ACL

Lenguaje de Control Avanzado

Version 1.43, F.44

Guia de Referencia

para Controlador-A

Catálogo #100266 Rev.01

ESHED ROBOTEC

INDICE

CAPITULO 3

Los Comandos ACL 3-1

Introducción

El ACL, Lenguaje de Control Avanzado, es un lenguaje y entorno de programación de robótica avanzado, multitare, desarrollado por ESHED ROBOTEC (1982) Ltd.
El ACL esta almacenado en memorias EPROM dentro del Controlador-A y se puede acceder a él con cualquier PC estándar o terminal, por medio de una línea RS232 de comunicaciones.

El ATS, Software para Terminal Avanzado, es el entorno de interface de usuario para el controlador ACL. El ATS viene suministrado en un disquete y se ejecuta en un PC. Este software es un emulador de terminal que permite el acceso al ACL desde un PC.

La siguente figura muestra los componentes del sistema robótico:

Lascaracterísticas del ACL incluyen:

- Ejecución directa de comandos del robot.
- Control de entrada y salida de datos.
- Programación de usuario del robot.
- Ejecución simultánea de programas (total soporte multitarea).
- Ejecución sincronizada de programas.
- Sencilla gestión de ficheros.

Las características del ATS son:

- Configuración rápida y sencilla del controlador
- Definición de elementos periféricos
- Teclas rápidas para la introducción de comandos
- Gestión de Backup
- Gestión de impresión

DELIVERADAMENTE EN BLANCO

CAPITULO 1

ACL : Referencia rápida

Este capítulo es un breve sumario de los modos de los comandos y tipos de datos usados en ACL. Las breves descripciones sirven para comparar y elegir el comando más adecuado para las necesidades concretas de programación y operación. Ver descripciones más detalladas de los comandos en otro capítulo de este manual.

Los comandos están agrupados de acuerdo con los grupos listados abajo.

- Comandos Control de ejes
- Comandos Control de I/O
- Comandos Control Programa
- Comandos Definición y Manipulación de Posiciones
- Comandos Definición y Manipulación de Variables
- Comandos Flujo de Programa
- Comandos de Configuración
- Comandos de Reportajes
- Comandos de Interface de Usuario
- Comandos de Manipulación Programas
- Comandos de Edición
- Comandos de Comunicaciones RS232
- Comandos de Backup y Restore

Para mayor detalle referirse al capítulo 3.

Tipos de Comandos

El ACL tiene dos tipos de comandos:

- DIRECTOS: los comandos son ejecutados tan pronto como son introducidos y pulsado <Enter> desde el teclado.
- EDICION: los comandos son ejecutados durante la ejecución de los programas y rutinas en los cuales están escritos.

Algunos de estos comandos pueden ser ejecutados tanto en modo directo como de edición como se indica su explicación.

Referirse al capítulo 2 para mayor detalle.

Sistemas de Coordenadas

El ACL permite al sistema robótico ser operado y programado en dos sistemas de coordenadas diferentes:

- EJES (joints): pasos de encoder
- XYZ : coordenadas Cartesianas

Referirse al capítulo 2 para mayor detalle.

Tipos de Datos

Variables

El ACL utiliza dos tipos de variables:

- Variables de Usuario:
  - GLOBALES: se pueden usar en todos los programas
  - PRIVADAS: solo pueden usarse en el programa en el cual fueron definidas y utilizadas.
- Variables del Sistema.

  Las variables del sistema contienen los datos del estado de las entradas, salidas, encoders y otros elementos del sistema.

Referirse al capítulo 4 para mayor detalle.

Strings (Comentarios)

Algunos comandos ACL incluyen comentarios o cadenas de texto (strings).
Se reconocen strings de hasta 40 caracteres (incluido espacios).

Referirse al capítulo 2 para mayor detalle

Posiciones

El ACL utiliza seis tipos de posiciones:

- Absoluta Ejes (joint)
- Absoluta XYZ
- Relativa a otra Posición por Ejes
- Relativa a otra Posición por XYZ
- Relativa a la Posición Actual por Ejes
- Relativa a la Posición Actual por XYZ

Referirse al capítulo 2 para mayor detalle.

Parámetros

Los Parámetros ACL definen los valores de constantes físicas que adaptan al controlador a un determinado sistema robótico (tipo de robot, periféricos, velocidades, etc.).

Los parámetros están definidos por su número (del 1 al 320)

Referirse al capítulo 7 para mayor detalle.

Comandos de Control de Ejes

MOVE	SPEED	INT:_ON
MOVED	SHOW SPEED	INT_OFF
MOVEL		TON
MOVELD	EXACT	TOFF
MOVEC	MPROFILE	
MOVECD		
MOVES	CON	HOME
MOVESD	COFF	LSON
		LSOFF
OPEN	SET ANOUT	CLR
CLOSE	SHOW DAC	
JAW		~
		<ALT>+M
CLRBUF		TEST

COMANDO	FORMATO	DESCRIPCION	MODO	NOTAS

MOVE

MOVE <pos>	Mueve el robot a la posición especificada y a la velocidad actual	DIRECTO EDITOR	Leer detalladamente la explicación antes de usar este comando.
MOVE <pos> <tiempo>	Mueve el robot a la posición especificada en el tiempo especificado	DIRECTO EDITOR	
MOVED <pos> <tiempo>	Igual que MOVE excepto que el programa continuará con el próximo comando sólo cuando el robot haya alcanzado la posición	EDITOR	La ejecución de este comando es afectada por el comando EXACT

MOVEC

MOVEC <pos 1> <pos2>	Mueve el robot a la posición 1, con trayectoria circular, pasando por el posición0 2	DIRECTO EDITOR	
MOVECD <pos1> <pos2>	Igual a MOVEC, salvo que el programa sólo pasa al siguiente comando cuando se alcanza la posición	EDITOR	Referirse al comando EXACT

COMANDO	FORMATO	DESCRIPCION	MODO	NOTAS

MOVEL

MOVEL <pos>	Desplaza el robot a la posición por una trayectoria lineal	DIRECTO EDITOR	
MOVEL <pos> <tiempo>	Se desplaza linealmente a la posición en el tiempo especificado	DIRECTO EDITOR	
MOVELD <pos> {<tiempo>}	Como MOVEL salvo que el programa pasa al comando siguiente sólo al alcanzar la posición.	EDITOR	Ver comando EXACT

MOVES

MOVES <pvect> <inicio> <fin>	Desplaza el robot por todas las posiciones del vector, desde la pos. de inicio a la de fin. No hay aceleración/deceleración ni pausa entre posiciones.	DIRECTO EDITOR	
MOVES <pvect> <inicio> <fin> <tiempo>	Como MOVES pero en el tiempo especificado	DIRECTO EDITOR	
MOVESD <pvect> <inicio> <fin> {<tiempo>}	Como MOVES salvo que el programa pasa al comando siguiente sólo al alcanzar la posición.	EDITOR	Ver comando EXACT

CLRBUF

CLRBUF	Vacía el contenido del buffer de movimiento de todos los ejes	DIRECTO EDITOR	
CLRBUFA/B	Vacía el buffer de movimiento del grupo A o del grupo B.	DIRECTO EDITOR	
CLRBUF <eje>	Vacía el buffer de movimiento del eje concreto.	DIRECTO EDITOR	

OPEN

OPEN	Abre pinza hasta el fin de su movimiento, en lazo abierto	DIRECTO EDITOR	Comando normal de apertura de la pinza
OPEN <var>	Abre pinza con fuerza adicional. Aplica el valor de var al DAC. En lazo abierto	DIRECTO EDITOR	

CLOSE

CLOSE	Cierra pinza hasta el fin de su movimiento, en lazo abierto	DIRECTO EDITOR	Comando normal de cierre de la pinza
CLOSE <var>	Cierra pinza con fuerza adicional. Aplica el valor de var al DAC. En lazo abierto	DIRECTO EDITOR	¡ Cuidado, puede dañar la pinza ! 0=< <var> =< 5000

COMANDO	FORMATO	DESCRIPCION	MODO	NOTAS

JAW (no aplicable para ER VII con pinza sin encoder óptico)

JAW <var>	Control de la pinza en lazo cerrado. Ajusta la apertura de pinza al valor porcentual de <var> . Velocidad. máxima.	DIRECTO EDITOR	Puede dañar el motor. 0=< >var> =< 100
JAW <var> <tiempo>	Igual a JAW <var> pero el movimiento se ejecuta en el tiempo especificado.	DIRECTO EDITOR	

SPEED

SPEED <val> **SPEED <var>**	Ajusta el valor de la velocidad para todos los ejes al valor seleccionado.	DIRECTO EDITOR	1=< <val> =< 100 Por defecto 50
SPEED { A/B } <val>	Ajusta velocidad al grupo A o B	DIRECTO EDITOR	
SPEED <val> <eje>	Ajusta velocidad al eje C	DIRECTO EDITOR	

EXACT

EXACT A/B/C	Activa la modalidad de exactitud de posicionamiento en el eje A, B o C.	DIRECTO EDITOR	Sólo afecta a los comandos con terminación "D": MOVED, MOVELD, etc.
EXACT OFFA/B/C	Desactiva la modalidad de exactitud de posicionamiento del eje correspondiente	DIRECTO EDITOR	Igual que el anterior.

MPROFILE

MPROFILE PARABOLE A/B/C	Ajusta el perfil de movimiento del grupo seleccionado a parabólico.	DIRECTO EDITOR	PARABOLE por defecto
MPROFILE TRAPEZE A/B/C	Ajusta el perfil de movimiento del grupo seleccionado a trapezoide.	DIRECTO EDITOR	

INT

INT_ON <eje1>.....<eje4>	Activa el servocontrol integral de los ejes especificados	DIRECTO EDITOR	Por defecto
INT_OFF <eje1>.....<eje4>	Desactiva el servocontrol de los ejes especificados.	DIRECTO EDITOR	

HOME

HOME {<n>}	Busca la posición de referencia de todos los ejes, o del eje especificado	DIRECTO EDITOR	Para hacer referencia desde la botonera de enseñanza teclear: RUN 0
HHOME <n>	Busca la posición de referencia hardware del eje especificado. Busca el tope rígido.	DIRECTO EDITOR	Se utiliza en accesorios sin microinterruptor de HOME: p.e. :base lineal deslizante

COMANDO	FORMATO	DESCRIPCION	MODO	NOTAS

CLR

CLR <n>	Borra el contenido del codificador del eje especificado.	DIRECTO	1=< <n> =< 11
CLR *	Borra el contenido del codificador de todos los ejes.	DIRECTO	

COFF

COFF	Desactiva el servocontrol en todos los ejes.	DIRECTO	
COFFA /B	Desactiva el servocontrol del grupo especificado.	DIRECTO	
COFF <eje>	Desactiva el servocontrol del eje especificado.	DIRECTO	

CON

CON	Activa el servocontrol en todos los ejes.	DIRECTO	
CONA /B	Activa el servocontrol del grupo especificado.	DIRECTO	
CON <eje>	Activa el servocontrol del eje especificado.	DIRECTO	

TON

TON {<n>}	Activa la protección térmica del motor, en todos los ejes o en el eje especificado.	DIRECTO	Por defecto.

TOFF

TOFF {<n>}	Desactiva la protección térmica del motor, en todos los ejes o del eje especificado.	DIRECTO	

SET

SET ANOUT [n] = <DAC>	Ajusta el valor del DAC, del eje especificado, al valor seleccionado. Se usa para mover ejes en lazo abierto.	DIRECTO	-5000 =< DAC =< 5000 Usarlo con cuidado, puede dañar el motor.

Comandos de Control de Programa y Tiempo Real

RUN	**PRIORITY**	**POST**
A (ABORT)		**PEND**
STOP	**SET TIME**	**QPOST**
SUSPEND		**QPEND**
CONTINUE	**DELAY**	
	WAIT	
	TRIGGER	

RUN

RUN <prog>	Ejecuta el programa especificado.	DIRECTO EDITOR	
RUN <prog> <prioridad>	Ejecuta el programa especificado con la prioridad solicitada.	DIRECTO EDITOR	

A (ABORT)

A ó <Ctrl + A>	Aborta todos los programas que se estén ejecutando y detiene el movimiento de ejes.	DIRECTO	
A <prog>	Aborta la ejecución del programa especificado.	DIRECTO	

STOP

STOP	Aborta todos los programas en ejecución.	EDITOR	
STOP <prog>	Aborta la ejecución del programa especificado.	EDITOR	

SUSPEND

SUSPEND <prog>	Detiene la ejecución del programa especificado.	DIRECTO EDITOR	Ver CONTINUE

CONTINUE

CONTINUE <prog>	Continua la ejecución del programa especificado, previamente detenido por el comando SUSPEND.	DIRECTO EDITOR	Ver SUSPEND

PRIORITY

PRIORITY <prog> <var>	Ajusta la prioridad de ejecución del programa especificado al valor de var	EDITOR	1=< <var> =<10 Por defecto 5

COMANDO FORMATO	DESCRIPCION	MODO	NOTAS

SET

SET <var> = TIME	Fija el valor de la variable <var> al valor que tiene en ese momento la variable del sistema TIME.	DIRECTO EDITOR	

DELAY

DELAY <var>	Suspende la ejecución del programa durante el tiempo especificado.	EDITOR	El valor de <var> serán décimas de segundo.

WAIT

WAIT <var1> <cond> <var2>	Suspende la ejecución de un programa hasta que se cumpla la condición especificada.	EDITOR	La condición puede ser: <, >, =, <=, >=, <>

TRIGGER

TRIGGER <prog> BY IN/OUT <n> {<estado>}	Ejecuta un programa condicionado al estado de una entrada/salida	EDITOR	1=< n=< 16 <estado> =0 --->OFF =1 ---->ON

PEND

PEND <var1> FROM <var2>	Suspende la ejecución de un programa hasta que otro programa pone un valor distinto de 0 en <var2>.	EDITOR	Funciona conjuntamente con POST, para sincronizar programas.

POST

POST <val> TO <var2>	Asigna el valor de val a la variable especificada.	EDITOR	Funciona conjuntamente con PEND, para sincronizar programas.

QPEND

QPEND <var1> FROM <vect>	Igual a PEND, pero el valor se toma desde un cola (vector)	EDITOR	Ver QPOST

QPOST

QPOST <val> TO <vect>	Igual a POST, pero el valor se toma desde un cola (vector)	EDITOR	Ver QPEND

Comandos de Manipulación y Definición de Posición

DEFP	**SETPV**	**UNDEF**
DIMP	**SETPVC**	**DELP**
HERE	**SHIFT**	**SET var=PVAL**
HERER	**SHIFTC**	**SET var=PVALC**
		SET var=PSTATUS
TEACH	**SETP**	
TEACHR		

DEFP

DEFP {A/B} <pos> **DEFPC <pos> <eje>**	Crea (define) una posición en el grupo A o B, o en el eje del grupo C.	DIRECTO EDITOR	<pos> es un nombre de hasta 5 caracteres.

DIMP

DIMP {A/B} <vect [n]>	Crea (define) un vector de n posiciones en el grupo especificado.	DIRECTO EDITOR	<vect> es un nombre de hasta 5 caracteres.

HERE

HERE <pos>	Graba en coordenadas de ejes una posición absoluta del robot.	DIRECTO EDITOR	En unidades de codificador. <pos> ha de definirse primero con DEFP o DIMP.

HERER

HERER <pos>	Graba en coordenadas de ejes una posición del robot relativa a la posición actual.	DIRECTO	
HERER <pos2> <pos1>	Graba en coordenadas de ejes la posición <pos2> relativa a la <pos1>	DIRECTO EDITOR	

TEACH

TEACH <pos>	Graba en coordenadas cartesianas una posición absoluta del robot	DIRECTO	Valores X Y Z en décimas de mm; valores de muñeca en décimas de grado.

COMANDO FORMATO	DESCRIPCION	MODO	NOTAS

TEACHR

TEACHR <pos>	Graba en coordenadas. cartesianas una posición relativa a la actual del robot.	DIRECTO	
TEACHR <pos2> <pos1>	Graba en coord. cartesianas una posición <pos2> relativa a <pos1>.	DIRECTO	

SETPV

SETPV <pos>	Graba en coord. de ejes una posición del robot.	DIRECTO	
SETPV <pos> <eje> <val>	Graba en coord. de ejes el valor de un eje en una posición previamente grabada.	DIRECTO EDITOR	1=< <eje> =< 11

SETPVC

SETPVC <pos> <coord> <val>	Graba en coord. cartesianas el valor de un eje en una posición previamente grabada.	DIRECTO EDITOR	

SHIFT

SHIFT <pos> BY <eje> <val>	Cambia el valor de una posición asignando un nuevo valor, en coord. de ejes, al eje especificado.	DIRECTO EDITOR	1=< <eje> =< 11
SHIFTC <pos> BY <val> <coord>	Cambia el valor de una posición asignando un nuevo valor , en coord. cartesianas, al eje especificado.	DIRECTO EDITOR	

SETP

SETP <pos2> = <pos1>	Copia el valor de la posición <pos1> a la posición <pos2>	DIRECTO	

UNDEF

UNDEF <pos>	Inicializa los valores de la posición especificada.	DIRECTO	Se borran los valores de la posición, pero esta sigue definida.

DELP

DELP <pos> **DELP <pvect>**	Borra posiciones y vectores de posiciones de la RAM de usuario.	DIRECTO EDITOR	

SET

SET <var> = PVAL <pos> <eje>	Asigna el valor, en coord. ejes, del eje especificado de la pos. <pos> a la variable <var>	DIRECTO EDITOR	
SET <var> = PVALC <pos> <coord>	Igual al anterior pero en coordenadas cartesianas.	DIRECTO EDITOR	Debe ser una posición del robot.
SET <var> = PSTATUS <pos>	Asigna un valor a <var> de acuerdo con el estado de la posición especificada.	DIRECTO EDITOR	

Comandos de Manipulación y Definición de Variables

DEFINE **DIM** **DELVAR**
GLOBAL **DIMG**
SET (ver página siguiente)

DEFINE

DEFINE <var1>.....<var2>	Crea (define) variables locales. Hasta 8 variables en un comando.	EDITOR	Variable local: sólo reconocida por el programa donde se definió.

GLOBAL

GLOBAL <var1>.....<var8>	Crea (define) variables globales. Hasta 8 variables en un comando.	DIRECTO EDITOR	Variables globales: reconocidas por todos los programas.

DIM

DIM <var[n]>	Crea (define) un vector de n variable locales.	EDITOR	<var> máximo de 5 caracteres.

DIMG

DIMG <var[n]>	Crea (define) un vector de n variables globales.	DIRECTO EDITOR	

DELVAR

DELVAR <var>	Borra la variable especificada de la RAM de usuario.	DIRECTO EDITOR	

Funciones Lógicas y Matemáticas

SET <var1> = <var2>
SET <var1> = NOT <var2>
SET <var1> = COMPLEMENT <var2>
SET <var1> = ABS <var2>
SET <var1> = <var2> <oper> <var3>

SET

SET <var1> = <var2>	Asigna el valor de var2 a var1	DIRECTO EDITOR	
SET <var1> NOT <var2>	Asigna el valor negativo lógico de var2 a var1		
SET <var1> = COMPLEMENT <var2>	Asigna el valor complementario de var2 a var1		
SET <var1> = ABS <var2>	Asigna el valor absoluto de var2 a var1		
SET <var1> = ABS <var2> <oper> <var3>	Asigna a var1 el resultado de la operación entre var2 y var3		<oper> puede ser: +, -, <, *, /, sen, cos, tan, atan, exp, log, mod, or, and.

Comandos de Flujo de Programa

IF	**FOR**	**LABEL**
ANDIF	**ENDFOR**	**GOTO**
ORIF		
ELSE		**GOSUB**
ENDIF		

IF

IF <var1> <cond> <var2>	Comprueba la relación condicional	EDITOR	<cond>: <, >, =, <=, >=, <>

ANDIF

ANDIF <var1> <cond> <var2>	Combina lógicamente una condición con otro comando IF.	EDITOR	

ORIF

ORIF <var1> <cond> <var2>	Combina lógicamente una condición con otro comando IF.	EDITOR	

ELSE

ELSE	Sigue a IF y precede a ENDIF. Ejecuta los comandos si IF es falso.	EDITOR	

ENDIF

ENDIF	Indica el fin del condicionador IF	EDITOR	

FOR

FOR <var> = <val1> TO <val2>	Comando de bucle. Ejecuta el contenido entre FOR y ENDFOR para todos los valores de la variable.	EDITOR	

ENDFOR

ENDFOR	Fin del bucle FOR	EDITOR	

COMANDO	FORMATO	DESCRIPCION	MODO	NOTAS

LABEL

LABEL <n>	Etiqueta para el uso de saltos incondicionales por medio de GOTO	EDITOR	Ver GOTO 0 =< n =< 9999

GOTO

GOTO <n>	Ejecuta un salto incondicional hasta el LABEL <n>	EDITOR	

GOSUB

GOSUB <prog>	Ejecuta el programa <programa> como una subrutina: programa principal suspendido hasta termino de subrutina.	EDITOR	

Comandos de Control I/O

LSON **ISABLE** **SET OUT**
LSOFF **ENABLE** **IF IN**
FORCE
SHOW DIN **TRIGGER (ver p.1-6)**
SHOW DOUT

LSON

LSON	Conecta los interruptores de los ejes a las entradas del controlador. El acceso de entrada normal esta inactivo.	DIRECTO EDITOR	HOME cambia automáticamente de LSON a LSOF

LSOFF

LSOFF	Desconecta los interruptores de los ejes de las entradas del controlador. Modo normal de operación	DIRECTO	Por defecto

DISABLE

DISABLE IN/OUT <n>	Desconecta las I/O físicas de las lógicas	DIRECTO	1 =< n =< 16

ENABLE

ENABLE IN/OUT <n>	Activa la I/O especificada al control normal	DIRECTO	Por defecto. Cancela DISABLE

FORCE

FORCE IN/OUT <n> <estado>	En modo DISABLE, fuerza la I/O especificada al estado 0 ó 1	DIRECTO	

SET

SET OUT[n] = <estado>	Fija el estado de la salida especificada a 0 ó a 1	DIRECTO	<estado> = 0 ----> OFF = 1 ----->ON

IF

IF OUT[n] =<estado>	Ver comando IF	EDITOR	

COMANDO	FORMATO	DESCRIPCION	MODO	NOTAS

Comandos de Manipulación de Parámetros

LET PAR SHOW PAR

LET

LET PAR <n> = <val>	Cambia el valor del parámetro especificado del sistema.	DIRECTO	¡Uso con cuidado !

