trabajar con varias librerías con los mismos datos exportados pero con valores diferentes

## SALVAR LIBRERÍAS

**libSave ()**: Guarda las variables y programas asignados al identificador de librería.

nErr=lib:libSave()

nErr=lib:libload("ref2")

movej(lib:pA,flange,nom\_speed)

lib:nX=10

nErr=lib:libSave()

nErr=lib:libload("ref1")

put(lib:nX)

Guarda la librería en memoria con el nombre lib

Carga la librería ref2 con el identificador lib

Usa el pA de Ref2

Modifica NX de ref2

Guarda nX en el disco

Carga la librería ref1 con el identificador lib

Muestra el valor de nX de la librería ref1

nErr=lib:libSave("ref3")

nErr=lib:libload("ref2")

nErr=lib:libSave("ref3")

Guarda la librería en memoria con un nuevo nombre ref3

Copia « ref2 » to « ref3 »

## ERRORES DE GESTIÓN DE LIBRERÍAS

Todas las instrucciones de librerías devuelven un valor de error. Si el valor es 0 no hay errores.

`nErr= lib:libLoad("data")`

11 : Error de carga = interfaz publica no corresponde con la librería.

12 : Dato o programa invalido en la librería.

`nErr = lib:libLoad("data") lib:libSave("data") libDelete("data")`

Error en la cadena asignada:

20 : Unidad no valida.

21 : Camino invalido.

22 : Nombre incorrecto.

`nErr = lib:libLoad("data") lib:libSave() lib:libSave("data")`

30 : Error de escritura / lectura (floppy, ftp, ....)

`nErr = lib:libSave("data")`

31 : Error de escritura = la librería ya existe (no hay posibilidad de sobreescritura → primero borrar la librería vieja)

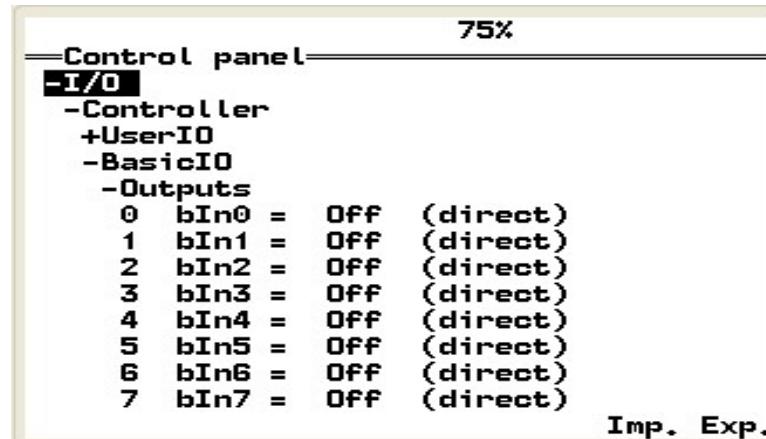
# LIBRERÍA IO (VAL<7)

I/O se usa como una libreria.

```
wait (io:btnSTART==true)
```

La librería IO está automáticamente creada por el sistema y actualizada cuando se realizan cambios en el panel de control.

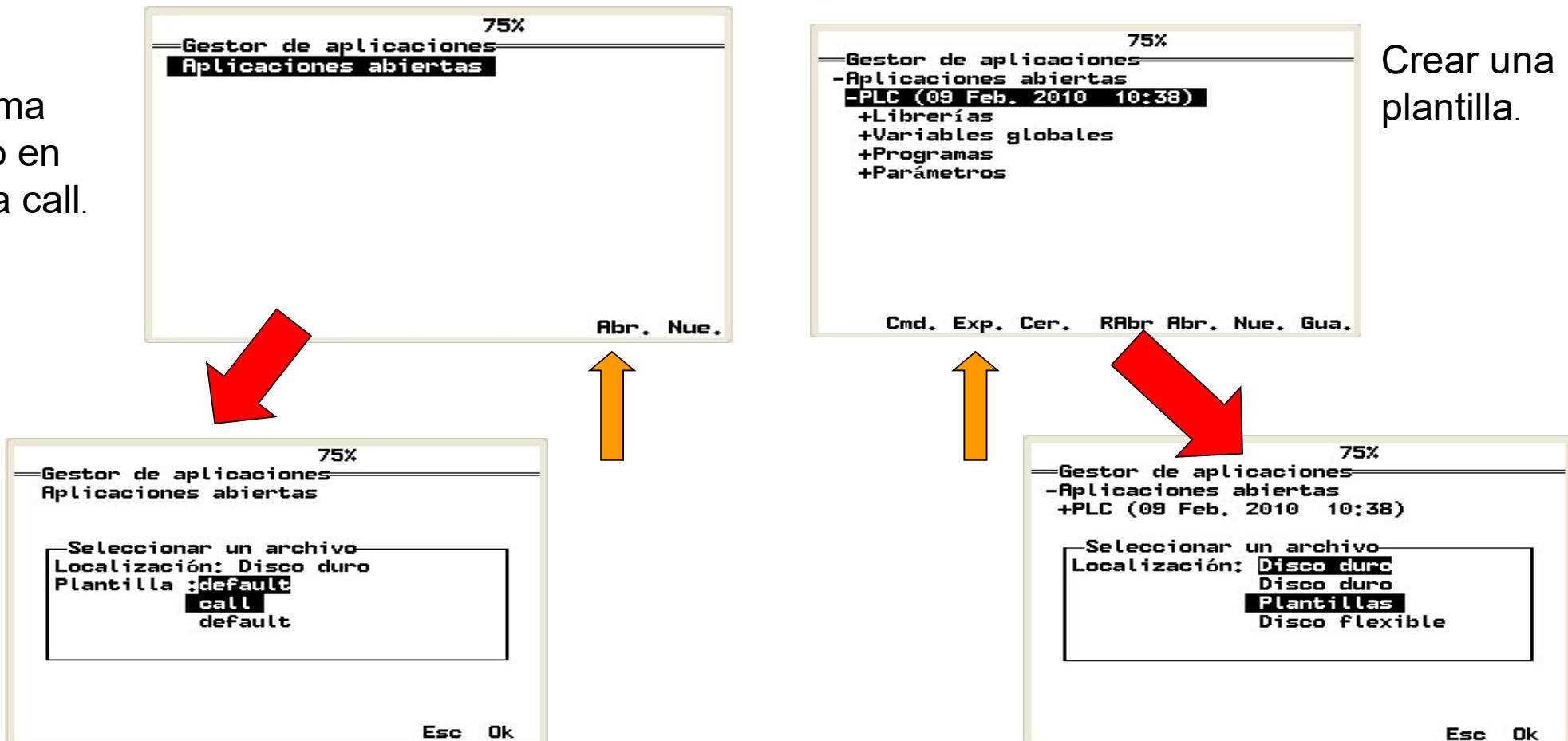
Al considerarse librería, si la modificamos, la asignación de esa io solo no funcionara en dicho sistema, tendremos que exportar / importar para poderla utilizar en otro equipo o en el simulador.



# PLANTILLAS

Plantillas: son programas base con subrutinas y datos predefinidos.

Crear  
programa  
basado en  
plantilla call.



## FORMACIÓN VAL3

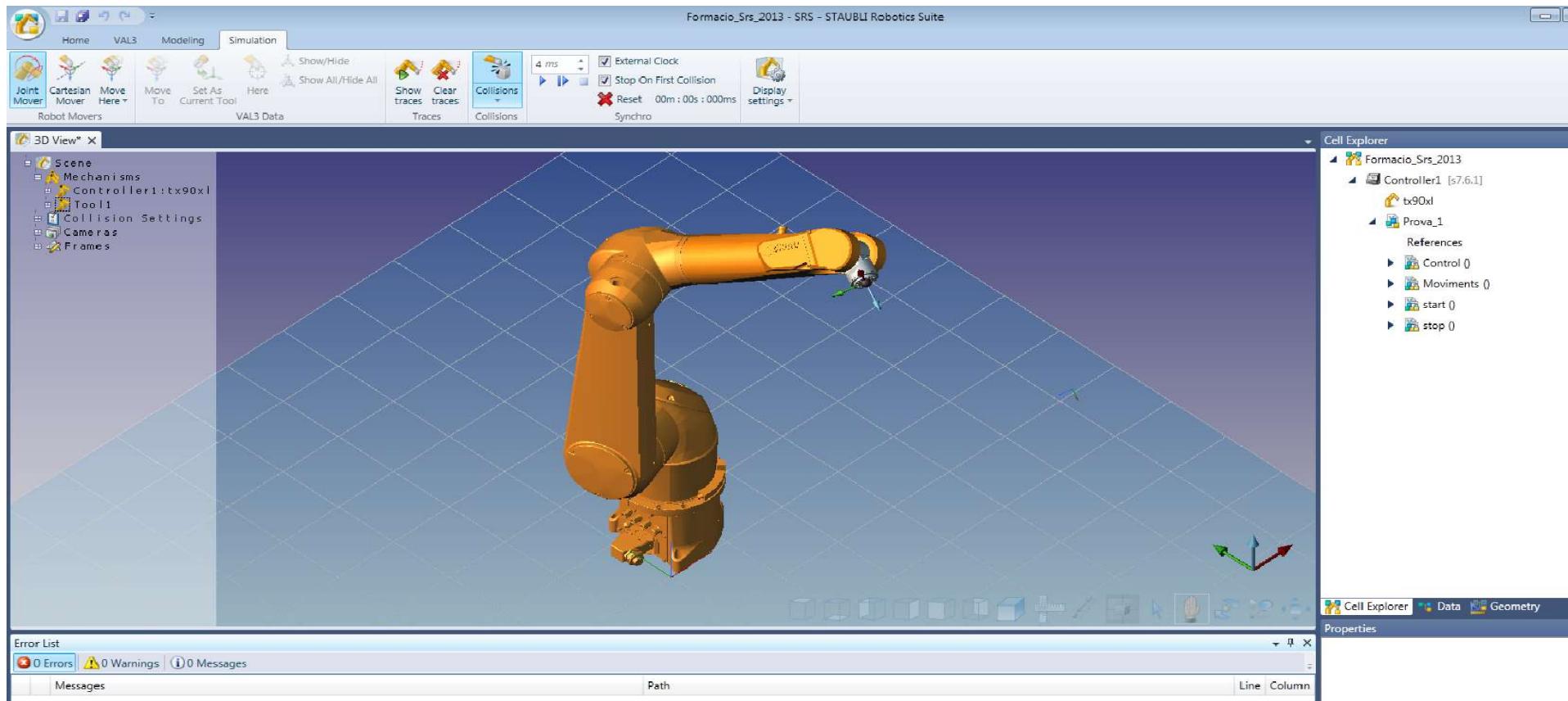


*Stäubli  
Robotics  
Studio*



# STÄUBLI ROBOTICS STUDIO ( SRS )

Herramientas de desarrollo sobre PC



# PERFILES



Tantos perfiles como sean necesarios.

Si previamente a llamar al editor de perfiles seleccionamos la opción “attach to ...” y “Target”, los perfiles serán los del controlador.

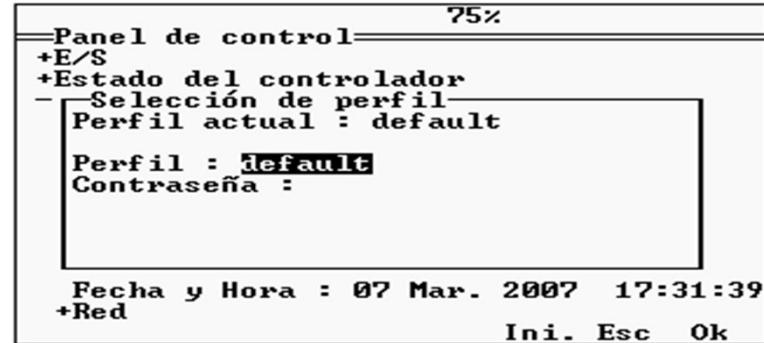
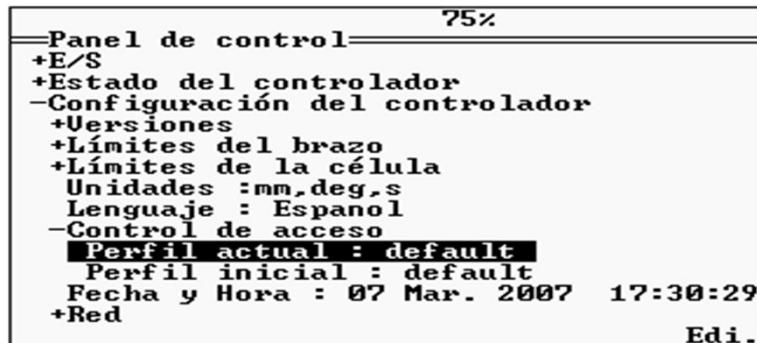


Access Right	Status
password	
connectionPassword	
writeAccess	<input checked="" type="checkbox"/>
readAccess	<input checked="" type="checkbox"/>
armWriteAccess	<input type="checkbox"/>
recovery	<input checked="" type="checkbox"/>
ioWriteAccess	<input type="checkbox"/>
123KeysControl	<input type="checkbox"/>
manualMode	<input checked="" type="checkbox"/>
testMode	<input checked="" type="checkbox"/>
localMode	<input checked="" type="checkbox"/>
remoteMode	<input checked="" type="checkbox"/>
monitorSpeed	<input checked="" type="checkbox"/>
powerButton	<input checked="" type="checkbox"/>
moveHoldKey	<input checked="" type="checkbox"/>
stopKey	<input checked="" type="checkbox"/>
menuKey	<input checked="" type="checkbox"/>
hardwareFaultAck	<input checked="" type="checkbox"/>

password	<input type="text"/>
connectionPassword	<input type="text"/>
writeAccess	<input checked="" type="checkbox"/>
readAccess	<input checked="" type="checkbox"/>
armWriteAccess	<input type="checkbox"/>
recovery	<input checked="" type="checkbox"/>
ioWriteAccess	<input type="checkbox"/>
123KeysControl	<input type="checkbox"/>
manualMode	<input checked="" type="checkbox"/>
testMode	<input checked="" type="checkbox"/>
localMode	<input checked="" type="checkbox"/>
remoteMode	<input checked="" type="checkbox"/>
monitorSpeed	<input checked="" type="checkbox"/>
powerButton	<input checked="" type="checkbox"/>
moveHoldKey	<input checked="" type="checkbox"/>
stopKey	<input checked="" type="checkbox"/>
menuKey	<input checked="" type="checkbox"/>
hardwareFaultAck	<input checked="" type="checkbox"/>

