PLC
CAIPE SCD 80/800

Manual de uso

Av. Hipólito Irigoyen 2164 1º piso (B1869BUR)
++54 (011) 4218 1840 - **www.caipe.com**
Avellaneda, Buenos Aires, Argentina

TECNOLOGIA ARGENTINA

INDICE

INTRODUCCION

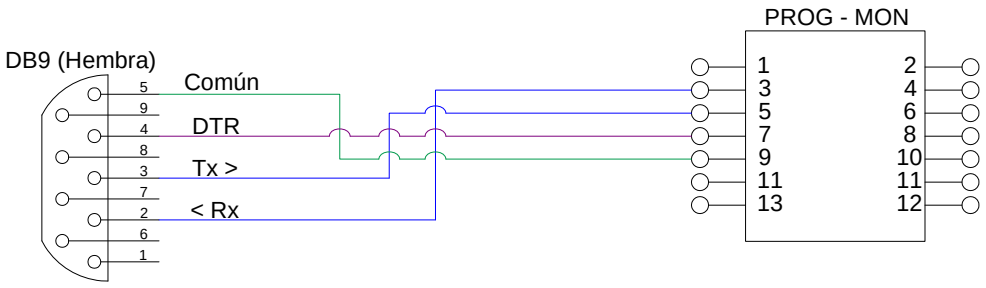
Está pensado para hacer control y adquisición de datos. Por medio de los módulos de entradas y salidas puede operar con los elementos de uso industrial, como interruptores, sensores inductivos, capacitivos, ópticos, termocuplas, PT100s, potenciómetros, lazos de corriente 4 - 20mA, celdas de carga, variadores de velocidad, etc. Control de temperatura, pesaje y dosificación, control de producción, lógica combinacional y secuencial, registro de eventos y señalización, tableros de comando y automatización. Su poderosa comunicación le permite armar redes de PLCs y monitorear y/o modificar variables de proceso en tiempo real. Se pueden armar redes de PLCs-PC, donde la PC es la cabeza de la red, o PLCs-PLCs, donde se pueden compartir datos o tener un PLC maestro. No se requiere hardware adicional, excepto para las redes donde es necesario usar un NODO por cada PLC. Bajo norma RS-232 y a 4800 baudios se pueden establecer comunicación half dúplex o full dúplex.
El equipo incluye una fuente de 24v que es para propósitos generales del uso del PLC, 7 indicaciones luminosas (4 de comunicación, fuente, funcionamiento y estado de batería), puerto para teclado-display, puerto para teclado-display alfanumérico e impresora (solo modelo con reloj de tiempo real), 2 puertos de comunicación serie RS-232, puerto para 2 expansiones, alojamiento para 2 módulos de entrada/salida y alimentación 220/110v.

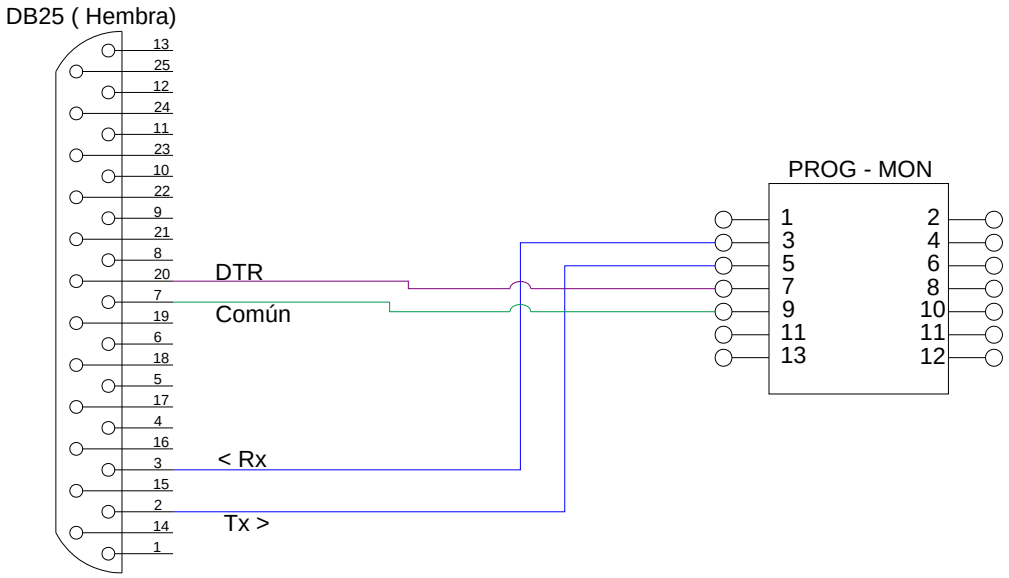
MODULOS DE EXPANCIÓN

Dependiendo del modelo de ROM el PLC CAIPE SCD 80/800 soporta los siguientes módulos de expansión, sin importar la repetición de los mismos.

Módulos	ROM 933	ROM 954	ROM 98	ROM 100	ROM 101	ROM Radio
Entradas						
16 Entradas Digitales	x	x	x	x	x	x
8 Entradas Analógicas 12bits				x	x	x
1 Entrada de celda amortiguada				x	x	
1 Entrada Contaje Rápido		x	x	x	x	x
Salidas						
12 Salidas Digitales	x	x	x	x	x	x
2 Salidas Ana. 2 canales 8bits		x	x	x	x	x
8 Salidas Ana. 8 canales 8bits		x	x	x	x	
1 Salida Pulsos					x	
Módulos combinados						
4 Entradas Dig. + 8 Salidas Dig.		x	x	x	x	x
4 Entradas Ana. 12bits + 8 Entr. Dig			x	x		

CABLE DE PROGRAMACIÓN





PROTOCOLO DE COMUNICACIÓN

En la red de PLCs CAIPE se usa el modelo maestro-esclavo. Esto significa que un único dispositivo inicia la comunicación (maestro, que se halla en la cabecera de la red). De los PLCs esclavos en la red solo uno contestará (aquel cuyo identificador coincida).

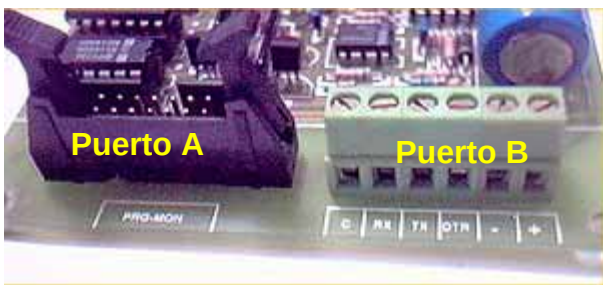
Enlace

En el enlace, la PC inicia la comunicación con el bloque de petición, y el PLC cuyo identificador coincide con el solicitado responde con su bloque de acknowlege. Los bloques intercambiados son de una longitud de 20 bytes, con un xor de los datos trasmitidos en el último byte.
El PLC tolera un tiempo máximo de 50 mseg. entre dato y dato que trasmita la PC, pasado ese tiempo se descarta el bloque.
El PLC solo responde cuando la petición es correcta.

Formato de la RS-232

Puerto A:	Baud rate:	**4800**
	Longitud de la palabra:	**8 bits**
	Numero de stop bits:	**2**
	Paridad:	**par (even)**
	DTR:	**(Deshabilitado o desconectado)**
Puerto B:	Baud rate:	**4800**
	Longitud de la palabra:	**8 bits**
	Numero de stop bits:	**1**
	Paridad:	**n (none)**
	DTR:	**(Deshabilitado o desconectado)**

NOTA: el dato siempre es de 8 bits y modelos anteriores de PLC solo tienen puerto A a menor velocidad.



Comandos

El protocolo de comunicaciones soporta dos tipos de comandos: de acceso por bloques y de acceso puntual.
Los comandos de acceso por bloques direccionan toda la memoria RAM del PLC (es decir las direcciones absolutas desde 0 hasta 8191), con bloques de 16 bytes consecutivos tomadas desde 0.Por ejemplo para direccionar 16 byte a partir de la dirección de la memoria usuario 288 (120 eh hex) se pide una comunicación con el bloque nro. 18 (12 en hex), o sea 288/16.
La memoria del PLC se divide en baja (de 0 a 575) y alta (de 2528 a 8191).La memoria baja a ser direccionada por los comandos de acceso por bloques tiene la siguiente disposición:

Direcciones de la memoria usuario de 0 a 501	0 0x1F5	AREA DE I/0 Y DE VARIABLES INTERNAS DEL PLC
Direcciones de la memoria usuario de 501 a 511 (área reservada)	0x1F6 0x1FF	AREA DE CONSTANTES Y ACUMULADORES
Direcciones de la memoria usuario de 512 a 575	0x200 0x23F	AREA DE ESTADO DE TEMPORIZADORES Y CONTADORES
	0x240 0x2BF	AREA DE ESTADO DE CUENTA DE TEMPORIZADORES Y CONTADORES
	0x300 0xFFF	VARIABLES INTERNAS DEL BIOS (área reservada)

El área de estado de cuenta de timers y contadores, son 128 bytes que forman 8 bloques de 16 bytes cada uno, donde cada contador o timer ocupa 2 bytes para su valor de cuenta con un numero que va desde 0 a 9999.Los bits mas altos deben enmascararse en la recepción con ceros pues no tienen significado. Por ejemplo:

Cuenta de Timer o Contador 512: **0x240 bbbbbbbb (parte baja)**
0x241 xxbbbbbb (parte alta)

La memoria alta del PLC comienza en 2528 y continúa hasta 8191. El área del teclado display alfanumerico comienza en 2528, y el final de la memoria ocupada por éste se especifica en los bytes 0x80Dy 0x80E, en parte alta y parte baja.
NOTA: al leer estos datos se notará que las direcciones son del tipo 4xxx o 5xxx. Esto se debe a que corresponden a direcciones absolutas en la memoria del PLC. Por lo tanto deducimos que la dirección 0 del usuario corresponde a la dirección 0x4000 en la memoria del PLC.

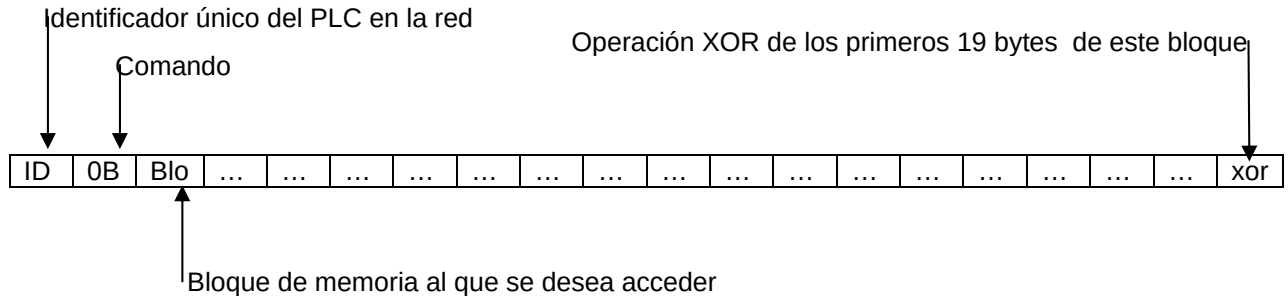
El final XXXX se especifica en 0x80D y 0x80E (2061 y 2062)	2528 XXXX	ZONA RESERVADA TECLADO DISPLAY ALFANUMERICO
Desde XXXX (lo ocupado por el teclado-display alfanumérico) hasta la dirección 0x1FFF	8192	AREA DE MEMORIA ALTA USUARIO

Los comandos de acceso puntual solo direccionan sobre el área de I/O y variables internas de la memoria usuario del PLC,(entre 0 y 501).El valor que se le transfiere como dirección es el de la parte baja y la parte alta del numero de variable. Por ejemplo si se desea acceder a la dirección 288 (120 en hex) se le pasa un 1 en el byte de parte alta y un 20 en el de la parte baja.