Comandos de Configuración

CONFIG
INIT CONTROL
INIT EDITOR
INIT PROFILE

CONFIG

CONFIG	Se entra en el menú de configuración del sistema	DIRECTO	Borra todo el contenido de la RAM de usuario.

INIT

INIT CONTROL	Inicializa todos los parámetros del sistema	DIRECTO	Se debe ejecutar al cambiar los parámetros.
INIT EDITOR	Inicializa todos los programas, posiciones y variables de la RAM de usuario.	DIRECTO	Borrará el contenido de la RAM de usuario
INIT PROFILE	Inicializa el perfil de movimiento según el valor del parámetro 76	DIRECTO	Ver apéndice B: parámetro 76

Comandos Informativos

ATTACH ?	**DIR**	**VER**
CONFIG ?	**LIST**	**FREE**
DISABLE ?	**STAT**	
	SHOW	

ATTACH ?

ATTACH ?	Presenta el estado actual de ATTACH.	DIRECTO	Ver comando ATTACH.

CONFIG ?

CONFIG ?	Presenta la configuración del sistema..	DIRECTO	

DISABLE ?

DISABLE ?	Presenta la lista de las I/O desactivadas por el comando DISABLE	DIRECTO	

LIST

LIST {<prog>}	Presenta el listado de todos los programas o de uno especificado	DIRECTO	
LISTP	Lista todas las posiciones definidas	DIRECTO	
LISTPV <pos>	Lista las coordenadas de la posición especificada	DIRECTO	
LISTPV POSITION	Lista las coordenadas de la posición actual del robot.	DIRECTO	
LISTVAR	Lista todas las variables definidas.	DIRECTO	

STAT

STAT	Lista los programas de usuario activos: nombre, prioridad y estado	DIRECTO	

SHOW

SHOW DIN	Presenta el estado de todas las entradas.	DIRECTO	ON = 1 ; OFF = 0
SHOW DOUT	Presenta el estado de todas las salidas.	DIRECTO	
SHOW ENCO	Presenta el estado de todos los encoder.		
SHOW DAC <eje>	Presenta el valor del DAC especificado en milivoltios		

COMANDO	FORMATO	DESCRIPCION	MODO	NOTAS

SHOW PAR <n>	Presenta el valor del parámetro especificado		
SHOW SPEED	Presenta el valor de la velocidad actual.	DIRECTO	

VER

VER	Presenta la versión de la EPROM	DIRECTO	

FREE

FREE	Presenta la cantidad de memoria RAM disponible..	DIRECTO	

Comandos Interfase de Usuario y Pantalla

QUIET	**HELP**	**PRINT**
NOQUIET		**PRINTLN**
ECHO		**READ**
NOECHO		**GET**

QUIET

QUIET	No se presentan en pantalla los comandos directos ejecutados en un programa.	DIRECTO	

NOQUIET

NOQUIET	Se presentan en pantalla los comando directos ejecutados en un programa	DIRECTO	

ECHO

ECHO	Presenta en pantalla todos los caracteres transmitidos al controlador.	DIRECTO	

NOECHO

NOECHO	No presenta el estado de todas las salidas.	DIRECTO	

HELP

HELP	Presenta ayuda en pantalla.	EDITOR	

DO HELP

DO HELP	Presenta ayuda.	DIRECTO	

PRINT

PRINT "string"	Presenta en pantalla la cadena de caracteres (string).	EDITOR	
PRINT <arg1>.....<arg4>	Presenta los valores de los argumentos <arg1>.....<arg4>	EDITOR	
PRINTLN	Igual a PRINT pero inserta un retorno de página.	EDITOR	

COMANDO	FORMATO	DESCRIPCION	MODO	NOTAS

READ

READ "string" <var>	Presenta en la pantalla el string y espera la introducción del valor de <var> por el teclado.	DIRECTO	

GET

GET <var>	Espera que se pulse un carácter por el teclado y lo guarda en la variable <var>.	EDITOR	

Comandos de comunicaciones RS232

SENCOM **PRCOM** **CLRCOM**
GETCOM **PRLNCOM**
READCOM

SENCOM

SENCOM <n> <var>	Transmite un byte cuyos valores especifica la variable por el puerto seleccionado	EDITOR	n = puerto RS232 1 <= n =< 8

GETCOM

GETCOM <n> <var>	Recibe un byte por el puerto especificado y almacena el valor en la variable	EDITOR	1 <= n =< 8

PRCOM

PRCOM <n> <arg1> {<arg2> <arg3>}	Transmite texto (cadena y/o variables) por el puerto especificado	EDITOR	1 <= n =< 8

PRNLCOM

PRNLCOM <n> <arg1> {<arg2> <arg3>}	Igual a PRCOM pero añade retorno de carro	EDITOR	1 <= n =< 8

READCOM

READCOM <n> <var1> {<var2> <var3>}	Recibe caracteres numéricos ASCII, seguidos de retorno de carro, desde el puerto especificado y asigna el valor a <var>	EDITOR	1 <= n =< 8

CLRCOM

CLRCOM <n>	Borra el buffer del puerto especificado	EDITOR	1 <= n =< 8

Funciones de Edición

S ***** **END**
P **@** **<exit>**
L
DEL
<enter>

S

S	Va a la primera línea del programa	EDITOR	
S <n>	Lleva el PROMT a la línea n del programa	EDITOR	

P

P	Va a la línea anterior del programa	EDITOR	

L

L <n1> <n2>	Lista desde la pos n1 a la n2	EDITOR	

DEL

DEL	Borra la linea actual del programa	EDITOR	

<Enter>

<Enter>	Va a la siguiente línea del programa	EDITOR	

*

*** "string"**	Líneas de comentario de usuario	EDITOR	

@

@ <comando DIRECTO>	Permite la ejecución de un comando directo desde el modo editor	EDITOR	

END

END	Fin de programa. El sistema lo graba automáticamente	EDITOR	

COMANDO	FORMATO	DESCRIPCION	MODO	NOTAS

EXIT

EXIT	Para salir de edición. Comprueba la validez del programa	EDITOR	

Comandos de Manipulación de Programas

COPY
RENAME
REMOVE
EDIT

COPY

COPY <prog1> <prog2>	Copia el programa1 a un nuevo programa2	DIRECTO	

RENAME

RENAME <prog1> <prog2>	Cambia el nombre del programa	DIRECTO	

REMOVE

REMOVE <prog>	Borra el programa de la memoria	DIRECTO	

EDIT

EDIT	Activa la modalidad de edición	DIRECTO	

Comandos de Reserva y Restauración Externa

SEND
RECEIVE
APPEND

SEND

SEND	Se envían listas del contenido de la memoria al ordenador	DIRECTO	Ver capítulo 3

RECEIVE

RECEIVE	Carga programas, posiciones y variables desde el disco a la memoria	DIRECTO	
RECEIVE <prog>	Carga el programa prog desde el disco.	DIRECTO	

APPEND

APPEND	Añade programas desde el disco a la memoria	DIRECTO	

Comandos de la Botonera de enseñanza (teach pendant).

ATTACH
ATTACH OFF

ATTACH

ATTACH <pvect>	Coloca un vector de posiciones en la botonera de enseñanza	DIRECTO	
ATTACH OFFA/B/C	Desactiva el vector de posiciones de la botonera	DIRECTO	

COMANDO	FORMATO	DESCRIPCION	MODO	NOTAS

Comando de Localización de Problemas

TEST

TEST

TEST	Ejecuta un programa de test de hardware	DIRECTO	Desde la botonera: RUN 999

Comandos de Sistema Generales

DO
AUTO
~

DO

DO <comando EDIT>	Ejecuta un comando de edición en modo directo	DIRECTO	

AUTO

AUTO	Nombre de fichero reservado: se ejecutará al poner el controlador en marcha		

END

END	Fin de programa.	EDITOR	

Intencionadamente en blanco

CAPITULO 2

Comandos: Tipos y Formatos

En este capítulo se describen los distintos modos de operación y programación del ACL, así como los tipos y formatos de los comandos y datos usados en el leguaje de programación ACL.

Tipos de Comandos

Una vez el ATS ha sido cargado, usted puede comunicarse con el controlador desde el teclado de su PC. Podrá ahora crear o editar sus programas o controlar directamente el robot y ejes periféricos, dependiendo del modo activo de operación.

ACL tiene dos tipos de comandos:

- Comandos DIRECTOS, que son ejecutados tan pronto como son introducidos por el teclado (y pulsado <Enter>).
- Comandos de EDITOR, que son ejecutados durante la ejecución de los programas y rutinas donde se utilizan.

Algunos de estos comandos se pueden ejecutar en ambos modos de operación.

Modo DIRECTO

Cuando el modo directo esta activado, todos los comandos directos introducidos desde el teclado son ejecutados al momento.

Cuando este modo esta activo en la pantalla aparece cursor:

>_

Los comandos de modo DIRECTO pueden ser incluidos en los programas para su ejecución en modo EDITOR, anteponiendo el carácter @. Este carácter le indica al controlador que se trata de un comando DIRECTO que tiene que ser ejecutado en la ejecución del programa.

Usar el comando DELAY para asegurarse de la total ejecución del comando.

Los comandos de modo EDITOR pueden ser ejecutados en modo DIRECTO anteponiendo al comando la orden DO.

Control Manual por Teclado

En modo DIRECTO se puede asumir el control total del robot y ejes periféricos desde el teclado activando el modo Manual. Este modo es útil cuando no se dispone de teclado de enseñanza.

Para activar el modo Manual teclee lo siguiente:

<Alt> + M

Los comandos que se pueden usar en modo Manual son comparables a los que maneja la botonera de enseñanza.

Referirse al comando <Alt>+M en el capítulo 3.

Control con Botonera de Enseñanza

La botonera de enseñanza es un terminal de mano que permite al operador el control directo del robot y de los ejes periféricos. Además de controlar el movimiento de los ejes se utiliza también para el almacenamiento de posiciones, envío de los ejes a determinadas posiciones, activación de programas y otras funciones.

La botonera de enseñanza posibilita el control de los ejes incluso cuando el controlador esta en modo EDITOR.

El funcionamiento de la botonera de enseñanza esta descrito completamente en el manual de Usuario del robot.

Modo EDITOR

El modo EDITOR se utiliza para crear y editar programas ACL.

Cuando el modo EDITOR esta activo, en la pantalla se muestra la línea de programa actual y el cursor, indicando que el comando puede ser insertado. Por ejemplo:

143:?_

El modo EDITOR se activa tecleando el comando EDIT y el nombre del programa a editar. Por ejemplo:

>edit prog1

El sistema responderá:

PROG1 NEW PROGRAM

DO YOU WANT TO CREATE THAT NEW PROGRAM (Y/N)>

Teclee:

Y <Enter>

El sistema responderá:

PROGRAM PROG1

36:?_

Si usted no especifica un nombre de programa el sistema le indicará que introduzca uno.

Si usted indica un nombre de programa ya existente el sistema lo editará indicándole:

WELCOME TO ACL EDITOR, TYPE HELP WHEN IN TROUBLE

PROGRAM PROG1

36:?_

El cursor se posicionará en la primera línea del programa. Si usted teclea <Enter> se posicionará en la línea siguiente y así sucesivamente.

Los nombres utilizados para los programas deben ser combinaciones de hasta cinco caracteres alfanuméricos. Por ejemplo: MILL3, 234, COGE, etc..

Funciones de Edición

ACL proporciona los siguientes comandos para la edición de programas:

S	Va a la primera línea de programa
P	Va a la línea precedente
L n1 n2 seleccionadas	Muestra las líneas de programa
DEL	Borra la línea de programa actual
<Enter>	Va a la próxima línea de programa
EXIT comprueba	Abandona el modo de EDITOR y la validez del programa

Referirse al capítulo 3 para mayor detalle.

ATS utiliza las siguientes teclas para el modo de EDITOR

←	(Backspace) Borra caracteres
→	Restaura caracteres
<Ins>	Inserta caracteres
<Del>	Borra caracteres
<Esc>	Borra el comando tecleado
<Ctrl>+→	Restaura el comando borrado
↑	Repetir el último comando entrado

Sistemas de Coordenadas

ACL permite operar y programar en dos modos diferentes de sistemas de coordenadas: **Ejes** (joints) y **Cartesianas** (XYZ).

Coordenadas Cartesianas (XYZ)

Las coordenadas cartesianas o XYZ son un sistema geométrico usado para especificar la posición de la punta de la herramienta (pinza) definiendo las distancias XYZ , en milímetros, desde el origen del sistema de coordenadas (el centro de la base del robot)

Para la completa definición de la posición también son necesarios los datos de Inclinación (Pitch) y Giro (Roll) de la pinza en grados.

Coordenadas Ejes (Joints)

Las coordenadas de ejes especifican la posición de la punta de la herramienta en pasos de encoder. Cuando los ejes se mueven los encoders ópticos generan una serie de impulsos eléctricos cuya cantidad es proporcional a la cantidad de movimiento realizado. El controlador cuenta los impulsos y los procesa, conociendo en cada momento la posición de todos los ejes del robot. Un movimiento del robot o una posición se puede definir como un específico número de pasos de encoder para cada eje, relativos a la posición de Home o a otra posición.

Cuando se mueve el robot en coordenadas EJES cada eje se mueve individualmente de los demás.

La posición de cualquier periférico conectado al controlador siempre se refiere a este sistema de coordenadas.

Tipos de Datos

El lenguaje ACL utiliza cuatro tipos den datos:

- Variables
- Cadenas de caracteres (string)
- Posiciones
- Parámetros

Variables

Las variables son localizaciones de memoria que almacenan valores enteros, entre -2147483647 a +2147483647 (datos de 32 bits).

ACL usa dos tipos de variables: variables de usuario y variables del sistema.

Variables de Usuario

Las variables definibles por el usuario pueden ser globales o locales.

- **Variables Globales**

  Se pueden usar en todos los programas.

  Con el comando GLOBAL se define una variable global.

  Con el comando DIMG se define un vector de variables globales.

- **Variables Locales**

  Sólo se pueden utilizar dentro del programa en el cual han sido definidas.

  Con el comando DEFINE se define una variable local.

  Con el comando DIM se define un vector de variables locales.

En una sola orden se pueden definir hasta doce variables.

Los nombres usados para definir las variables pueden ser una combinación de hasta cinco caracteres alfanuméricos. El primer carácter debe de ser una letra. Los nombres de los vectores de variables también incluye un numero entre corchetes, que define el número de variables de que consta el vector.

Aquí tenemos unos ejemplos de comandos con variables:

DEFINE X	Define una variable local llamada X.
GLOBAL VAR99	Define una variable global llamada VAR99.
DIM A[20]	Define un vector de variables locales, de 20 unidades.
SET Z= 10	Asigna a la variable Z el valor 10.
SET OUT[3]= Y	El estado de la salida 3 es determinado por el valor de la variable Y.

SET Y = IN[1] por	El valor de la variable Y esta determinado el estado de la entrada 1.
WAIT IN[J]= 1	Condición para la entrada de valor J.

Las variables de usuario tienen un atributo de lectura/escritura. Se pueden ejecutar operaciones sobre las variables y cambiar sus valores con todos los comandos disponibles de ACL.

Variables del Sistema.

Las variables definidas por el sistema contienen valores que indican el estado de las entradas, salidas, encoders y otros elementos de control del sistema. Estas variables del sistema le permitirá realizar todas las aplicaciones que necesiten información en tiempo real del estado del sistema.

El ACL para el Controlador A tiene 9 variables del sistema:

IN[16]	TIME	MFLAG
OUT[16]	LTA	ERROR
ENC[11]	LTB	ANOUT[11]

Los números indican la dimensión de las variables.

Los valores de las variables IN, ENC, TIME, LTA, LTB son actualizadas en cada impulso de reloj; MFLAG es actualizada continuamente durante el movimiento de un eje. Para el usuario, son variables de sólo lectura.

Las variables OUT y ANOUT son variable de lectura/escritura.

referirse al capítulo 4 para mayor detalle.

Listados de variables

El comando LISTVAR mostrará un listado de todas las variables, tanto de usuario como del sistema. Las variables locales tienen entre paréntesis el nombre del archivo en el que están definidas.

El comando SENDVAR produce un listado codificado para la descarga de variables. El formato es el siguiente:

Prefijo: tipo de variable ($1 para local; %v para global).

Número secuencial.

Nombre de la variable.

Nombre del programa (si es variable local)

Valor inicial

Comentarios (cadenas de caracteres)

Una cadena (string) es un argumento de hasta 10 caracteres, que se utilizan en los siguientes comandos de ACL:

PRINT “.....”

PRINTLN “.....”

PRCOM “.....”

PRLNCOM “.....”

@ *Comando*

* *Comentario*

Estas líneas de comandos, pueden consistir de un máximo de 40 caracteres y espacios (es decir, 4 cadenas).

Si una cadena es mayor de 10 caracteres, es automáticamente dividida en subcadenas, cada una de las cuales esta limitada a 10 caracteres. Por ejemplo:

PRINT “HOLA, COMO SE ENCUERNTRA ESTA MAÑANA?”

Este texto tiene 4 cadenas (argumentos):

“HOLA, COMO” “SE ENCUENT” “RA ESTA MA” “ÑANA?”

El máximo número de cadenas almacenables en memoria es un valor definible en la configuración del controlador.

Posiciones

Las posiciones están almacenadas en un área de memoria reservada para estos datos. Los datos de posiciones incluyen un valor entero para cada eje en el rango -32768 y +32768 para definir las coordenadas y una palabra en el rango -32768 y +32768 para indicar el tipo y nombre de la posición.

Tipo de posiciones

El ACL tiene seis tipos de posiciones listadas a continuación. Los comandos usados para grabar cada tipo de posición aparecen entre paréntesis.

- **Absoluta Ejes** (HERE, SETPV, SHIFT): los datos de la posición son las coordenadas en valores de pasos de encoder.
- **Absoluta XYZ** (TEACH, SETPVC, SHIFTC): los valores de la posición están dados en valores de coordenadas Cartesianas.
- **Relativa a otra Posición por Ejes** (HERER): los datos de la posición son la diferencia entre los valores de los encoders de una posición y los valores de encoders de la otra posición. ACL permite anidar hasta 32 posiciones relativas.
- **Relativa a otra Posición por XYZ** (TEACHR): los datos de la posición son la diferencia entre los valores de coordenadas de ambas posiciones. ACL permite anidar hasta 32 posiciones relativas.

- **Relativa a la Posición actual por Ejes** (HERER): los datos de la posición son calculados añadiendo los valores de los encoders de la posición a los valores de encoders de la posición actual.
- **Relativa a la Posición actual por XYZ** (TEACHR): los datos de la posición son calculados añadiendo los valores de las coordenadas Cartesianas de la posición a los valores de la posición actual.

Definición de posiciones

Los comandos DEFP, DEFPB y DEFPC se utilizan para definir posiciones y los comandos DIMPA, DIMPB y DIMPC se usan para definir vectores de posición.

Definir una posición es reservar un espacio en la memoria del controlador y dar un nombre de localización.

Son posibles dos tipos de nombres:

- Nombres numéricos de hasta cinco dígitos (como 3, 22, 101). A estas posiciones se puede acceder directamente desde la botonera de enseñanza.
- Nombres alfanuméricos (como P, POS10, A2). El nombre debe ser una combinación de hasta cinco caracteres y debe comenzar con una letra. A estas posiciones que no son vectores de posición no se puede acceder desde la botonera de enseñanza.

  Los vectores de posición deben tener nombres alfanuméricos y deben comenzar con una letra. Su definición también incluye un índice (número entre corchetes), que define el número de posiciones del vector.