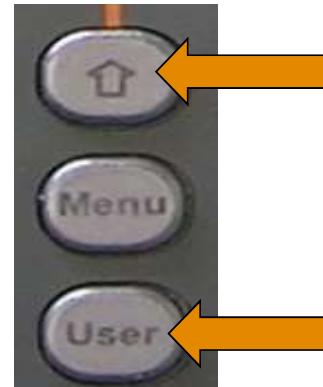
Password	Contraseña del perfil
ConnectionPassword	Contraseña red para transferencia de ficheros y mantenimiento remoto (opción)
writeAccess	Autorización de guardar modificaciones(configuraciones y programas)
readAccess	Autorización para abrir aplicaciones
armWriteAccess	Autorización para calibrar: ajuste del motor, aprendizaje de los puntos de referencia
recovery	Autorización de la recuperación de calibración después de la desconexión de la batería (sólo sobre RX)
ioWriteAccess	Autorización para forzar salidas desde el panel de control
123KeyControl	Autorización para la configuración de las teclas 1, 2, 3 (electro válvulas 1 y 2 por defecto)
manualMode	Autorización del modo de trabajo « manual »
testMode	Autorización del modo de trabajo « test »
localMode	Autorización del modo de trabajo «auto local »
remoteMode	Autorización del modo de trabajo «auto remote »
monitorSpeed	Autorización de cambio a % de velocidad
powerButton	Autorización para corte de potencia en modo remoto
moveHoldKey	Activación de la tecla « move/hold » en modo auto/remoto
stopKey	Autorización de paro de una aplicación
menuKey	Autorización de navegación desde el menú principal CS8
HardwareFaultAck	

## USO DE PERFILES DESDE EL MCP

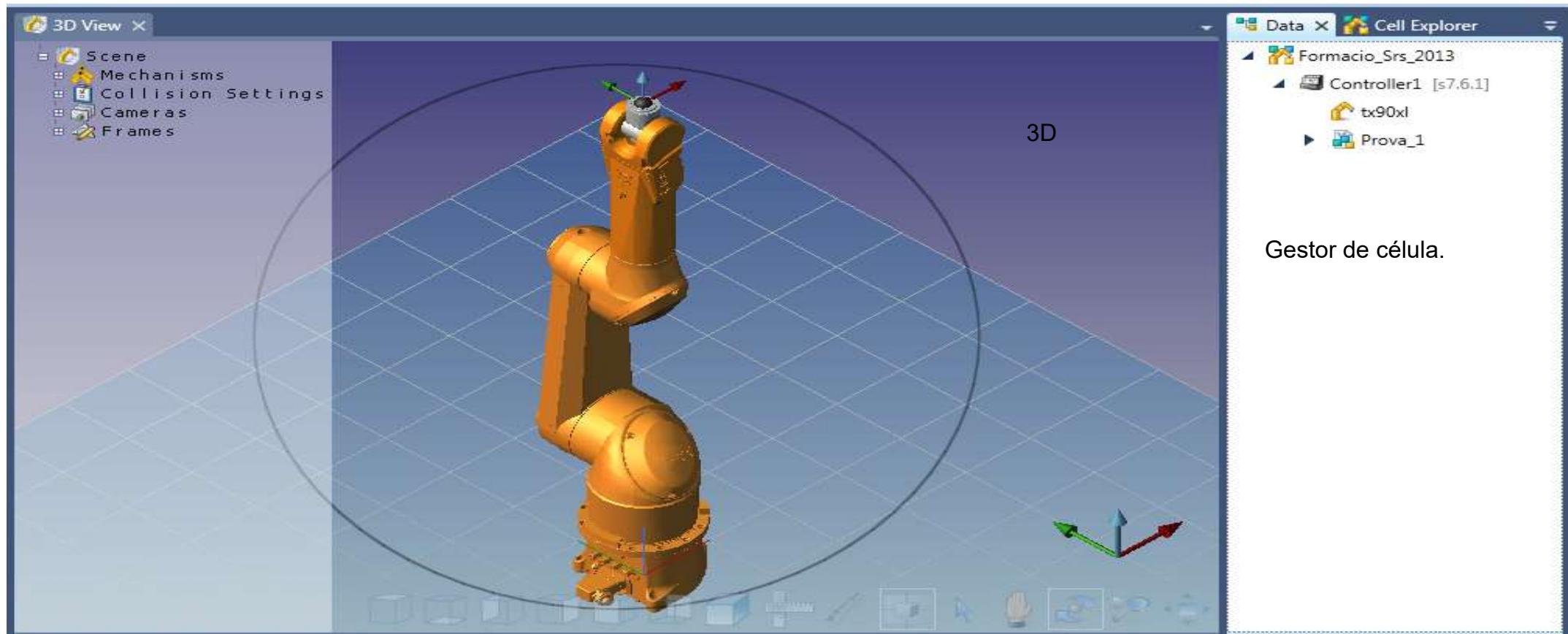


### Desde el panel de control:

- Posibilidad de definir el perfil al arranque del controlador.
- Cambio de perfil actual.
  - Menú → Panel de control → Configuración del controlador → Control de acceso → Perfil actual.
  - Shift + User (En el emulador CS8 usar F11 en lugar de Shift.).

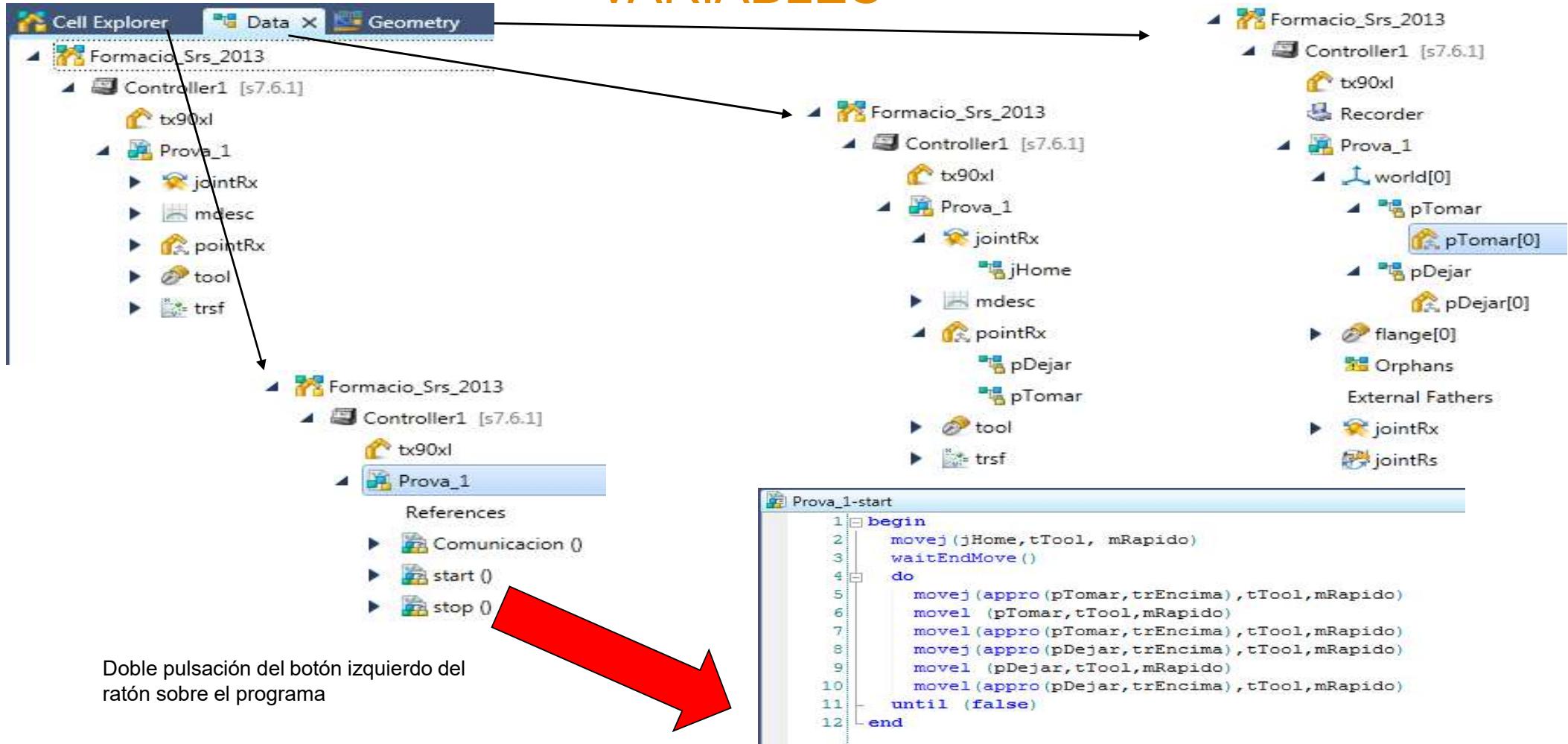


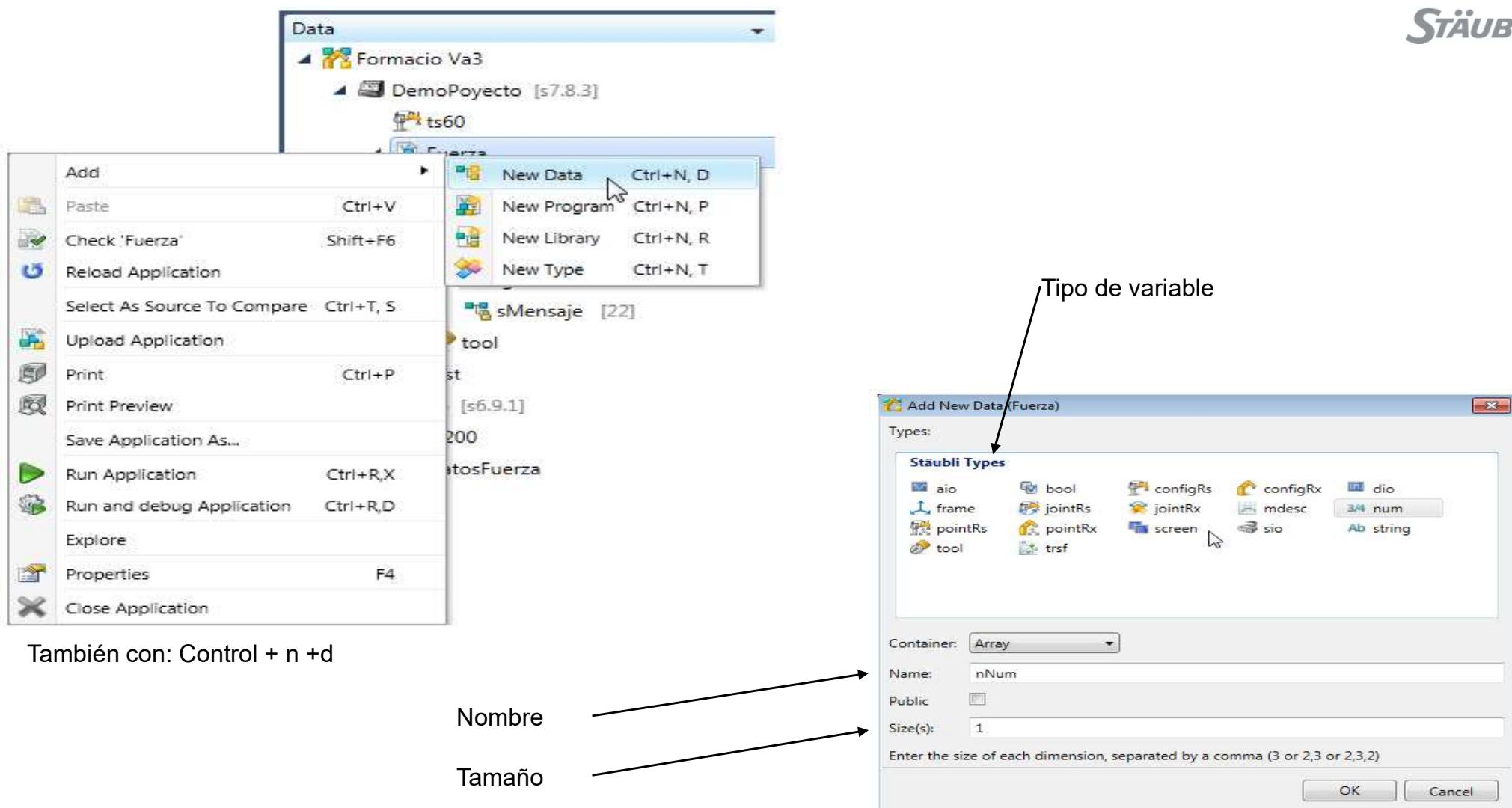
# VAL3 STUDIO

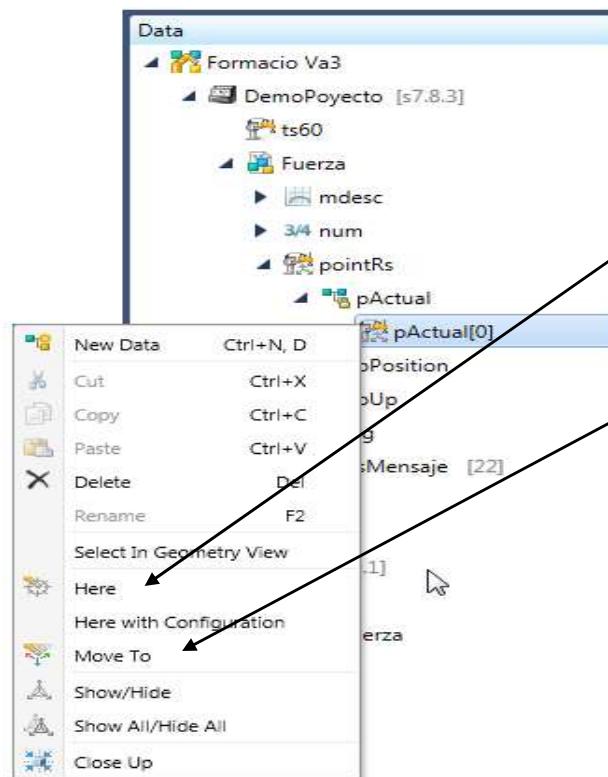


Entorno de programación offline.