Comandos de acceso por bloque

LECTURA DE BLOQUE EN LA MEMORIA BAJA (comando 0B hex):
Permite leer un bloque de 16 bytes de la memoria baja del PLC (de 0 a 0x2BF) y parte de la alta (0x9E0 a 0xFFF). Se le debe especificar para ello el, identificador del PLC, el comando y el número de bloque, con el siguiente formato:

Bloque de petición:



ID	0B	Blo	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	xor

Bloque de acknowlege (respuesta):

ID	0B	Blo	Dat 1	Dat 2	Dat 3	Dat 4	Dat 5	Dat 6	Dat 7	Dat 8	Dat 9	Dat 10	Dat 11	Dat 12	Dat 13	Dat 14	Dat 15	Dat 16	xor

El PLC en su bloque de acknowlege responde con los 16 bytes del bloque. El número de bloque se calcula como se especifico antes. Por ejemplos: para leer la dirección de la memoria usuario 342, se debe pedir comunicación con el bloque 21 (342/16).

ESCRITURA DE BLOQUE EN LA MEMORIA BAJA (comando 0A hex):

Permite escribir un bloque de 16 bytes de la memoria baja del PLC (de 0 a 0x2BF) y parte de la alta (0x9E0 a 0xFFF). Se le debe especificar para ello el , identificador del PLC, el comando ,el numero de bloque, y seguido los 16 bytes a escribir con el siguiente formato:

Bloque de petición:

ID	0B	Blo	Dat 1	Dat 2	Dat 3	Dat 4	Dat 5	Dat 6	Dat 7	Dat 8	Dat 9	Dat 10	Dat 11	Dat 12	Dat 13	Dat 14	Dat 15	Dat 16	xor

Bloque de acknowlege (respuesta):

ID	0B	Blo	AA	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	xor

El PLC en su bloque de acknowlege responde con un 0xAA indicando transferencia efectuada o con un 0xEE si no se pudo realizar.

LECTURA DE BLOQUE EN LA MEMORIA ALTA (comando 1B hex):

Permite leer un bloque de 16 bytes de la memoria alta del PLC (de 0x1000 a 0x1FFF), de la misma manera que como se especifico para leer bloques en la parte baja.

Bloque de petición:

ID	0B	Blo	AA	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	xor

Bloque de acknowlege (respuesta):

ID	0B	Blo	Dat 1	Dat 2	Dat 3	Dat 4	Dat 5	Dat 6	Dat 7	Dat 8	Dat 9	Dat 10	Dat 11	Dat 12	Dat 13	Dat 14	Dat 15	Dat 16	xor

ESCRITURA DE BLOQUE EN LA MEMORIA ALTA (comando 1A hex):
Permite escribir un bloque de 16 bytes de la memoria alta del PLC (de 0x1000 a 0x1FFF). Se le debe especificar para ello todo tal, cual se definió para el comando 0A (hex) con la diferencia de que los bloques se toman desde 0x1000.

Bloque de petición:

ID	0B	Blo	Dat 1	Dat 2	Dat 3	Dat 4	Dat 5	Dat 6	Dat 7	Dat 8	Dat 9	Dat 10	Dat 11	Dat 12	Dat 13	Dat 14	Dat 15	Dat 16	xor

Bloque de acknowlege (respuesta):

ID	0B	Blo	AA	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	xor

Comandos de acceso puntual

LECTURA BINARIA (comando 6):

Lee a lo sumo 8 valores de un byte en direcciones de la memoria usuario del PLC. Estas direcciones se le especifican al PLC en el bloque de interrogación, como parte baja y parte alta respectivamente. El PLC en su bloque de acknowlege responde, con los 8 valores de esos lugares memoria ubicados en los 2 bytes de los lugares correspondientes a donde se le indico su dirección.

Bloque de petición:

ID	6	Dir 1L	Dir 1H	Dir 2L	Dir 2H	Dir 3L	Dir 3H	Dir 4L	Dir 4H	Dir 5L	Dir 5H	Dir 6L	Dir 6H	Dir 7L	Dir 7H	Dir 8L	Dir 8H	0	xor

Bloque de acknowlege (respuesta):

ID	6	Dat 1	0	Dat 2	0	Dat 3	0	Dat 4	0	Dat 5	0	Dat 6	0	Dat 7	0	Dat 8	0	0	xor

Cuando en una comunicación se desea leer menos de 8 bytes se deben especificar las direcciones a las que se desea acceder y seguida de la ultima, enviar un 255 (ff en hex) en la parte alta de la dirección.

ESCRITURA BINARIA (comando 7):

Escribe a lo sumo 4 bytes en la memoria usuario del PLC. Se especifica dirección (parte alta y baja) y valor de cada byte. Por cada byte a escribir el PLC devuelve en su bloque de acknowlege un 170 (AA en Hex) en el lugar del dato si se realizo la transferencia o un 238 (EE en hex) si hubo un error.

Bloque de petición:

ID	7	Dir 1L	Dir 1H	Dat	x	Dir 2L	Dir 2H	Dat	x	Dir 3L	Dir 3H	Dat	x	Dir 4L	Dir 4H	Dat	x	0	xor

Bloque de acknowlege (respuesta):

ID	7	Dir 1L	Dir 1H	AA o EE	x	Dir 2L	Dir 2H	AA o EE	x	Dir 3L	Dir 3H	AA o EE	x	Dir 4L	Dir 4H	AA o EE	x	0	xor

Cuando se desee escribir menos de 4 byte lo que se debe hacer es especificar en la parte alta de la próxima dirección a la última un 255 (ff en hex).

LECTURA ANALÓGICA (comando 8):

Su uso es similar a como se describió en el comando 6, con la diferencia de que los datos son de 2 bytes.

Bloque de petición:

ID	8	Dir 1L	Dir 1H	Dir 2L	Dir 2H	Dir 3L	Dir 3H	Dir 4L	Dir 4H	Dir 5L	Dir 5H	Dir 6L	Dir 6H	Dir 7L	Dir 7H	Dir 8L	Dir 8H	0	xor

Bloque de acknowlege (respuesta):

ID	6	Dat 1L	Dat 1H	Dat 2L	Dat 2H	Dat 3L	Dat 3H	Dat 4L	Dat 4H	Dat 5L	Dat 5H	Dat 6L	Dat 6H	Dat 7L	Dat 7H	Dat 8L	Dat 8H	0	xor

Cuando en una comunicación se desea leer menos de 8 bytes se deben especificar las direcciones a las

ESCRITURA ANALÓGICA (comando 9):

Su uso es similar a lo que se especifico para el comando 7, con la diferencia de que los datos son de 2 bytes.

Bloque de petición:

ID	9	Dir 1L	Dir 1H	Dat 1L	Dat 1H	Dir 2L	Dir 2H	Dat 2L	Dat 2H	Dir 3L	Dir 3H	Dat 3L	Dat 3H	Dir 4L	Dir 4H	Dat 4L	Dat 4H	0	xor

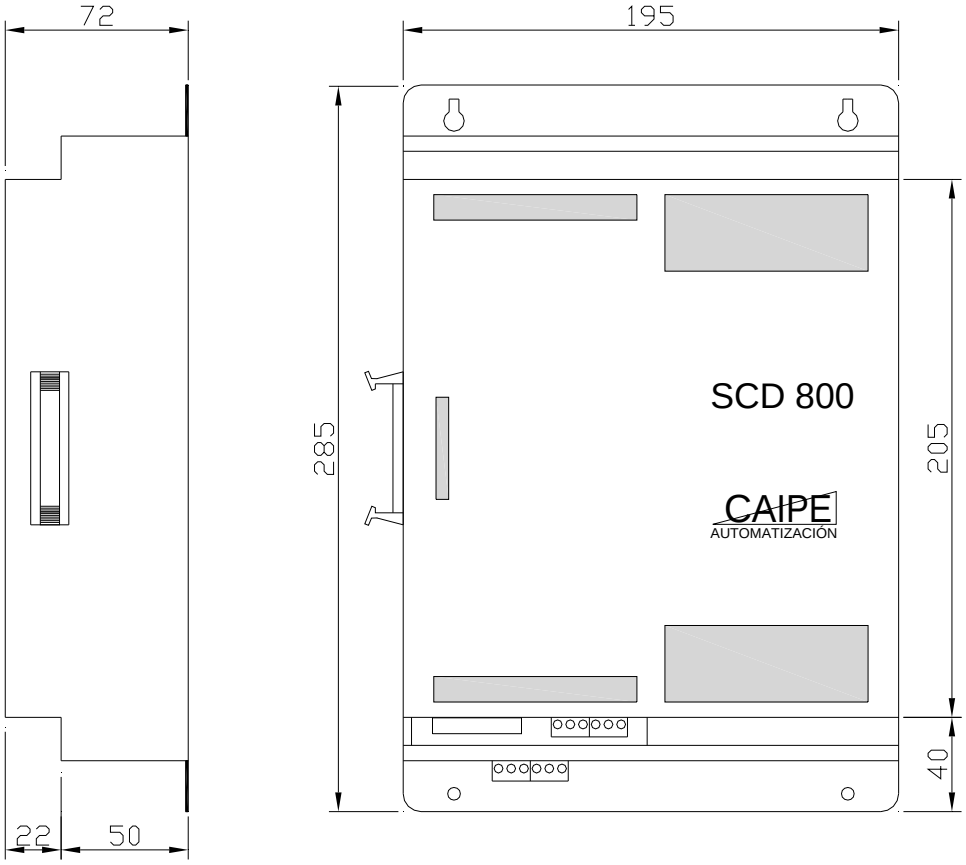
Bloque de acknowlege (respuesta):

ID	9	Dir 1L	Dir 1H	AA o EE	x	Dir 2L	Dir 2H	AA o EE	x	Dir 3L	Dir 3H	AA o EE	x	Dir 4L	Dir 4H	AA o EE	Dir 8H	0	xor

DIMENCIONES

Las medidas se encuentran en mm

CAIPE SCD 800



CAIPE SCD 80

GENERALIDADES

Cuando el usuario desarrolla un programa de aplicación, debe escribir el mismo mediante el teclado de la computadora (editar). La computadora debe traducir el lenguaje utilizado por el usuario al lenguaje binario que es el que realmente entiende el microprocesador del ordenador (compilar). Finalmente, debe ser enviada toda esta información (con el protocolo correspondiente) de la computadora al controlador (comunicar). Es importante también poder visualizar el estado instantáneo de las variables internas del controlador en ejecución (monitorear). Con el objeto de depurar y optimizar el programa.
El entorno integrado editor/compilador/comunicación llamado “CEDITOR”, permite cargar, editar, compilar, y enviar un programa para el SCD-Serie 80. Análogamente, el “EDIT90” permite cargar, editar, compilar y enviar un programa para el SCD-Serie 90.
Este entorno integrado para computadoras PC compatibles hace más sencillo el trabajo de programación y depuración de programas del controlador.
Por otra parte, el programa denominado “LADER” (accesorio de los anteriores), es un editor de diagramas de contacto tipo escalera que permite, como se verá más adelante, la edición gráfica de programas y/o subrutinas.

CARGA DEL EDITOR - CONEXIONES

Los programas de edición y compilación para los controladores marca Caipe® SCD-Series 80 y 90, llamados “CEDITOR.EXE” (de “compilador – editor”), y “EDIT.EXE” (de “editor de la serie 90”) respectivamente, deben ir acompañados en el mismo subdirectorio desde donde se los haga correr de “CEDIT80.HLP” y “CEDIT90.HPL” respectivamente; estos archivos con extención .HPL proporcionan ayudas accesibles durante la operación de los programas.
La conección hacia/desde la serie 80 se hace mediante el pórtico RS232 de la computadora: se conmuta la conección hacia cada uno de los dos pórticos del PLC (programación/lectura de un programa por un lado, y monitoreo/envío de datos por otro) en forma automática, de acuerdo a la función de comunicación a la que se invoque. La conección hacia la serie 90 se lleva a cabo desde un pórtico paralelo (impresora) de la computadora. Explicaremos el uso del editor basándonos en la serie 80, dado que las diferencias con la serie 90 pueden deducirse valiéndose de las funciones de ayuda del EDIT90.EXE.