  Desde la botonera de enseñanza se puede acceder a las posiciones del vector cuando este esta "ligado" a la botonera por medio del comando ATTACH. Se llama a cada posición por medio de su número índice.

La memoria de posiciones esta separada para cada grupo de ejes A para los ejes del brazo magnético, B para los accesorios y C para ejes individuales. El máximo número de posiciones para cada grupo es definible en la configuración del controlador.

Por defecto las posiciones se definen para el grupo A.

Ejemplos de definición de posiciones:

DEFP PA A	Define una posición llamada PA para el grupo
DIM PP AA[10] para	Define un vector 10 de posiciones llamado AA el grupo A
DEFPB POSB	Define una posición POSB para el grupo B.
DEFPC PC 8	Define una posición PC para grupo C eje 8.
DIMPC AC[30] 8	Define un vector de 30 posiciones AC para el grupo C eje 8.

Grabación de Posiciones

Para grabar los valores delas posiciones se utilizan los comandos HERE, HERER, TEACH, TEACHR, SETPV y SETPVC.

Grabar una posición es escribir sus valores en la posición reservada de la memoria.

El siguiente recuadro es un sumario de los comandos para grabar posiciones.

Posiciones Grabadas	**para Todos los Grupos de Ejes en Coordenadas Ejes**		**sólo para el Robot (grupo A) en Coordenadas Cartesianas**	
Absoluta; valores actuales	HERE pos	DIRECTO EDIT	no comando	
Absoluta; valores definibles	SETPV pos	DIRECTO	TEACH pos	DIRECTO
Relativa a la posición actual	HERER pos	DIRECTO EDIT	TEACHR pos	DIRECTO
Relativa a otra posición	HERER pos2 pos1	DIRECTO EDIT	TEACHR pos2 pos1	DIRECTO
Absoluta; el usuario cambia los valores de una posición	SETPV pos axis var	DIRECTO EDIT	SETPVC pos coord var	DIRECTO EDIT
Absoluta; el usuario cambia los valores de una posición por el valor de offset	SHIFT pos BY axis var	DIRECTO EDIT	SHIFTC pos BY coord var	DIRECTO EDIT

Aunque las posiciones estén grabadas en coordenadas de Ejes o Cartesianas, los ejes se pueden mover en ambas coordenadas.

Es recomendable definir (aunque no necesariamente grabar) las posiciones que se van a utilizar, antes de editar un programa.

Ejemplos de grabación de posiciones:

HERE 1 Graba la posición actual de los valores de encoders de los ejes del robot, nombre 1.

TEACHR P1 Graba la posición P1 en coordenadas Cartesianas.

Listados de Posiciones

El comando LISTP muestra una lista de todas las posiciones definidas y su grupo correspondiente.

El comando LISTPV muestra los valores de los encoders y de las coordenadas Cartesianas de la posición especificada.

El comando SENDPOINT produce un listado codificado para la descarga de posiciones. Código:

Prefijo ($p)

Número secuencial

Grupo (1/2/3: A, B, C respectivamente)

Nombre de posición

Valores de coordenadas

Número de eje (si es del grupo C)

Tipo de posición

Parámetros

Los parámetros son localizaciones reservadas de memoria que se usan para guardar valores de constantes físicas necesarias para adaptar el controlador a cada particular sistema robótico.

Los parámetros están referidos por su número (del 1 al 320). Por ejemplo:

SHOW PAR 300 Muestra el valor del parámetro 300

LET PAR 294=8000 Cambia el valor del parámetro 294

Ver capítulo 7 para mayor detalle.

Anotaciones utilizadas en este manual

Las siguientes anotaciones se utilizan en las explicaciones de los comandos en este manual:

{ } Las llaves encierran una lista de la cual debe escoger una opción.

[ ] Los corchetes encierran diferentes opciones. ACL requiere la utilización de corchetes en los comandos de vectores de posición, vectores de variables y entradas y salidas.

/ Separan diferentes opciones de una lista. Por ejemplo:

ATTACH OFF {A/B/C}, significa:

ATTACH OFFA ó

ATTACH OFFB ó

ATTACH OFFC

Otras notas

- Los comandos de ACL se pueden introducir tanto en mayúsculas como en minúsculas.
- <Entre> debe pulsarse siempre para ejecutar un comando salvo en algunos casos, como por ejemplo:

<Ctrl>+ A	Abortar
<Alt>+ M	Modo manual
<Ctrl>+ C	Cancela comandos como LIST, SHOW ENCO y SEND

CAPITULO 3

Los Comandos ACL

Los comandos ACL pueden catalogarse, en su mayoría, en dos grupos esenciales: (1) comandos DIRECTOS, para su ejecución inmediata; y (2) indirectos, o comandos EDITOR, para escribir programas y rutinas para su ejecución posterior.

En este capítulo se presentan los comandos ACL por orden alfabético. La modalidad (DIRECTA y/o EDICION) en la que son operativos los comandos se presentan a la izquierda de cada encabezamiento de nombre de comando. Cada registro incluye también el formato del comando, su descripción, ejemplos de uso y observaciones adicionales cuando son necesarias.

Los campos opcionales dentro de los comandos aparecen entre llaves {}.
Las elecciones dentro de los comandos se indican con una barra /. Por ejemplo, ATTACH OFFA/B/C significa:

ATTACH OFFA o,
ATTACH OFFB o,
ATTACH OFFC

Algunos formatos de comando piden que se especifiquen nombres de variables y posiciones, mientras que otros piden valores. Estos campos definidos por el usuario se indican en cursiva entre angulares; por ejemplo:

Programa <prog>	Posición <pos>
Variables <var>	Vector de posición <pvect>
Valor <valor>	Tiempo <tiempo>
Parámetros <par>	Estado <estado>
Eje <eje>	Argumento <arg>

Al definir programas, posiciones y variables, los nombres que se asignen pueden ser una combinación de letras y números hasta cinco caracteres. El primer carácter de los nombres de variables y vectores debe ser una letra. Los nombres de las posiciones de vectores también incluyen un índice (número dentro de corchetes).
Este manual no indica comandos <Enter>, ya que todos los comandos ACL excepto dos, requieren <Enter>. Estas dos excepciones son:

<Ctrl>+A Abortar
Control Manual On/Off

Formato: A {<prog>}

Donde: <prog> es un programa de usuario.

Descripción:

de movimiento	A	Aborta inmediatamente todos los programas usuario en ejecución y detiene el de los ejes.
	A <prog>	Aborta sólo la ejecución del programa especificado

El comando A <prog> aborta el programa de usuario pero no afecta al comando de movimiento ya lanzado por el programa.

Ejemplos:

A	Aborta todos los programas.
A NEW	Aborta el programa NEW.

ANDIF
DIRECTO

ANDIF
DIRECTO

Formato: ANDIF <var1> <cond> <var2>

Donde: <var1> y <var2> son constantes o variables definidas por el usuario, y <cond> puede ser <, >, =, <=, >=, < >.

Descripción: Es un comando tipo IF. ANDIF combina lógicamente con una condición y otros comandos IF.

Ejemplo: condiciones

```
IF A=B                  Cerrará la pinza si se cumplen las
   ANDIF C>2            A=B y C>2 (ambas), de lo contrario abrirá
   CLOSE                la pinza
ELSE
   OPEN
ENDIF
```

Notas: Ver comando IF.

Formato: APPEND

Descripción: Carga el contenido de un fichero de reserva (.CBU) de usuario desde ordenador a la RAM de Usuario del controlador. Similar al comando RECEIVE, pero no borra ni modifica los programas existentes.

El fichero debe estar en el formato generado por un comando SEND.

Se aceptan nuevos programas.

Se aceptan nuevas variables.

Se aceptan nuevas posiciones. A las posiciones existentes, cuyos valores de ubicación no hayan sido fijados todavía, se les asignan los nuevos valores.

Notas: Ver comandos SEND y RECEIVE.
Ver Menú de Gestión de Archivos de ATS (<shift> + F10)

Formato: ATTACH <pvect>

ATTACH OFFA/B/C

ATTACH ?

Descripción: ATTACH <pvect> Asigna el vector de posición especificado a la botonera de enseñanza de acuerdo con el grupo al que pertenece el vector de posición.

Cuando se asigna un vector a la botonera, todas las referencias a ese grupo afectarán al vector añadido.

Sólo se puede asignar a un grupo un vector cada vez. Si se añade otro vector de posición, este cancelará el añadido previamente.

ATTACH OFFA/B/C Desactiva el vector de posición asignado a la botonera, de acuerdo con el grupo especificado.

ATTACH ? Presenta en pantalla el estado actual de ATTACH:
mostrará qué vectores están asignados a la botonera de enseñanza.

Ejemplos: DIMP ALPHA [20]
ATTACH ALPHA

Define un vector de posición denominado ALPHA, que contiene 20 posiciones (grupo A). Asigna el vector ALPHA a la botonera.
Una referencia de la botonera a la posición 15 (p. e.) se referirá ahora a la posición ALPHA [15].

ATTACH OFFA Desactiva el vector del grupo A de la botonera.

Descripción: Es un nombre de fichero reservado para un programa de usuario, que se ejecutará automáticamente cuando se enciende o se resetea el controlador. El fichero AUTO se crea en modalidad de EDICION como cualquier otro programa de usuario.

Ejemplo:

```
  AUTO    program
**********************
HOME            Cuando se encienda el controlador, el robot
RUN OPER        buscará automáticamente su posición HOME
END            (origen) y después ejecutará el programa OPER.
```

Formato: CLOSE {<var>}

Donde: <var> es una constante o variable definida por el usuario.
0≤<var>≤5000

Descripción:

CLOSE	Cierra la pinza hasta el fin de su movimiento
CLOSE <var>	Una constante o variable asignada al DAC de la pinza para mantener la impulsión del motor de la pinza y obtener una fuerza de agarre adicional. La fuerza motriz es proporcional a <var>.

El comando CLOSE desconecta el eje de la pinza del servo circuito de control (control en lazo abierto).

Aviso! ***Hay que usar la opción de variable con extrema precaución para evitar daños en el motor y su engranaje. Hay que usar este comando durante breves periodos y fijar el valor de <var> tan bajo como sea posible.***

Ejemplos:

CLOSE	Cierra la pinza
CLOSE 1000	Fija el valor DAC de la pinza a 1000
CLOSE PRESS	Fija la fuerza activa de la pinza al valor de la variable PRESS.

Notas: Consultar el comando OPEN.

DAC: Convertidor Digital - Analógico

Formato: CLR <n>

Donde: <n> es el número de encoder (codificador óptico), 1≤ n ≤11.

Descripción: Borra (pone a cero) los valores de contaje del encoder n.

Si n = *, se borrar todos los codificadores.

Ejemplos:

CLR 3	Borra el encoder 3.
CLR *	Borra todos los encoders.

Formato: CLRBUF{A/B}

CLRBUF <eje>

Donde: <eje> es un eje del grupo C.

Descripción:

CLRBUF	Vacía el buffer (memoria intermedia) de movimiento de todos los ejes. Se puede utilizar para detener el robot una vez en movimiento y para continuar el programa con otros comandos.
CLRBUFA/B	Vacía el buffer de movimiento del grupo A ó B
CLRBUF <eje>	Vacía el buffer de movimiento de un eje específico.

Ejemplos:

```
CLRBUF

IF IN[3]=1          Si la entrada 3 esta activada;
   STOP MAIN        para el programa MAIN;
   CLRBUFA          borra el buffer del grupo A;
   MOVE HOME        se mueve a la posición HOME
ENDIF

CLRBUF 4            Borra el buffer del eje 4
```

Formato: CLRCOM <n>
CLRCOM 0

Donde: <n> es un puerto RS232 de comunicaciones, $1 \leq n \leq 8$
0 = todos los puertos RS232.

Descripción: CLRCOM <n> borra el buffer del puerto RS232 especificado.

CLRCOM 0 borra los buffers de todos los puertos RS232.

Este comando puede utilizarse para resetear los puertos serie de comunicaciones cuando hay un error, como un XOFF sin el correspondiente XON, interrupciones o interrupción de las comunicaciones RS232.

Ejemplo: CLRCOM 2

Notas: Ver comando SENCOM.

COFF
DIRECTO

COFF
DIRECTO

Formato: COFF{A/B}

COFF {<eje>}

Descripción: Desactiva el servo control de todos los ejes o del eje o grupo especificado.

Ejemplos:

COFF	Control OFF en todos los ejes.
COFFA	Control OFF en los ejes del grupo A.
COFFB	Control OFF en los ejes del grupo B.
COFF 10	Control OFF en el eje 10.

Formato: CON{A/B}

CON {<eje>}

Descripción: Activa el servo control de todos los ejes o del eje o grupo especificado.

Ejemplos:

CON	Control ON en todos los ejes.
CONA	Control ON en los ejes del grupo A.
CONB	Control ON en los ejes del grupo B.
CON 10	Control ON en el eje 10.

Formato: CONFIG {?}

Descripción: CONFIG Activa el fichero de configuración que define la configuración del controlador.

Aviso! Este comando borra todos los programas y posiciones de la RAM de Usuario.

La configuración contiene:

Número de entradas y salidas.
Número de servo ejes.
Tipo de robot.
Tamaño de memoria seleccionado para las líneas del programa de usuario.
Tamaño de memoria seleccionado para las variables de usuario.
Tamaño de memoria seleccionado para las posiciones de usuario.
Tamaño de memoria seleccionado para los comentarios de usuario.

CONFIG? Presenta la configuración actual.

Ejemplos: >config

```
!!!AVISO, SE BORRARAN TODOS LOS PROGRAMAS DE
USUARIO.
ESTA SEGURO??? [YES/NO] > YES
FASE DE CANCELACION DEL TRABAJO..... >
INTRODUCIR NUMERO DE ENTRADAS [16] (0-16) >
INTRODUCIR NUMERO DE SALIDAS[16] (0-16) >
INTRODUCIR NUMERO DE CODIFICADORES[11] (0-11) >
INTRODUCIR NUMERO DE DACS [|1] (0-11) >
TIPO DE ROBOT (0-5-7) [5] (0-7) >
INTRODUCIR NUMERO DE SERVO CIRCUITOS, GRUPO A [5] (5-8)
>
SERVO PINZA INSTALADA EN EL EJE [6] (0-8) >
INTRODUCIR NUMERO DE SERVO CIRCUITOS, GRUPO B [2] (0-2)
>
```

```
INTRODUCIR NUMERO TOTAL DE SERVO CIRCUITOS [8] (8-8) >
INTRODUCIR CANTIDAD RAM DE USUARIO EN KB [32] (16-128)
>
HAY TARJETA DE VISION? [NO] >
INTRODUCIR NUMERO DE PROGRAMAS DE USUARIO [100] >
INTRODUCIR NUMERO DE LINEAS DE PROGRAMA USUARIO
[575] >
INTRODUCIR NUMERO DE PUNTOS DE USUARIO, GRUPO A
[450] >
INTRODUCIR NUMERO DE PUNTOS DE USUARIO, GRUPO C
[150] >
INTRODUCIR NUMERO DE COMENTARIOS DE USUARIO [100] >
Ejecutando configuración, por favor espere 10 segundos
>
>
O.K.
>
```

Durante la configuración se ejecuta automáticamente el INIT EDITOR (EDITOR DE INICIALIZACION, ver comando INIT).

Durante la configuración, el sistema muestra los valores existentes (por defecto) y permite cambiarlos. Si no se quiere cambiar nada, se acepta pulsando <Enter>.

El número mínimo de entradas y salidas es 16.

El número máximo de codificadores y DAC's es 11 (el número máximo de ejes.)

¿ Qué tipos de robots hay disponibles ?

Tipo 0: desconocida,	Ejes independientes, cinemática del brazo sin cálculos XYZ, sin rutina HOME.
Tipo 5: y	Compatible con la cinemática **SCORBOT- ERV ER V+**.
Tipo 7: **VII**.	Compatible con la cinemática **SCORBOT-ER**

El número de ejes para los grupos A y B son definibles por el usuario. Cuando el robot es del Tipo V, el grupo A debe ser el robot e incluir un mínimo de 5 ejes.

Si se utiliza una servo pinza, se debe instalar como el eje que sigue al grupo A; por ejemplo, si un grupo A incluye 5 ejes, la pinza debe ser instalada como el eje 6. Si no hay servo pinza instalada, introducir 0 como

eje de la pinza. Se puede entonces usar el eje 6 para impulsar otros servo dispositivos.

El número total de servo circuitos debe ser mayor que el número de ejes definidos para los grupos A y B y la pinza, y no puede superar el número de codificadores y DAC's (11). Se hace referencia a los ejes restantes como grupo C, que siempre contiene ejes independientes.

En este ejemplo, los ajustes por defecto son, 5 ejes en el grupo A, 2 ejes en el grupo B, y la pinza, totalizando 8 ejes.

La cantidad de KBytes de RAM de Usuario estándar es de 128.

La tarjeta de visión es sólo para uso futuro. No hay que intentar contestar de forma positiva.

El número de programas de usuario, líneas de programa de usuario, variables de usuario, puntos de usuario y comandos depende del tamaño de la memoria y de la distribución de todos estos conceptos. Ver en Apéndice A, el espacio de memoria necesario para cada concepto y calcular según las necesidades.

A continuación, un ejemplo de un informe de configuración .

```
>config?

******* LA CONFIGURACION ACTUAL ES:
ENTRADAS - 16
SALIDAS - 16
CODIFICADORES - 8
SALIDAS ANALOGICAS - 8
TIPO DE ROBOT - 5
SERVO EJE GRUPO A - 5
SERVO PINZA - 6
SERVO EJE GRUPO B - 2
NUMERO TOTAL DE SERVO EJES 8
128 KBYTE DE MEMORIA DE USUARIO INSTALADA PROTEGIDA
POR BATERIA.
PROGRAMAS DE USUARIO - 150
LINEAS DE PROGRAMA DE USUARIO - 3000
VARIABLES DE USUARIO - 600
PUNTOS DE USUARIO, GRUPO A - 2380
PUNTOS DE USUARIO, GRUPO B - 2380
PUNTOS DE USUARIO, GRUPO C - 0
COMENTARIOS DE USUARIO - 550
>
```

Notas: Ver Apéndice A.

CONTINUE CONTINUE
DIRECTO/EDITOR DIRECTO/EDITOR

Formato: CONTINUE <prog>

Donde: <prog> es un programa de usuario.

Descripción: Reanuda la ejecución del programa a partir del punto donde previamente había sido suspendido por el comando SUSPEND.

Ejemplo: CONTINUE ALPHA Reanuda la ejecución del programa ALPHA.

Notas: Ver comando SUSPEND.

COPY
DIRECTO

Formato: COPY <prog1> <prog2>

Donde: <prog1> y <prog2> son programas de usuario.

Descripción: Copia el programa de usuario <prog1> con un nuevo nombre de programa <prog2>. Existen ahora dos copias del mismo programa con nombres diferentes.

El controlador avisar si ya estuviese en uso el <prog2>.

Ejemplo: COPY ALPHA BETA Copia el programa de usuario ALPHA al programa BETA.

Formato: DEFINE <var1> {<var2> ... <var12>}

Donde:<var1>, <var2>, ... <var12> son variables de usuario.

Descripción: Define una variable local. Una variable local sólo es reconocida por el programa específico en el que esta definida.
Para usar una variable tiene que estar previamente definida.

Ejemplos:

DEFINE I	Define una variable local con el nombre I
DEFINE L ALL KEY	Define variables locales con nombres L, ALL y KEY

Notas: Este es un comando de definición. Cuando se escribe, el comando se ejecuta pero no crea una línea de programa, incluso en modalidad de EDICION.

Hay que cerciorarse de que el primer carácter del nombre de la variable sea una letra y que no tenga más de 5 caracteres.

DEFP
DIRECTO

Formato: DEFP{A/B} <pos>

DEFPC <pos> <eje>

Descripción: Crea una posición en el grupo A, en el grupo B, o en un eje del grupo C.

Si no se ha especificado un grupo, se supone el grupo A.

Ejemplos:

DEFPA 9531	Crea una posición en el grupo A llamada 9531.
DEFP 13	Crea una posición en el grupo A llamada 13.
DEFPB DOD	Crea una posición en el grupo B llamada DOD.
DEFPC PP 9	Crea una posición en el eje 9 del grupo C llamada PP.

Notas: Este es un comando de definición. Cuando se escribe, se ejecuta el comando pero no se crea una línea de programa, ni en modalidad EDICION.

DEL
EDITOR

Formato: DEL

Descripción: Comando de edición: borra la línea actual de programa (en edición)

Ejemplo:

```
190 LABEL 1
191 MOVE 10
192 DEL          Borra la línea de programa 191
```

DELAY
EDITOR

Formato: DELAY <var>

Donde: <var> es una constante o variable definida por el usuario.

Descripción: Demora la ejecución de un programa por número <var> de tics. Cada tic es igual a 10 milisegundos.

Ejemplos:

DELAY 100	Demora el programa durante 1 segundo.
SET T=500 DELAY T	Demora el programa durante 5 segundos.

Formato: DELP <pos>

DELP <pvect>

Donde: <pos> es una posición definida por el usuario.
<pvect> es un vector de posiciones definidas por el usuario.