# VARIABLES

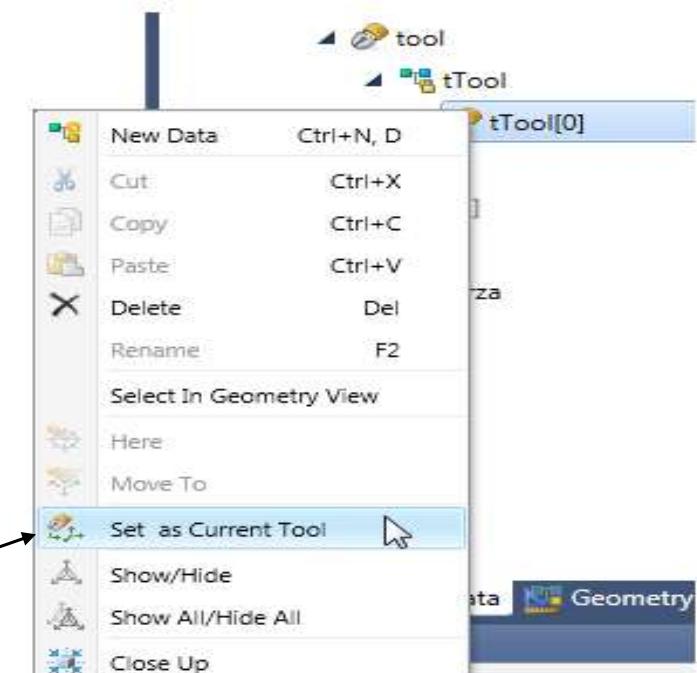






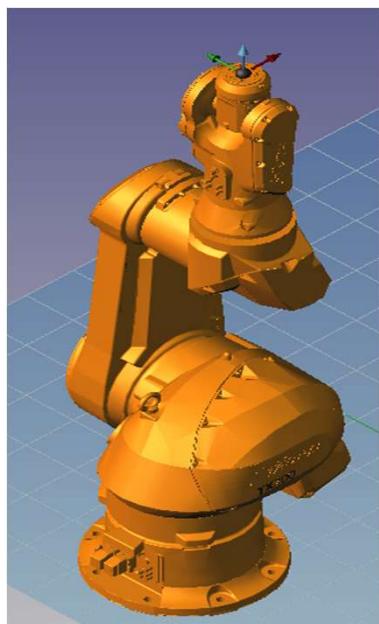
Permite guardar la posición actual (en el 3D)

Moverá el robot a la posición en cuestión.

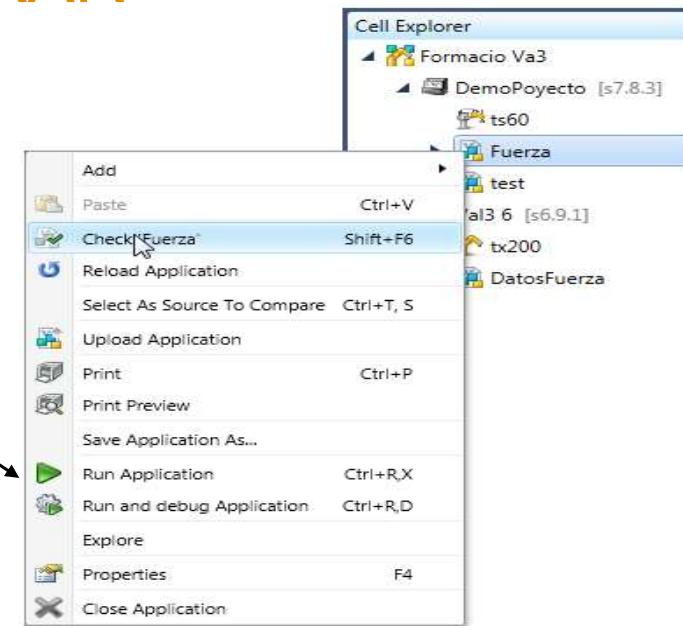
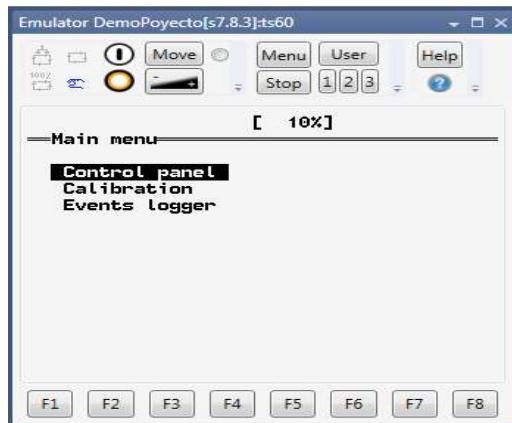


Establecer la pinza como actual.

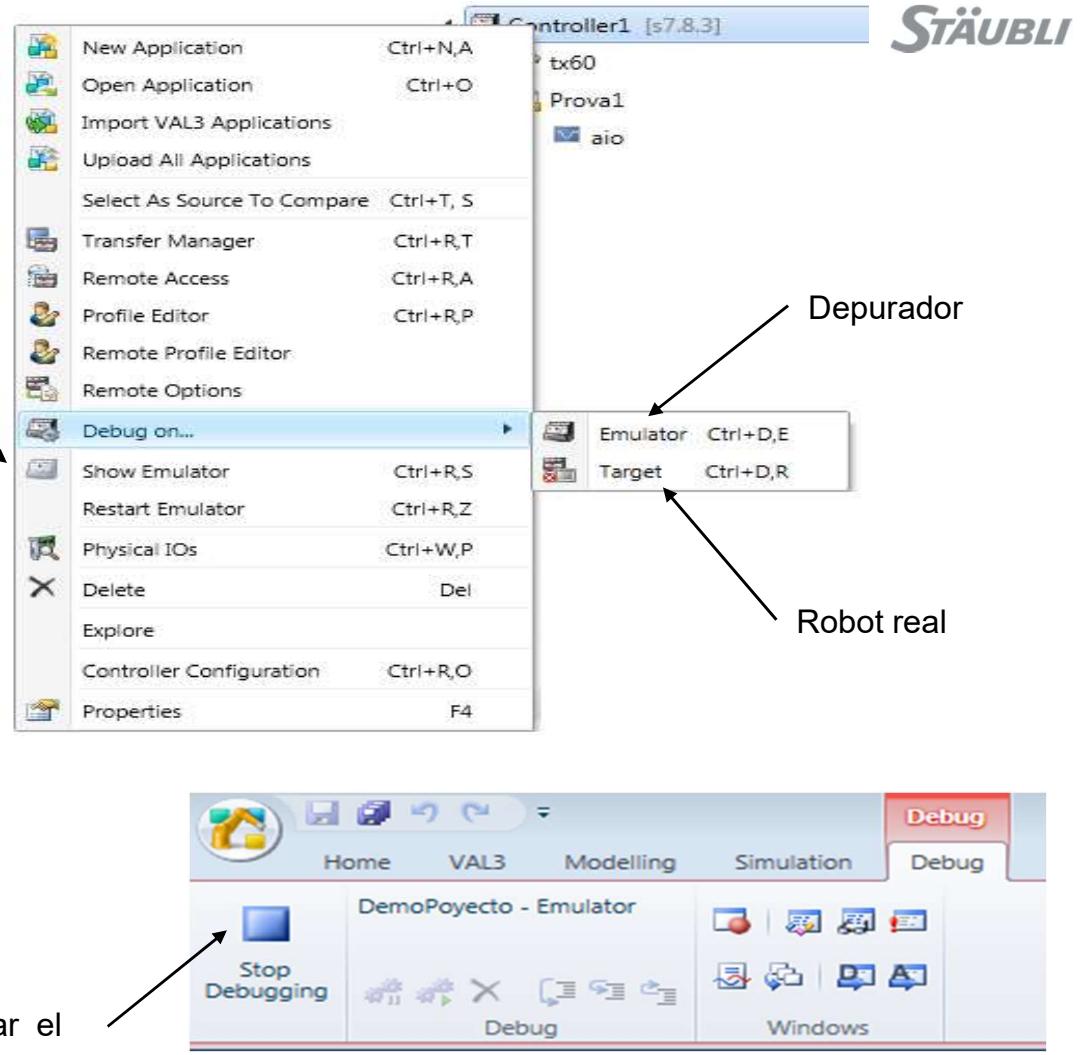
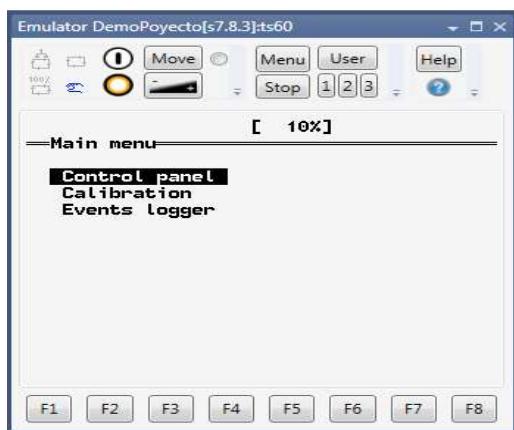
# SIMULAR Y DEPURAR



Simular programa



Simular y depurar el programa.



Es necesario parar el depurador para poder editar el programa

Durante la depuración es posible visualizar y editar diferentes parámetros

The screenshot displays three panels within a software application:

- Tasks**: A table showing two tasks:

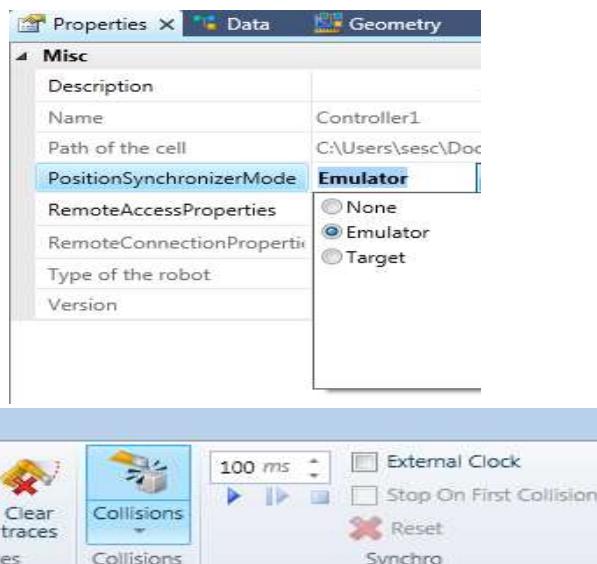
	Name	State	Priority	Created By	Location	Runtime Error	
	Move	Running	15	Disk://Prova_1/Prova_1.pjx	Moviments(4): movel (pTomar,tTool,mRapido)		
	Control	Running	2	Disk://Prova_1/Prova_1.pjx	Control(4): delay (1)		
- Watch**: A tree view of variables:

Name	Value	Type	Additional information
pActual	x=729.580713 y=509.510968 z=-178.189259 rx=174.434894 ry=-... trsf config	pointRx trsf configRx	Disk://Prova_1/Pro...
x	729.580713		
y	509.510968		
z	-178.189259		
rx	174.434894		
ry	-...		
rz	...		
trsf	trsf		
config	shoulder=lefty elbow=epositive wrist=wpositive		
- Dio Control Panel**: A table of digital inputs:

Description	Locked	Physical link	Value
bIn0	<input checked="" type="checkbox"/>	BasicIO-1\%I0	0
bIn1	<input type="checkbox"/>	BasicIO-1\%I1	0
bIn2	<input type="checkbox"/>	BasicIO-1\%I2	0
bIn3	<input type="checkbox"/>	BasicIO-1\%I3	0
bIn4	<input checked="" type="checkbox"/>	BasicIO-1\%I4	0
bIn5	<input type="checkbox"/>	BasicIO-1\%I5	0

## Opciones del simulador

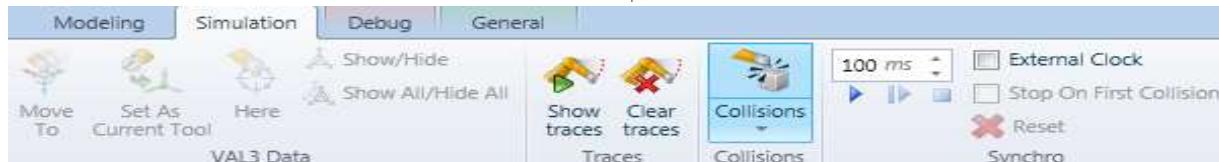
Propiedades del controlador



None: Robot 3D sin vinculación.

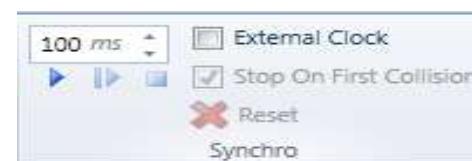
Emulator : Robot 3D vinculado al emulador.

Target: Robot 3D vinculado al controlador real.

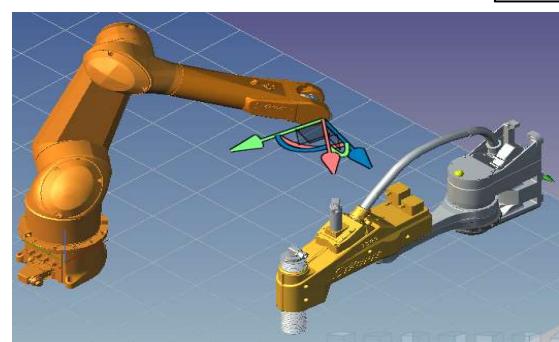
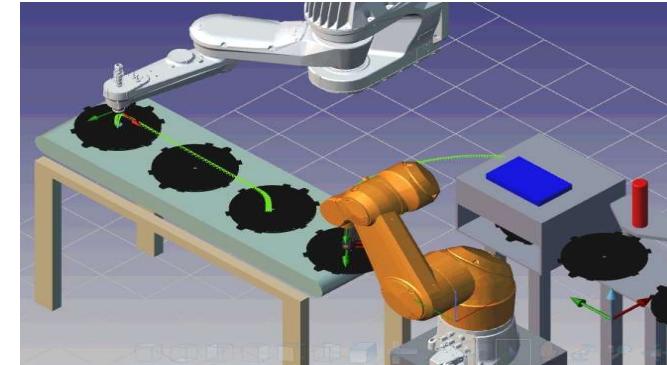
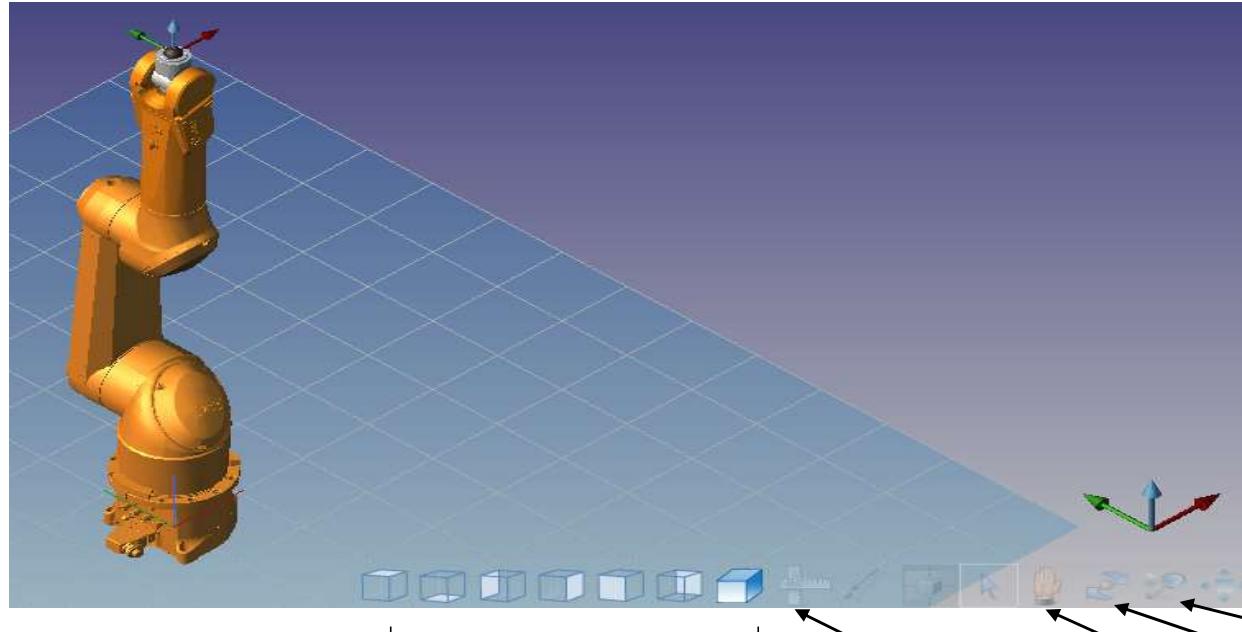