Estando la computadora en el sistema operativo (MS-DOS, DR-DOS ó PC-DOS) y suponiendo que introducimos el diskette del editor en el drive “A”, se debe ingresar el siguiente comando:

A:\> CEDITOR

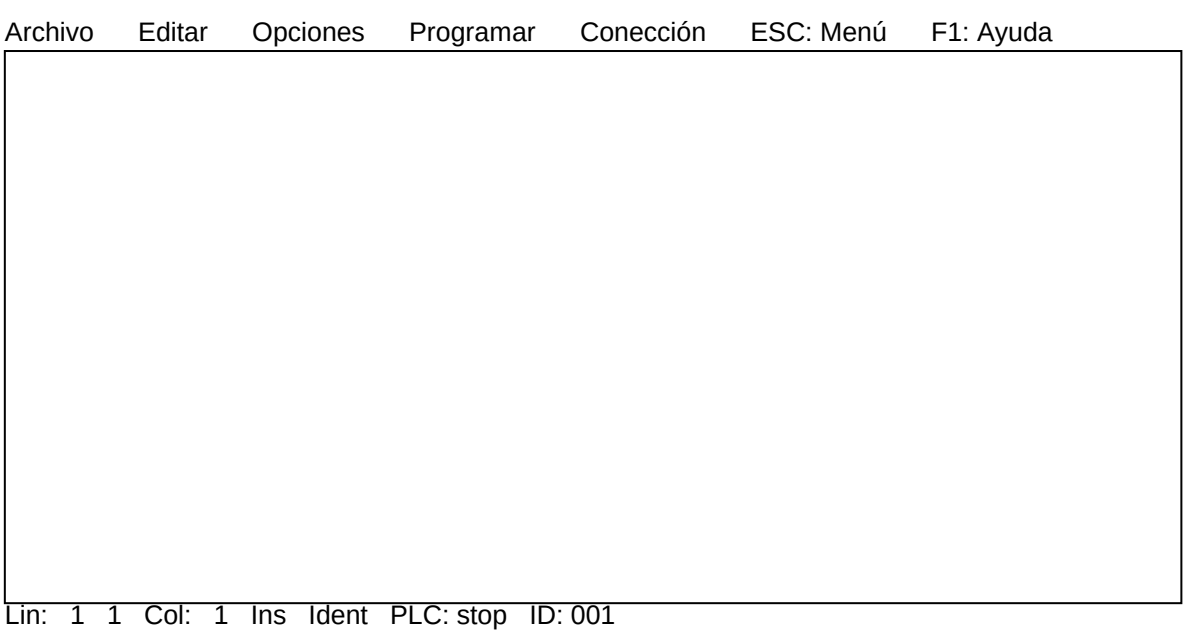
y luego pulsar la tecla <ENTER>.
Después de unos instantes de inicialización, el sistema presentará el título principal
Se debe pulsar una tecla para continuar. El sistema se encuentra ya en condiciones de trabajar. Para mejorar la interpretación de este capítulo sugerimos leer los items siguientes, teniendo el editor en operación en la computadora.

El cursor

Lo primero que se destaca al activar el editor, es un carácter que titila. Este carácter se denomina “cursor”, e indica la posición en donde se imprimirá el siguiente carácter que se pulse.

EL EDITOR - PROGRAMA PRINCIPAL

El programa principal activo en pantalla es el EDITOR. Se visualizará lo siguiente:



En la parte superior de la pantalla, vemos el menú de las distintas opciones principales accesibles desde eleditor. Cada opción tiene asociado un submenú que se activa cuando el usuario pulsa la tecla <ESC> (como muestra la indicación de ayuda ESC: Menú) y, usando las flechas de movimiento del cursor hacia los costados, selecciona el submenú deseado. Dentro del submenú elegido, se puede seleccionar una opción mediante el uso de las teclas de movimiento del cursor hacia arriba o hacia abajo, pulsando luego la tecla <ENTER>. Pueden también elegirse estas opciones si, aún sin haber accedido antes a un submenú se presiona/n la/s tecla/s indicada/s en los submenúes. Asi mismo, para activar un submanú puede pulsarse a la vez las tecla <ALT> y la inicial del submenú; por ejemplo, para acceder al submenú ARCHIVO se deben pulsar al unísono <ALT> y <A>. (Excepción: para acceder a CONECCIÓN pulsar <ALT> + <X>). Ya se verá como usar la ayuda que, como se indica a la derecha proporciona F1.
En la pare inferior de la pantalla del editor se muestra la “barra de estado” horizontal, que permite visualizar distintas señales del estado del editor (y del controlador), a saber:

a) Lin: Col:

Indican la línea (del listado completo y parcial de la pantalla actual) y la columna en donde se encuentra el cursor.

b) Ins Ident

Si aparecen estas indicaciones, señalan que el editor está en “modo inserción” y en “modo indentar”. Más adelante explicaremos el significado de estos modos.

c) (NOMBRE) .PRG

Muestra, a la derecha de la barra de estado, el nombre del archivo en edición. Si no hay ninguno seleccionado, no se muestra nombre alguno.

d) PLC: run (stop)

Muestra el estado en que se encuentra el controlador (PLC). Este estado lo deduce la computadora de acuerdo al estado de la interfase RS232C. Si el controlador está desconectado (OFF LINE), puede darse el caso de que esta indicación no coincida con la realidad. Cuando se indica “run”, significa que el controlador está en ejecución, mientras que se se indica “stop” este se encuentra detenido.

e) ID: 00 nº

Indica el número de identificación del controlador que recibe el programa dentro de una red.

Ayuda proporcionada por F1

F1, acompañada de otras teclas, permite acceder al sistema de ayuda activa que tiene incorporado el editor, y permite consultar, en cualquier momento, distintos aspectos del editor mismo, del lenguaje del controlador (instrucciones), y diversas operaciones del sistema:

a) Pulsando <F1>, se visualiza la lista de comandos del editor: movimientos del cursor, controles de bloques, movimientos de página, etc.

b) Oprimiendo la combinación <CTRL> + <F1>, estando ubicado el cursor en la línea de una instrucción del programa en edición, se visualiza la explicación de dicha instrucción.

c) Pulsando la combinación <ALT> + <F1>, se accede a un menú que permite optar (utilizando las teclas de control del cursor y seleccionando con <ENTER>) entre tres listados de ayuda, a saber:

1. Listado de comandos (instrucciones).

2. “Mapa” de variables.

3. Detalle del tramo de dicho mapa correspondiente a acumuladores y constantes.

* Se sale del sistema de ayuda pulsando cualquier tecla.

EDICIÓN DE PROGRAMAS

Para editar un programa, se escriben las sucesivas instrucciones del mismo desde el teclado, dejando un espacio entre la instrucción propiamente dicha y la primera variable (si la hay), y entre variables (si hay más de una). Además, debe dejarse por lo menos un espacio a la izquierda de cada instrucción, debido a que todo lo que se encuentre en el extremo izquierdo será considerado como etiqueta (label).
Si se desea editar comentarios después de la instrucción, se los escribe directamente después de esta dejando un espacio (o más).
Para editar títulos o comentarios en un renglón que no incluya instrucciones, deben comenzarse los mismos con el símbolo “punto y coma” (;) o, solo para las primeras cinco líneas con “punto” (.), para que se repitan en cada hoja del listado.

Antes de teclear la primera instrucción del programa (si la misma no tiene etiqueta), es conveniente pulsar la tecla <TAB> para obtener la primera tabulación; si luego de escrita la línea se pulsa <ENTER>, el sistema pasa al próximo renglón y, si se está en modo INDENTAR (se verá mejor más adelante), el mismo se tabula automáticamente.

Si se desea editar la etiqueta correspondiente a un salto, estando el cursor en la línea elegida y en la posición definida por el tabulador;, se pulsa <HOME> (o bienla combinación <SHIFT> + <TAB>), con lo cual el cursor se correrá hacia el extremo izquierdo permitiendo escribir la etiqueta. Dicha etiqueta puede tener la cantidad de caracteres alfanuméricos que se desee, con la condición de comenzar con un alfabético y no contener espacios (el sistema interpreta, al aparecer un espacio, que lo que sigue es una instrucción).

En cualquier momento de la edición se puede acceder a todas las opciones de operación mediante los comandos aquí explicitados.

COMANDOS DEL EDITOR

Los comandos asociados al editor (se pueden ver en la pantalla oprimientdo F1) son:

a- Movimiento del Cursor.

Mover Cursor Texto	= Teclas de control del cursor
Tope de Ventana	= CTRL + T
Borde de Ventana	= CTRL + B
Siguiente Palabra	= CTRL + → (mov. de cursor a la derecha)
Palabra Anterior	= CTRL + ← (mov. de cursor a la izquierda)
Comienzo de Línea	= HOME
Fin de Línea	= END
Retroceder Tabulador	= SHIFT + TAB
Línea Arriba	= CTRL + W
Línea Abajo	= CTRL + X

b- Movimiento por Página.

Comienzo de Archivo	= CTRL + HOME
Fin de Archivo	= CTRL + END
Página Anterior	= PG UP
Página Siguiente	= PG DN

c- Comandos del Editor.

Salir del Editor	= ALT + S
Inserta/Reescribe	= INS
Borrar Carácter	= DEL
Borrar Carácter	= BACKSPACE
Borrar Palabra (desde la posición del cursor en adelante)	= CTRL + D
Borrar Línea	= ALT + D
Busca una Cadena de Caracteres	= F7
Sigue Buscando	= ALT + F7
Indentar Si/No	= ALT + I
Configurar PLC	= ALT + F
Ayuda Instrucciones PLC	= CTRL + F1
Mapa de Memoria y otros	= Alt + F1

OPCIONES DEL PROGRAMA PRINCIPAL

Ya se ha visto que se puede acceder a las opciones o submenúes del programa principal de dos maneras: a través de la tecla <ESC> y el uso de las teclas de movimiento del cursor, o pulsando la convinación simultánea de las teclas asociadas a la opción o submenú.

ARCHIVOS: <ALT> + <F4>

En este submenú se encuentran las posibilidades de cargar o guardar programas, listarlos por impresora, mezclar archivos, etc. Según el siguiente detalle:

CARGA: <ALT> + <C>

Transfiere el archivo que se indique al editor en uso, reemplazando al archivo anterior.

GRABA: <ALT> + <C>

Transfiere lo cargado en el editor al archivo que se indique.

MEZCLA: <ALT> + <M>

Mezcla en el buffer del editor en uso, a partir de la línea a la cual apunta el cursor, el archivo que se indique. Se toma la configuración de módulos del archivo a mezclar.

NUEVO: <ALT> + <N>

Borra el contenído del buffer en edición, y lo abre nuevamente para editar.

LISTAR: <ALT> + <T>

Lista por impresora el buffer de edición, incluyendo los errores del programa (si los hay).

SALIR: <ALT> + <S>

Sale del editor, volviendo al sistema operativo de computadora.

NOTA: Cuando se opta por los comandos de carga o mezcla, se visualiza en “nombre” la leyenda: “*.PRG”. Pulsando <ENTER>, el editor le mostrará una lista de los archivos de programas disponibles en el directorio. Con los cursores puede recorrerse la lista; para acerlo rápidamente, puede usar <Pg Up>, <Pg Dn>, <HOME> o <END>.
Pulsando nuevamente <ENTER> se selecciona.

EDITOR: <ALT> + <E>

Este submenú de opciones permite trabajar cor bloques de programa (creación, borrado, movimiento, búsqueda, etc, de los mismos), para facilitar la edición de programas.
Los comandos asociados son los siguientes:

MARCA COMIENZO: <F5>

Marca el comienzo de un bloque a determinar, para moverlo, copiarlo, borrarlo, o formar párrafo.

MARCA FINAL <F6>

Marca el final de un bloque a determinar, para moverlo, copiarlo, borrarlo, o formar párrafo.

MUEVE: <F3>

Inserta un bloque determinado just antes de la línea en donde se encuentra el cursor, eliminando al bloque del lugar donde se encontraba inicialmente.

COPIAR: <F4>

Repite un bloque determinado, insertándolo justo antes de la línea donde se encuentra el cursor.