Descripción: Borra las posiciones y los vectores de posición de la tabla de posiciones.

Sólo se puede borrar una posición si ningún programa de la RAM de Usuario la utiliza. El controlador avisar si se intenta borrar una posición en uso.

No se pueden borrar posiciones individuales dentro de vectores.

Ejemplos: DELP A953 Borra una posición o un vector llamado A953

DELP DOD Borra una posición o un vector llamado DOD

Nota: Este es un comando de definición. Cuando se escribe, ejecuta la acción requerida pero no crea línea de programa.

Formato: DELVAR <var>

Donde: <var> es una variable definida por el usuario.

Descripción: Borra la variable <var> de la tabla de variables.

Sólo se puede borrar una variable si ningún programa de la RAM de Usuario la utiliza. El controlador avisar si se intenta borrar una variable en uso.

Ejemplos:

DELVAR I	Borra la variable I.
DELVAR PRESS	Borra la variable PRESS.

Nota: Este es un comando de definición. Cuando se escribe, el comando se ejecuta pero no se crea línea de programa, ni en modalidad de EDICION.

Formato: DIM <var[n]>

Donde: <var[n]> es un vector de n variables locales.

Descripción: Define un vector de variables locales de n elementos. Los elementos creados se llaman var[1], var[2], ... var [n].

Una variable local sólo es reconocida por el programa en el que esta definida.

Ejemplos:	DIM LOCV[20]	Crea un vector llamado LOCV que contiene 20 variables locales, LOCV[1]...VLOC[20]

Nota: Este es un comando de definición. Cuando se escribe, se ejecuta el comando pero no se crea línea de programa, ni en modalidad de EDICION.

Hay que asegurarse de que el primer carácter del nombre del vector sea una letra. Máximo número de caracteres, 5.

Formato: DIMG <var[n]>

Descripción: Define un vector de variables globales de n elementos. Los elementos creados se llaman var[1], var[2] ... var[n].

Una variable global puede ser utilizada por cualquier programa.

Ejemplos:

DIMG	GLOB[8] Crea un vector llamado GLOB que contiene 8 variables globales llamadas GLOB[1]...GLOB[8]

Nota: Este es un comando de definición. Cuando se escribe, se ejecuta el comando pero no se crea una línea de programa, ni en modalidad de EDICION.

Hay que asegurarse de que el primer carácter del nombre del vector sea una letra. Máximo número de caracteres, 5.

DIMP
DIRECTO

Formato: DIMP{A/B} <pvect[n]>

DIMPC <pvect[n]> <eje>

Donde: <pvect[n]> es un vector de n posiciones

Descripción: Define un vector que contiene n posiciones, llamadas pos[1], pos[2] ... pos[n] para el grupo A o B o para un eje del grupo C.

Si no se especifica un grupo, se supone el A.

Ejemplos:

DIMP PICK[30]	Crea un vector para el grupo A que contiene 30 posiciones llamadas PICK[1]...PICK[30].
DIMP BB[10]	Crea un vector para el grupo B que contiene 10 posiciones llamadas BB[1]...BB[10].
DIMPC CONV[25] 11	Crea un vector para el eje 11 que contiene 25 posiciones.

Nota: Este es un comando de definición. Cuando se escribe, se ejecuta el comando, pero no se crea una línea de programa, ni en modalidad de EDICION.

Hay que asegurarse de que el primer carácter del nombre del vector sea una letra. Máximo número de caracteres, 5.

Formato: DIR

Descripción: Presenta el directorio de todos los programas de usuario, con su prioridad. Si se añade la serie "no válido" al nombre del programa, entonces el programa no es válido sintácticamente.

Ejemplos: DIR

nombre :	validez	:identidad :	prioridad
DEMO :		:1	: 5
IO :		: 2	: 5
IOA :		: 3	: 5
TWOIO :		: 4	: 5
INOUT :		: 5	: 5
PICP :		: 6	: 5

Notas: Validez: Consultar el comando EXIT.

Identidad: Número de programa asignado por el controlador. Este número se usa para acceder los programas desde la botonera de a enseñanza..

Prioridad: Ver comandos PRIORITY y RUN.

Formato: DISABLE IN/OUT <n>

DISABLE ?

Donde: IN y OUT son entradas y salidas, y <n> es el índice I/O, 1≤n≤16.

Descripción: DISABLE IN/OUT<n>

Una especificada

Desconecta la I/O física de la I/O lógica. vez desactivada la I/O, su último estado permanece sin cambios.

Sin embargo, el comando FORCE permite posicionar/reposicionar una salida mientras esta desactivada.

DISABLE ? Presenta todas las I/Os desactivadas.

Ejemplos: DISABLE IN 8 Desconecta la Entrada 8.

DISABLE OUT 12 Desconecta la Salida 12.

Nota: Consultar los comandos ENABLE y FORCE.

DO
DIRECTO

Formato: DO <comando EDITOR>

Donde: <comando EDITOR> es cualquier comando de modalidad de EDICION.

Descripción: Ejecuta un comando EDITOR (indirecto) en modalidad DIRECTA.

Ejemplos:

DO CLRCOM 2	Resetea el puerto RS232 número 2
DO PRINTLN	Inserta un retorno de carro

ECHO
DIRECTO

ECHO
DIRECTO

Formato: ECHO

Descripción: Recibe en pantalla el eco de todos los caracteres transmitidos al controlador. Esta es la modalidad por defecto.

El comando NOECHO detiene el eco de los caracteres transmitidos.

Ejemplos: NOECHO

ECHO

EDIT
DIRECTO

Formato: EDIT <prog>

Donde: <prog> es cualquier programa definido por el usuario.

Descripción: Activa la modalidad de EDICION y llama a un programa de usuario denominado <prog>. Si no se encuentra <prog>, crea automáticamente un nuevo programa con ese nombre y espera confirmación del usuario.

Ejemplos: >edit ALPHA

Bienvenido al editor ACL, teclee HELP si tiene problemas.

PROGRAM ALPHA
****

127:?	El editor ya esta listo para recibir líneas de programa.

ELSE
EDITOR

Formato: ELSE

Descripción: El comando ELSE sigue a un comando IF y precede a un ENDIF.

ELSE marca el comienzo de una subrutina de programa que define las acciones a ejecutar cuando sea falso un comando IF.

Ejemplos: 1

```
IF J>2                    Si J>2 y A=B, el controlador activa la Salida
  ANDIF A=B               Si no (ELSE), activa la 5.
  SET OUT[1]=1
ELSE
  SET OUT[5]=1
ENDIF
```

Nota: Consultar comando IF.

Formato: ENABLE IN/OUT <n>

Donde: IN y OUT son entradas y salidas, y <n> es el Índice I/O, 1≤n≤16.

Descripción: Retoma el control normal del sistema de la entrada o salida especificada.

Ejemplos:

ENABLE IN 8	Reactiva la Entrada 8 al software.
ENABLE OUT 12	Reactiva la Salida 12 al software.

Nota: Consultar el comando DISABLE.

Descripción: El sistema escribe automáticamente END como última línea del programa. No es necesario que el usuario introduzca este comando.

Es necesario cuando se editan programas desde un editor de texto.

Formato: ENDFOR

Descripción: Fin del bucle FOR.

Termina la subrutina que ha de ejecutar el comando FOR.

Ejemplos:

```
FOR I=1 TO 16
  SET OUT[I]=1
ENDFOR
```

Este bucle se ejecuta 16 veces, y activa las 16 salidas.

Nota: Consultar el comando FOR.

ENDIF
EDITOR

Formato: ENDIF

Descripción: Fin de la subrutina IF de un programa.

Ejemplos:
```
IF XYZ=1
  ANDIF Z[1]=X
  ORIF B<C
    MOVE POS[1]
  ELSE
     MOVE POS[2]
ENDIF
```

Nota: Consultar el comando IF.

<ENTER> <ENTER>
EDITOR **EDITOR**

Descripción: En modo EDITOR, va a la siguiente línea del programa y muestra su número.

En modo DIRECTO, confirma e introduce el comando.

Formato: EXACT {OFF}A/B/C

Descripción: Fija la modalidad de operación de los comandos MOVED, MOVELD, MOVECD y MOVESD.

En modalidad EXACT, el comando MOVED (o cualquier comando MOVE que termine con una D) instruye al controlador para que compruebe que los ejes han alcanzado sus destinos (dentro de una tolerancia dada).

En modalidad EXACT OFF, los comandos MOVE + D demoran la ejecución del programa hasta que el controlador asume (de acuerdo con el tiempo) que los ejes están en las posiciones correctas.

	EXACT A/B/C	Fija la modalidad EXACT para el movimiento del grupo especificado.
	EXACT OFFA/B/C	Desactiva la modalidad EXACT para el grupo especificado.

Ejemplos: EXACT A

EXACT OFFA

Nota: Los parámetros 261-271 determina la tolerancia exacta.

EXIT EDITOR

Formato: EXIT

Descripción: Sale de la modalidad de EDICION y comprueba la sintaxis y la lógica del programa. Busca errores, como los comandos FOR sin ENDFOR, IF sin ENDIF, y GOTO sin la etiqueta LABEL adecuada.

Si se encuentra un error, aparece un mensaje:

PROGRAMA NO VALIDO

Si el programa esta bien hecho, aparece otro mensaje:

EL PROGRAMA ES VALIDO

Formato: FOR <var> = <valor1> TO <valor2>

Donde: <var> es una variable definida por el usuario y, <valor1> y <valor2> son constantes o variables definidas por el usuario.

Descripción: Ejecuta una subrutina para todos los valores de <var1>, comenzando por el <valor1> y terminando por el <valor2>.

La última línea de la subrutina debe ser el comando ENDFOR.

Ejemplos:

```
FOR L=M TO N
   MOVED POS[L]
ENDFOR
```

```
FOR I=1 TO 16        Activará todas las salidas, primero la 1,
   SET OUT[I]=1      luego la 2 ,.....hasta la 16
ENDFOR
```

FORCE
DIRECTO

Formato: FORCE IN/OUT <n> <estado>

Donde: IN y OUT son entradas y salidas, <n> es el índice I/O, y <estado> es 1 (ON) 0 (OFF).

Descripción: Fuerza la I/O especificada al estado solicitado.

Este comando sólo es operativo para las I/Os que hayan sido desactivadas con el comando DISABLE.

Ejemplos: FORCE IN 5 1 Activa la Entrada 5 en estado ON.

FORCE OUT 11 0 Activa la Salida 11 en estado OFF.

Notas: Ver comando DISABLE.

FREE
DIRECTO

Formato: FREE

Descripción: Presenta una lista del espacio disponible en la RAM de Usuario.

Líneas de programa libres.
Variables libres.
Puntos libres del grupo A.
Puntos libres del grupo B.
Puntos libres del grupo C.
Bytes libres para comentarios.

Ejemplos: FREE

```
484 LINEAS LIBRES
-----------------
63 VARIABLES LIBRES
-----------------
310 PUNTOS DEL GRUPO A LIBRES
-----------------
308 PUNTOS DEL GRUPO B LIBRES
-----------------
11 PUNTOS DEL GRUPO C LIBRES
-----------------
900 BYTES LIBRES para comentarios.
>
```

GET
EDITOR

Formato: GET <var>

Donde: <var> es una variable definida por el usuario.

Descripción: Cuando un programa encuentra un comando GET, hace una pausa y espera que se pulse un carácter desde el teclado del ordenador. Se asignará a la variable <var> el valor ASCII del carácter pulsado.

Ejemplos: PRINTLN "SELECT PROGRAM: P Q R"

```
GET VP              (VP es la variable)
IF VP=80            (80 es el valor ASCII de P)
    RUN P
ENDIF
IF VP=81            (81 es el valor ASCII de Q)
    RUN Q
IF VP=82            (82 es el valor ASCII de R)
    RUN R
ENDIF
```

Formato: GETCOM <n> <var>

Donde: <n> es un puerto RS232 de comunicaciones,1≤n≤8
<var> es una variable definida por el usuario.

Descripción: Complementa al comando SENCOM.
Recibe un byte por el puerto especificado.
El valor del byte se guarda en la variable especificada.

Ejemplo:

```
PROGRAMA WAIT1

LABEL 1
GETCOM 1, RECV
PRINTLN "$ IN :" RECV
GOTO 1
END
```

por Este programa espera caracteres recibidos el puerto serie 1 y muestra su valor por la pantalla.

Formato: GLOBAL <var1> {<var2> ... <var12>}

Donde: <var1> {<var2> ... <var12>} son variables definidas por el usuario.

Descripción: Define una variable global. Una variable global puede usarse en cualquier programa de usuario.

Se pueden definir hasta doce variables en un comando.

Ejemplos: GLOBAL HB Crea una variable global llamada HB.

GLOBAL J BYE ME Crea variables globales llamadas J, BYE y ME.

Notas: Este es un comando de definición. Cuando se escribe, el comando se ejecuta pero no crea línea de programa, ni en modalidad de EDICION.

Hay que asegurarse de que el primer carácter del nombre de la variable sea una letra.

Formato: GOSUB <prog>

Donde: <prog> es un programa definido por el usuario.

Descripción: Transfiere el control del programa desde el programa principal al <prog>, comenzando en la primera línea de <prog>. Cuando se alcanza el comando END del <prog>, la ejecución del programa principal se reanuda con el comando que sigue al comando GOSUB.

Ejemplos:

```
SET Z=10
GOSUB SERVE
MOVE P3
```

Tras ejecutar el comando SET, y antes de ejecutar el comando MOVE, se ejecuta por completo el programa SERVE.

Formato: GOTO <etiqueta n>

Donde: <etiqueta n> es cualquier número, 0≤ n≤ 9999

Descripción: Salta a la línea inmediatamente seguida del comando LABEL <etiqueta n>.

LABEL <etiqueta n> debe estar incluido en el mismo programa que el comando GOTO.

Ejemplos:

```
LABEL 5                 Este programa se ejecuta en un bucle interminable
MOVE POS13              hasta que se aborta manualmente.
SET A=B+C
GOSUB MAT
GOTO 5

LABEL 6                 Este programa se ejecuta 500 veces y después
GOSUB BE                se para.
SET K=K+1
IF K<500
GOTO 6
ENDIF
```

Formato: {DO} HELP

Descripción: En modalidad DIRECTA:
HELP ofrece una pantalla de ayuda on-line para comandos DIRECTOS y DO HELP ofrece una pantalla de ayuda on-line para comandos EDITORES.

En modalidad de EDICION:
HELP ofrece una lista y breve explicación de todos los comandos EDITORES.

Formato: HERE <pos>

Donde: <pos> es una posición definida por el usuario.

Descripción: Graba, en valores ejes (unidades de encoder), la ubicación actual de los ejes y le asigna el nombre <pos>.

<Pos> debe definirse primero usando los comandos DEFP o DIMP

Ejemplos:

HERE POINT	Enseña una posición llamada POINT.

```
DIMP P[20]
HERE P[5]
```

Formato: HERER <pos2> {<pos1>}

Donde:<pos2> y <pos1> son nombres de posiciones definidas por el usuario.

Descripción: HERER <pos2> Fija en valores de ejes (unidades de codificador) la ubicación de <pos2>. <Pos2> será relativa a la posición actual. Se deben introducir los valores de coordenadas según el ejemplo de abajo.

HERER <pos2> <pos1>Registra en valores de codificador la ubicación de <pos2> relativa a <pos1>. <Pos1> debe registrarse antes de <pos2>. Después, cuando se mueva <pos1>, <pos2> se moverá automáticamente con ella.

Este comando también puede utilizarse en modalidad de EDICION.

Ejemplos: >HERER AA[3] AA[3] es relativa a la posición actual del robot por los siguientes valores:

```
1--[1]>0          pulsos de codificador en la base
2--[-1]>5000      pulsos de codificador en el hombro
3--[0]>250 500    pulsos de codificador en el codo
4--[2]>3030       pulsos de codificador en inclinación muñeca
5--[2]>0 0        pulsos de codificador en giro muñeca
DONE
>
```

Los valores entre corchetes son los "por defecto" si están vacíos no hay valor en memoria.

>HERER AA[5] AA[4]

La ubicación de la posición AA[5] se registra como relativa a la posición AA[4].

Formato: HOME {<n>}

HHOME {<n>}

Donde: <n> es el número de un eje, 1≤n≤11.

Descripción: HOME es el programa que está fijo en la EPROM y se activa mediante el comando HOME:

HOME	Lleva todos los ejes del robot a su posición origen buscando un microinterruptor en cada eje. La búsqueda de origen se ejecuta sólo si se introdujo Tipo V ó Tipo VII durante la configuración.
HOME n	Lleva el eje n a su posición origen, usando la búsqueda de microinterruptor. El comando HOME <n> permite crear un programa dedicado para cualquier configuración origen especial.
HHOME n	Lleva el eje n a la posición origen. El programa busca un tope rígido en lugar de un microinterruptor.

Ejemplos:

HOME 7 HOME 8 7, 8, 9 y 10. HOME 9 HOME 10	Busca un microinterruptor origen en los ejes
HHOME 7	Busca un tope rígido origen en la base deslizante lineal conectada al eje 7.

Notas: Hay que asegurarse de que la base móvil esté los suficientemente cerca del tope mecánico cuando se realice la operación home en la LSB.

Notas: La potencia utilizada para impulsar el motor en un comando HHOME es determinada por los parámetros del sistema 201-211. Hay que asegurarse de que este valor DAC no dañe los ejes conectados.

Para ejecutar el programa HOME del robot desde la botonera hay que teclear: RUN 0 [ENTER]

IF
EDITOR

Formato: IF <var1> <cond> <var2>

Donde: <var1> es una variable definida por el usuario,
<var2> es una variable o constante, y
<cond> es una de las siguientes condiciones:<,>,=,<=,>=,< >

Descripción: El comando IF comprueba la relación entre <var1> y <var2>. Si cumple las condiciones especificadas, el resultado es verdadero, y se ejecuta una subrutina o comando. Si no es verdadero, se ejecuta otra subrutina o comando.

Ejemplos:

```
IF C[1]=3          Si C[1]=3,
   MOVE AA[1]      entonces pasa a AA[1].
ELSE               Si C[1] ≠ 3,
   GOSUB TOT       se ejecuta el programa (subrutina) TOT.
ENDIF

IF QY ≤ 10         Si la variable QY es menor o igual que 10
   SET OUT[3]=1    activar salida 3.
ELSE               En caso contrario
   SET OUT[3]=0    desactivar salida 3.
ENDIF
```

INIT
DIRECTO

Formato: INIT CONTROL

Descripción: Inicializa todos los parámetros del sistema de control. Debe ejecutarse después de cualquier cambio en los parámetros de control del sistema.

Formato: INIT EDITOR

Descripción: Inicializa los programas, variables y espacio de la RAM de Usuario.

Aviso! Este comando borra el contenido de la RAM de Usuario

Notas: El comando CONFIG realiza esta operación automáticamente. Por tanto, INIT EDITOR se utiliza con muy poca frecuencia.

Formato: INIT PROFILE

Descripción: Inicializa los perfiles de velocidad (tanto paraboloide como trapezoide) de acuerdo con el valor del parámetro 76.

Notas: Ver explicación del parámetro 76 en el Apéndice B.

INT
DIRECTO/EDITOR

Formato: INT_ON <eje1> {<eje2> ... <eje4>}

INT_OFF <eje1> {<eje2> ... <eje4>}

Donde: <eje1> ... <eje4> son servo ejes del sistema.

Descripción: Activa y desactiva el control de realimentación integral de los servo ejes especificados.

Se pueden especificar hasta cuatro ejes en un comando. La conmutación se hace en el momento de la puesta en marcha.

Ejemplos: INT_OFF 2

INT_ON 5

Notas: Hay que asegurarse de usar un espacio subrayado entre INT y ON/OFF.

Formato: JAW <var> {<tiempo>}

Donde: <var> y <tiempo> son constantes o variables definidas por el usuario.

Descripción: <Var> es el porcentaje requerido de abertura de la pinza.

El comando JAW dimensiona la abertura de la pinza en el tamaño de <var> dentro del tiempo especificado. Si se omite el <tiempo>, el movimiento tiene lugar a máxima velocidad.

Aviso! Hay que asegurarse de seleccionar una <var> adecuada. Un tamaño incorrecto originara un suministro de potencia constante y excesivo al motor, y puede dañarlo.

El comando JAW no es aplicable para la pinza SCORBOT-ER VII sin encoder.

Ejemplos: JAW 40 Abre la pinza hasta el 40 por ciento de su abertura.

JAW 0Cierra la pinza.

Notas: El comando JAW activa el servo control para el eje de la pinza mientras que los comandos OPEN/CLOSE desconectan el eje de la pinza del servo circuito de control.

A menos que sea necesario el comando JAW para una aplicación específica, se recomienda utilizar los comandos OPEN y CLOSE.

Formato: L <línea1> <línea2>

Descripción: Presenta una lista de líneas de programa, desde la primera línea especificada hasta la segunda línea especificada.