Seleccionar el modo de reloj:

- Interno: Cada emulador sincroniza su reloj con su generador de movimientos. El periodo es solo para refrescar el 3D.
- Externo: los generadores de movimiento de los emuladores son sincronizados por el SRS en el período especificado. En cada señal de reloj, el SRS permite a los generadores de movimiento realizar un paso adelante (un paso = período de tiempo determinado), recuperar la posición de los robots, refrescar la posición de los robots en el 3D y actualizar el valor del temporizador. Todo esto lo que permite es: medidas tiempo de ciclo realistas y detección de colisiones más precisa .



# 3D STUDIO



Vistas:

- Planta.
- Inferior.
- Vista lateral izquierda.
- Vista lateral derecha.
- Alzado.
- Posterior.
- Standard.

Centrar vista actual.

Zoom.

Girar plano.

Desplazar en plano.

Medida.



The image shows a screenshot of the STÄUBLI software interface. At the top, there's a menu bar with tabs for "3D View", "Debug", and "Windows". Below the menu, a toolbar has buttons for "Cartesian View", "Joints View", "Cartesian Move To", and "Edit Position". A callout arrow points from the "Edit Position" button to a "Move to ..." dialog box. This dialog box contains fields for X: 701,48, Y: 0,86, Z: -200,55, Rx: 174,21, Ry: -6,86, and Rz: 84,63. Another callout arrow points from the "Edit Position" button to a "Controller1:tx90xl" panel. This panel displays six joints (J1-J6) with their current angles: J1 (136) at 94,72°, J2 (57) at -34,90°, J3 (39) at 38,98°, J4 (100) at -103,47°, J5 (68) at 62,09°, and J6 (136) at -135,91°. A third callout arrow points from the "Edit Position" button to a "Windows" context menu. The menu includes options like "Floating", "Dockable", "Tabbed Document", "Auto Hide", "Hide", "Auto Hide All", "Close All Documents", "Layouts", "Reset", "Save as ...", "Save as default for new cell", and "Load ...".

Controller1:tx90xl

X: 68,93  
Y: -288,89  
Z: 1229,60  
Rx: 11,82  
Ry: 60,53  
Rz: -153,95

Es posible visualizar y mover el robot.

Move to ... Controller1:tx90xl

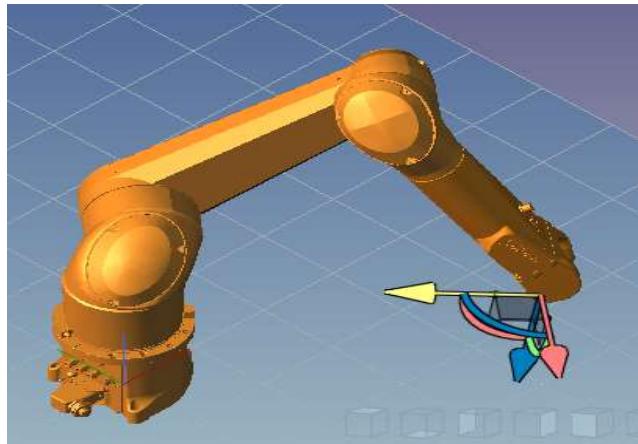
X: 701,48  
Y: 0,86  
Z: -200,55  
Rx: 174,21  
Ry: -6,86  
Rz: 84,63

Controller1:tx90xl

J1 (136) 94,72°  
J2 (57) -34,90°  
J3 (39) 38,98°  
J4 (100) -103,47°  
J5 (68) 62,09°  
J6 (136) -135,91°

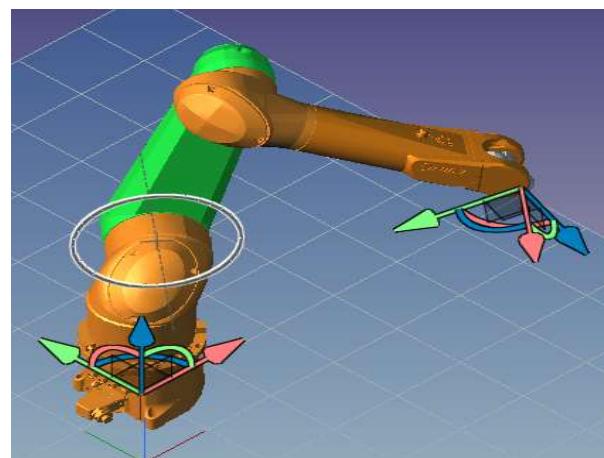
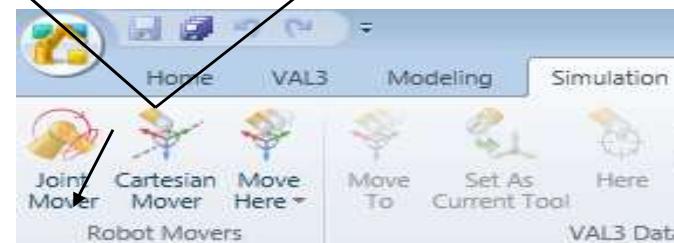
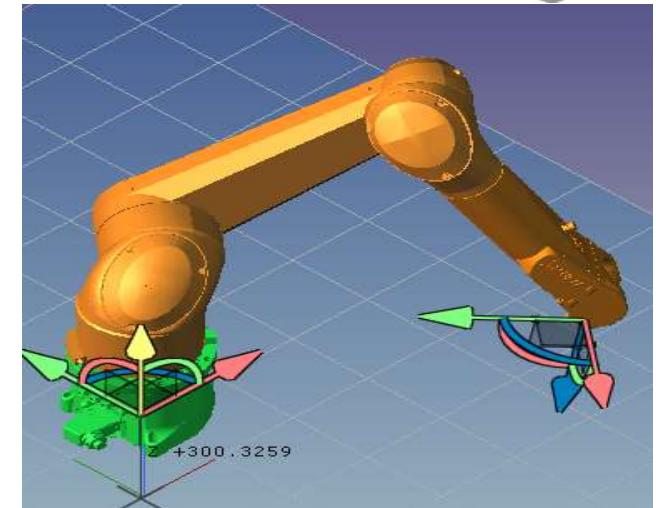
Windows

Floating  
Dockable  
Tabbed Document  
Auto Hide  
Hide  
Auto Hide All  
Close All Documents  
Layouts  
Reset  
Save as ...  
Save as default for new cell  
Load ...



Desplazamiento en tool: Seleccionando los ejes de coordenadas de la pinza y posteriormente manteniendo seleccionado uno mientras desplazamos el ratón.

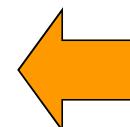
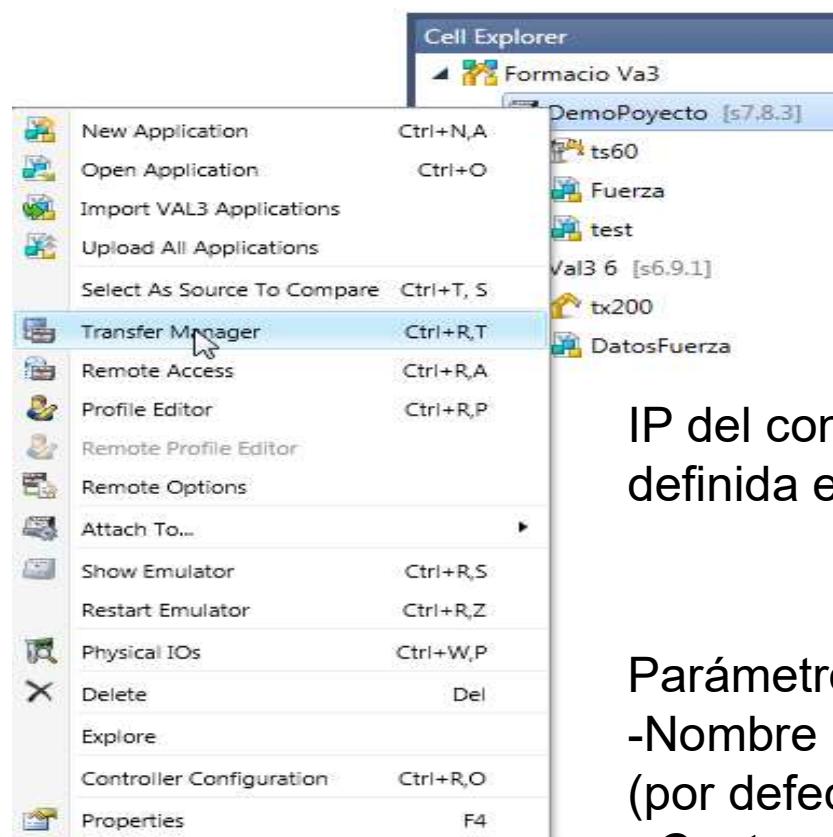
Desplazar base robot:  
Seleccionando los ejes de coordenadas de la base del robot y posteriormente manteniendo seleccionado uno mientras desplazamos el ratón.



Desplazamiento en Join: Seleccionando el ejes en cuestión durante unos segundos, aparecerá un eje de rotación con el que podremos mover el eje.

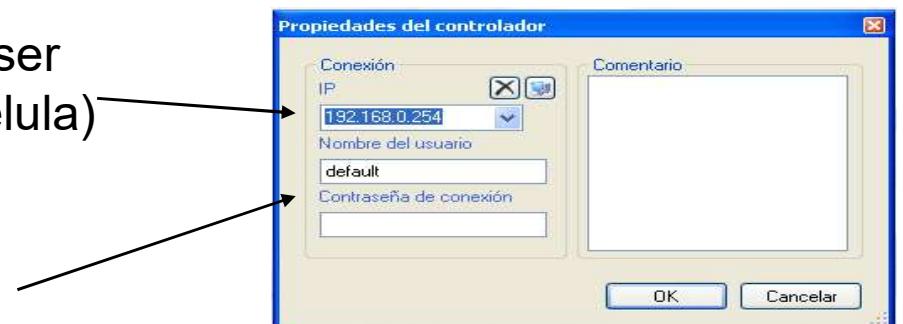
# GESTOR DE TRANSFERENCIAS

STÄUBLI



Seleccionar célula o programa.

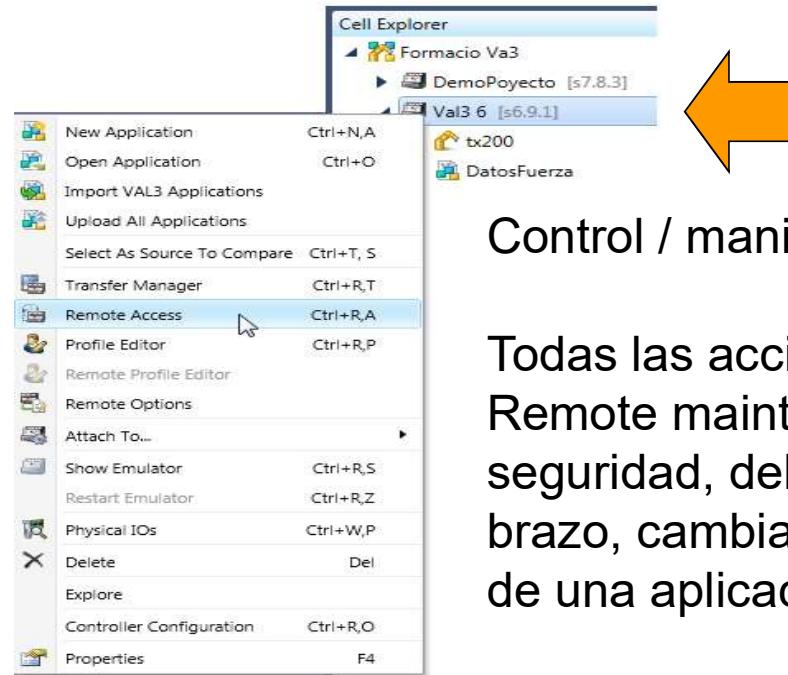
IP del controlador (puede ser definida en el gestor de célula)



Parámetros de acceso:

- Nombre usuario: default (por defecto)
- Contraseña: nado (por defecto)

# MANTENIMIENTO REMOTO

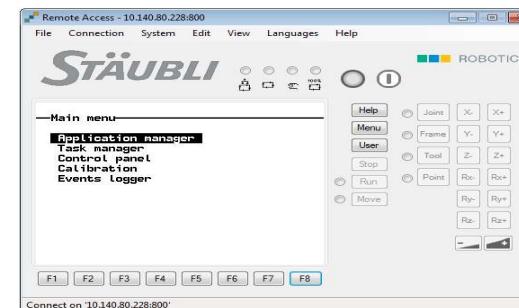
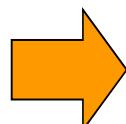


Seleccionar célula.