BORRAR: <F8>

Borra un bloque determinado ocupando el espacio que ocupaba el mismo anteriormente.

DESMARCAR: <F9>

Cancela las marcas que definen un bloque.

BUSCA: <F7>

Busca un texto especificado (instrucción o comentario) en el buffer del editor. Una vez encontrado el mismo, mueve el cursor a la posición donde se lo halló. Se recomienda llevar el cursor al comienzo del buffer del editor antes de buscar, pues el sistema busca siempre hacia delante.
Puede especificarse que en el texto a buscar, deba tenerse en cuenta, o no, las características de redacción de dicho texto (mayúsculas y minúsculas); siempre son tenidos en cuenta los espacios intermedios y los anteriores al texto.

SIGUE BUSCANDO: <ALT> + <F7>

Luego de utilizar el comando BUSCA (<F7>), localiza la siguiente vez en que aparece el texto en cuestión.

IMPRIME BLOQUE: <ALT> + <K>

Lista por impresora un bloque determinado del programa.

OPCIONES: <ALT> + <O>

Este submenú agrupa diversas opciones y modos del editor, a saber:

CONFIGURAR TARJETAS PLC: <ALT> + <F>

Introduce en el archivo en edición, la configuración real del controlador que ejecutará el programa asociado. Como ya se indicó, es necesario introducir esta información durante la edición del programa. Con las teclas del cursorpuede seleccionarse la ubicación del módulo a ingresar, y con la tecla <RETURN> seleccionar el tipo de módulo.

TIPO DE MÓDULO	
4 Entradas de termocupla	1
8 Salidas y 4 entradas discretas	2
8 Entradas analógicas de 10 bit	3
1 Salida analógica de 11 bit	4
4 Entradas anaológicas y 8 discretas	5
4 Entradas analógicas de 10 bit/termo	6
16 Entradas discretas	7
12 Salidas discretas	8
2 Termocuplas, 2 analógicas y 4 discretas	9
Celda de carga	10
Contaje rápido tipo uno	11
Entrada analógica 12 bit	12
Doble salida analógica	13
Balanza	14
Indefinido	0

INSERTAR: <INS>

Enciende o apaga el modo de inserción de caracteres. Este modo, cuando está activo, hace que al teclear un nuevo caracter, los caracteres que se encuentran a la dercha del cursor se desplacen un lugar, para dar lugar al nuevo caracter. Cuando se desactiva el modo, los caracteres ingresados se escriben sobre los ya existentes (si los hubiese).

INDENTAR: <ALT> + <I>

Enciende o apaga el modo de indentación (tabulación automática). Estando este modo activado, cuando se pulsa <ENTER>, en vez de ir al inicio del siguiente renglón, se ejecuta una tabulación automática para

respetar el espacio requerido por el lenguaje del controlador. Cuando se encuentra activado este modo y se quiere ingresar una etiqueta de salto, pulsar luego de <ENTER> la tecla <HOME>, o bien la combinación <SHIFT> + <TAB>.

LÍNEAS DE RETORNO: <F10> Y <SHIFT> + <F10>

Merecen un párrafo propio, se explican más adelante.

PROGRAMAR: <ALT> + <P>

Este submenú de acceso a las distintas opciones de traducción de lenguajes en la programación (compilar, descompilar, y visualizar errores) a saber:

COMPILAR: <ALT> + <L>

Traduce el programa cargado en el buffer de edición a código máquina del controlador (código objeto). Se puede grabar el programa objeto, enviar el mismo al controlador, y/o que el controlador lo ejecute en forma automática, luego de compilar.
Puede también generarse una lista tipo *cross-reference* de variables y números de línea en que se encuentran las mismas.
Si se quiere activar o deshacer alguna de estas opciones, se deberá pulsar:

<1> Para no grabar el archivo en binario.
<2> Para solamente generar el archivo binario.
<3> Para no ejecutar el programa.
<4> Para grabar el archivo *cross-reference*.

Pulsando otra vez <1>, <2> y <3>, se activan nuevamente las opciones.

NOTA: Los programas en código de máquina, al grabarse en disco, toman automáticamente la extensión ".BIN", a diferencia de los programas fuente que toman la extensión ".PRG".

DESCOMPILAR: <ALT> + <Q>

Retraduce al lenguaje de edición del controlador un programa en código máquina para su verificación. El programa retraducido se carga en el buffer de edición.

ERRORES: <CTRL> + <E>

Si durante la compilación se detectaron errores de edición, se visualizan los mismos en la parte inferior de la pantalla.
Una vez posicionado en un determinado error con el cursor, pulsando <RETURN> se vuelve al editor en la línea de error, para corregirlo luego de pulsar <ESC>. Para pasar a analizar otro error (si lo hay) debe pulsarse <CRTL> <E>.

CONEXIÓN: <ALT> + <X>

Este submenú maneja las opciones relacionadas con la comunicción entre la computadora y el controlador, a saber:

PÓRTICO RS 232C (sin tecla asignada)

Permite conmutar el canal de comunicación RS 232C hacia el controlador entre el pórtico 1 y 2.
Pulsando <ENTER> se conmuta entere el pórtico 1 ó 2, indicándose COM1 ó COM2 respectivamente a la derecha de la opción.

NÚMERO ID DEL PLC: <F2>

Permite ingresar un número del 1 al 99, para identificar a un PLC dentro de una red de hasta 99 controladores. Si no se ingresa ningún número, el EDITOR incorpora automáticamente el Nº 1.

ENVIAR AL PLC: <ALT> + <V>

Envía el programa en código máquina hacia el controlador.

RECIBIR DEL PLC: <ALT> + <B>

Pide el programa al controlador y lo coloca en memoria para su posterior análisis (grabar a un archivo .BIN y/o descompilar).

CORRER PROGRAMA: <CRTL> + <R>

Permite conmutar la modalidad del controlador, entre *ejecución del programa* o *detenido*.
El indicador “PLC” de la línea de estado del editor, muestra si el controlador está ejecutando el programa en su memoria o no.

NOTA: Al estar el controlador en EJECUCIÓN esta se detendrá si activamos la opción ENVIAR AL PLC o RECIBIR DEL PLC.

MONITOR: <ALT> + <F10>

Esta opción permite visualizar por pantalla (a través del canal de comunicación de red) el estado instantáneo de las variables internas del controlador, mientras el mismo se encuentra en ejecución. Permite también imponerle al controlador algunos valores del mapa de variables desde la computadora (se excluyen las variables ligadas a entradas y salidas físicas).

Al activar el comando que corresponde a esta opción se reconfigura la pantalla del editor, apareciendo dos nuevas barras horizontales superior e inferior y dos bloques (A y B) de direcciones de memoria. Si no es la primera vez que se activa el comando los bloques serán los últimos que se hayan usado.
Como canal de comunicación de red permite el envío de “paquetes” de información de a 16 direcciones de memoria (16 bytes) podemos ver al unísono dos bloques distintos (consecutivos o no) de los 36 posibles. Según sea la posición del cursor (bloque A o B) y a través de las teclas <Page Down> y <Page Up> se van colocando en pantalla los bloques a visuaizar; puede también, seleccionarse el bloque , mediante la función F6 (go to) seguida de una de las direcciones deseadas.

Cada región de bloque indica: la posición de memoria, el contenido de la misma en decimal y en código hexadesimal, tomado como variable de 8 bit, y además el contenido de esa posición en decimal, como parte baja de una variable de 16 bit cuya parte alta es la posición siguiente. También se muestra el estado del bit 0 de esa posición de memoria.
Permite ademas visualizar 8 bloques adicionaleas para monitorear el estado de cuenta instantáneo de los 64 timers/contadores (8 por bloque).
La barra horizontal superior indica además del título de esa opción (*MONITOR*), la forma de desplazar el cursor entre bloques y dentro de cada bloque.
A través del comando <ENTER> se establece la característica de la comunicación con el controlador, a saber:

a) Recepción continúa con el controlador en ejecución.
b) Una única recepción, mientras el controlador sigue en ejecución.
c) Enviar estado de variables al controlador. La edición de estas variables a enviar se hace a través de:

F2: decimal de 8 bit
F3: hexa en 8 bit
F4: decimal en 16 bit

F5: bit 0 del byte

Una vez elegida la característica de la comunicación, con la tecla <ENTER> se establece la misma.

Si, en modo de recepción, se colocan en ambos bloques las mismas direcciones de memoria, se efectuará una única recepción por vez, acelerando de esta forma el proceso.

Con la tecla <HOME> se inicializan o recetan los valores en pantalla.

NOTA: Para establecer el canal de comunicación de red y utilizar potencialmente esta opción del submenú CONEXIÓN, debe conectarse el pórtico serie de la computadora al canal de comunicación de red de la CPU del controlador, desconectándolo del canal de programación.

BAUDIOS PROG:

Es opción permite seleccionar la velocidad de carga y lectura de programas. El valor a adoptar depende del modelo de CPU, identificándose el mismo de la siguiente forma: Si su cable de programación posee una caja con una llave inversora, debe establecer 14400 baudios; de no existir tal caja, setear 19200 baudios.

BAUDIOS MONITOR:

Esta opción permite seleccionar la velocidad de monitoreo de variables. Actualmente 4800 baudios y equipos de radio 2400 baudios.

SEÑALIZACION DE ZONAS DE PROGRAMA

Supongamos que, durante la edición de un programa, estamos agregando líneas en una zona intermedia del listado: sea que, en un momento dado, queremos consultar otra zona. Bastará pulsar <F10> para que la zona de la que uno se irá momentáneamente quede, invisiblemente, señalizada, pudiéndose retornar a ella desde cualquier otro lugar del listado si se pulsa <SHIF> + <F10>.

Al señalizar otra zona, quedan registradas las marcas anteriores, pudiéndose volver a ellas pulsando <SHIFT> + <F10> tantas veces como sea necesario. Pueden memorizarse hasta 10 marcas.

GENERALIDADES DE LAS INSTRUCCIONES:

Clasificamos el tipo de instrucciones disponibles en cuatro grupos fundamentales:

INSTRUCCIONES LÓGICAS:
Se utilizan fundamentalmente para resolver circuitos por lógica funcional de contactos.

INSTRUCCIONES ARITMÉTICAS:
Realizan en forma directa operaciones aritméticas simples y de comparación.

INSTRUCCIONES DE MANEJO Y MOVIMIENTO DE VARIABLES:
Permiten optimizar el programa del usuario, mediante el movimiento, transformación y conversión de variables de memoria.

INSTRUCCIONES DE MANEJO DE PERIFÉRICOS:
Permiten la entrada y salida de datos desde y hacia periféricos.

NOMENCLATURA A UTILIZAR:

Para poder especificar los distintos tipos de instrucciones y sus ejemplos, utilizaremos la siguiente nomenclatura:

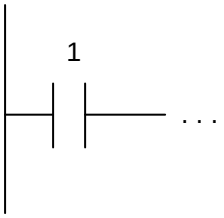
NOMENCLATURA	ESPECIFICACIÓN
v	Lugar o dirección de memoria (usaremos este ultimo termino) de 8 BIT, correspondiente a E/S, variable interna, etc.
(v)	Contenido de la dirección x.
n	Constante de 8 BIT.
c	Constante de 16 BIT.
i	Dirección de entrada.
o	Dirección de salida.
p	Dirección hacia la que se realiza un salto (etiqueta o LABEL).
Acc	Acumulador principal.
t	Tiempo en segundos.
B	BIT unidad de memoria.
< >	Campo de direcciones posibles.
((v+1) (v))	Numero de 16 BIT que tiene, en v, su parte baja (PB), y en v+1 su parte alta (PA).
(v)->Acc	Indica que el contenido de la dirección x se carga en el acumulador principal.

INSTRUCCIONES LÓGICAS:

Carga de acumulador LDv

Carga el acumulador principal Acc con el contenido de la dirección de memoria v. Esa dirección v puede pertenecer al área de entradas / salidas discretas o analógicas, al área de variables internas, o al estado de timers o contadores. En un diagrama funcional de contactos, cada línea o escalón con un primer contacto normal abierto, comienza con la instrucción LD.