Ejemplo:

```
16 : ?L  3  13
*******  Listing 3 to 13  *******
3:  GOSUB MVMAX
4:  IF MVMAX
5:  IF VA>=VB
6:  MOVE O
7:  DELAY 1
8:  SET  TI=LTA - LTB
9:  IF  TI> 100
10: MOVE  00 TI
11: ELSE (9)
12: MOVE  00
13: ENDIF (9)
******* End of listing *****
```

Nota: Cuando se use este comando, los comandos ENFOR, ENDIF y ELSE irán seguidas de un número que corresponderá con su respectivos comandos FOR y IF.

Formato: LABEL <etiqueta n>

Donde: <etiqueta n> es cualquier número, 0_ etiqueta n_ 9999.

Descripción: Marca el comienzo de una subrutina de programa que se ejecuta al leer el comando GOTO.

Ejemplos:

```
LABEL 12
MOVEL 1
MOVE 15 20
OPEN
MOVE JJ
GOTO 12          El programa saltará hasta LABEL 12
```

Formato: LET PAR <n> <valor>

Donde: <n> es el número del parámetro y,
<valor> es una constante o variable definida por el usuario.

Descripción: Fija el parámetro del sistema seleccionado <n> al <valor>.

Ejemplos: LET PAR 21=400 (pone el parámetro 21 a un valor de 400)

LET PAR 299=50 (pone el parámetro 299 a un valor de 50)

Notas:

cambiar se **Aviso! Hay que tomar extremas precauciones al intentar los parámetros del sistema. No cambiar los parámetros mientras están ejecutando programas.**

Una vez fijados los nuevos parámetros del sistema, activarlos ejecutando el comando:

INIT CONTROL

Apéndice Consultar la descripción de los parámetros del sistema en el B.

LIST
DIRECTO

Formato: LIST {<prog>}

Donde: <prog> es un programa definido por el usuario.

Descripción: Presenta todas las líneas del programa de usuario <prog>. Si se omite <prog>, se visualizan todos los programas de usuario.

Ejemplos:

LIST OUT	Presenta todas las líneas del programa OUT.
LIST	Presenta todas las líneas de todos los programas de usuario.

Formato: LISTP

Descripción: Presenta una lista con todas las posiciones definidas.

Ejemplos: LISTP

```
PUNTOS DEFINIDOS
nombre del punto:     grupo:    (eje)
- - - - - - - - - - - - - - - - - - - - - - - - - -
0:                    A
P[10]:                A
PICP[10] :            A
AA :                  A
00 :                  B
B1 :                  B
B2 :                  B
BBA[50]:              B
C1 :                  C:        10
C2 :                  C:        11
C3[100]:              C
```

Formato: LISTPV <pos>

LISTPV POSITION

Descripción: LISTPV <pos> Presenta en mixtas (unidades de codificador) la ubicación de la posición especificada.
Si <pos> es una posición de robot (grupo A), aparecen ambas coordenadas, cartesiana y mixta.

LISTPV POSITION Presenta la ubicación actual del brazo del robot.

Ejemplos: LISTPV P1

Posición P1

1:0	2: 5	3: 13926	4:05:0
X: 5197	Y:0 Z: 9963	P: 594	R:-3

Los valores de eje de P1 aparecen en unidades de codificador. Si P1 es una posición de robot, los valores cartesianos también aparecen en décimas de milímetro (X,Y,Z) y décimas de grado (inclinación y giro).

LISTVAR
DIRECTO

Formato: LISTVAR

Descripción: Presenta variables de usuario y de sistema.

Los corchetes indican variables de vector; por ejemplo, IN[16].

Los paréntesis redondos indican una variable local, usada sólo por el programa especificado; por ejemplo, I(DEMO).

Ejemplos: LIST VAR

```
VARIABLES DEL SISTEMA
**********************
IN[16]
ENC[11]
TIME
LTA
LTB
MFLAG
ERROR
OUT[16]
ANOUT[11]

VARIABLES DE USUARIO
**********************
I (DEMO)
J (DEMO)
I (IO)
I (INOUT)
G1
G2
```

Formato: LSON, LSOFF

Descripción:

LSON	Conecta los interruptores limitadores a los sensores de entrada del controlador. En modalidad LSON, las entradas están inactivas. Todas las referencias a entradas (incluyendo IN[n]) se refieren en realidad a los interruptores limitadores.
LSOFF	Desconecta los limitadores de los sensores de entrada del controlador. En modalidad LSOFF, las entradas están activas. Modo normal.
HOME	Conmuta automáticamente a LSON y de nuevo a LSOFF.

Ejemplos:

```
LSON
SHOW DIN
```

Aparece el estado de todos los limitadores.

```
LSOFF
```

Se ruega leer cuidadosamente esta sección. El comando MOVE puede ser origen de confusiones ya que su ejecución puede no estar sincronizada con el flujo de programa. El comando MOVED suele ser más adecuado para la mayoría de las aplicaciones.

MOVE

Formato: MOVE <pos> {<tiempo>}

Donde: <pos> es un nombre definido por el usuario y<tiempo> es una constante o variable definida por el usuario.

Descripción: MOVE <pos> Desplaza el robot a la posición especificada, de acuerdo con la velocidad actual.

MOVE <pos> <tiempo>
Desplaza el robot a la posición dentro del tiempo especificado. El tiempo se da en unidades de centésimas de segundo.

El comando MOVE deposita un comando de movimiento en el buffer de movimiento. El programa que lanza el comando MOVE no espera que termine el movimiento. La rutina de programa continua independientemente de cuando se ejecuta el comando MOVE.

Si el programa consta de varios comandos MOVE consecutivos, el programa los lee hasta que el buffer de movimiento este lleno, independientemente de su ejecución real. Por tanto, los programas que contengan comandos distintos de MOVE pueden no ejecutarse en la secuencia de programa aparente.

La duración (el tiempo) de la ejecución del comando MOVE es la especificada (o la calculada a partir de la velocidad actual) independientemente de cuanto se acerque el eje a la posición objetivo.

Hay tres métodos para asegurar la secuencialidad en un programa que contenga comandos MOVE:

1. Hay que usar el MOVE con su opción <tiempo>, seguido por un comando DELAY de igual tiempo. Por ejemplo:

MOVE <pos1> <tiempo1>

DELAY <tiempo1>
MOVE <pos2> <tiempo2>
DELAY <tiempo2>

2. Hay que usar comandos secuenciadores como WAIT. Por ejemplo:

MOVE <pos1>
WAIT IN[1]=1

3. Usar el comando MOVED.

MOVED

Formato: MOVED <pos> {<tiempo>}

Donde: <pos> es una posición definida por el usuario, y<tiempo> es una constante o variable definida por el usuario.

Descripción: A diferencia del comando MOVE, el comando MOVED asegura la ejecución secuencial de las operaciones definidas en el programa.

Un comando MOVED se deposita en el buffer de movimiento solo cuando se ha ejecutado por completo el comando MOVED anterior.

Un comando MOVED solo se termina cuando los ejes han llegado a su posición objetivo dentro de la precisión especificada, no importa el tiempo que cueste, e incluso cuando se ha definido <tiempo>.

Para asegurar la ejecución del comando MOVED dentro de un periodo de tiempo definido, se lanza el comando EXACT OFF. Por ejemplo:

EXACT OFFA	
MOVED <pos1>	Los ejes alcanzan <pos1> y <pos2> dentro
MOVED <pos2>	de un periodo de tiempo definido.
EXACT A	
MOVED <pos3>	Los ejes alcanzan <pos3> dentro de la precisión requerida, independientemente del tiempo.

Resumen MOVE, MOVED

1. Comando MOVE: Fácil de programar, no garantiza la secuencialidad ni la precisión.

2. Comando MOVED: Garantiza la secuencialidad y la precisión, pero no el tiempo.

3. Comando MOVED Garantiza la secuencialidad y el tiempo, comando EXACT OFF: pero no la precisión.

Ejemplos:

```
MOVE 3                  El robot se desplaza a la posición 3 y
MOVE AA                 después a la posición AA.
PRINT "COMMAND          La línea "COMMAND GIVEN" probablemente
GIVEN"                  se imprimirá antes de que se complete el
                        movimiento real.

MOVE 3                  Los tres comandos de movimiento se depositan
MOVE AA                 casi simultáneamente en el buffer
MOVE POS[1]             de movimiento. El robot se desplaza a la
SET OUT[1] = 1          posición 3, después hacia AA y después
DELAY 1000              a POS[1]. Al mismo tiempo que se produce el
                        movimiento a la posición 3, la Salida 1 se
                        activa (ON) y el programa se demora durante
                        10 segundos. Este programa acaba unos 10
                        segundos después de su activación,
                        independientemente de la ubicación de los ejes.

MOVE 3 500              El robot se mueve a la posición 3 en 5
DELAY 500               segundos, después AA en8 segundos, después
MOVE AA 800             a POS[1] en 2 segundos. Después se
DELAY 800               activa la Salida 1 y se produce una demora
MOVE POS[1] 200         de 10 segundos. El tiempo total para la
DELAY 200               ejecución de un programa es 25 segundos
SET OUT[1]=1            más una fracción de tiempo despreciable
DELAY 1000              para ejecutar comandos.

MOVED 3                 Todos los comandos se ejecutan
                        secuencialmente.
SET OUT[1]=1            Todas las posiciones se alcanzan dentro
DELAY 1000              de la precisión tolerada.
MOVED AA                Se observaron los ejes haciendo pausas
MOVED POS[1]            en algunas posiciones.
```

```
EXACT OFFA          Se recomienda este formato de programa,
MOVED 3             que asume que la posición 3 y AA están
MOVED AA            a lo largo de una trayectoria y que la
EXACT A              posición POS[1] es donde se recoge un
MOVED POS[1]        objeto. La posición 3 y AA se alcanzan
CLOSE               en un tiempo especificado,
independientemente
SETOUT[1]=1         de la precisión, pero con una demora posible
                    Todos los comandos de este programa se
                    activan en secuencia.
```

Notas: Consultar el comando EXACT.

Formato: MOVEC <pos1> <pos2>

MOVECD <pos1> <pos2> solo EDITOR

Descripción: Desplaza el robot por una trayectoria circular, desde su ubicación actual hasta <pos1>, a través de <pos2>. La ubicación de <pos2> y <pos1> determina el tiempo necesario para completar el movimiento.

El tiempo requerido para completar un movimiento puede definirse mediante un comando SPEED precedente. La velocidad del movimiento circular determina la precisión de la trayectoria circular.

Los demás aspectos de los comandos MOVEC/MOVECD son similares a los de los comandos MOVE/MOVED.

Aviso! Hay que tener cuidado al registrar posiciones para comandos MOVEC. Las limitaciones mecánicas u obstáculos, como el propio robot, pueden hacer que el arco resultante no sea válido.

tales

Ejemplos:

MOVEC 1 2	Se desplaza por una trayectoria circular desde la posición actual a la posición 1 vía posición 2.
SPEED 20	Se desplaza por una trayectoria circular desde
MOVEC 2 1	la posición actual a la posición 2, pasando por la posición 1, a una velocidad 20.

Notas: Este comando es válido solamente para posiciones de robot (grupo A).

MOVEL /MOVELD
DIRECTO/EDITOR

MOVEL / MOVELD
DIRECTO/EDITOR

Formato: MOVEL <pos1> {<tiempo>}

MOVELD <pos1> {<tiempo>} sólo EDITOR

Descripción: Desplaza el robot por una trayectoria lineal, desde su ubicación actual hasta <pos1>.

La velocidad del movimiento lineal determina la precisión de la trayectoria.

Aviso! Hay que tener cuidado al registrar posiciones para comandos MOVEL. Las limitaciones mecánicas u obstáculos, como el propio robot, pueden hacer que el arco resultante no sea válido.

tales

Los demás aspectos de los comandos MOVEL/MOVELD son similares a los de los comandos MOVE/MOVED.

Ejemplos:

MOVELD TR	Se desplaza por una línea recta hasta la posición TR.

Notas: Este comando es válido solamente para posiciones de robot (grupo A).

MOVES / MOVESD
DIRECTO/EDITOR

MOVES /MOVESD
DIRECTO/EDITOR

Formato: MOVES <pvect> <primpos> <ultpos> {<tiempo>}

MOVESD <pvect> <primpos> <ultpos> {<tiempo>} sólo EDITOR.

Donde: <primpos> es la primera posición alcanzada, y <ultpos> es la última posición alcanzada.

Descripción: Desplaza los ejes a través de cualquier número de posiciones de vector consecutivas, desde <primpos> hasta <ultpos>, sin pausas.

Se aplica un perfil de movimiento a todo el movimiento. Aceleración y desaceleración sólo se producen en las posiciones primera y última. Como el tiempo asignado a cada segmento es igual, la velocidad del movimiento es determinada por la distancia entre las posiciones de vector. Cuanto mayor sea la distancia entre dos posiciones de vector consecutivas, más rápido se moverá el robot a través de ese segmento de la trayectoria.

Los demás aspectos de los comandos MOVES/MOVESD son similares a los de los comandos MOVE/MOVED.

Ejemplos:

MOVED PATH[1]	Se desplaza a la posición de comienzo 1.
MOVESD PATH 2 20	Se desplaza en una trayectoria continua
MOVESD PATH 19 1	a través de las posiciones 2 a 20. Después ejecuta la misma trayectoria en dirección opuesta.

Formato: MPROFILE PARABOLE A/B/C

MPROFILE TRAPEZE A/B/C

Descripción: Asigna un perfil de movimiento al grupo especificado.

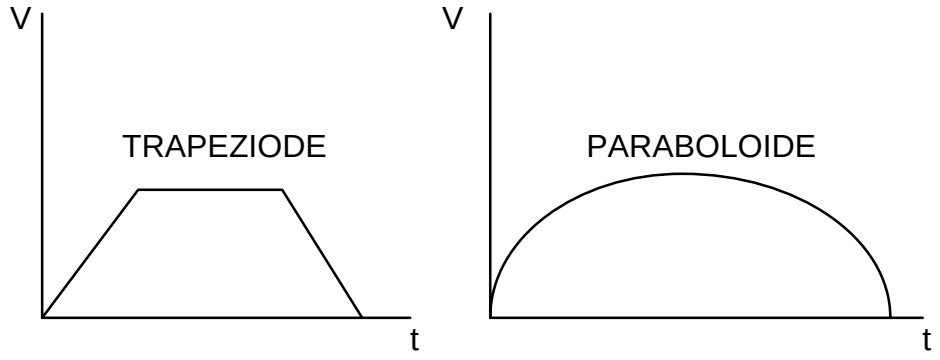
Hay dos perfiles de control de trayectoria disponibles: paraboloide y trapezoide. El comando TRAPEZE hace que los ejes aceleren y desaceleren rápidamente al comienzo y final del movimiento, con una velocidad constante durante la trayectoria. El comando PARABOLE hace que los ejes aceleren lentamente hasta alcanzar la velocidad máxima, decelerando después al mismo ritmo. Ver diagrama:

Se pueden asignar diferentes perfiles de control para diferentes grupos de control. Por ejemplo, perfil paraboloide para el grupo A, perfil trapezoide para el grupo B.

Ejemplos: MPROFILE TRAPEZE A

Cambia el perfil de movimiento del robot a TRAPEZE.

Notas: La velocidad mínima de aceleración y desaceleración viene determinada por el parámetro 76.



NOECHO NOECHO
DIRECTO **DIRECTO**

Formato: NOECHO.

Descripción: Cancela la modalidad ECHO.

no

En modalidad NOECHO, los caracteres transmitidos al controlador aparecen por pantalla.

El comando ECHO cancela la modalidad NOECHO.

NOQUIET NOQUIET
DIRECTO **DIRECTO**

Formato: NOQUIET

Descripción: DIRECTOS
Durante la ejecución de un programa, todos los comandos dentro del programa (es decir, los comandos introducidos con @ en modalidad de EDICION) se visualizan a medida que se ejecutan.

Esta es la modalidad por defecto.

Notas: Consultar el comando QUIET.

Formato: OPEN {<var>}

Donde: <var> es una constante o variable definida por el usuario
0 < var < 5000.

Descripción: OPEN Abre la pinza hasta el final de su movimiento

Si la pinza está conectada al circuito de control, el comando OPEN lo desconecta antes de ejecutar el movimiento de la pinza.

OPEN <var> Una variable o una constante fijada en el DAC de la pinza para mantener la impulsión del motor de la pinza para obtener fuerza de agarre adicional. La fuerza motriz es proporcional a <var>.

El comando OPEN desconecta la pinza del servo circuito de control.

Aviso! Hay que usar la opción de la variable con la máxima precaución para evitar daños en el motor y su engranaje. Hay que usar este comando durante breves periodos, y fijar el valor de <var> lo más bajo posible

Ejemplos:

OPEN	Abre la pinza
OPEN 1000	Fija el valor DAC de la pinza en 1000
OPEN PRESS	Fija el valor DAC de la pinza en el valor de PRESS.

Notas: Ver comando CLOSE.

ORIF
EDITOR

Formato: ORIF <var1> <cond> <var2>

Donde: <var1> son constantes o variables definidas por el usuario, y <cond> puede ser, <,>,=,<=,>=,< >

Descripción: Comando tipo IF. ORIF combina lógicamente con una condición y otros comandos IF.

Ejemplos:

```
IF A=B
   ORIF A=D
   CLOSE
ELSE
    OPEN
ENDIF
```

Notas: Consultar el comando IF.

Formato: P

Descripción: En modalidad de EDICION, va a la línea anterior del programa

Formato: PEND <var1> FROM <var2>

POST <var3> TO <var2>

Donde:<var1> y <var2> deben ser variables definidas por el usuario y <var3> es una variable o una constante.

Descripción: Los comandos PEND y POST son útiles para sincronizar la ejecución simultánea de programas.

Cuando un programa encuentra un comando
PEND <var1> FROM <var2>,
ocurre una de las dos cosas siguientes:

1. Si <var2> tiene un valor de cero, se suspende la ejecución del programa hasta que otro programa en ejecución "envíe" un valor distinto de cero por medio del comando POST <var1> TO <var2.

2. Si <var2> tiene un valor distinto de cero, se asigna ese valor a <var1> y el valor de <var2> se fija en cero.

Ejemplos:

```
PROGRAM DOACT
GLOBAL SIGN              La ejecución del programa DOACT
SET SIGN=0               quedará suspendida hasta que se
PEND VALUE FROM SIGN     active el programa SEND y asigne
RUN ACT                  a SIGN un valor de 1.
END

PROGRAM SEND
POST 1 TO SIGN
END
```

PRCOM
EDITOR

Formato: PRCOM<n> <arg1> {<arg2> <arg3>}

Donde: <n> es un puerto RS232 de comunicaciones,1<=n<=8 y <arg> es o bien una variable o una cadena de caracteres entrecomillada

Descripción:

Envía cadenas de caracteres y variables por el puerto RS232 especificado. El texto que sigue a PRCOM <n> puede contener hasta 30 caracteres y espacios sin incluir comillas. El texto puede contener un total de 3 argumentos y/o variables. Una variable es un argumento, sin importar su longitud. Una cadena de hasta 10 caracteres es un argumento. Cadenas que excedan de 10 y 20 caracteres son tratadas como 2 y 3 argumentos respectivamente.

Ejemplo:

PRCOM 6 "TESTING"	El texto "TEXTING" será transmitido por el puerto serie número 6.

Formato: PRINT <arg1> {<arg2>...<arg4>}

Donde: <arg> es o una variable o una serie dentro de comillas ("").

Descripción: Presenta en pantalla valores de variables y series.

El texto que sigue a PRINT puede contener hasta 40 caracteres y espacios, sin incluir las comillas. El texto puede contener un total de 4 argumentos y/o variables.

Una variable es un argumento, independientemente de su longitud.

Una serie de hasta 10 caracteres es un argumento. Las series que superan 10, 20 y 30 caracteres se tratan, respectivamente, como argumentos dos, tres y cuatro.

Ejemplos: SET NA=5
PRINT "EL ROBOT TIENE" NA "EJES"

En pantalla aparecerá:
"El robot tiene 5 ejes".
El texto "el robot tiene" constituye los argumentos 1 y 2 (contiene 13 caracteres); la variable NA es el argumento 3; el texto "ejes" es el argumento 4.

Formato: PRINTLN <arg> {<arg2> ... <arg4>}

Descripción: Igual que PRINT, pero inserta un retorno del carro antes del texto visualizado.

Ejemplos:
```
SET X=7
SET Y=15
SET J=8
SET K=20
PRINTLN "EL NIVEL DEL TANQUE #" X "ES," Y
PRINT "PULGADAS"
PRINTLN "EL NIVEL DEL TANQUE #" J "ES," K
PRINT "PULGADAS"
```

Aparecerá:
```
EL NIVEL DEL TANQUE #7 ES, 15 PULGADAS
EL NIVEL DEL TANQUE #8 ES, 20 PULGADAS
```

Formato: PRIORITY <prog> <var>

Donde: <prog> es un programa definido por el usuario.<var> es una constante o variable definida por el usuario.

Descripción: Cuando se ejecutan varios programas concurrentemente, los que tengan mayor prioridad se ejecutaran primero.

Los programas de usuario con la misma prioridad comparten tiempo del CPU empleando un algoritmo de distribución igual.

La prioridad por defecto de los programas de usuario es 5.