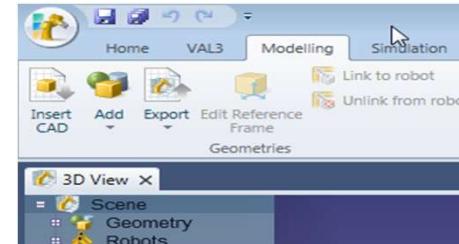
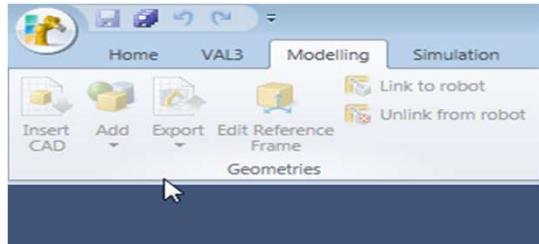
Control / manipulación remota del controlador.

Todas las acciones realizadas en el MCP se pueden hacer con el Remote maintenance a excepción de las que comporten violación de la seguridad, debido a que no estamos delante del robot (Dar potencia al brazo, cambiar los modos de trabajo, movimientos, lanzamiento y paro de una aplicación.)

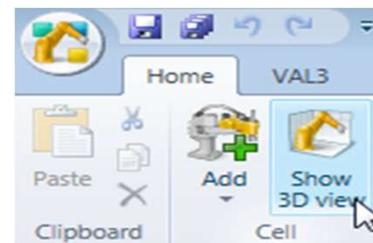
2 personas pueden actuar simultáneamente y ver las acciones del otro.



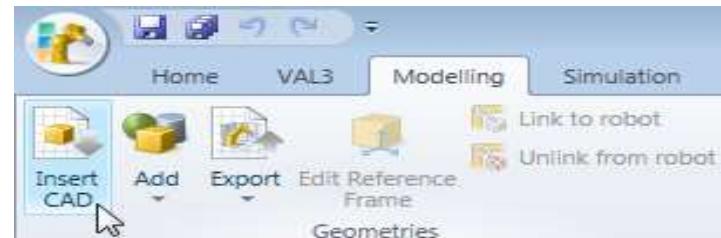
# AÑADIR ESCENARIO 3D



Mostrar la pantalla 3d.



Importar el 3D.



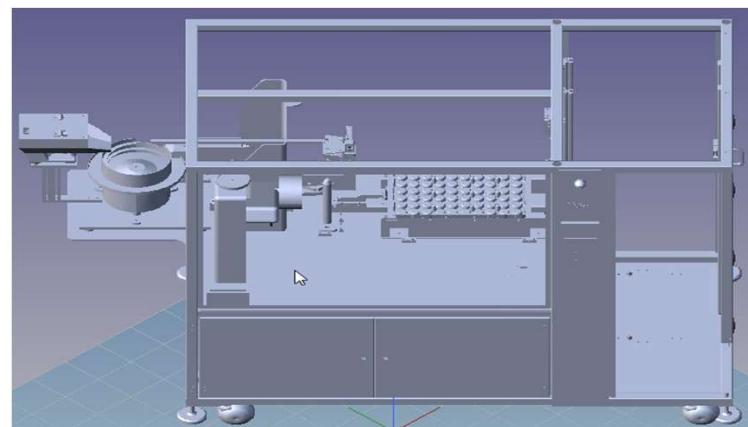
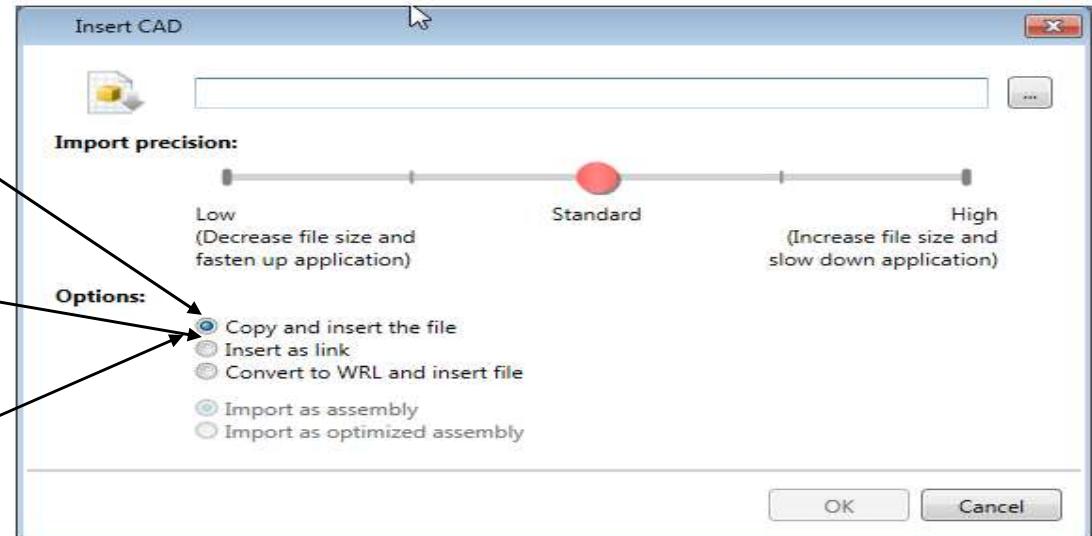
Formatos aceptados

Geometry Files (\*.iges, \*.igs, \*.step, \*.stl, \*.stp, \*.wrl)

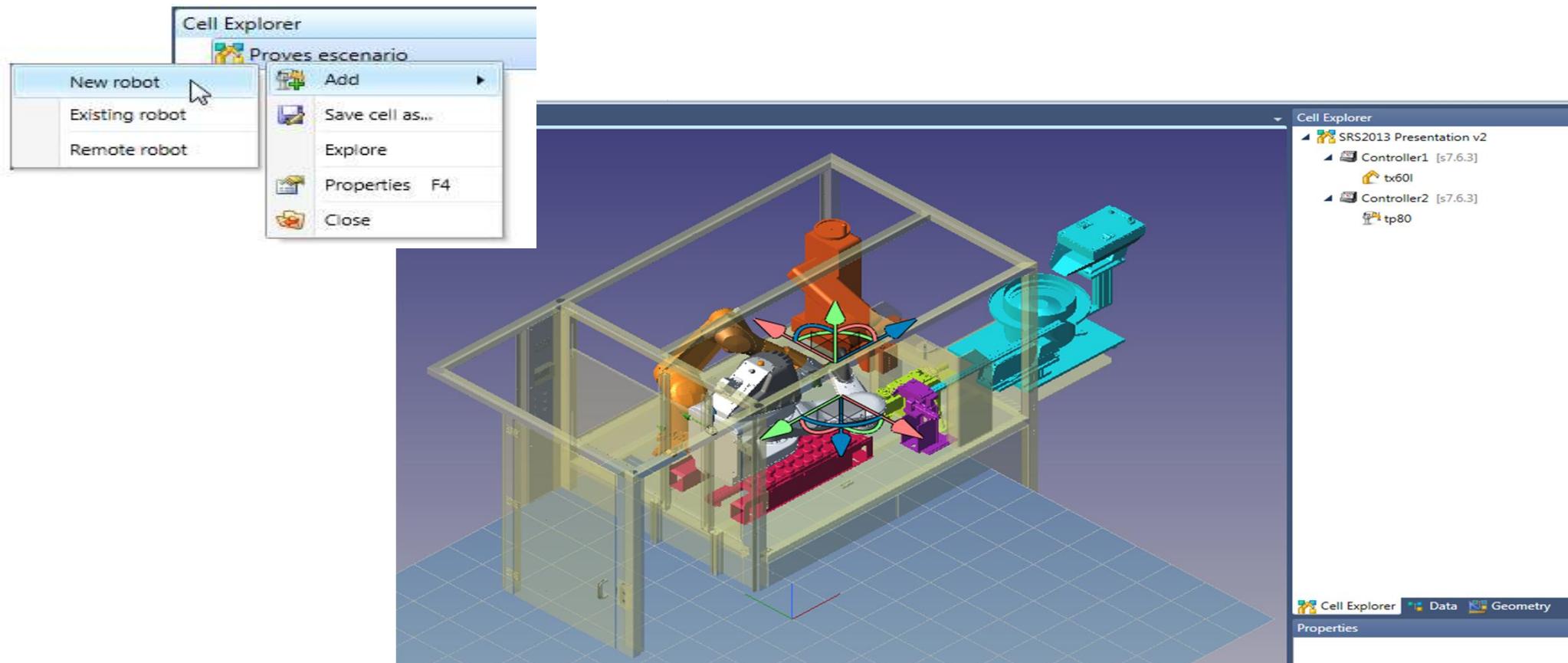
Guarda una copia en nuestra célula.

El 3d queda enlazado con la carpeta original

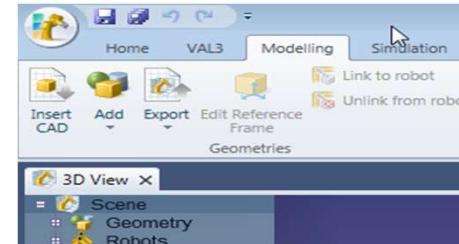
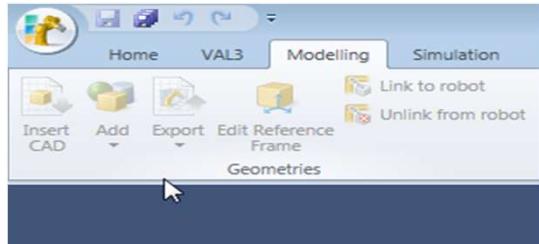
Transforma a WRL y copia en nuestra célula.



Una vez tenemos el escenario WRL cargado, podemos posicionar los robots o añadir nuevos robots.



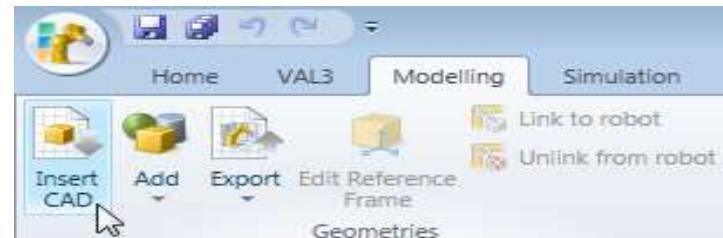
# AÑADIR PINZA 3D



Mostrar la pantalla 3d.



Importar el 3D.



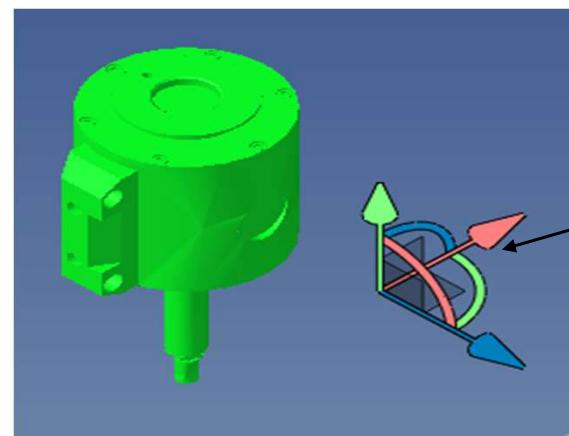
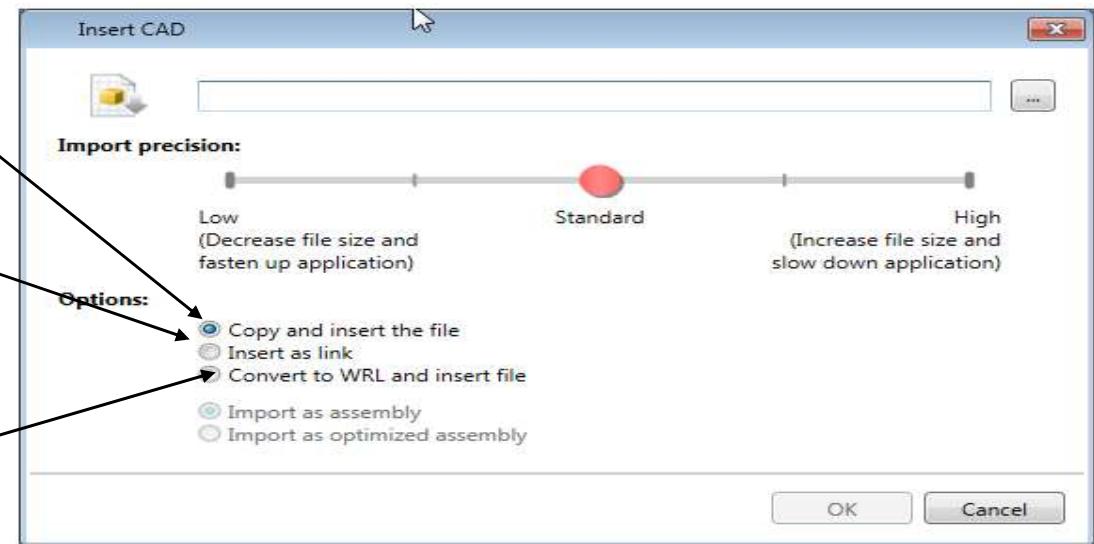
Formatos aceptados

Geometry Files (\*.iges, \*.igs, \*.step, \*.stl, \*.stp, \*.wrl)

Guarda una copia en nuestra célula.

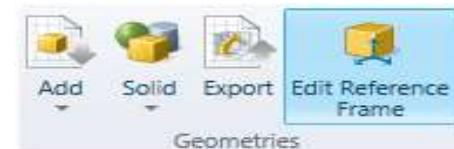
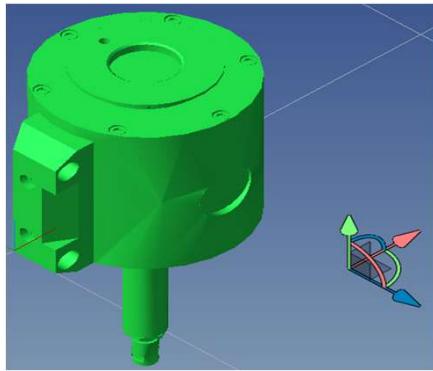
El 3d queda enlazado con la carpeta original

Transforma a WRL y copia en nuestra célula.