EJEMPLO NEMÓNICO




```
LD 1        LD v       v <0…575>
```

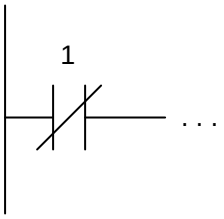
Carga el acumulador con el bit 0 de cada una de las variables desde v hasta (v+7), en orden creciente.

	b7	b6	b5	b4	b3	b2	b1	b0		Acc	
V	X	X	X	X	X	X	X	0	———	0	b0
V+1	X	X	X	X	X	X	X	1	———	1	b1
V+2	X	X	X	X	X	X	X	1	———	1	b2
V+3	X	X	X	X	X	X	X	1	———	1	b3
V+4	X	X	X	X	X	X	X	0	———	0	b4
V+5	X	X	X	X	X	X	X	0	———	0	b5
V+6	X	X	X	X	X	X	X	1	———	1	b6
V+7	X	X	X	X	X	X	X	0	———	0	b7

Carga de acumulador con negación LDN v

Carga el acumulador principal Acc con el contenido negado de la dirección de memoria v. Dicho contenido no es afectado. En un diagrama funcional de contactos, cada línea o escalón con un primer contacto normal cerrado comienza con esta instrucción.

EJEMPLO NEMÓNICO



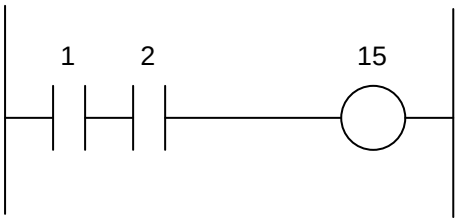

```
LDN 1       LD x       x <0…575>

(v)                        Acc
```

Salida OUT v

Descarga el contenido del acumulador principal Acc en la dirección de memoria v. Esa dirección v puede pertenecer al área de entradas / salidas o al de variables internas. El contenido del acumulador no es afectado.

EJEMPLO



NEMÓNICO

```
LD 1
AND 2        OUT v v  <0..506>
OUT 15
```

Acc (v)

Negación NOT

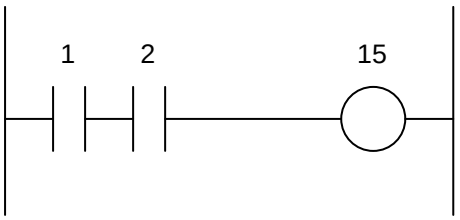
Niega el estado del acumular.

Acc _____ Acc

Operación lógica "Y" AND v

Operación lógica "Y" entre el contenido del acumulador principal Acc y el contenido de la dirección v. El resultado queda en el acumulador. El contenido de v no es afectado. En un diagrama funcional de contactos esta instrucción coloca un contacto normal abierto en serie con lo definido anteriormente.

EJEMPLO



NEMÓNICO

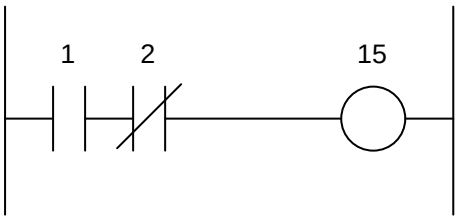
```
LD 1
AND 2        AND v v  <0..575>
OUT 15
```

(v) "AND" Acc_____Acc

Operación lógica "Y" con variable negada ANDN v

Operación lógica "Y" entre el contenido del acumulador principal Acc y el contenido negado de la dirección v. el resultado queda en el acumulador. El contenido de v no es afectado. En un diagrama funcional de contactos, esta instrucción coloca un contacto normal cerrado en serie con lo definido anteriormente.

EJEMPLO



NEMÓNICO

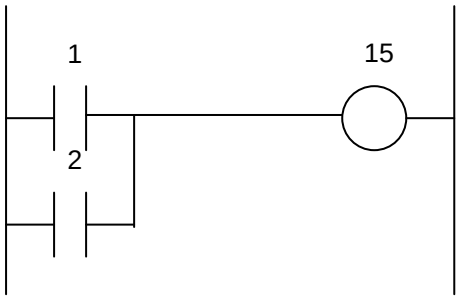
```
LD 1
ANDN 2       ANDN v    v  <0..575>
OUT 15
```

(v) "AND" Acc_____Acc

Operación lógica "O" OR v

Operación lógica "O" entre el contenido del acumulador principal Acc y el contenido de la dirección v. El resultado queda en el acumulador. El contenido de v no es afectado. En un diagrama funcional de contactos, esta instrucción coloca un contacto normal abierto en paralelo con lo definido anteriormente.

EJEMPLO



NEMÓNICO

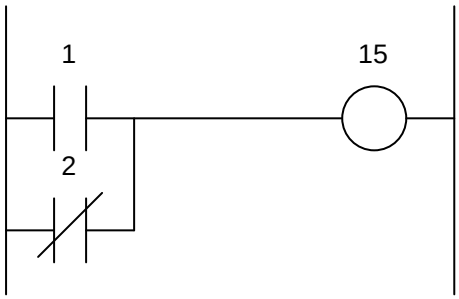
```
LD 1
OR 2          OR v    v  <0..575>
OUT 15
```

(v) "OR" Acc _____ Acc

Operación lógica "O" con variable negada ORN v

Operación lógica "O" entre el contenido del acumulador principal Acc y el contenido negado de la dirección v. El resultado queda en el acumulador. El contenido de v no es afectado. En un diagrama funcional de contactos, esta instrucción coloca un contacto normal cerrado en paralelo con lo definido anteriormente.

EJEMPLO



NEMÓNICO

```
LD 1
ORN 2         ORN v    v  <0..575>
OUT 15
```

(v) "OR" Acc _____ Acc

Operación lógica "O exclusiva" XOR v

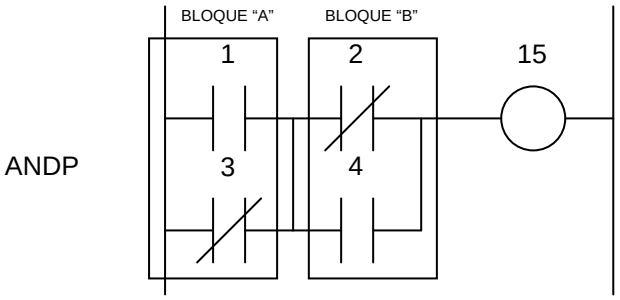
Operación lógica "O exclusiva" entre el contenido del acumulador principal Acc y el contenido de la variable v. El resultado queda en el acumulador. El contenido de la dirección v no es afectado.

(v) "XOR" Acc _____Acc

Operación lógica "Y" entre acumuladores (bloques) ANDP

Operación lógica "Y" entre el contenido del acumulador principal Acc y el secundario Acc1. El resultado queda en el acumulador principal Acc. En un diagrama funcional de contactos, esta instrucción hace la operación lógica "Y" (AND) entre bloques se una misma línea o escala. Cada bloque se define en forma independiente y luego, con esta instrucción, se realiza la "serie" lógica.

EJEMPLO



NEMÓNICO

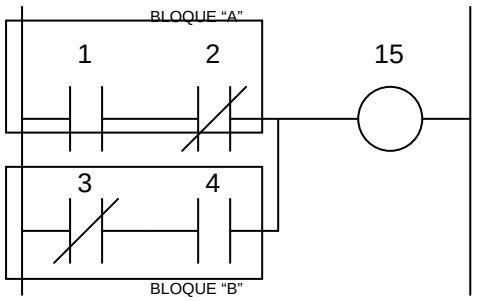
```
LD 1
ORN 3
LDN 2
OR 4
ANDP
OUT 15
```

Acc "AND" Acc1 (anterior) ——— Acc

Operación lógica "O" entre acumuladores (bloques) ORP

Operación lógica "O" entre el contenido del acumulador principal Acc y el secundario Acc1. El resultado queda en el acumulador principal Acc. En un diagrama funcional de contactos, esta instrucción hace la operación lógica "O" (OR) entre bloques de un mismo escalón. Cada bloque se define en forma independiente y luego, con esta instrucción, se realiza el "paralelo" lógico.

EJEMPLO



NEMÓNICO

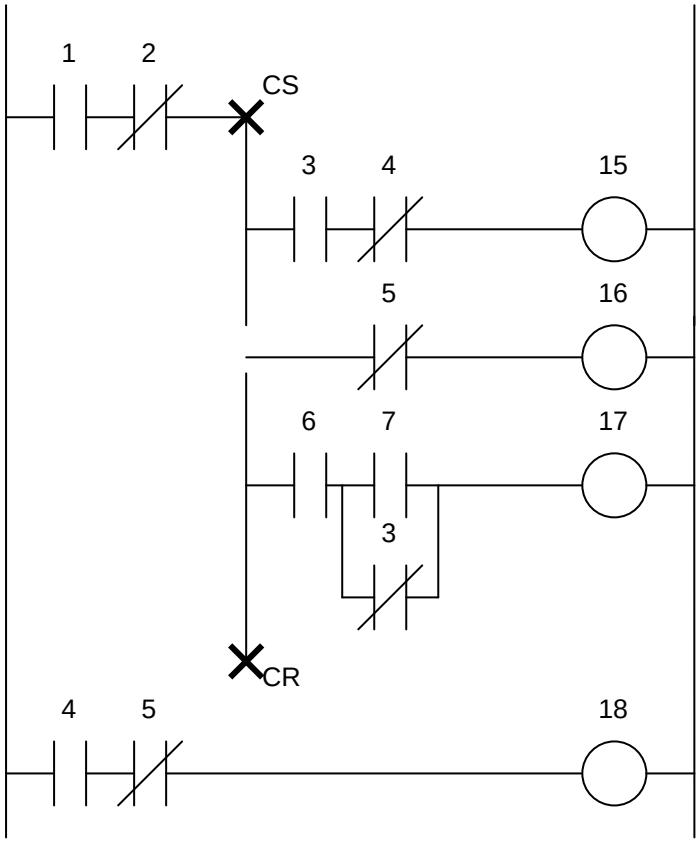
```
LD 1
ANDN 2
LDN 3          ORP
AND 4
ORP
OUT 15
```

Acc "OR" Acc1 (anterior) ——— Acc

Control maestro (control set) CS

Esta instrucción permite definir un bloque o condición principal y a partir del CS, colocarla en serie con una cantidad de ramas en paralelo (hasta encontrar un control reset CR).

EJEMPLO



NEMÓNICO

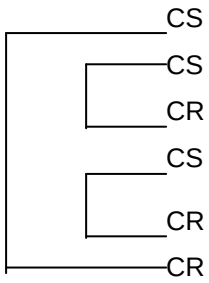
```
LD 1
ANDN 2
CS
LD 3
ANDN 4
OUT 15
LDN 5
OUT 16
LD 6
LD 7
ORN 3
ANDP
OUT 17
CR
LD 4
ANDN 5
OUT 18
```

Las instrucciones afectadas por el Control Maestro son: LD, LDN, OR, ORN, CP, CPI y CPV.

Fin del control maestro (control reset) CR

Desactiva la condición de CONTROL MAESTRO (control SET). Los CS y CR pueden anidarse, debiendo ser la cantidad de CS igual a la de CR. La cantidad máxima de CS/CR que se pueden anidar es de 4.

EJEMPLO



NEMÓNICO

```
CR
```

Fin de programa END

Debe colocarse en los bloques finales del programa, obligando a este a recomenzar.