El comando PRIORITY cambia tanto las prioridades por defecto como las actuales de <prog> en el valor de <var>.

Las prioridades van de 1 a 10, siendo 10 la más alta.
Si el valor de <var> es mayor de 10, se fija la prioridad en 10.
Si es menor de 1, se fija la prioridad en 1.

Ejemplos: PRIORITY PALET 7Asigna al programa PALET una prioridad 7

Notas: Consultar el comando RUN.

Formato: PRLNCOM <n> <arg1> {<arg2> <arg3>}

Donde: <n> es un puerto RS232 de comunicaciones,1<=n<=8
<arg> es o bien una variable o una cadena de caracteres entrecomillada

Descripción: Igual que el comando PRCOM, pero añade un retorno de carro después del envío del texto por el puerto.

Ejemplo:

PRLNCOM 7 "EL VALOR ES" VAL[I]

El texto "EL VALOR ES" y el valor de la variable VAL[I] serán transmitidos por el puerto 7 seguido de un retorno de carro.

Si, por ejemplo, el valor de VAL[I] es 26, la cadena "26" (no el carácter ASCII 26) será enviado por el puerto 7.

Formato: QPOST <var3> TO <var2>

QPEND <var1> FROM <var2>

Donde: <var1> es una variable definida por el usuario.
<var2> es un vector definido por el usuario.
<var3> es una constante o variable definida por el usuario.

Descripción: QPOST Pone en cola los valores a procesar.

QPEND Toma los valores de la cola en el mismo orden en que los introdujo el comando QPOST.

Si se acaba la cola, QPEND suspende la ejecución del programa hasta que un comando QPOST introduce un valor.

El tamaño máximo de la cola es igual a la dimensión del vector <var2> menos 1. Si la cola está llena, QPOST suspende la ejecución del programa hasta que un comando QPEND toma un valor de la cola.

Importante! Se debe inicializar una cola antes de usarla poniendo todos sus elementos a cero.

Ejemplos:

```
PROGRAM INITQ          Define e inicializa la cola.
DIMG QUEUE[10]
DEFINE I
FOR I=1 TO 10
   SET QUEUE[1]=0
ENDFOR
```

```
PROGRAM DOACT          Toma un valor de la cola.
DEFINE VALUE
LABEL 1                El programa ACT se ejecutará cuando
QPEND VALUE FROM QUEUE       el comando SEND deposite valores en
RUN ACT                QUEUE. Si no se ha enviado ningún
GOTO 1                 valor, se suspenderá DOACT hasta la
END                    llegada de un valor.
```

```
PROGRAM SEND           Pone un valor en la cola.
POST 1 TO QUEUE
END
```

QUIET
DIRECTO

Formato: QUIET

Descripción: Cancela la modalidad NOQUIET. En modalidad QUIET, los comandos DIRECTOS dentro del programa (los precedidos por @) no aparecerán durante la ejecución del programa.

Notas: Ver comando NOQUIET.

READ
EDITOR

Formato: READ <arg1> {<arg2> ... <arg4>}

Donde: <arg> es o una variable o una *string* dentro de comillas ("").
string: es una cadena (serie) de caracteres.

Descripción: Cuando READ encuentra un argumento que es una string, el texto aparecerá como una sentencia PRINT.
Cuando READ encuentra un argumento que es una variable, aparecerá en pantalla un "?", indicando que esta esperando la introducción de un valor.
Sólo se considerarán los valores numéricos. La pulsación de <ENTER> sin especificar un valor, introducirá un valor cero.

Cualquier respuesta a "?" es distinta de un valor numérico se interpreta como un comando. Si se introduce un comando, se ejecutará y el comando READ pedirá de nuevo que se introduzca un valor presentando el mensaje:
ENTER value>> (introducir valor)

Ejemplos READ "introducir valor de x" X

Aparecerá por pantalla:
Introducir valor de x?

Si se introduce 254, se asignará a X el valor 254.

Nota: Ver comando PRINT.

Formato: READCOM<n> <var>

Donde: <n> es un puerto RS232 de comunicaciones,1<=n<=8
<var> es una variable.

Descripción: Comando complementario al PRLNCOM
Cuando el comando READCOM es encontrado, espera por el puerto especificado una cadena que contenga códigos ASCII seguidos de un retorno de carro.
Ese valor numérico es asignado a la variable especificada.

Ejemplo:

```
READCOM !, RPART
IF RPART > 9999
PRINTLN "NO PUEDO REALIZAR MAS DE 9999 PIEZAS"
```

Formato: RECEIVE {<prog>}

Descripción: Los comandos RECEIVE cargan datos desde un fichero de reserva de usuario a la RAM de Usuario del controlador a través de la línea RS232C.

Aviso! Este comando borra el contenido de la RAM de Usuario del controlador. El fichero a recibir debe estar en el formato creado por el comando SEND. Tras introducir el comando RECEIVE, el controlador responde:

AVISO, TODOS LOS PROGRAMAS, PUNTOS Y VARIABLES DE USUARIO SE BORRARAN!!!
ASEGURESE DE TENER UNA RESERVA.
Esta determinado a realizar la inicialización?? (SI/NO)
Si la respuesta es sí , el controlador responde lo siguiente:

POR FAVOR ENVIE FICHEROS

(Consultar la documentación del terminal para instrucciones exactas acerca del envío y recepción de ficheros).
El ordenador envía ahora el fichero línea por línea al controlador. Después de cada línea, el ordenador espera que el controlador transmita un signo dos puntos ":". Esto indica que se puede enviar la siguiente línea.

La última línea del fichero a transmitir debe ser,
(END)

A lo que el controlador responde:
FIN DE LA CARGA

RECEIVE <prog> acepta el contenido de un fichero de reserva en el formato generado por los comandos SEND. Acepta sólo un programa e inserta su contenido al <prog> especificado. No afecta al resto de programas y posiciones almacenados en la RAM de Usuario.

Notas: Si se esta usando ATS, se puede reservar y restaurar más fácilmente la AM de Usuario utilizando el menú Backup Manager (Gestor de Reservas).

Consultar el comando SEND.

Formato: REMOVE <prog>

Descripción: Borra un programa de usuario de la RAM de usuario y deja libre el espacio asignado a ese programa.

todo

El sistema pedirá verificación:

Esta seguro? (si /no)

También se borran las variables locales asignadas a este programa.

Ejemplos: REMOVE PALET Borra el programa PALET.

Formato: RENAME <prog1> <prog2>

Descripción: Cambia el nombre del programa de usuario de <prog1> a <prog2>.

Si el nombre <prog2> ya existe, el comando no se ejecuta, y aparece un mensaje de error:

<prog2> ya existe

Una vez cambiado un nombre de programa, ya no existe el nombre original <prog1>.

Ejemplos: RENAME PAL NEW El programa PAL se llama ahora NEW. El programa PAL ya no aparece en la lista del directorio.

Formato: RUN <prog> {<var>}

Donde: <prog> es un nombre de programa definido por el usuario, y,<var> es una constante o variable definida por el usuario.

Descripción: Comienza la ejecución de una tarea a partir de la primera línea del programa <prog>.

Cuando un programa de usuario encuentra un comando RUN <prog>, ambos programas se ejecutan concurrentemente.

<Var> es la prioridad del programa, y va de 1 (la más baja) a 10 (la más alta).

En la modalidad de EDICION, se puede asignar una prioridad introduciendo una variable o bien con el comando PRIORITY, o bien con el comando RUN. Si no se asigna prioridad, la prioridad del programa se fija automáticamente por defecto en 5.

En modalidad DIRECTA, si no se especifica una prioridad al introducir el comando RUN, el programa asume la prioridad por defecto o la última fijada por los comandos PRIORITY o RUN. Si se especifica prioridad, la prioridad por defecto del programa cambia al nuevo valor.

Programas con la misma clasificación de prioridad se ejecutaran concurrentemente.

Ejemplos:

RUN DEMO	El programa DEMO se ejecuta en la prioridad por defecto.
RUN IOS 9	El programa IOS se ejecuta en prioridad 9.

S
EDITOR

Formato: S {<línea n>}

Descripción: Un comando utilizado durante la edición de programas de usuario.

Se Desplaza el editor a la primera línea del programa actualmente editado.

S <línea n>Desplaza el editor a la línea especificada.

Formato: SENCOM<n> <var>

Donde: <n> es un puerto RS232 de comunicaciones,1<=n<=8
<var> es una variable o constante

Descripción: Envía un byte a través del puerto RS232 especificado.
El valor del byte se especifica por una variable o por una constante.

Ejemplo: PROGRAMA ESC

```
DEFINE I
CLRCOM 2
FOR I=1 TO 5
   SENCOM 2, 27
    DELAY 20
ENDFOR
END
```

Este programa borra el buffer del puerto 2 y envía el valor 27 (código <esc> de ASCII) 5 veces por ese puerto.

SEND
DIRECTO

Formato: SEND
SEND <prog>
SENDPROG
SENDVAR
SENDPOINT
SENDPAR

Donde: <prog> es un programa definido por el usuario.

Descripción: Los comandos SEND producen listas en un Formato compatible con los comandos APPEND y RECEIVE. La transmisión de una lista creada por SEND a un ordenador host produce una copia de reserva de los datos.

SEND	Produce una lista de todos los programas de usuario, variables, puntos, parámetros y asignaciones en un **Formato** compatible con RECEIVE y APPEND.
de	SEND sirve para crear una reserva completa la RAM de Usuario.
SEND <prog>	Produce una lista del programa de usuario especificado en un formato compatible con
el	comando RECEIVE <prog>.
SENDPROG	Produce una lista de todos los programas de usuario, cabeceras de programas de usuario, variables de usuario y posiciones de usuario
en APPEND.	un formato compatible con RECEIVE y
	Hace una reserva completa de los programas de usuario con excepción de parámetros del sistema.
SENDVAR	Produce una lista de todas las variables definidas por el usuario en un formato compatible con RECEIVE y APPEND.

SENDPOINT	Produce una lista de todas las posiciones definidas por el usuario, en un formato compatible con RECEIVE y APPEND.
SENDPAR del RECEIVE	Produce una lista de todos los parámetros sistema, en un formato compatible con y APPEND.

Después de introducir el comando SEND, el controlador empieza a transmitir datos. Cuando llega al final de la lista, transmite la línea:

(END)

(Para obtener instrucciones exactas sobre el envío y recepción de ficheros, consultar la documentación del terminal).

Nota: Si se esta usando ATS, se puede reservar y restaurar más fácilmente la RAM de Usuario utilizando el menú Gestor de Reservas.

Formato: SET <var1> = <var2>
SET <var1> = NOT <var2>
ET <var1> = COMPLEMENT <var2>
SET <var1> = ABS <var2>
SET <var1> = <var2> <oper> <var3>
SET <var1> = PVAL <pos> <eje>
SET <var1> = PVALC <pos> <coord>
SET <var1> = PSTATUS <pos>

Donde: <var1> <var2> y <var3> son constantes o variables definidas por el usuario;
<oper> es algo de lo siguiente: + - * / COS SIN TAN ATAN EXP LOG MOD AND OR;
<pos> es una posición definida por el usuario;
<eje> es un número de eje;
<coord> es una coordenada cartesiana: X, Y, Z, P o R.

Descripción:

1. SET <var1> = <var2>

Asigna el valor de <var2> a <var1>

2. SET <var1> = NOT <var2>

Asigna el valor lógico negativo de <var2> a <var1>.
Si <var2> _0, entonces <var1> = 1;
Si <var2> > 0, entonces <var1> = 0.

3. SET <var1> = COMPLEMENT <var2>

Asigna el valor complementario de eje de <var2> a <var1>.
Se toma el valor de <var2>, se invierte individualmente cada bit y el resultado se asigna a <var1>.

4. SET <var1> = ABS <var2> Pone el valor absoluto de <var1> a <var2>

5. SET <var1> = <var2> <oper> <var3>

Donde <oper> es algo de lo siguiente: +,-,*,/,MOD:	Se ejecuta la función matemática de <var2> y <var3> y se almacenan los resultados en <var1>.
Donde <oper> es, AND, OR:	La operación del bit AND/OR se ejecuta en <var2> y <var3> y el resultado se almacena en <var1>.

Aviso! Antes de realizar cualquier función trigonométrica o logarítmica, debe leer cuidadosamente esta descripción**.** El controlador es una máquina de valores enteros y los valores fraccionarios deben convertirse a valores enteros.

Donde <oper> es una función trigonométrica: COS, SIN TAN.	<var2> actúa como un multiplicador y <var3> esta en grados: La función trigonométrica de <var3> se computa y después se multiplica por <var2>. Hay que recordar que el controlador utiliza aritmética integral, así que el multiplicador debe ser lo suficientemente grande para dar la precisión esperada.
Donde <oper> es una de las funciones: ATAN, EXP, LOG	El valor de <var3> se divide por 10000 antes de ejecutar el cómputo. El resultado se multiplica por <var2>. El resultado de la función ATAN se expresa en radianes.

6. SET <var> = PVAL <pos> <eje>

Asigna a <var> el valor de mixtas del eje especificado en la posición especificada.

7. SET <var1> = PVALC <pos> <coord>

Asigna a <var1> una de las coordenadas cartesianas de la posición especificada.

<Pos> debe ser posición de robot (grupo A).

<Coord> puede ser algo de lo siguiente:
X,Y,Z,P,R

X,Y,Z se especifica en décimas de milímetro.

P (de paso) y R (de giro) se especifican en décimas de grado.

8. SET <var1> = PSTATUS <pos>

Asigna a <var1> un valor de acuerdo con el estado de la posición especificada.

Estado de Posición	Código
Ubicación no definida	0
Posición fija	1
Relativa por mixtas a otra posición	2
Relativa por compensación cartesiana a otra posición	3
Relativa por mixtas a la posición actual	12
Relativa por XYZ (compensación cartesiana) a la posición actual	13

Ejemplos:

SET A=B

SET A=NOT B — Si B es 0 entonces A se pone a 1; para cualquier otro valor distinto de B, A se pone a 0.

SET A=COMPLEMENT B

Si B es 0, entonces A se pone a -1.

SET A=ABS B — Si B es -1, entonces A se pone a 1.

SET A=B AND C — Si B o C son 0, entonces A se pone a 0; Si ni B ni C son 0, entonces A se pone a 1.

SET A=1000 COS 60

COS 60 = 0,5; — Multiplicar por 1000; A se pone a 500.

SET ST=PSTAT P1

Si P1 es una posición fija, entonces a ST se

le asignaran valor de 1.

SET y Variables del Sistema

Formato: SET ANOUT [<eje>] = <DAC>

Descripción: Pone el valor de salida analógica del eje especificado en <DAC>. -5000≤DAC≤5000.

Se usa para control de circuito abierto.

Aviso! Usar con cuidado para evitar daños en el motor.

Formato: SET OUT [<n>] = <estado>

Descripción: Fija el estado de la puerta de salida especificada.
<estado> = 1 (ON) _0 (OFF).

Formato: SET <var> = TIME

Descripción: Pone el valor de <var> en TIME.

TIME es una variable de sistema que se expresa en tics de 10 milisegundos a partir del momento de la activación del controlador.

Formato: SETP <pos2> = <pos1>

Donde: <pos2> y <pos1> son posiciones definidas por el usuario.

Descripción: Copia el valor de <pos1> en <pos2>. Ambas posiciones tienen ahora los mismos valores de ubicación.

Este comando es útil para preparar <pos2> de forma que se pueda utilizar el comando SETPV para cambiar un valor de esa posición.

Ejemplos: SETP POINT=PLACE

Formato: SETPV <pos>(sólo DIRECTO)

SETPV <pos> <eje> <valor>

Descripción: SETPV <pos> (similar al comando TEACH).

Fija la ubicación de la posición utilizando valores mixtos (unidades de codificador) para cada eje. Se pide la introducción de valores de todos los ejes en la posición especificada.

SETPV P
1--[100] >
2--[130] >
3--[250] >
4--[120] >
5--[100] >
Done.

El valor entre corchetes es el valor anterior utilizado para fijar una posición. La pulsación de <ENTER> confirma el valor actual.

Si la posición solicitada no es válida, aparece el siguiente mensaje:

BAD POINT COORDINATE
(malas coordenadas del punto)

SETPV <pos> <eje> <val>

Este comando esta pensado sólo para modificar posiciones. Permite cambiar el valor codificador
de un eje en una posición registrada.

Si se quiere que un programa cree una nueva posición, introducir un comando HERE para registrar una posición fija. Pueden fijarse entonces los valores de todos los ejes.

SETPV <pos> <eje> <valor> no avisa de coordenada de punto inválida hasta que prueba y no lo alcanza.

Ejemplos: SETPV PS 3 1000 — Fija el valor del eje 3 en la posición PS en 1000.

```
HERE PQ
SETPV PQ 1 500
SETPV PQ 2 300
SETPV PQ 3 200
SETPV PQ 4 100
SETPV PQ 5 400
```

Formato: SETPVC <pos> <coord> <valor>

Donde: <pos> es una posición de robot (grupo A) definida por el usuario.
<coord> es una coordenada cartesiana: X,Y,Z,P o R
<valor> esta en décimas de milímetro (XYZ) o grado (P,R).

Descripción: Este comando esta pensado sólo para modificar posiciones. Permite cambiar una coordenada cartesiana en una posición previamente registrada con el comando SETPV.

SETPVC es al comando TEACH lo que
SETPVC <pos> <eje> <valor> es para el comando SET <pos>.

SETPVC no avisa de una coordenada de punto inválida hasta que prueba y no lo alcanza.

Ejemplos:
```
SET XV=1000
FOR I=1 TO 20
  SETP V[I]=START
  SETPVC V[I] X XV
  SET XV=XV + 100
 ENDFOR
```

Formato: SHIFT <pos> BY <eje> <valor>

SHIFTC <pos> BY <coord> <valor>

Donde: <pos> es una posición definida por el usuario;
<pos> para SHIFTC debe ser una posición de robot (Grupo A).
<eje> especifica el número de eje;
<coord> especifica la coordenada cartesiana: X,Y,Z,P o R.
<valor> para SHIFTC es en décimas de un mm (X,Y,Z) o un grado (P,R).

Descripción: Cambia la ubicación de posición del eje o coordenada dados en el valor especificado.

SHIFT: coordenadas mixtas.

SHIFTC: coordenadas cartesianas.

Ejemplos: SHIFTC OBJ1 BY Z 1000

Cambia la posición OBJ1 hacia arriba en 100 mm.

Formato: SHOW DIN

Descripción: Se presenta el estado de las 16 entradas individuales en el Formato siguiente:

```
>SHOW DIN
1 -> 16: 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0
O.K.
```

Formato: SHOW DOUT

Descripción: Presenta el valor de todos los codificadores cada 0,5 segundos en el Formato siguiente:

```
enc1   enc2   enc3   enc4   enc5   enc6   enc7   enc8
1000   1000   1000   2371   2371   100    100    1000
```

Los valores actualizados continuarán destellando hasta que se pulse <CTRL+C>.

Formato: SHOW DAC <n>

Donde: <n> = número de eje, 1 _<N> _1.

Descripción: Presenta el valor DAC para el eje especificado en milivoltios.

Formato: SHOW PAR <n>

Descripción: Presenta el valor del parámetro de sistema <n>

Formato: SHOW SPEED

Descripción: Muestra los ajustes de velocidad actuales.

Ejemplos:

```
>SHOW SPEED
GRUPO A VELOCIDAD: 40
GRUPO B VELOCIDAD: 50
O.K.

>SHOW DAC 7
DAC 7=0
O.K.

>SHOW PAR 261
PAR 261=10
O.K.
```

Formato: SPEED{A/B} <valor>
SPEEDC <valor> <eje>

Descripción: Fija el valor de velocidad actual.

La velocidad se especifica en porcentajes. La velocidad máxima es 100; la mínima es 1. La velocidad por defecto es 50.

Se puede fijar la velocidad de todos los ejes utilizando el comando SPEED. Se puede fijar la velocidad de un grupo específico o de un eje del grupo C utilizando los comandos SPEEDA, SPEEDB o SPEEDC.

Los comandos de movimiento que no incluyen un argumento tiempo se ejecutan de acuerdo con el ajuste de velocidad.

Ejemplos: SPEED 60

SPEEDA 30

Nota: Los valores de velocidad mínimo y máximo son determinados por los parámetros 298 y 299.

La modalidad manual utiliza movimientos muy lentos cuyas velocidades son relativas a los ajustes de velocidad actuales. El parámetro 297 determina el ajuste de velocidad manual.

Para visualizar los ajustes de velocidad actuales, teclear:

SHOW SPEED

Formato: STAT

Descripción: Presenta el estado de los programas de usuario activos. La lista incluye prioridad del programa y estado de operación.

Un programa se reportará como "pendiente" si esta esperando que se complete un movimiento.

Ejemplos:

nombre del trabajo	prioridad	estado
BOOM	000005	DELAY
DEMO	000005	PEND
OC	000005	DELAY
MVC	000005	PEND
L	000005	SUSPENDED

STOP
EDITOR

Formato: STOP {<prog>}

Descripción: STOP Aborta todos los programas en ejecución y detiene el movimiento de todos los ejes.