Centro / base de la pinza

Modificar / editar el centro / base de la pinza

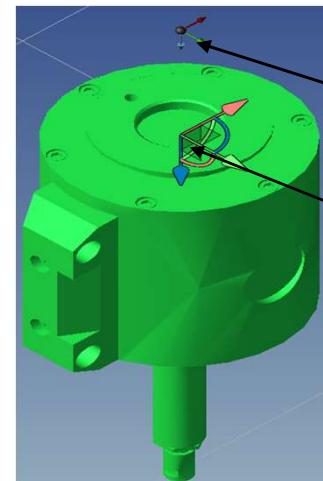


Opciones de selección



Iniciar / finalizar edición

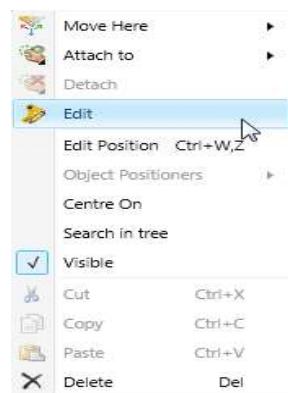
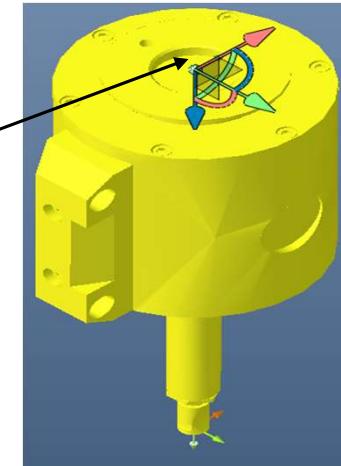
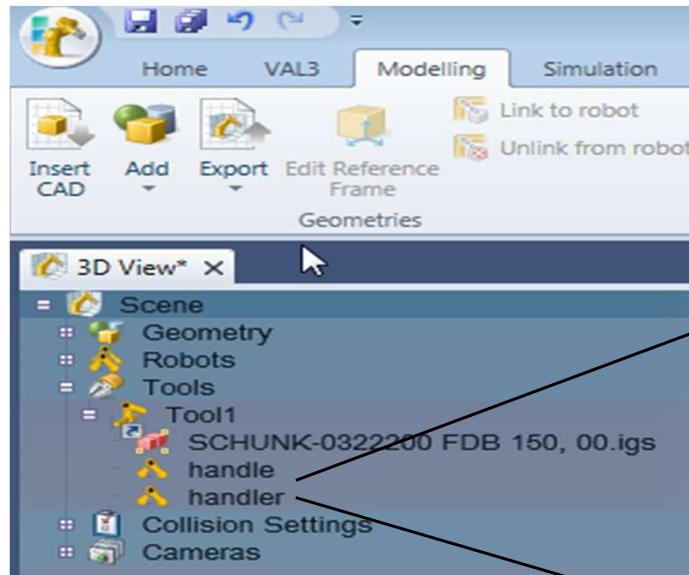
Crear la pinza.



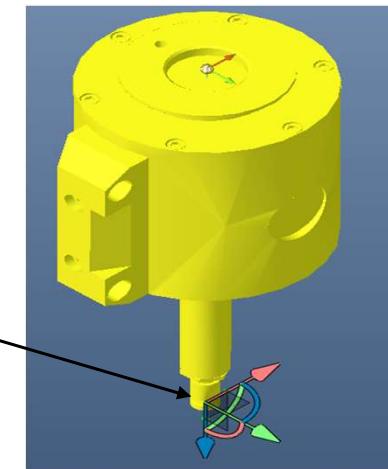
Base de la pinza (handle)

Punta de la pinza (handler)

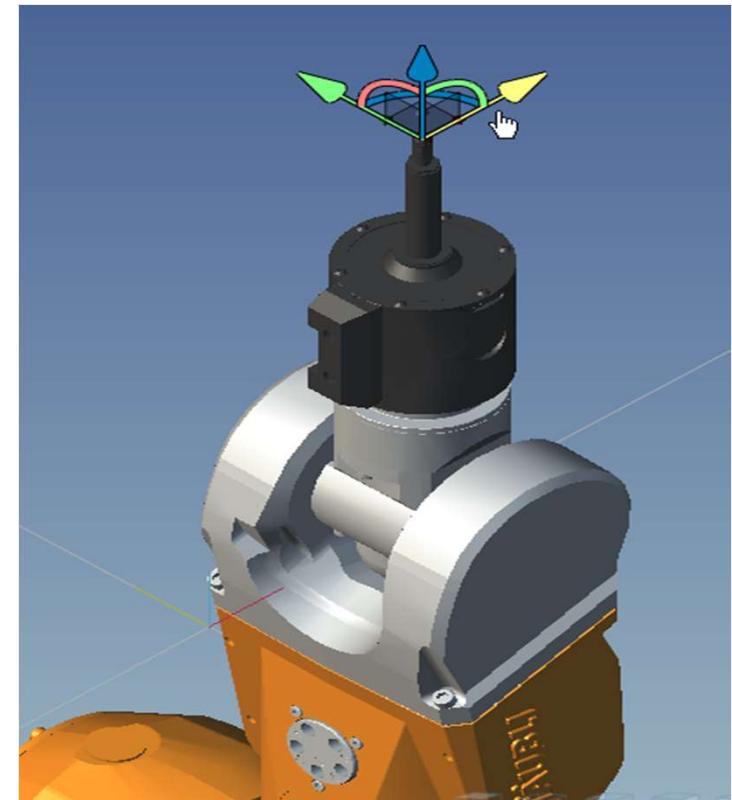
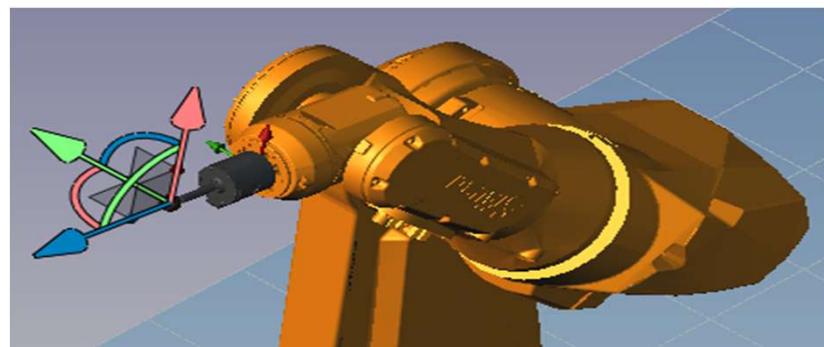
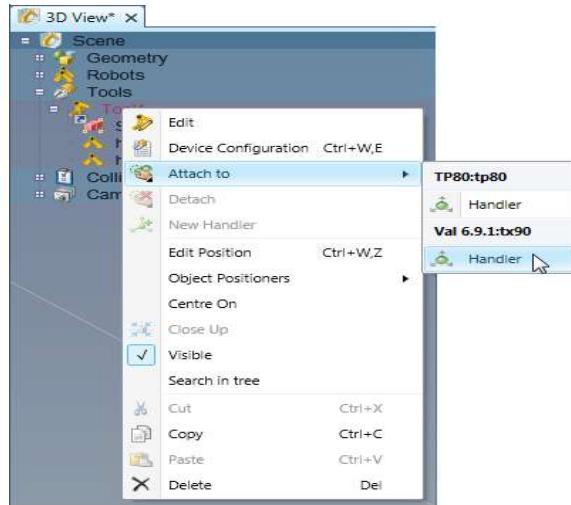
Modificar la punta y base de la pinza.



Al finalizar, recordar,  
desactivar la edición

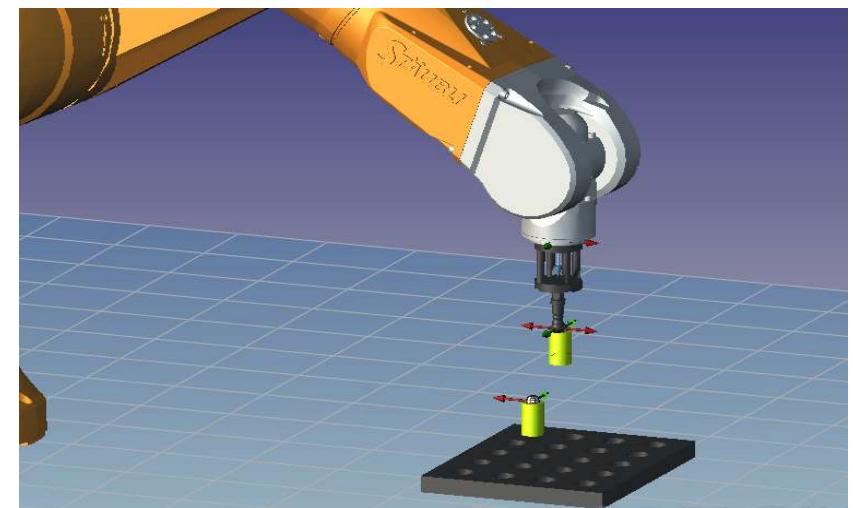
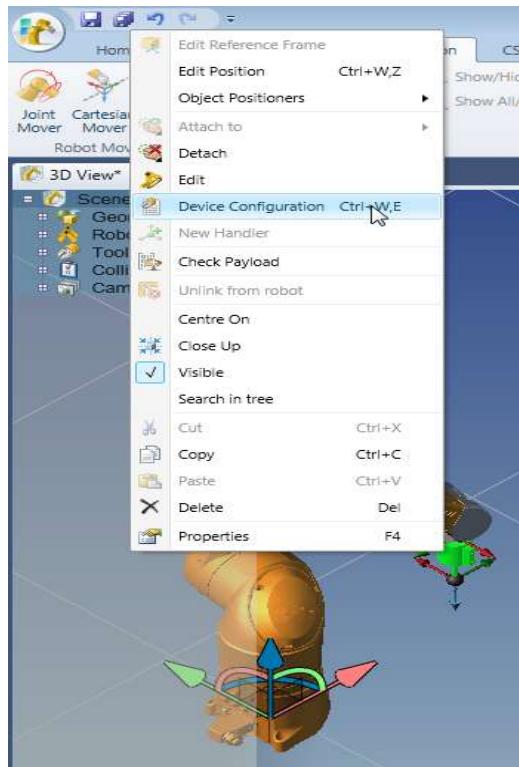


Adjuntar la pinza al robot.

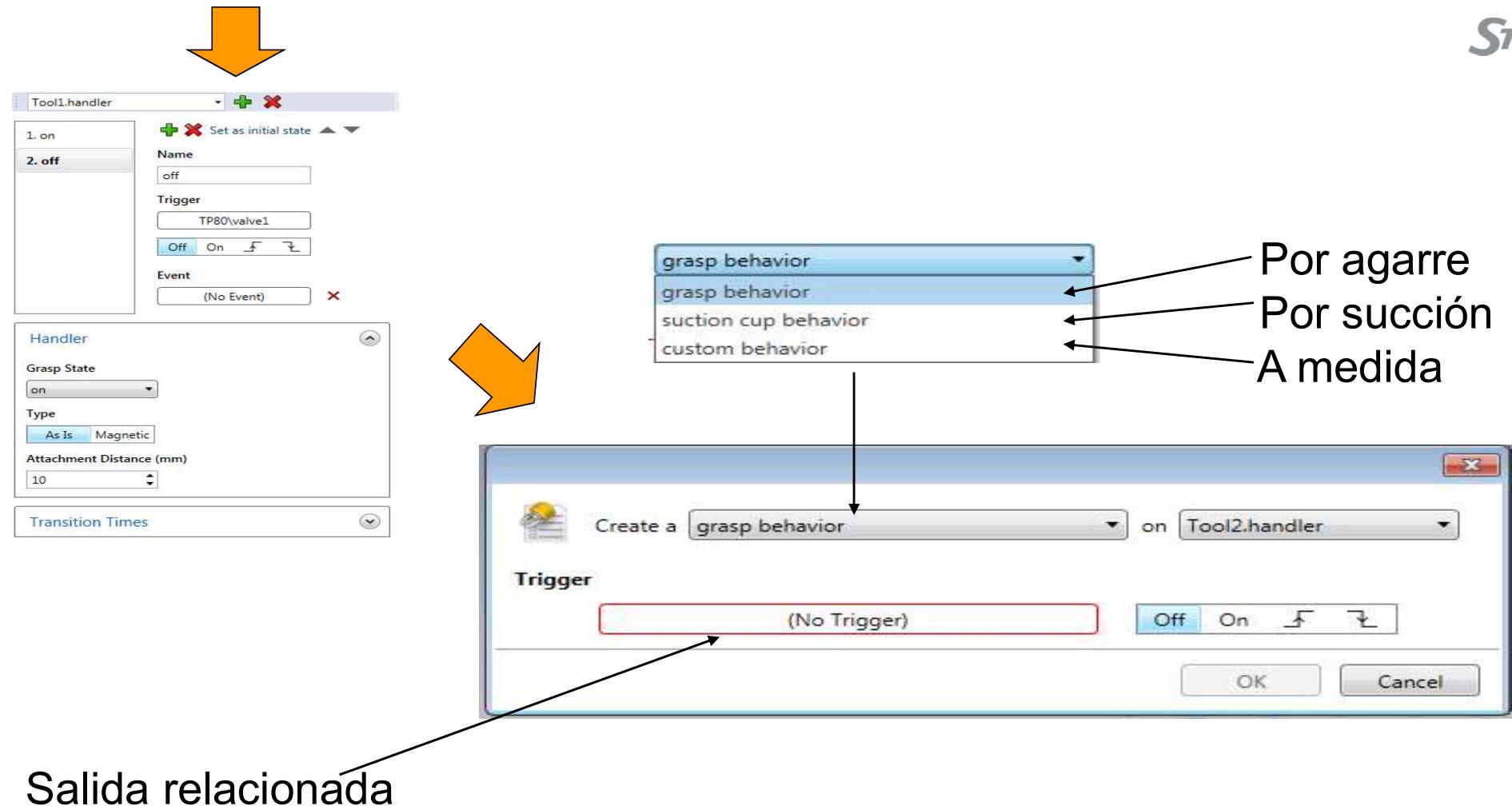


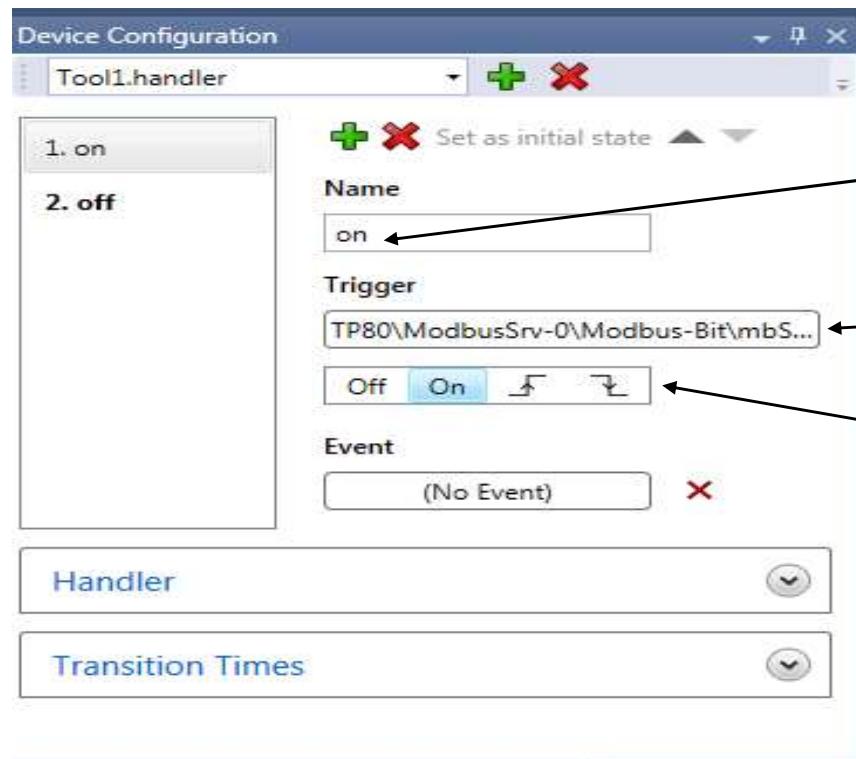
Movimientos cartesianos con pinza.