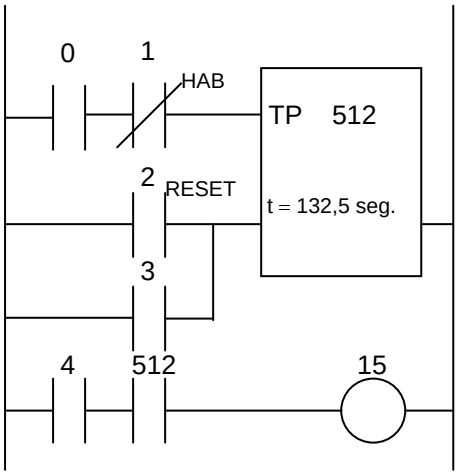
NEMÓNICO

```
END
```

Temporizador TP v t

Si bien esta instrucción no realiza una operación lógica, la incluiremos dentro de este grupo por ser de gran uso de los diagramas funcionales de contactos. El SCD-Serie 80 incluye la posibilidad de programar hasta 64 temporizadores, dependiendo del numero que se le asigne a v, que sea con o sin retención de estado ante un corte de suministro de alimentación; pueden programarse con retardos de 0,1 seg. a 999,9 seg.
La instrucción necesita de dos cargas previas en los acumuladores: un comando de habilitación y uno de reset.
Cuando la lógica o condición de habilitación se pone en “1”, el temporizador se pone en funcionamiento hasta que se cumple un tiempo asignado, o bien, se active la lógica o condición de reset, que siempre tiene prioridad. Si se desactiva la condición de habilitación antes de cumplirse el tiempo asignado, el temporizador entre en espera, hasta que se active nuevamente la habilitación y complete su tiempo.
Cuando el tiempo se cumple, en la dirección asociada se ubica un “1” (en el BIT 0) y permanece hasta que se desactive la habilitación o se active la condición de reset., permaneciendo en “0” en caso contrario.

EJEMPLO



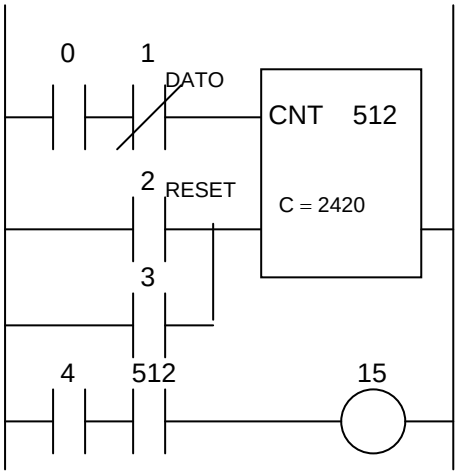
NEMÓNICO

```
LD 0       TP v t  v  <512…575> ver nota
ANDN 1             t <0,1...999,9 seg. >
LD 2
OR 3
TP 512 132.5        Nota:
LD 4                512 a 551: sin retención
AND 512             552 a 575: con retención
OUT 15
```

Contador CNT v c

Incluimos esta instrucción dentro de este grupo, por ser, también, muy utilizada en los diagramas funcionales de contacto. El SCD-Serie 80 incluye la posibilidad de programar hasta 664 contadores, dependiendo del numero que se le asigne a v, que sea con o sin retención de estado, ante un corte del suministro de alimentación. Los contadores queden programarse con predeterminaciones de 1 a 9999.
La instrucción necesita de dos cargas previas en los acumuladores: un comando de entrada de datos (se cuentan los flancos ascendentes de los pulsos), y un comando de reset.
Cuando la cantidad de datos contados coincide con el valor predeterminado, en la dirección asociada al contador se ubica un "1" (en el BIT 0), y permanece hasta que se active la condición de reset. Se pone además el acumulador principal en "1", habiendo permanecido en "0" hasta ese momento.
El valor de cuenta se sigue incrementando ante la aparición de más pulsos en la entrada de datos (igual que un contacto mecánico), y solo cambia el estado por la aparición de la condición de reset.
El valor instantáneo de cuenta puede obtenerse en cualquier momento (ver la instrucción OUTC).

EJEMPLO



NEMÓNICO

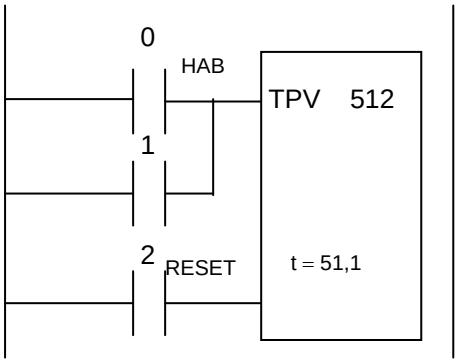
```
LD 0       TP v c  v  <512…575> ver nota
ANDN 1           c  <0...9999>
LD 2
OR 3
CNT 512 2420     Nota:
LD 4             512 a 551: sin retención
AND 512          552 a 575: con retención
OUT 15
```

Temporizador variable TPV v y

Es similar al temporizador TP, pero la predeterminación de tiempo se encuentra en las direcciones de memoria y (parte baja), e (y+1) (parte alta). Como la base de tiempo de los temporizadores es de 0,1 seg., al número decimal que se desee introducir como predeterminación de tiempo, se lo debe multiplicar por 10, y ubicarlo en y e (y+1).

O sea, sí: t = 51,1 seg. en y —— 511
t = 328,7 seg. en y —— 3287
t = 40 seg. en y —— 400

EJEMPLO



200	1 1 1 1 1 1 1 1	PB
201	0 0 0 0 0 0 0 0	PA

511 en binario

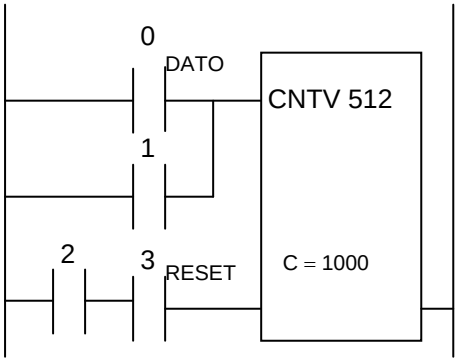
NEMÓNICO

```
LD 0      TPV x y      x <512...575>
OR 1                   y <0...510>
LD 2
TPV 512 200
```

Contador variable CNTV v y

Es similar al contador CNT, pero el valor de comparación lo toma de las variables (v) (parte baja) y (v+1) (parte alta).

EJEMPLO




```
LD 0
OR 1
LD 2
AND 3
CNTV 512 260
```

NEMÓNICO

```
CNTV x y    x <512...575>
            y <0…510>
```

260	1 1 1 1 1 1 1 1	PB	1000 en binario
261	0 0 0 0 0 0 0 0	PA	

INSTRUCCIONES ARITMÉTICAS

Suma ADD v y

Suma el contenido de dos direcciones de memoria (variables internas) v e y, de 16 BIT cada una (número máximo de cada sumando: 65535).
Al elegir v e y se debe cuidar de no elegir direcciones consecutivas pues el equipo, al definir v, toma automáticamente la parte baja del número de 16 bits en v, y la parte alta en v+1. Lo mismo sucede para y. el resultado se ubica siempre en la dirección 502 (parte baja) y en 503 (parte alta). La parte baja se ubica también en el acumulador principal Acc, para poder usarse rápidamente (si el resultado es de 8 bits (número menor a 255) su utilización es inmediata).
El ARRASTRE (un “1” si la suma da como resultado un número de 17 bits) aparece en la dirección 504.
Por último aparece un “0” en la dirección 505, cada vez que se realiza la suma.

EJEMPLO

2150 + 360 = 2510
ADD 160 162

160	0 1 1 0 0 1 1 0	PB	2150
161	0 0 0 0 1 0 0 0	PA	
162	0 1 1 0 1 0 0 0	PB	360
163	0 0 0 0 0 0 0 1	PA	
502	1 1 0 0 1 1 1 0	PB	
503	0 0 0 0 1 0 0 1	PA	2510
504	0 0 0 0 0 0 0 0	Arrastre	
505	0 0 0 0 0 0 0 0	0	

NEMÓNICO

```
ADD x y       v <0...510> variable de 16 bits
              y <0...510> variable de 16 bits
```

((v+1)(v))+((y+1)(y))

Acc (parte baja)
502 (parte baja)
503 (parte alta)
504 (arrastre)
505 (0)

Acc 1 1 0 0 1 1 1 0

Resta SUB v y

Resta el contenido de dos direcciones de memoria (variables internas) v e y, de 16 bits cada una (valor máximo de cada operando: 65535). Hay que tomar las mismas precauciones al elegir las direcciones, que en la suma ADD.

EJEMPLO

4000 – 230 = 3770

SUB 168 170

168	1 0 1 0 0 0 0 0	PB	4000
169	0 0 0 0 1 1 1 1	PA	
170	1 1 1 0 0 1 1 0	PB	230
171	0 0 0 0 0 0 0 0	PA	
502	1 0 1 1 1 0 1 0	PB	
503	0 0 0 0 1 1 1 0	PA	3770
504	0 0 0 0 0 0 0 0		

NEMÓNICO

```
SUB x y        v <0...510>
               y <0...510>
```

((v+1)(v))-(y+1)(y)) Acc (parte baja)
502 (parte baja)
503 (parte alta)
505 (1 si es negativa)

Acc 1 0 1 1 1 0 1 0

Multiplicación MUL v y

Multiplica el contenido de dos direcciones de memoria (variables internas) v e y, de 16 bits cada una (valor máximo de cada operando: 65535) con la restricción de que el resultado no puede superar el número 65535 (16 bits); de ocurrir esto último, se indica el rebalse con un 1 en la dirección 506, dándose como resultado 65535. Se deben tomar las mismas precauciones al elegir las direcciones v e y que en la suma ADD o resta SUB.

NEMÓNICO

```
((v+1)(v)*(y+1)(y))          Acc (parte baja)
                             502 (pate baja)
                             503 (parte alta)
                             506 (rebalse)
```

División DIV v y

Divide el contenido de dos direcciones de memoria (variables internas) v e y, de 16 bits cada una (valor máximo de cada operando: 65535).
Se deben tomar las mismas precauciones al elegir las direcciones de v e y que en la suma ADD, resta SUB y multiplicación MUL.

NEMÓNICO

```
((v+1)(v):(y+1)(y))          Acc (parte baja)
                             502 (parte baja)
                             503 (parte alta)
                             506 (resto parte baja)
```

Comparación CP v

Realiza la comparación (resta) entre el contenido del Acumulador principal Acc y el contenido de la dirección de memoria v. No se modifica el contenido de v, pero si el del acumulador Acc que toma el valor “1” si son iguales, o “0” si son distintos. Afecta los flags de comparación (direcciones 504 a 506).

NEMÓNICO

```
v   <0...575> variable interna, externa o T/C.
    Acc – (v)        Acc = 1 si son iguales
    Acc – (v)        Acc = si son distintos
```

Comparación entre dirección y constante CPI v n

Realiza la comparación entre una variable de 15 BIT (numero máximo 32767) contenida en las direcciones v (parte baja) y v + 1 (parte alta) y una constante entera de 15 BIT (numero máximo: 32767). No se modifica el contenido de v y v + 1 y el acumulador principal Acc toma el valor "1" si son iguales, y "0" si son distintos. Afecta los flags de comparación (direcciones 504 a 506) y los internos, para saltar con JPP. Antes de ejecutar la instrucción, se guarda el contenido del acumulador, pudiéndose luego realizar un ORP ó ANDP.

NEMÓNICO

```
Con v   <0...574>, variable interna, externa o T/C
    n   <0...575>, valor externo

<(v+1) (v)> - n            Acc = 1 si son iguales
                           Acc = 0 si son distintos
```

CPV v y

Compara el valor de la variable de 15 bits formada por la dirección v (parte baja) y v+1 (parte alta) con la variable de 15 bits y (parte baja) e y+1 (parte alta). Si la comparación e igual, el acumulador toma el valor 1, caso contrario queda en 0. No afecta el contenido de v, v+1, y ó y+1.
Antes del CPV se guarda el Acc, de modo que luego se puede hacer un ORP ó ANDP.
Afecta los flags y el acumulador de la misma manera que el CPI.

NEMÓNICO

```
Con v   <0...574>, variable de 15 bits interna, externa o T/C
    y   <0...574>, variable de 15 bits interna, externa o T/C

<(v+1) (v)> - <(y+1) (y)>              Acc = 1 si son iguales
                                       Acc = 0 si son distintos
```

INSTRUCCIONES DE MANEJO Y MOVIMIENTO DE VARIABLES

Carga de acumulador en forma inmediata LDI n

Carga el acumulador principal Acc con una constante entera entre 0 y 255 (8bit).