STOP <prog>Aborta sólo la ejecución del programa específico.

Ejemplos: STOP

STOP DEMO

Formato: SUSPEND <prog>

Descripción: Suspende la ejecución del programa especificado.

y El programa completa el comando que esta ejecutando actualmente, después va a la suspensión.

Un programa suspendido puede continuarse desde el punto de suspensión utilizando el comando CONTINUE.

Ejemplos: SUSPEND DEMO

Formato: TEACH <pos>

Donde: <pos> es una posición de robot definida por el usuario (grupo A).

Descripción: Fija en coordenadas cartesianas la ubicación actual de los ejes en <pos>. El controlador pide todos los valores.

Los valores X,Y,Z son en décimas de milímetro, mientras que los valores de paso (P) y giro (R) son en décimas de grado.

El valor entre corchetes es el valor previamente utilizado para instruir (teach) una posición.
La pulsación de <Enter> confirma el valor actual.

```
TEACH PP
X -- [2000] >
Y -- [0] >
Z -- [3400] >
P -- [-900] >
R -- [0] >
```

Si la posición introducida no es válida, aparece el siguiente mensaje:

BAD POINT COORDINATE

Nota: Ver el comando SETPV.

Formato: TEACHR <pos2> {<pos1>}

Descripción: Registra en coordenadas cartesianas la ubicación de <pos2> relativa a otra posición.

Si <pos1> no se especifica, <pos2> es relativa a la posición actual.
Si se especifica <pos1>, <pos2> es relativa a <pos1>.
Cuando se desplaza <pos1>, <pos2> se desplaza correspondientemente.
Los valores X,Y,Z son en décimas de milímetro, mientras que paso (P) y giro (R) son en décimas de grado.

Ejemplos: TEACHR OVER TABLE

X [0] >	La posición OVER estará 200 mm verticalmente
Y [0] >	por encima de la posición TABLE y será relativa
Z [0] > 2000	a esta.
P [0] >	
R [0] >	

Formato: TEST

Descripción: Ejecuta un programa de pruebas de hardware interactivo.

TEST comprueba el servo circuito de control desplazando todos los ejes definidos. Si falla un eje, aparece un mensaje:

TEST FAILURE AXIS <n> (Fallo del eje n)

Este mensaje también se da cuando los ejes definidos no existen realmente, o cuando A y B no esta conectados.

Tras terminar el servo test, TEST activa todas las salidas y después las desactiva.

Después TENTER entra en modalidad interactiva. Rastrea las entradas y para cada entrada activada (ON), TEST fija la salida correspondiente.

El comando A (Abortar) es la única forma de detener el programa TEST.

TON / TOFF
DIRECTO

Formato: TON {<n>}
TOFF {<n>}

Descripción: TON Activa la protección térmica del motor en todos los ejes o en un eje específico.

TOFF Desactiva la protección térmica del motor en todos los ejes o en un eje específico.

El sistema pide la confirmación de COFF.

Nota: **Aviso!** Cuidado en modalidad TOFF. Los motores no están protegidos por ninguna protección de software.

Formato: TRIGGER <prog> BY IN/OUT <n> {<estado>}

Donde: <prog> es un nombre de programa definido por el usuario.
<n> es el índice I/O, 1 ≤<n> ≤16
<estado> es 1 (ON) ó 0 (OFF).

Descripción: Ideal para respuestas sensoriales, el comando TRIGGER comienza la ejecución de <prog> cuando se pone en <estado> la entrada o salida específica.

Si se omite el <estado>, la ejecución de <prog> comienza tan pronto como la I/O especificada cambia sus estado.

TRIGGER es un comando de una sola pasada. Ejecuta un programa una sola vez. Se debe repetir el comando TRIGGER para reactivar el programa.

Ejemplos: TRIGGER DRILL BY IN 15 1

Comienza DRILL cuando se activa la Entrada 15.

TRIGGER TAKE BY OUT 8

Comienza TAKE cuando la Salida 8 cambia de estado.

Formato: UNDEF <pos>

UNDEF <pvect>

Descripción: Inicializa los valores de posición. Las posiciones todavía están definidas, pero sus valores de ubicación ya no están asignados.

Si se especifica un <pvect>, se inicializan los valores de todas las posiciones en el vector.

Este comando es útil con el comando APPEND ya que APPEND puede asignar un valor a una posición definida sólo cuando no tiene ya un valor de ubicación asignado.

Ejemplos:

UNDEF VECTV[5]	Borra el valor de la posición 5 del vector VECTV.
UNDEF VECTV	Borra los valores de todas las posiciones del vector VECTV.

Nota: Este es un comando de definición. Cuando se escribe, se ejecuta el comando pero no crea línea de programa, ni en modalidad de EDICION.

VER
DIRECTO

VER
DIRECTO

Formato: VER

Descripción: Presenta la versión y fecha de creación de la EPROM.

Ejemplos:> VER
-- ESHED ROBOTEC --
VERSION: 1.32
DATE: 10/01/90

WAIT
EDITOR

Formato: WAIT <var1> <cond> <var2>

Donde: <cond> puede ser, <,>,>=,<= _=

Descripción: La ejecución del programa se suspende hasta que la condición especificada sea verdadera.

Cuando un programa esta esperando que una entrada alcance un estado específico, este comando resulta muy útil, ya que WAIT usa muy poca potencia del CPU mientras espera un acontecimiento.

Ejemplos: WAIT IN[J]=1 Espera hasta que la Entrada J esta ON.

WAIT X<Y

Formato: * <comentario de usuario>

Donde <comentario de usuario> es una serie de hasta 40 caracteres y espacios.

Descripción: Este no es un comando de ejecución. Simplemente permite incluir comentarios en el texto del programa.

Ejemplos: *ESTE ES UN EJEMPLO DE COMENTARIO

***** PRINCIPIO DE PROGRAMA ******

*** *** F I N *** ***

Formato: @<comando DIRECTO>

Donde <comando DIRECTO> es una serie escrita en Formato de comando DIRECTO.

Descripción: Permite la ejecución de un comando DIRECTO desde un programa de usuario en ejecución.

Para asegurarse de que se ejecuta el comando antes de que continúe el Programa, introducir n comando de demora abreviado después de cada comando @.

Ejemplos: @ SHOW DIN Cuando el programa alcanza esta línea de comando, aparecerán los estados de todas las entradas.

```
@ ATTACH LOAD
 DELAY 10
 @ LISTPV POSITION
 DELAY 10
```

Formato: **<Alt> +M** ó (~)

Descripción: Activa y desactiva el control manual del robot desde el teclado. Al pulsar , se activa la modalidad manual, y aparece el siguiente mensaje:

MANUAL MODE!(modalidad manual)
JOINT MODE(modalidad mixta)

Esto indica que el controlador esta en modalidad manual de mixtas.

Las teclas J y X cambian las coordenadas a sistemas mixtos y cartesianos (XYZ), respectivamente. Sin embargo, el control de teclado es diferente si se activa la modalidad manual con el controlador en el sistema de coordenadas XYZ, según denota el segundo grupo del cuadro siguiente:

	TECLA	MOVIMIENTO	DIRECCION
Mixtas:	I/Q	mueve base	dcha./izda
	2/W	mueve hombro	arriba/abajo
	3/E	mueve codo	arriba/abajo
	4/R	mueve muñeca paso	arriba/abajo
	5/T	mueve muñeca giro	dcha./izda
	6/Y	abre/cierra pinza	
	7/U	mueve eje 7	+/-
	8/I	mueve eje 8	+/-
	9/O	mueve eje 9	+/-
	0/P	mueve eje 10	+/-
	-/[	mueve eje 11	+/-
XYZ:	I/Q	mueve	X+/-
	2/W	mueve	Y+/-
	3/E	mueve Z	arriba/abajo

C/F Activa/desactiva los servo-ejes (CON/COFF)

S Permite ajustar la velocidad (speed)

Se desactiva la modalidad manual pulsando <Alt>+M. Esto es especialmente útil cuando el teclado no incluye el carácter (~) .

CAPITULO 4

Elementos Predefinidos del Sistema

Además de los comandos y datos, ACL tiene un número de elementos del sistema predefinido que se usan durante la programación y operación del sistema robótico.

Procedimientos Internos del Sistema

HOME

El procedimiento de HOME (REFERENCIA), consiste en la búsqueda del microinterruptor de referencia de todos los ejes del robot.

Este procedimiento es activado bien ejecutando el comando de ACL HOME, o pulsando, desde la botonera de enseñanza RUN 0.

Si el robot en la configuración del controlador esta definido como tipo cero, se debe usar el comando HOME n o HHOME n para cada eje individual. Para los ejes del grupo B y C también se debe hacer home individualmente a cada eje (HOME ó HHOME).

Referirse al capítulo 3 para mayor detalle.

TEST

El procedimiento TEST ejecuta una rutina de comprobación hardware.

Esta rutina se activa bien ejecutando el comando TEST o tecleando RUN 999 desde la botonera de enseñanza.

Referirse al capítulo 3 para mayor detalle.

Nombres de Programas Reservados

El ACL para el Controlador A tiene dos nombres reservados para programas de usuario: AUTO y CRASH. Estos programas se crean y editan en modo EDIT, igual que cualquier otro programa. Si existen estos programas se ejecutarán automáticamente cuando ocurran unas determinadas condiciones.

AUTO

El programa AUTO se ejecuta automáticamente cuando el controlador se enciende. Comandos que se sugiere introducir:

- Activación/desactivación de Entradas y Salidas.
- Asignación de vector de posiciones a la botonera (ATTACH).
- Ejecución de algún programa

Ejemplo:

PROGRAMA AUTO

```
HOME                    Cuando se enciende el controlador, el robot
DIMP PV[10]             busca el HOME, se define un vector de 10
@ATTACH PV              posiciones, se asignan a la botonera y se
DELAY 10                ejecuta el programa OPER
RUN OPER
END
```

CRASH

El programa CRASH se ejecuta automáticamente cuando ocurre un impacto, salta la protección software térmico o se da el error "velocidad demasiado alta". Se recomienda incluir en este programa:

- Activación/desactivación de E/S.
- Mensajes de alarma

Ejemplo:

PROGRAMA CRASH

```
* SALIDA 16 = TIMBRE DE ALARMA
SET OUT[16]=1
PRINTLN " ROBOT PARADO"
PRINTLN "COMPROBAR Y CORREGIR PROBLEMA"
PRINTLN "REINICIAR APLICACION"
END
```

POSITION

POSITION es una posición definida del sistema, reservada para los valores de coordenadas de la posición actual del robot.

POSITION se puede utilizar para leer los valores de las coordenadas de la localización actual del robot y para asignar esos valores a variables o a otra posición.

Ejemplos

- LISTPV POSITION — Muestra los valores de las coordenadas actuales
- SETP 100 = POSITION — La posición 100 recibe los valores e las coordenadas que tiene el robot en ese instante.

  Equivalente — al comando HERE 100.
- SET var = PVAL POSITION 3 — La variable var recibe el valor de la coordenada, en pasos de encoders, del eje 3 (codo).
- SET var[1] = PVALC POSITION X
  SET var[2] = PVALC POSITION Y
  SET var[3] = PVALC POSITION Z
  SET var[4] = PVALC POSITION P
  SET var[5] = PVALC POSITION R

  Var[I] recibe el valor en coordenadas Cartesianas que tiene el robot en ese momento.

Se puede cambiar la posición del robot usando POSITION como se muestra en los siguientes ejemplos.

Atención ; el robot se moverá inmediatamente a la nueva POSICION; por lo que se debe hacer con pequeños cambios de coordenadas.

- SHIFTC POSITION BY 2 100
- SHIFTC POSITION BY Z 100
- SETPV POSITION 1 500
- SETPVC POSITION Y 300

Variables del Sistema

Las variables del sistema indican el estado de elementos de este como son las entradas, salidas y otros elementos del sistema. Con estas variables se pueden ejecutar comprobaciones y aplicaciones que requieren información en tiempo real sobre elementos del sistema.

ACL tiene nueve variables de sistema:

IN[16]	TIME	MFLAG
OUT[16]	LTA	ERROR
ENC[11]	LTB	ANOUT[11]

Los índices indican la dimensión de los vectores de las variables.

Estas variables se pueden manejar de la misma manera que las variables de usuario normales. Sin embargo estas variables no se pueden borrar y algunas son sólo de lectura.

Los valores de las variables IN; ENC; LTA; LTB; TIME; y MFLAG son actualizadas continuamente por el sistema , por lo que son variables de sólo lectura.

IN[N]

El valor de esta variable indica el estado de la correspondiente entrada.

El valor es actualizado en cada ciclo del microprocesador de acuerdo con el estado de la entrada. Cualquier valor introducido en esta variable será automáticamente sobrescrito.

Esta considerada como variable de sólo lectura.

n = índice de la entrada; puede ser una variable o una constante y no debe exceder el número de entradas configuradas (16 por defecto).

Ejemplos

Propósito	Comando	Display	Notas
Ver estado de la entrada	PRINTLN IN[3]	1 / 0	Activada / Desactivada
Control de programa	IF IN[I]=0 SET OUT[2]=1 ENDIF		

OUT[n]

El valor de esta variable esta determinado por el estado de la correspondiente salida.

El valor de la variable es aplicado a la salida correspondiente cada ciclo de reloj.

Es una variable de escritura/lectura.

n = índice de la salida; puede ser una variable o una constante; no puede exceder el número de salidas configuradas (16 por defecto).

Ejemplos

Propósito	Comando	Display	Notas
Ver el estado de la salida	PRINTLN OUT[7]	1 / 0	Activada / Desactivada
Comprobar y cambiar el estado del aparato conectado a la salida	IF OUT[5]=0 SET OUT[5]=1 ENDIF		Si la salida 5 desactivada (p. E. Una lámpara) activarla
Cambiar el estado de una salida	SET OUT[15]=1 - OUT[5]		

ENC[n]

El valor de esta variable indica el número de pasos de encoder del eje especificado en la posición actual del robot.

El valor de la variable se actualiza cada paso de reloj con el valor de los pasos de encoder del eje determinado. Se considera una variable de sólo lectura.

n = índice del eje; puede ser una variable o una constante; no puede exceder el número de salidas configuradas.

Ejemplos

Propósito	Comando	Display	Notas
Ver el estado del encoder	PRINTLN ENC[1]	0 a 6844	Pasos de encoder
Asignar el valor del encoder a una variable	SET X = ENC[5]		El valor del encoder 5 es escrito en la variable X

TIME

Esta variable contiene el valor del reloj interno del controlador. Cuando el controlador se enciende o se resetea el reloj se inicia a 0. Cada 10 milisegundos (tic) el reloj del controlador es incrementado en una unidad.

Es una variable de sólo lectura.

Propósito	Comando	Display	Notas
Determinar la duración de un movimiento	PROGRAM TIME SET TIMEA=TIME MOVED POS99 SET TIMEA=TIME - TIMEA PRINTLN " TIEMPO = " PRINT TIMEA "MSEG."		

LTA y LTB

Los valores de estas variables indican el tiempo (reloj interno) en el cual el grupo especificado de ejes alcanza la posición que se le ha ordenado ejecutar.

LTA se aplica al grupo de ejes A y LTB al grupo de ejes B.

Estas variables se utilizan cuando los comandos MOVE, MOVEC y MOVES son colocados en el buffer. Estas variables posibilitan entre otras cosas la sincronización en células de trabajo: coger piezas de cinta de transporte, sincronización de dos grupos de ejes, etc..

LTA y LTB son variables de sólo lectura.

Propósito	Comando	Display
Sincronizar la llegada de los ejes del grupo A y B **POSA5, una posición de l grupo A** **POSB3, una posición del grupo B**	PROGRAM SYNCR MOVE POSA5 SET V=LTA-TIME MOVE POSB3 V	

MFLAG

El valor de esta variable indica qué ejes están en ese momento en movimiento. Es una variable de sólo lectura.

Cuando un comando MOVE es ejecutado, los 32 bits de la variable MFLAG cambian su valor de acuerdo con el siguiente esquema

Bit 1 (menos significativo) a 1 si eje 1 en movimiento.

Bit 2 a 1 si eje 2 en movimiento.

Bit 3 “

.....

Bit 11 a 1 si eje 11 en movimiento.

....

Bit 32 (más significativo)

Los bits de 12 al 32 siempre están a cero.

Asumiendo que el controlador esta configurado con 5 ejes grupo A, una servopinza en el eje 6, dos ejes grupo B y 3 eje en grupo C, el valor de MFLAG indicará el movimiento según el siguiente diagrama.

Valor Bit	1	2	4	8	16	32	64	128	256	512	1024
Grupo	**Grupo A**					**Pinza**	**Grupo B**		**Grupo C**		
Eje	1	2	3	4	5	6	7	8	9	10	11

Ejemplos

Propósito	Comando	Display	Notas
Estado de movimiento de ejes	PRINTLN MFLAG	31	31=1+2+4+8+16 Todos los ejes del grupo A en movimiento
Estado de movimiento de ejes	PRINTLN MFLAG	543	543=31+512 Todos los ejes del grupo A y el eje 10 en movimiento

ERROR

Cuando ocurre algún error durante el funcionamiento, a la variable ERROR se le asigna un valor que indica el correspondiente error. También se muestra en pantalla un mensaje de error.

Referirse al capítulo 5 para ver los mensajes de error.

La variable ERROR debe tener un valor inicial de cero. Es una variable de escritura/lectura.

Ejemplo.

Propósito	Comando	Display	Notas
Para identificar errores en tiempo de ejecución	PROGRAM MOVE LABEL 1 MOVE POS66 MOVE POS67 GOTO 1 PROGRAM ERROR SET ERROR=0 *ESPERA UN ERROR WAIT ERROR<> 0 PRINTLN ERROR	 53	Se ejecutan simultáneamente el programa MOVE y ERROR 53=Nº identif. error

ANOUT[n]

Esta variable contiene el valor del DAC (convertidor analógico-digital) que se esta aplicando al motor del eje especificado por el índice n.

El valor de ANOUT[n] se actualiza en cada paso de reloj.

Cuando se ejecuta el comando SET ANOUT, el servo control del eje esta desactivado; COFF esta efectivo hasta que CON es activado.

ANOUT[n] es una variable de escritura/lectura.

N= índice del eje; puede ser una variable o una constante y no debe exceder el máximo valor de ejes configurados.

El rango de los DAC esta entre -5000 a +5000.

Atención! Usar con cuidado para no dañar el motor.

Ejemplo.

Propósito	Comando	Display	Notas
Para ver el valor actual del DAC de un eje	PRINTLN ANOUT[5]	2500	El DAC esta a la mitad de su máximo valor
Para ajustar el valor del DAC de un eje	SET ANOUT[1]=1000		Pone la salida analógica del eje 1 a 1000

Intencionadamente en blanco

CAPITULO 5

LA VARIABLE ERROR

La variable ERROR es una variable del sistema. Cuando ocurre un error en el sistema, esta variable toma un valor definido y en pantalla aparece un mensaje.

Errores en los movimientos del brazo

53 *** IMPACT PROTECTION eje n

El controlador ha detectado un error de posición demasiado grande. El sistema aborta todos los movimientos de ese grupo de ejes y los desactiva. El programa de usuario CRASH, si existe, se ha ejecutado.

Posibles causas:

1) Un obstáculo impide el movimiento del eje n.
2) El fusible del eje n se ha fundido.
3) Fallo del encoder.
4) Fallo mecánico.
5) El eje no esta conectado

- Determinar y corregir el error de posición. Activar el servo control de ejes (CON) y reinicializar el programa.

54 *** OUT OF RANGE eje n

Un intento de grabar una posición (HERE) mientras el robot estaba fuera de su campo de trabajo.

- Mover el robot dentro de su campo de acción y repetir la orden.

55 *** THERMIC OVERLOAD eje n

Por medio de una rutina software de protección térmica del motor, se ha detectado una condición peligrosa para ese motor. El robot aborta todos los movimientos de los ejes de ese grupo. El programa CRASH, si existe, se ejecuta. Causas posibles:

1) El brazo intenta alcanzar una posición que no puede acceder debido a un obstáculo. La protección de impacto no se activa debido a que el obstáculo esta muy cerca de la posición final. La realimentación integral incrementará la corriente del motor sobrecalentándolo, por lo que se activará la protección térmica.
2) Fallo en el eje correspondiente o el fusible se ha fundido.

3) El brazo del robot esta cerca de la posición final, pero no ha llegado debido a un fallo del driver. El software detecta entonces una situación anormal.

4) Los parámetros de Protección Térmica están mal seleccionados.

- Comprobar la posición, la tarjeta de control del eje y los parámetros. Restablecer el servo control (CON) y repetir el movimiento.

Errores durante la Ejecución de un Programa

302 ARITHMETIC OVERFLOW AT LINE n

El resultado de una operación aritmética esta fuera de rango (o no es válida).

303 NO POSITION ASSIGNED TO POINT pos

La posición ha sido definida usando el comando DEFP, pero sus coordenadas no han sido grabadas.

- Usar HERE u otro comando para asignar las coordenadas a la posición.