## MOVER PIEZAS



Hacemos clic derecho sobre la pinza, y seleccionamos “Device configuration”:



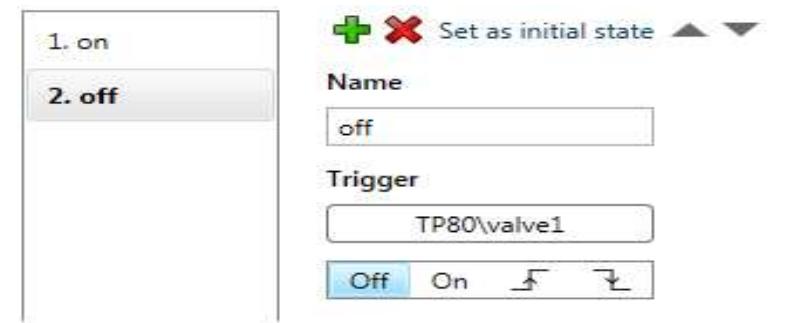


Acciones con la posible acción

Salida relacionada

¿Que hacemos con la salida?

El primero es el estado inicial



Handler

Grasp State ←

on

Type ←

As Is Magnetic

Attachment Distance (mm) ←

10

Estado para sujetar

Tipo:

- Como
- Magnética

Vinculada a la distancia

Transition Times ←

on - off

From 'on' To 'off' (ms)

500

From 'off' To 'on' (ms)

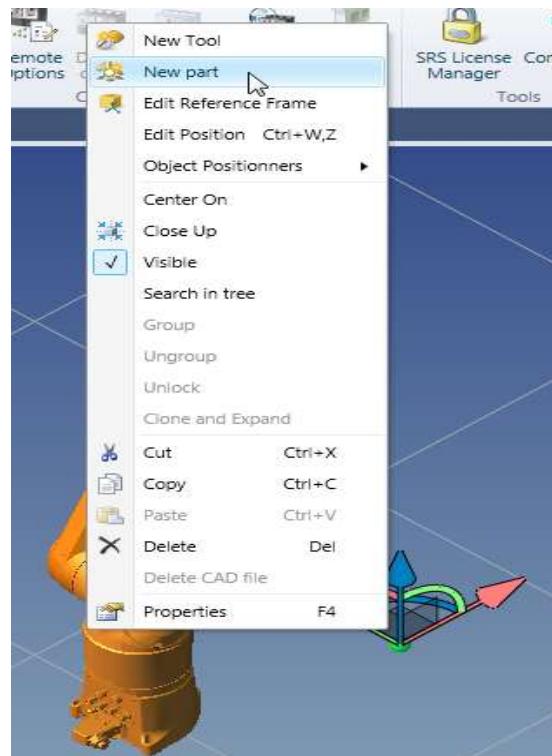
700

Tiempo de transición:

- On → off
- Off → On

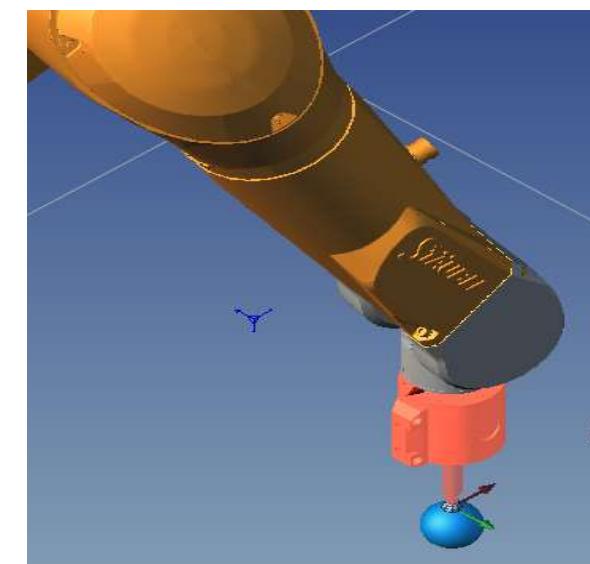
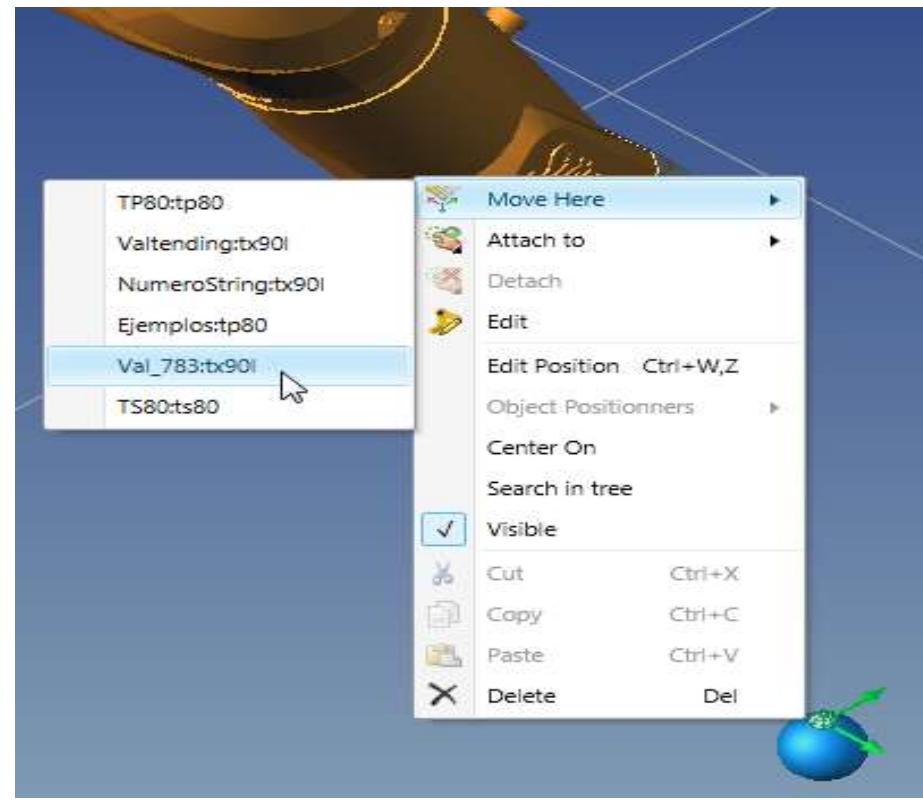
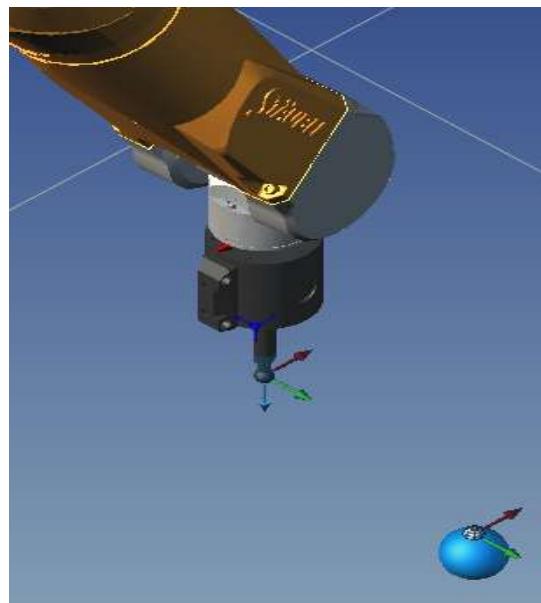
Una vez definida las características de la pinza, toca configurar el objeto que queremos manipular.

Para hacerlo, simplemente pulsamos botón derecho sobre la pieza, y en el desplegable, seleccionamos “New part”

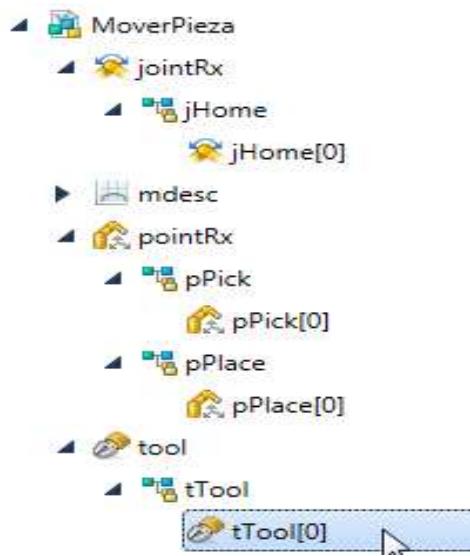


El plano de referencia de la pieza ha de estar con la z hacia adentro (eje de color azul)

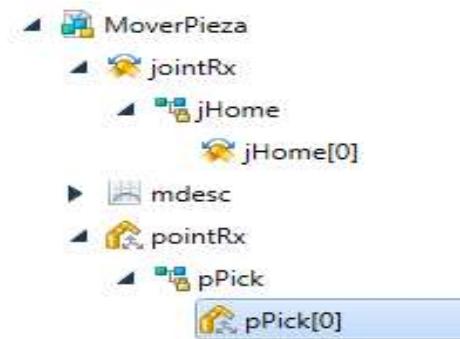
Es posible comprobar que el brazo llegue correctamente al objeto, haciendo clic derecho sobre el cilindro, y en el desplegable seleccionar “Move here”.



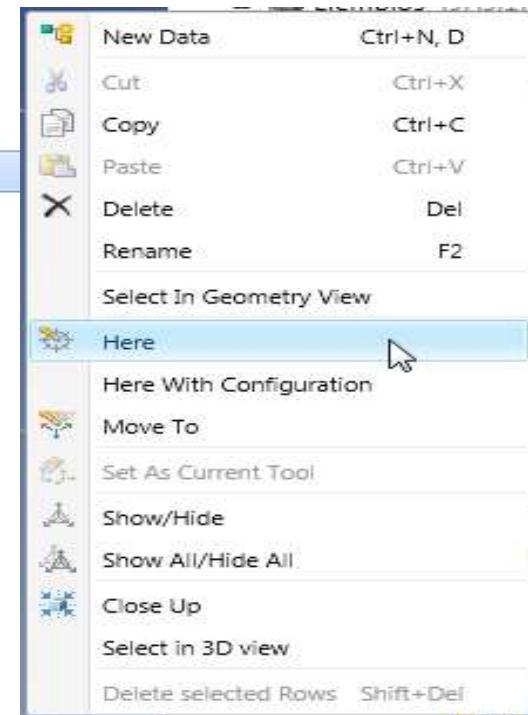
Para poder tomar la posición en el 3d, hay que definir la pinza actual y tomar la posición.



## Definir pinza

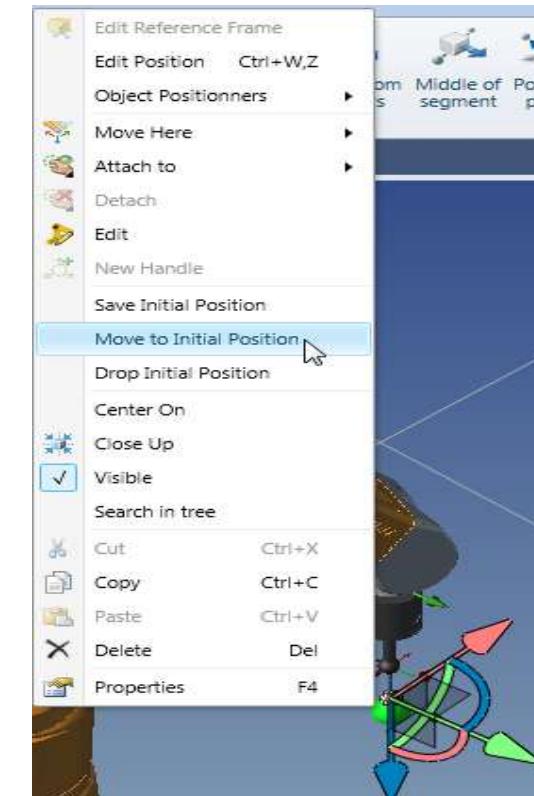
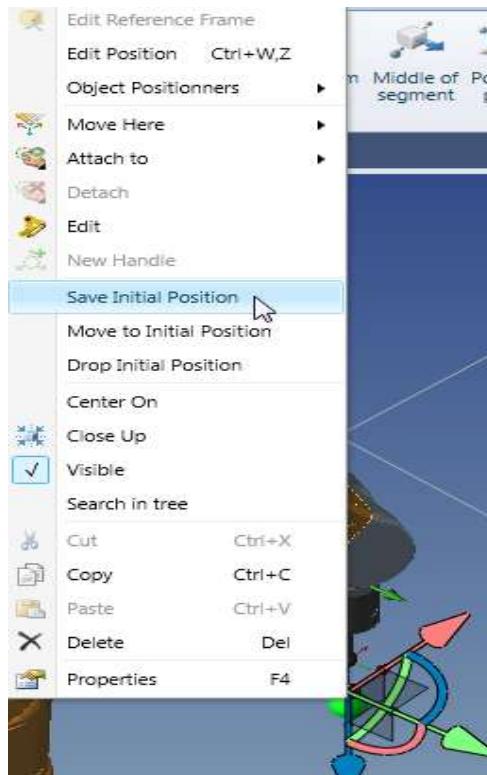