NEMÓNICO

```
Con n  <0...225> constante
```

Carga de acumulador con byte LDB v

Carga el acumulador principal Acc con el BIT menos significativo (BIT 0) de cada una de ocho direcciones consecutivas de memoria, comenzando con la dirección en v hasta (v+7), en orden creciente.

EJEMPLO

	b7	b6	b5	b4	b3	b2	b1	b0		Acc	
V	X	X	X	X	X	X	X	0	———	0	b0
V+1	X	X	X	X	X	X	X	1	———	1	b1
V+2	X	X	X	X	X	X	X	1	———	1	b2
V+3	X	X	X	X	X	X	X	1	———	1	b3
V+4	X	X	X	X	X	X	X	0	———	0	b4
V+5	X	X	X	X	X	X	X	0	———	0	b5
V+6	X	X	X	X	X	X	X	1	———	1	b6
V+7	X	X	X	X	X	X	X	0	———	0	b7

NEMÓNICO

```
Con v  <0...568>, variable interna, externa o T/C
```

Carga indirecta de acumulado LDX v

Carga el acumulador principal Acc con el contenido de la dirección identificada a partir de un número formado, en su parte baja, por el contenido de la dirección v, y en su parte alta, por el contenido de v + 1.

NEMÓNICO

```
Con v  <0...574>, variable interna, externa o T/C

        < (v+1) (v) >              Acc
```

Carga inmediata de dos variables consecutivas LDIO c v

Carga una constante entera (llamada i) entre 0 y 65535 (16 BIT), en las direcciones de memoria v (parte baja) y (v + 1) (parte alta). El contenido del acumulador principal Acc no es afectado.

NEMÓNICO

```
Con i  <0...65535>, constante entera
Con v  <0...505>, variable interna, externa o T/C
```

Carga inmediata de una variable en binario IOB n v

Carga una constante entera (denominada b), especificada en binario, entre 00000000 y 11111111, en la dirección de memoria v. El contenido del acumulador Acc no es afectado.
NOTA: La constante binaria debe expresarse con los 8 bits.

NEMÓNICO

```
Con b  constante binaria de 8 bits
Con v  <0...506>, variable interna, externa o T/C
```

Movimiento de variable de 16 BIT MOV v y

Carga el contenido de las direcciones de memoria v (parte baja) y (v + 1) (parte alta), en las direcciones y (parte baja) e (y + 1) (parte alta). Las direcciones v y (v + 1) conservan el mismo valor.
El valor máximo admitido de los operandos es de 65535.

NEMÓNICO

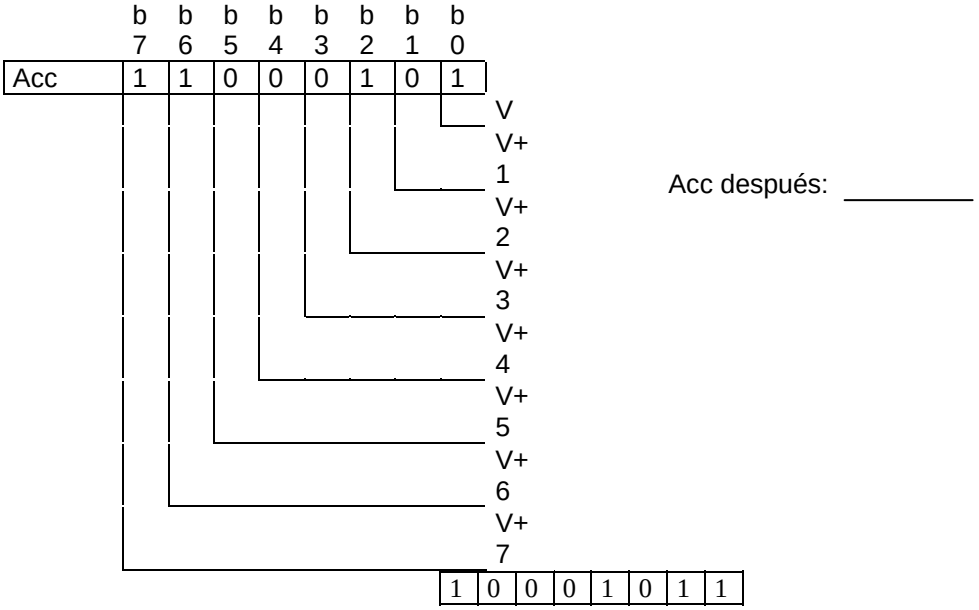
```
Con y  <0...504>, variable de 16 bits
Con v  <0...510>, variable de 16 bits

(v)                 (y)___Parte baja
(v+1)             (y+1)___Parte alta
```

Descarga de byte del acumulador OUTB v

Descarga cada uno de los 8 BIT del acumulador principal Acc en el BIT menos significativo (BIT 0) de 8 direcciones consecutivas de memoria, comenzando con la direccionada en v. Es la inversa de LDB. Se realiza además la rotación del BYTE. El acumulador principal Acc queda con el contenido de v + 7.

EJEMPLO



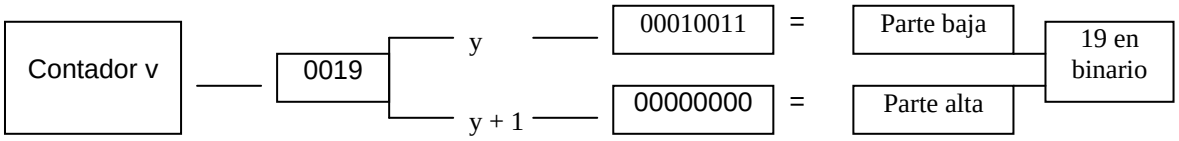
NEMÓNICO

```
Con v  <0...499>, variable interna o externa
```

Descarga del estado instantáneo de timer/contador OUTC v y

Descarga el valor de cuenta instantáneo de un temporizador o contador v en la dirección de memoria y (parte baja) e y + 1 (parte alta).
El acumulador principal Acc toma el valor “0” si el valor de cuenta es distinto de 0 y “1” si este ultimo es 0.

EJEMPLO



NEMÓNICO

```
Con v  <512...575>, contador
Con y  <0...505>, variable interna o externa
```

Descarga indirecta de acumulador OUTX v

Descarga del contenido del acumulador principal Acc, en la dirección de memoria identificada con el número que se forma con el contenido de v (parte baja) y v + 1 (parte alta).

NEMÓNICO

```
Con v  <0...506>, variable interna o externa
```

Acc_____ ((v+1) (v))

Forzado a “1” (seteado) SET v

A partir de colocar un “1” en el acumulador principal Acc, se fuerza (Setea) los 8 BIT de la dirección de memoria v, a “1”. Un “0” en el acumulador Acc no modifica el contenido de v. El acumulador Acc queda como indica la siguiente tabla:

b 7	b 6	b 5	b 4	b 3	b 2	b 1	b 0
1	1	0	0	0	1	0	1
1	1	1	0	0	0	1	0
0	1	1	1	0	0	0	1
1	0	1	1	1	0	0	0
0	1	0	1	1	1	0	0
0	0	1	0	1	1	1	0
0	0	0	1	0	1	1	1
1	0	0	0	1	0	1	1

Acc	b0 v (antes)		b0 v (después)	Acc
0	0		0	0
0	1	SET————	1	0
1	0		1	1
1	1		1	0

NOTA: La tabla indica en la columna v, solo el BIT menos significativo (BIT 0) del byte contenido en v. Los demás BIT del contenido en la dirección v, puede tener cualquier estado, pero al darse las condiciones necesarias de SET, todos los BIT de v van a “1”.

NEMÓNICO

```
Con v  <0...506>, variable interna o externa
```

Forzado a “0” (reseteado) RST v

Colocando un “1” en el acumulador principal Acc, se fuerza (resetea) los 8 BIT de la dirección de memoria v, a “0”. Un “0” en el acumulador principal Acc no modifica el contenido de la dirección v. Si es mayor o igual que 512 (contador o temporizador), se resetean tanto la cuenta como la variable asociada (nota: que v sea mayor o igual que 512 puede usarse solo en versiones de BIOS posteriores a la 8.5). El acumulador Acc queda como indica la siguiente tabla.

Acc	b0 v (antes)		b0 v (después)	Acc
0	0		0	0
0	1	RST v————	1	0
1	0		0	1
1	1		0	0

NOTA: La tabla indica en la columna v, solo el BIT menos significativo (BIT 0) del byte contenido en v. Los demás BIT del contenido de la dirección v, pueden tener cualquier estado, pero al darse las condiciones necesarias para el RESET, todos los BIT de v van a “0”.

NEMÓNICO

```
Con v  <0...575>, variable interna o externa, o
    asociada a un contador o temporizador.
```

Salto si el acumulador es “1” JP p

Si al leerse esta instrucción el acumulador principal esta en “1”, se salta a ejecutar una línea identificada con una etiqueta “p” (label) generada por el usuario, que puede ser alfanumérica, con la condición de comenzar con una letra y no contener espacios. Si el acumulador principal esta en “0” se ignora el salto. La subrutina que comienza con la línea identificada con la etiqueta p, debe escribirse en el programa siempre mas adelante que la instrucción JP1, y debe contener la indicación de retorno al programa principal mediante la instrucción RJP o END 9consulta las mismas).

NEMÓNICO

```
Con p  <etiqueta>
```

Salto incondicional a subrutina GOSUB p

Al pasar por esta instrucción, se salta a ejecutar una subrutina cuya primera instrucción está identificada con la etiqueta “p”. Deben tomarse para “p” y para la subrutina que ésta identifica, las mismas consideraciones que en JP1.

NEMÓNICO

```
Con p  <etiqueta>
```

Retorno de salto RJP

Retorna al programa principal, en la instrucción posterior a la que generó el salto correspondiente.

Salto sin retorno si el acumulador es 1 IF1 p

Esta instrucción es similar a JP1, pero no se retorna. Se deben tener en cuenta, respecto a “p” (label), sin retorno, las mismas consideraciones que en JP1. La línea identificada por “p” debe escribirse siempre más adelante que la instrucción GOTO.
NOTA: Las etiquetas “p” tienen las mismas características que en JP1 o GOSUB.

NEMÓNICO

```
Con p  <etiqueta>
```

Conversión de BCD a binario BIN v

Convierte un número decimal de hasta 4 dígitos expresado en BCD (16 bits), a binario, sobre las mismas direcciones en que se encuentra.

EJEMPLO

v	0	1	1	1	0	0	0	1	271 en BCD
v+1	0	0	0	0	0	0	1	0	

v	0	0	0	0	1	1	1	1	271 en binario
v+1	0	0	0	0	0	0	0	1	

NEMÓNICO

```
Con v  <80...505> variable de 16 bits BCD
```

Conversión de binario a BCD BCD v

Convierte un número decimal de hasta 4 dígitos expresado en binario (16 bits), a BCD, sobre las mismas direcciones en que se encuentra.

EJEMPLO

v	0	0	0	0	1	1	1	1	271 en binario
v+1	0	0	0	0	0	0	0	1	

v	0	1	1	1	0	0	0	1	271 en BCD
v+1	0	0	0	0	0	0	1	0	

NEMÓNICO

```
Con v  <80...505>
```

ROTACIÓN DERECHA DE BYTE SHR v n

Cada vez que el programa lee la instrucción, si n=0, desplaza cada uno de los bits del contenido de la dirección de memoria v un lugar, hacia el BIT menos significativo (BIT 0), colocando a éste (BIT 0 del contenido de v), en el BIT 0 del acumulador principal Acc. El estado que tenía el BIT 0 del acumulador,

pasa al BIT más significativo (BIT 7) del contenido de v. El efecto es el de una rotación a la derecha, utilizando el acumulador como lugar de paso del BIT 0 del contenido de la dirección v. Si n=4 la rotación se efectúa en forma similar, pero considerando “paquetes” de 4 bits.