304 AXIS DISABLED

Ha intentado mover un eje no permitido. Posibles causas:

1) No se puede ejecutar un movimiento porque el servo control de los ejes esta desactivado (COFF).

2) Un movimiento previo del brazo ha dado un error de Impacto o error de Trayectoria desactivando el servo control.

- Comprobar los movimientos del robot y corregir el/los comandos.

305 TOO DEEP NESTING

Demasiadas subrutinas- GOSUB - anidadas.

306 INVALID PROGRAM

El comando RUN, GOSUB, TRIGGER no puede ser ejecutado por fallo sintáctico o lógico en el programa. Por ejemplo el programa contiene un comando IF sin su correspondiente ENDIF.

309 INDEX OUT OF RANGE

Ha intentado usar un valor índice fuera de rango de un vector de variables o de posiciones.

310 BAD AXIS

El eje no esta en el grupo definido por el comando, eje no configurado, eje fuera de rango, etc.

311 LOOPING RELATIVE CHAIN OR DEPTH EXCEEDED 32 POINTS

Ha usado más de 32 puntos relativos unidos

312 BAD POINT COORDINATE [pos]

Ha intentado usar una posición no válida. Por ejemplo, una posición relativa definida fuera del rango del eje especificado.

- Grabar nuevas coordenadas para la posición.

315 INCOMPATIBLE POINTS

Posibles causas:

1) Ha intentado usar un comando HERE para posiciones en diferentes grupos de ejes o posiciones que ambas son del grupo C pero asignadas a diferentes ejes.
2) Ha intentado copiar un punto (SETP) desde un grupo de ejes a otro grupo.

316 NO GRIPPER CONFIGURATION

El comando de pinza (gripper) no se puede ejecutar porque no ha sido configurada la pinza.

317 BAD CARTESIAN POINT pos

El comando no se puede ejecutar o la posición no se puede grabar, debido a un valor no válido de la posición especificada (valores XYZ fuera de rango).

Intencionadamente en blanco

CAPITULO 6

CONFIGURACION DE LA MEMORIA DE USUARIO

El Controlador-A estándar tiene 128KB de RAM con batería, que se redistribuye durante la configuración del controlador.

Por ejemplo, cuando la configuración se realiza en ATS, mediante la combinación de las teclas <Ctrl>+F1, la opción SCORBOT ER VII y se seleccionan 8 ejes, la distribución por defecto de la memoria para cada tipo de elemento será la siguiente:

150	Programas
3000	Líneas de programa
600	Variables
2380	Posiciones grupo A
2380	Posiciones grupo B
0	Posiciones grupo C
550	Comentarios

Sólo los grupos A y B se definen en una configuración por defecto del controlador. Grupo A son los ejes del 1 al 5. Eje 6 es para la pinza eléctrica. El resto de ejes son del grupo B. Para definir ejes del grupo C se debe usar el comando CONFIG (configuración personalizada).

El número de elementos se calculan de acuerdo con la cantidad de memoria requerida para cada elemento:

Elemento	Memoria
Cabecera de programa	11 bytes cada una
Líneas de programa	10 bytes cada una
Variables de usuario	16 bytes cada una
Posiciones de grupo A o B	[(número de ejes del grupo +1) x 2 + 8] bytes cada una
Posiciones de grupo C	14 bytes cada una
Comentarios/string	12 bytes cada uno

Los parámetros y variables de usuario pueden ocupar hasta 2.5KB.

La suma de todos los elementos no debe sobrepasar los 128KB.

El sistema requiere un número mínimo de algunos elementos, ver lista. Si se seleccionan un número menor automáticamente el sistema seleccionará el mínimo.

Elemento	Mínimo
Programas usuario	2
Líneas programa	50
Variables usuario	10
Comentarios/string	50

Ejemplo:

Controlador configurado para 11 ejes:

El Grupo A se define como 5 ejes - ejes 1 a 5 (el robot)
La pinza se define como eje 6
El Grupo B se define como 4 ejes - ejes 7 a 10
El Grupo C se deja así sólo con un eje - eje 11

Se definirán los siguientes elementos:

100 Programas	= 100x 11 = 1100 bytes
550 Líneas	= 550 x 10 = 5500 bytes
200 Variables	= 200 x 16 = 3200 bytes
450 Pos. Grupo A	= 450 x 20 = 9000 bytes
450 Pos. Grupo B	= 450 x 18 = 8100 bytes
150 Pos. Grupo C	= 150 x 14 = 2100 bytes
100 comentarios	= 100 x 12 = 1200 bytes
Total BBRAM usuario	30200 bytes
BBRAM reservada por el sistema	2560 bytes
Total BBRAM	32760 bytes

El total de memoria necesaria para esta configuración es 32760 bytes. La configuración es válida porque 32760 < 131072; (131072 = 128 x 1024).

CAPITULO 7

PARAMETROS

Muchas de las funciones del controlador dependen del valor de los parámetros del sistema. Los parámetros del sistema determinan operaciones tales como:

* Servo control
* Envoltura de trabajo
* Protección de eje
* Límites de velocidad
* Operación de la pinza
* Botonera de enseñanza (instructor) y operación manual
* Cálculos cinemáticos cartesianos

Aviso!

- No cambiar los valores paramétricos con el robot en movimiento. Cuidado al manipular los parámetros.
- Sólo los usuarios avanzados deben cambiar parámetros.
- Guardar los parámetros antes de cambiarlos.
- Asegurarse que los parámetros de protección de impacto están bien seleccionados. Trabajar sin la protección a impacto activada puede dañar el brazo del robot.

Comandos de acceso

Los comandos que se usan para acceder a los parámetros del sistema son los siguientes:

SHOW PAR n	Presenta el valor del parámetro n
LET PAR <n> =	Cambia el valor de parámetro n a
= <nuevo valor>	<nuevo valor>.
SENDPAR	Presenta todos los parámetros del sistema.
INIT CONTROL	Debe lanzarse después de cambiar cualquier parámetro; de otra forma, no
se	procesan los cambios adecuadamente.
INIT PROFILE	Debe lanzarse después de cambiar el parámetro 76.

Para ver el valor actual de un parámetro usar SHOW PAR n

Descripción de Parámetros

El Controlador -A tiene dos tipos de parámetros:

- Parámetros aplicables al elemento independientemente del eje al que esta conectado. Por ejemplo PAR 176 define el valor DAC aplicado al motor de la pinza al comienzo del movimiento.
- Parámetros que son aplicados a cada eje individualmente. Estos parámetros están en lotes de un rango de números, a intervalos de 20, en la tabla de parámetros del controlador. El rango se indica por el término PAR n+ eje; por ejemplo PAR 180+eje (PAR 180+2= PAR 182).

Los parámetros referentes a valores DAC están en la escala de ±5000. (Equivalente a ±24V en el motor).Otros valores de parámetros están indicados en pasos de encoder, tics de reloj, medidas lineales, etc..

Hay que observar que algunos parámetros sólo son válidos para una configuración de robot específica. El efecto de otros puede cambiar dependiendo del brazo robot utilizado. Se debe leer cuidadosamente la documentación antes de cambiar parámetros.

Algunos parámetros sólo son operacionales en las versiones de EPROM 1.32 y posteriores o versión F.44 y posteriores. Asegurarse de comprobar la versión de la EPROM antes de intentar utilizar estos parámetros. Para comprobar la versión de la EPROM se usa el comando VER.

El controlador SCORBOT-ER VII esta equipado con la EPROM F.44 o posterior, para poder utilizar la pinza servo con realimentación.

Parámetros del Servo Control del Eje

Estos parámetros determinan el servo circuito de cada eje (1 a 11).

PAR 21 ... 31

Constantes de realimentación proporcional por eje.

PAR 41 ... 51

Constantes de realimentación diferencial por eje.

PAR 61 ... 71

Constantes de realimentación integral por eje.

PAR 81 ... 91

Compensaciones absolutas para la salida DAC por eje.

El valor absoluto aumenta el resultado de realimentación DAC. Ajusta la salida DAC mínima bajo servo control.

Escala: 0 - 5000.

Valor típico 0 a 200

Por ejemplo, los servo parámetros que definen el movimiento del eje 3 son 23, 43, 63, 83.

Parámetros de Servo Control Global

Estos parámetros definen los cambios de escala en las constantes de servo realimentación cuando el brazo se esta moviendo o terminando un movimiento.

PAR 78

Si el parámetro 78 no es cero, las constantes de realimentación proporcional y diferencial se doblan al final del movimiento.

PAR 79

Relación de las constantes de realimentación integral reducida con los ejes en movimiento. Durante el movimiento, las constantes integrales se dividen por el PAR 79.

Parámetros Limitadores del Eje

Estos parámetros determinan los límites del movimiento del eje en unidades codificadoras.

PAR 101 ... 111

Límite superior por eje.

PAR 121 ... 131

Límite inferior por eje.

Parámetros de Error de Posición de Eje

Estos parámetros definen el error de posición máximo permitido para la terminación de un comando MOVED, MOVELD ó MOVECD. Estos parámetros son activos sólo en modalidad EXACT.

PAR 261 ... 271

Error de posición máximo por eje en unidades de codificador.

Si el eje es la servo pinza, PAR 266 define la fluctuación máxima del valor del encoder mientras la pinza esta bloqueada. En este caso la versión de la EPROM debe ser F.44 o mayor.

Parámetro Suavizador

PAR 77

Este parámetro define un factor de suavidad para los movimientos lineales y circulares. Si el PAR 77 es mayor que 0, se hace una media de los puntos de la trayectoria lineal/circular para reducir la vibración del brazo.

Se aumenta la suavidad si se aumenta el PAR 77.

Escala: 0 - 4.

Hay que observar que el brazo puede saltar ligeramente al final del movimiento si el parámetro tiene un valor demasiado alto.

Sólo válido para la versión de EPROM 1.32 y posteriores.

Parámetro Perfil de Velocidad

PAR 76

Determina los perfiles de velocidad de los movimientos del robot. Este parámetro determina la velocidad mínima al acelerar o decelerar. Un valor más alto aumenta la velocidad de aceleración - deceleración.

Valores típicos: 0-10

El comando INIT PROFILE se debe ejecutar siempre que se cambie el PAR 76.

Parámetros de Protección Térmica

La protección térmica desactiva la potencia del motor si se inmoviliza un motor durante largo tiempo. Estos parámetros definen el cálculo y umbral de error para la alarma térmica de cada eje.

Los parámetros 141 ... 151 definen las características básicas del motor, mientras que los parámetros 161 ... 171 definen el umbral de error para la salida DAC térmica.

El valor DAC actual junto con las características del motor resultan en un valor DAC térmico teórico, equivalente a la cantidad de potencia no traducida en movimiento.

PAR 141 ... 151

Característica voltaje/velocidad de un motor de marcha libre, usado para calcular la carga térmica media del motor, definido por:

$$r = \frac{\text{salida DAC lógica x } 65{,}53}{\text{número de cuentas de codificador por tic (10 ms)}}$$

Un tic de reloj es 10 ms. Se asume DAC= 5000

En nuestro caso:

$$r = \frac{327650}{\text{número de cuentas de codificador por 10 ms}}$$

Con DAC 5000, un motor de SCORBOT-ER Vplus con un codificador de 20 slots tiene una velocidad de 8555rpm, resultando 32 cuentas por tick y esto da una característica típica del motor de 3000.

El valor actual de la salida analógica combinada con el valor característico del motor da como resultado un valor teórico de salida analógica térmica que es equivalente a la cantidad de potencia no transmitida en movimiento.

PAR 161 ... 171

Define el umbral de error térmico; es decir, máxima potencia térmicamente permitida de DAC para un motor inmovilizado.

Cuando el valor teórico de la salida analógica térmica excede el valor de PAR 160+ eje, un error térmico es anunciado y la tensión del motor inmovilizado es anulada.

Escala: 1000 - 5000

Valor típico: 4000

Parámetros de Protección de Impacto

Estos parámetros definen la condición de un error de impacto (se especifican parámetros por cada eje).

PAR 181 ...191

Número de veces que se ha de comprobar el bloqueo real del motor antes de anunciar una condición de impacto.

PAR 240

Define la respuesta a un impacto entre grupos de control

Si PAR 240 = 0 : se paran todos los motores

Si PAR 240 ≠ 0 : sólo se paran los motores del Grupo al que pertenece el motor impactado.

Sólo para versiones EPROM 1.42 en adelante.

PAR 241 ... 251

Número mínimo de pulsos de codificador por tic (10 ms) cuando se suministra al eje la potencia especificada en el PAR 281 ... 291.

Si la potencia DAC es mayor que la especificada en PAR 281 ... 291 y la cantidad de movimiento es menor que la especificada en PAR 241 ... 251, se sospecha un impacto.

Si PAR 240 + eje =0: La protección de impacto no es aplicada.

¡Atención! Trabajar sin protección de impacto puede dañar el brazo.

PAR 281 ... 291

El valor de salida del DAC sobre el cual el algoritmo de impacto comienza a buscar movimiento de impacto (normalmente debe ser mayor de 4000).

Parámetros Limitadores de Velocidad

Estos parámetros definen las velocidades efectivas del robot en unidades de codificador por tic de reloj. Un tic de reloj es igual a 10 ms.

Unidades para PAR 297, 298 y 299 son, (paso de codificador/ tic de reloj) x 1000.

PAR 298

Límite inferior de velocidad

PAR 299

Límite superior de velocidad

PAR 297

Velocidad de transición entre el algoritmo manual de velocidad muy baja y el algoritmo manual normal.

El algoritmo de baja velocidad acumula solicitudes de movimiento, lo que resulta en una correlación ojo/mano mejorada.

El algoritmo normal no acumula solicitudes de movimiento manual. As, se evitan los tirones de alta velocidad.

PAR 294

Define la velocidad manual máxima de movimientos cartesianos. Las unidades son, (0,1 mm /tic de reloj) x 1000.

Valor típico de **ER V**: 8000

Valor típico de **ER VII**: 13000

PAR 296

Factor de reducción de velocidad para ejes periféricos. El valor es un porcentaje de la velocidad del robot.

El PAR 296 sólo es válido para la versión de EPROM 1.32 y posteriores.

Parámetros de Velocidad Manual

Estos parámetros definen la velocidad manual relativa de cada eje al usar la botonera de enseñanza (instructor).

PAR 221 ... 231

Define el valor de velocidad relativa para la operación manual con botonera de enseñanza de cada eje. La escala difiere de los ajustes de velocidad normales.

El valor se puede considerar que define la velocidad relativa entre ejes y no una velocidad física real que sólo se determina de acuerdo con el valor de velocidad real durante el movimiento.

Parámetro Manual de Teclado

Parámetro de repetición de teclado para movimiento manual.

PAR 300

Número de veces que hay que repetir un golpe de teclado para producir un suave movimiento continuo.

Parámetros Interfase de Codificador

Estos parámetros determinan una relación de división entre la cuenta de codificador real y la cuenta de codificador utilizada para el control ACL. Estos parámetros pueden ser necesarios debido al límite ±32767 en valores de codificador ACL.

PAR 301 ... 311

El valor de estos parámetros define realmente el número de veces que se divide por 2 cada valor de codificador. Por ejemplo:

Un valor 1 resulta en una división por 2

Un valor 2 resulta en una división por 4

Un valor 3 resulta en una división por 8

Parámetros de la Pinza

Estos parámetros determinan la operación de la pinza.

PAR 72

Tiempo de cierre de la pinza (en unidades de centésimas de segundo). Valor típico: 125 (1,25 segundos).

PAR 72 es válido sólo para pinza eléctrica sin realimentación. Válido sólo para versión EPROM 1.32 y posterior.

PAR 73

Número de cuentas de codificador para cerrar la pinza desde su posición totalmente abierta.

Si PAR 73=0: no hay encoder en el motor de la pinza.

PAR 73 válido sólo para pinza con realimentación por encoder.

PAR 74

Lectura de codificador para pinza cerrada. Válido sólo para pinza con relimentación.

PAR 75

Voltaje DAC máximo para el motor de la pinza en el comando abertura/cierre.

Rango: 0 - 5000.

Al usar una pinza standard SCORBOT-ER Vplus, no se debe fijar un valor superior a 3500 en el parámetro 75.

PAR 275

La potencia de sujeción constante aplicada al motor de la pinza después de la terminación de un comando CLOSE.

Este valor determina una salida DAC constante para el motor de la pinza. Se debe tener cuidado de no fijar este valor demasiado alto.

Valor típico: 1000.

Sólo válido para pinzas eléctricas sin encoder. Válido sólo para versiones 1.32 y posteriores.

PAR 276

Define el valor del DAC para ser aplicado a la pinza al comienzo del movimiento.

Sólo válido para versiones F.44 y posteriores.

PAR 277

Define la duración del parámetro 276. Las unidades son en centésimas de segundo.

Sólo versiones F.44 y posteriores.

Parámetros de Posicionamiento en Origen (HOME)

PAR 200

Si PAR 200 = 0 (por defecto): se ejecuta una rutina de HOME doble de alta precisión.

Si PAR 200 = 1 la rutina >HOME es más rápida y de menor precisión.

Sólo para versiones EPROM 1.42 en adelante.

PAR 201 ... 211

Voltaje DAC máximo permitido al posicionar en origen. Si, al posicionar en origen, el valor DAC es demasiado grande, se interpretar como un motor mecánicamente bloqueado. 5000 es el voltaje DAC máximo.

PAR 212

Usado para velocidad baja durante el homing.

Define el número de impulsos de reloj en el cual el movimiento es parado entre cada intervalo mientras busca el microinterruptor límite.

Si PAR 212=0: (por defecto) no disminuye la velocidad durante el homing.

Sólo válido para versiones 1.42 y posteriores.

PAR 295

Velocidad relativa de movimiento al posicionar en origen un eje individual utilizando el comando HOME <eje>. El valor es un porcentaje de la velocidad de posicionamiento en origen en codificador del eje 1 (es decir, 100 = la misma velocidad de posicionamiento en origen que el codificador del eje 1).

Hay que observar que la velocidad de posicionamiento en origen también esta determinada por los parámetros interfase del codificador del robot.

El PAR 295 es necesario debido a los diferentes codificadores instalados en el robot y el diverso equipo periférico.

Parámetros de Cálculo Cartesiano

Estos parámetros definen las distancias, codificadores y relaciones de transmisión del brazo mecánico, y la posición origen del robot. Estos parámetros también se usan para calcular la posición cartesiana del brazo. La estructura del robot debe articularse verticalmente.

El acoplamiento o desacoplamiento de los ejes esta determinado de acuerdo con el tipo de robot especificado en los procedimientos de configuración.

- SCORBOT ER Vplus es un robot desacoplado. En un robot desacoplado, un movimiento del hombro (eje 2) no cambia el ángulo del codo (eje 3) y de la muñeca (pitch). Cuando se mueve el codo, la inclinación de la muñeca mantiene el ángulo.
- SCORBOT ER VII es un robot acoplado. En un robot acoplado, todos los ejes están acoplados de tal forma que un movimiento del eje 2 cambia el ángulo del eje 3 pero mantiene el ángulo entre el eje 2 y 3 constante. Lo mismo se aplica a otros ejes.

Hay que observar que los movimientos de paso y giro del SCORBOT-ER Vplus se crean combinando la rotación de los motores 4 y 5. En el SCORBOT-ER VII se usa un motor independiente para mover cada uno de estos ejes.

Parámetros de Escala de Rotación

PAR 33

Número de cuentas de codificador para +90 grados del eje 1.

PAR 34

Número de cuentas de codificador para +90 grados del eje 2.

PAR 35

Número de cuentas de codificador para +90 grados del eje 3.

PAR 36

ER Vplus: Diferencia en número de cuentas de codificador (codificador 4 - codificador 5) para +90 grados de paso de eje.

ER VII: Número de cuentas de codificador para +90 grados del eje 4.

PAR 37

ER V: Número total de cuentas de codificador (codificador 4 + codificador 5) para +90 grados de giro de eje.

ER VII: Número de cuentas de codificador para +90 grados del eje 5.

Parámetros Posición de Referencia Horizontal

Estos parámetros definen la compensación de codificador desde la posición origen a una posición donde todos los ejes están en posición horizontal, incluyendo un plano de pinza horizontal (posición H). Esta posición puede incluir también una compensación del eje 1.

PAR 52

Lectura del codificador 1 en la posición H.

PAR 53

Lectura del codificador 2 en la posición H.

PAR 54

Lectura del codificador 3 en la posición H.

PAR 55

Lectura del codificador 4 en la posición H.

PAR 56

Lectura del codificador 5 en la posición H.

Sólo válido para SCORBOT **ER VII**.

Sólo son válidos para la versión EPROM 1.32 y posteriores.

Parámetros de Longitud

PAR 92

Coordenada Y (offset desde el centro a lo largo del eje Y) de la punta de la pinza en la posición de HOME.

Sólo válido para EPROM 1.42 en adelante.

PAR 93

Coordenada X del eje de rotación del brazo 2 cuando el robot esta en la posición de Home.

PAR 94

Coordenada Z del eje de rotación del brazo 2.

PAR 95

Longitud del brazo 2.

PAR 96

Longitud del brazo 3 .

PAR 97

Distancia desde el final de la pinza hasta el eje de rotación de paso .