## Tomar posición

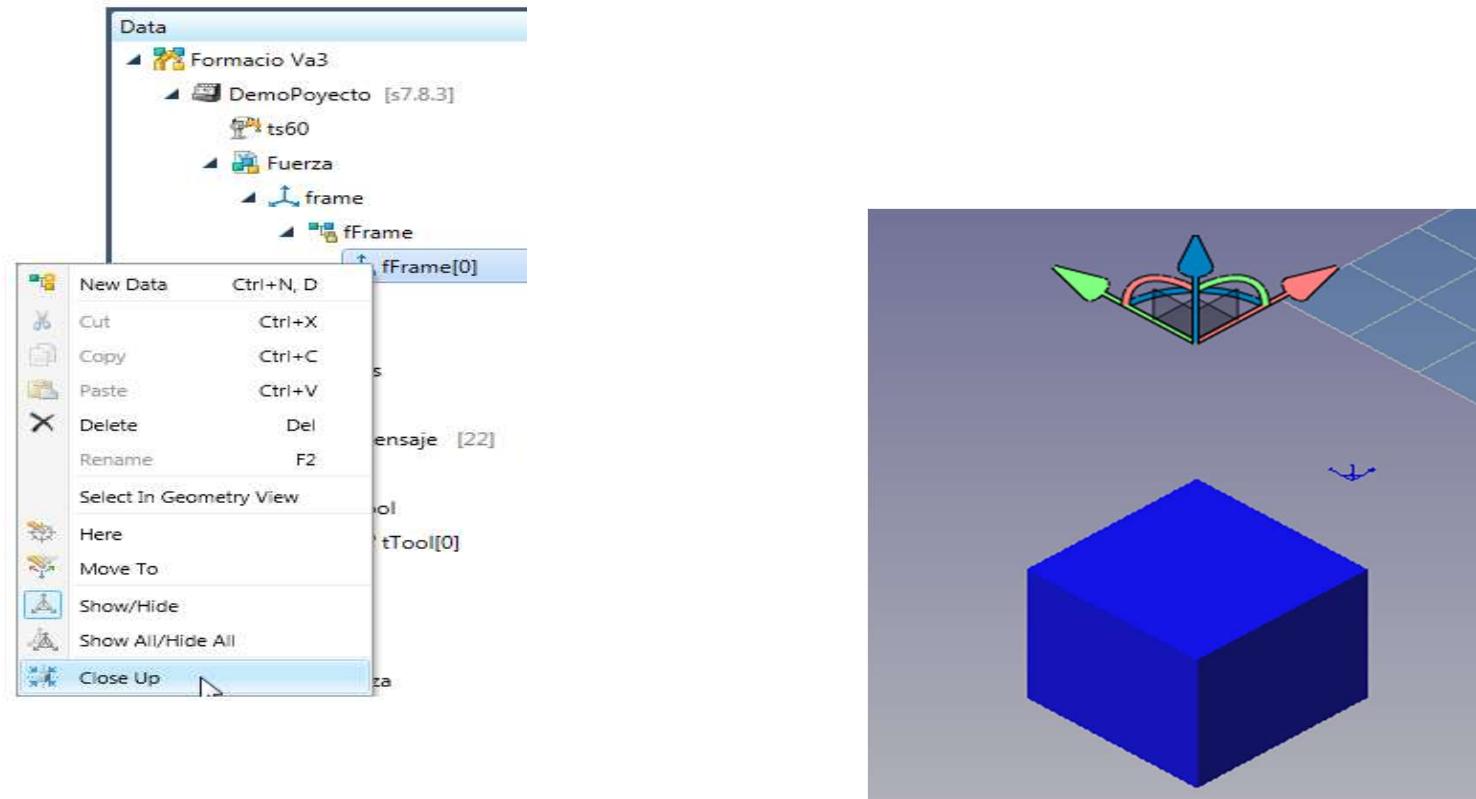


Puede ser necesario, llevar la pieza siempre a una posición antes de ejecutar nuestra aplicación, para ello hay que:

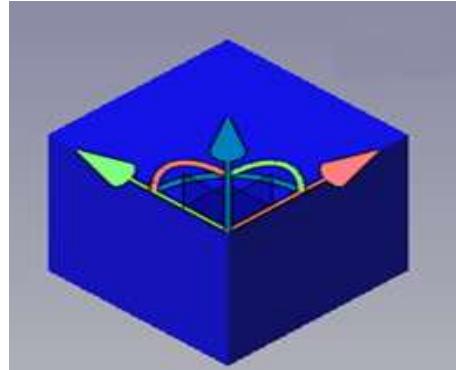
- Guardar la posición de la pieza
- Llevar la pieza



# PLANOS AUXILIARES EN EL 3D

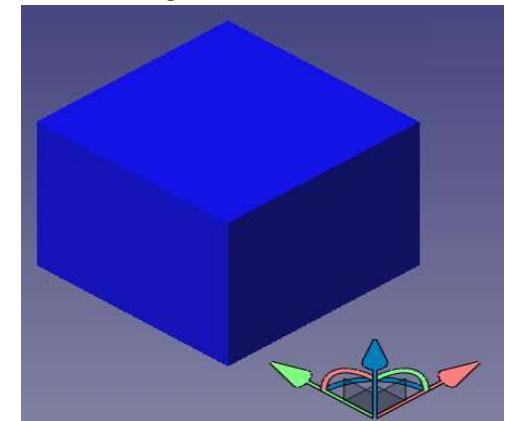


Definiremos /colocaremos el plano en la posición deseada.



Colocamos el plano en una posición pero no hay ninguna vinculación plano objeto, si movemos el objeto, el plano quedara en la misma posición.

Lo que si que hay, es una vinculación plano robot. Si movemos el robot se nos mueve el plano.



# LINKS MANUALES / TUTORIALES SRS

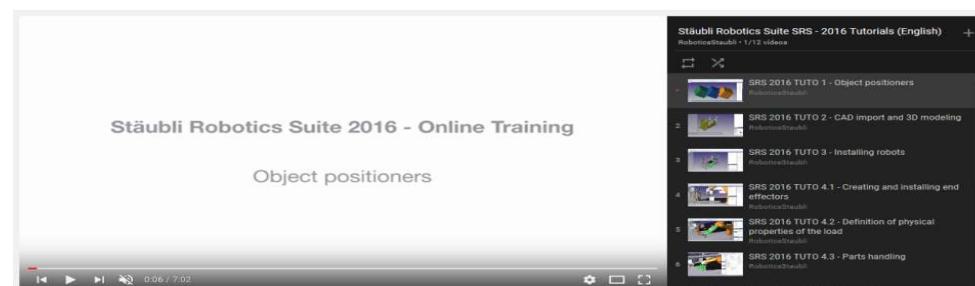
Tutoriales SRS ([www.staubli.com](http://www.staubli.com) → srs tutorial)

<https://www.staubli.com/en/robotics/robot-software/pc-robot-programming-srs/video-srs-2016/>



Tutoriales SRS (Canal Youtube: Robotics Staubli)

[https://www.youtube.com/user/RoboticsStaubli/playlists?sort=dd&shelf\\_id=10&view=50](https://www.youtube.com/user/RoboticsStaubli/playlists?sort=dd&shelf_id=10&view=50)



## FORMACIÓN VAL3



# Anexos

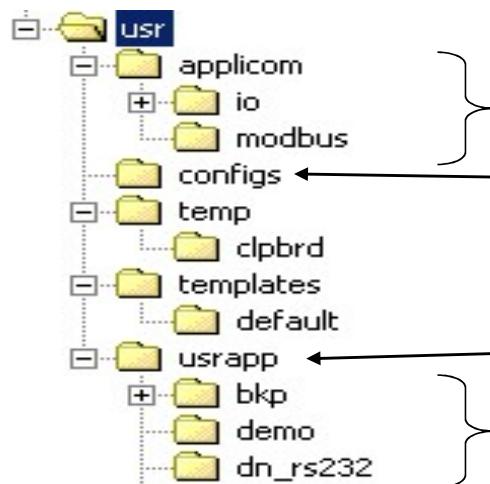


## ESTRUCTURA DE ARCHIVOS



Misma estructura en el disco duro del CS8 que en el emulador.

- Partición /SYS : archivos de sistema NO TOCAR
- Partición /LOG : Registro de eventos ERRORS.LOG
- Partición /USR : archivos del usuario.



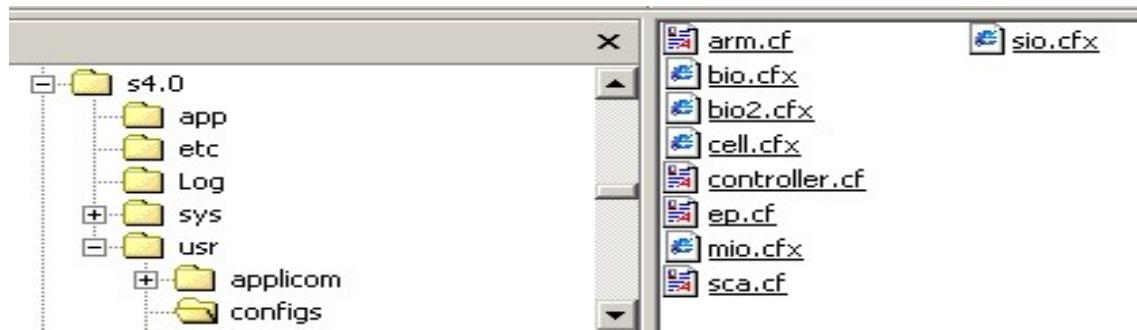
Configuración de Bus de campo/modbus.

Configuraciones brazo, controlador y célula.

Aplicaciones de usuario

1 directorio por aplicación

# FICHEROS DE CONFIGURACIÓN



ARM.CFX : parámetros de calibración del brazo

CONTROLLER.CF : modo de marcha y velocidad al arranque del CS8

EP.CF : Lista de aplicaciones en carga / auto arranque

Network.CFX: IP, Puertos, etc.

CELL.CFX : configuración de la célula.

BIO, MIO,SIO .CFX : definición de las E/S

## ARCHIVOS DE LA APLICACIÓN

Archivos a guardar para ejecutar una aplicación en el emulador:

En **/USR/USRAPP/** :

- Directorio de aplicación.
- Directorio IO (librería de Entradas/Salidas).
- Directorios de librerías utilizadas por la aplicación.

En **/USR/CONFIG/** :

- CELL.CFX (Configuración de la célula).

Copiar estos archivos en el directorio equivalente del emulador

Otros archivos a guardar:

En **/USR/CONFIG/** : ARM.CFX (valores de calibración del brazo)

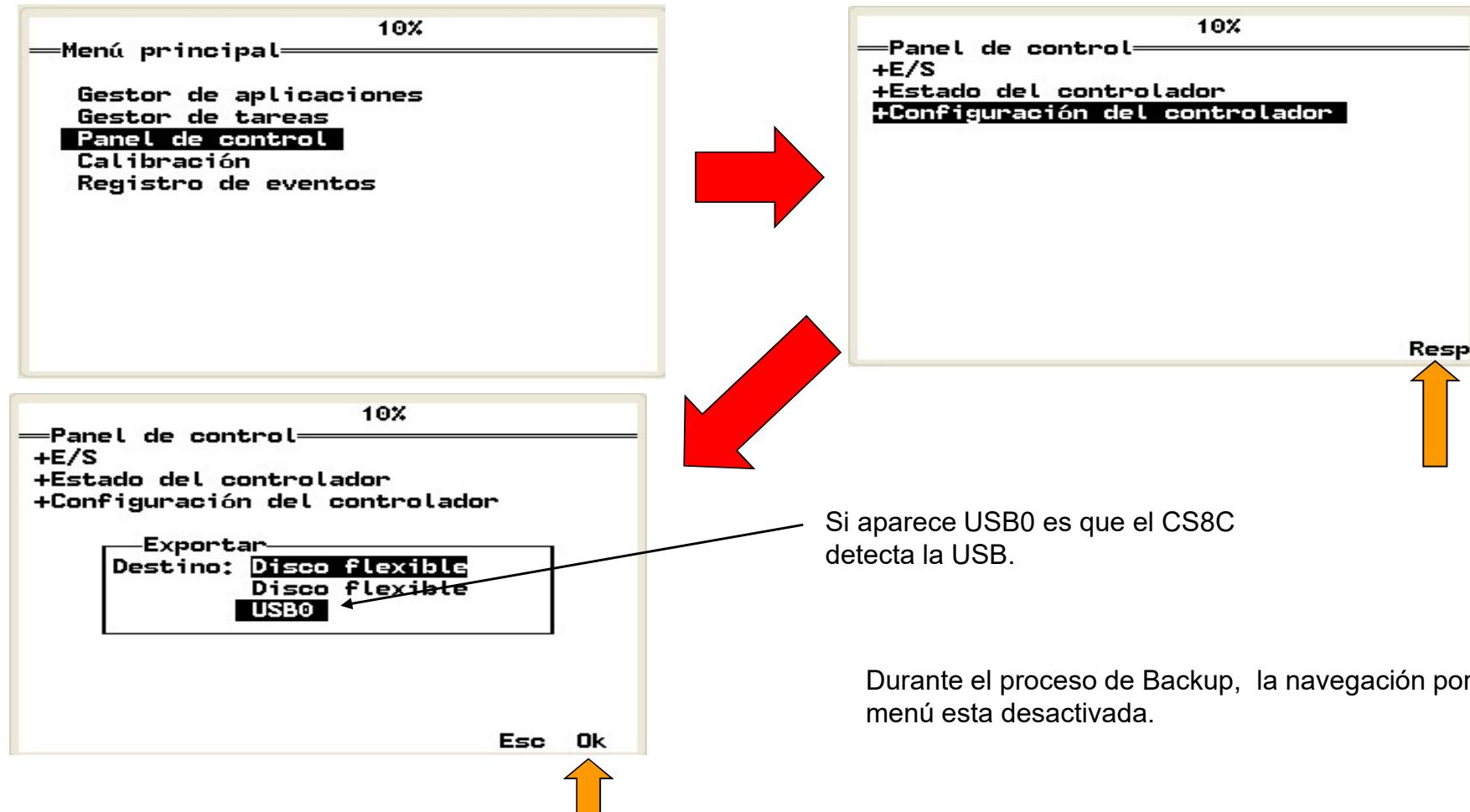
Si hay Bus de Campo:

En **/USR/APPLICOM/IO/** : CONFIGTAG.XML

En **/USR/APPLICOM/MODBUS/** : MODBUS.XML

# BACKUP

Verificar fecha y hora: Tecla menú → panel de control → configuración del controlador → Fecha y hora



Stäubli Española S.A.U.  
C/ Salvador Gil Vernet nº1  
08192 SANT QUIRZE DEL VALLÈS (Barcelona)  
Telf. +34 93 720 54 08  
Fax +34 93 712 42 56  
<http://www.staubli.com>

FAST MOVING TECHNOLOGY

**STÄUBLI**

# Thank you for your attention!

[www.staubli.com](http://www.staubli.com)