EJEMPLO

	b 7							b 0		b 7							B 0
v	1	0	1	0	1	0	1	0	SHR v	0	1	0	1	0	1	0	1
Acc	X	X	X	X	X	X	X	0	Acc	X	X	X	X	X	X	X	0

NEMÓNICO

```
Con v  <0...506> variable interna o externa
Con n  <0 ó 4> parámetro de modo de rotación
```

Rotación izquierda de byte SHL v n

Esta instrucción es similar a SHR, pero la rotación se efectúa en sentido inverso (hacia la izquierda).

EJEMPLO

	b 7							b 0		b 7							B 0
v	1	0	1	0	1	0	1	0	SHL v	0	1	0	1	0	1	0	0
Acc	X	X	X	X	X	X	X	0	Acc	X	X	X	X	X	X	X	1

NEMÓNICO

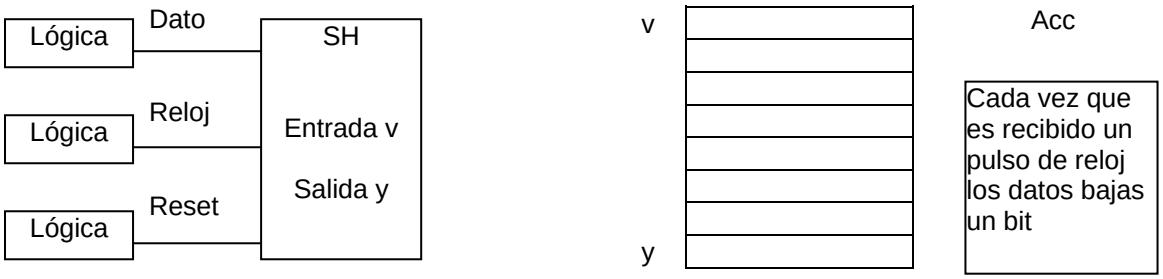
```
Con v  <0...506> variable interna o externa
Con n  <0 ó 4> parámetro de modo de rotación
```

Registro de desplazamiento SH vy

Esta instrucción utiliza 3 cargas previas en los acumuladores: un canal de entrada de datos, un reloj (del cual se detectan sus flancos ascendentes), y un comando de reset.

Cada vez que se dan las condiciones de un pulso de reloj (flanco ascendente), el registro de desplazamiento lee el canal de dato, y lo transfiere a la dirección de memoria v, desplazándose el contenido que tenía v, a (v+1); el de (v+1), a (v+2); así hasta (v+n) = y.
El contenido que tenía la dirección de memoria y (llamada salida), es cargado en el acumulador. O sea: cada flanco de reloj desplaza los contenidos de las direcciones entre v (entrada de dato), e y (salida), en forma creciente. No hay limitación en la cantidad de registros de desplazamiento a utilizar, pero debe tenerse cuidado de no solapar las direcciones de memoria usadas. Cuando aparece un “1” en el comando de reset, todos los contenidos de las memorias entre v e y se ponen en cero.

EJEMPLOS



NEMÓNICO

```
Con v  <373...500> variable de entrada
Con y  <374...501> variable de salida, siendo v < y
```

NOTA: En el ejemplo se visualizan solamente los bits menos significativos de cada dirección, pero el desplazamiento se realiza para todo el byte.

INSTRUCCIONES DE MANEJO DE PERIFÉRICOS

Carga de teclado TCL 80

Permite interpretar la lectura de un teclado de hasta 16 teclas (números del 0 al 9, y 6 teclas de función), conectado a la CPU mediante el conector de teclado y display “DISP-TCL”.
Cuando una tecla es pulsada, el acumulador principal se pone en “1” durante una vuelta de programa. En caso contrario, Acc = 0. El código de la tecla apretada es cargado en la dirección de memoria 502, según el siguiente detalle:

TECLA	DIRECCIÓN 502
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111
8	00001000
9	00001001
Fun. A	00001010
Fun. B	00001011
Fun. C	00001100
Fun. D	00001101
#	00001110
*	00001111

Si bien puede conectarse un solo modulo de teclado al controlador, existe un modo alternativo de hacerlo, el cual permite establecer aislación galvánica entre el PLC y el teclado (conveniente, por ejemplo, en el caso de que ambos se encuentren distanciados).
En esta opción, es necesario contar con cinco entradas de numeración consecutiva de una tarjeta de ocho.

La línea “DO” del teclado se conectara a la entrada que, entre las cinco a usar, posea la menor numeración; “D1”, a la siguiente, y así sucesivamente; la línea “DA” se conectara a la entrada de mayor numeración del grupo. La conexión quedara identificada reemplazando el numero “80” de la instrucción, por el número de la entrada de numeración menor.

EJEMPLO *NEMÓNICO*

Si se conecta el modulo a las siguientes entradas: `Con 80 <variable auxiliar>`

D0 ----> entrada 16
D1 ----> entrada 17
D2 ----> entrada 18
D3 ----> entrada 19
DA ----> entrada 20

... el modulo se leerá con la instrucción: TCL 16

Salida de display

El SCD - Serie 80 permite el envío de información a módulos de display de leds de 7 segmentos hasta 4 dígitos (0 a 9999) cada uno expresados en BCD en las variables (v) y (v+1).
En caso de usarse un único modulo, el mismo puede conectarse a la CPU mediante el conector de display y teclado "DISP-TCL"; De ser necesarios mas, cada modulo adicional necesitara de dos salidas a transistor sin optpacopplar.
Se vera a continuación como se manejan los módulos se display según como estén conectados.

Manejo del display conectado a la CPU:

Debe tenerse en cuenta lo siguiente
1. La información a enviar al display se debe ubicar en las direcciones 81, 82 y 83 según el siguiente detalle:

DECENA	UNIDAD
MILLAR	CENTENA
Puntos decimales	

En la tabla 1 (ver la pagina siguiente), se detalla la equivalencia entre el numero enviado y lo que se visualizara en el display.

En la dirección 83, y según la tabla 2, se carga la ubicación en el display de los puntos.

2. La habilitación del envío de la información, se realiza colocando un "1" en el BIT 0 de la dirección 84. Esta condición hace que, en el próximo barrido (scan):
a) la información salga por display:
b) se resetee (se ponga a "0" el BIT 0) la dirección 84.

Manejo de cada display conectado a salidas a transistor

La instrucción OUTD v o o¹ envía la información al módulo a través de dos salidas o y o¹. La información a enviar debe ubicarse en las direcciones v y (v+1), según el siguiente detalle:

Respecto a la equivalencia entre el valor enviado y lo que se visualizará, y la ubicación de los puntos, valen, como en el caso anterior, las tablas 1y 2.

Una vez ejecutada la instrucción OUTD, la misma no debe volver a ejecutarse hasta, por lo menos, 100ms después. El siguiente esquema de programa asegura tal condición:

```
        LDI 1
        LD 512
        TP 512. 1
        NOT
        IF1 NODISP
        OUTD x o ó
NODISP (sigue el programa)
```

TABLA 1:

UBICADO EN MEMORIA	ENVIADO	VISUALIZADO
0000	0	0
0001	1	1

0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	I
1100	12	II
1101	13	U
1110	14	-
1111	15	Dígito apagado

TABLA 2:

DIRECCIÓN 83 ó v+2	n	DIGITOS
00000000	0	
00000001	1	*
00000010	2	*
00000011	3	* *
00000100	4	*
00000101	5	* *
00000110	6	* *
00000111	7	* * *
00001000	8	*
00001001	9	* *
00001010	10	* *
00001011	11	* * *
00001100	12	* *
00001101	13	* * *
00001110	14	* * *
00001111	15	* * * *

* : Punto encendido

para cargar la dirección 83:

```
LDI n
OUT 83
```

para cargar v+2:

```
LDI n
OUT v+2
```

Instrucciones de incremento y decremento de variables:

	NEMONICO	
INC	v n m	Incremento
DEC	v n m	Decremento

v: variable a incrementar o decrementar;
n: cantidad a incrementar o decrementar cada vez que se lee la instrucción, según el siguiente detalle:

0 => incr./decr. de a 1
— — — — — —
— — — — — —
9 => incr./decr. de a 10 (Cabe acotar que se admiten valores intermedios);

m: magnitud de la variable a considerar, de acuerdo a la siguiente convención:

0 => variable de 16 bits
1 => variable de 8 bits

Téngase en cuenta que, para el caso mas habitual, en el cual se desea incrementar o decrementar de a uno una variable de 16 bits, ambos parámetros secundarios de la instrucción son cero.

FUNCIONES ESPECIALES

FUN1: Set clock: FUN1 v 0 0

Donde v es la primera de un grupo de 5 direcciones consecutivas; desde ellas, al ejecutarse la función, se enviara al reloj de tiempo real los datos ubicados en la dirección v y las 4 siguientes, de acuerdo al siguiente esquema:

(v):	Fecha	} En BCD
(v+1):	Mes	
(v+2):	Año	
(v+3):	Hora	
(v+4):	Minutos	

FUN2: Get clock: FUN2 v 0 0

Donde v es la primera de un grupo de 5 direcciones consecutivas; hacia ellas, al ejecutarse la función, el reloj enviara la información de hora y fecha, distribuyéndose los datos de dichas 5 direcciones de forma similar al caso ya visto de Set clock (FUN1):

(v):	Fecha	} En BCD
(v+1):	Mes	
(v+2):	Año	
(v+3):	Hora	
(v+4):	Minutos	

FUN3: Manejo de la impresora

Al ejecutarse esta función, la impresora de 24 columnas conectadas a la CPU 80R imprimirá (en el caso mas general) una línea; se vera a continuación como especificar lo que se desea imprimir.

Comenzaremos viendo la sintaxis de la función:

```
FUN3   v   0   c
```

"v": es la dirección en el cual, para algunos comandos posibles 9 como ya se vera), comienza la tabla de datos a imprimir.

"c": es la dirección que especifica el tipo de línea a imprimir, a saber:
0: se deja en el papel una línea en blanco;
1: se imprime una línea de texto de hasta 24 caracteres;
2: se imprime una línea de puntos;
3: impresión de fecha y hora.

Para el comando 1, el código del primer carácter se ubica en la dirección v.

Para el comando 3, la información se ubica desde la dirección v, según el esquema de las funciones de manejo de reloj (FUN1 y FUN2).

Como puede llevarle al sistema más de una vuelta de programa el envío de toda la información a la impresora, para tener esto en consideración en el programa de usuario, el acumulador sale en 1 solamente cuando se tuvo éxito, saliendo en 0 en caso contrario.

Veremos ahora cada comando en particular:

0) línea en blanco: Tratándose de dejar una línea en blanco, carece de sentido hablar de datos, por lo tanto, como dirección "v", podemos colocar. Por ejemplo, un 0.

1) línea de texto: En la dirección "v" comenzara la tabla de hasta 24 caracteres a imprimir. El código de cada carácter ocupara una dirección. Los caracteres pueden ser:

a) Cifras del 0 al 9 (se interpretan directamente);
b) Caracteres alfabéticos, numéricos, o simbólicos, representados en códigos ASCII (se anexa la tabla de los necesarios):
c) código de fin de línea: si bien el sistema tiene en cuenta que la tabla correspondiente a una línea no puede tener más que 24 caracteres, existe la posibilidad de, si es conveniente. Truncar la tabla, finalizándola con el código de fin de línea (0D hexa); el resto de la línea quedara en blanco.

2) línea de puntos: Tampoco aquí se necesitan datos, por lo tanto podemos, como en el caso de la línea en blanco, colocar como dirección "v" un 0.

3) Fecha y hora: Para imprimir fecha y hora se especifican los datos de forma análoga a "set clock" y "get clock" (5 direcciones consecutivas). De esta forma, un método cómodo de imprimir fecha y hora será hacer, por ejemplo:

```
FUN2   300   0   0   (se descarga la información actualizada en las direcciones 300 a 304)
FUN3   300   0   3   (se imprime dicha información; la codificación es la misma)
```

Tabla de códigos de caracteres para las líneas de texto

La siguiente tabla presenta los códigos de los caracteres que se pueden incluir en las líneas de texto. Cabe mencionar que, si bien se incluyen códigos para las cifras del 0 al 9, el sistema también interpreta, como ya se indicó, estas cifras directamente. Esto último facilita la impresión de números, en particular de resultados de operaciones aritméticas.

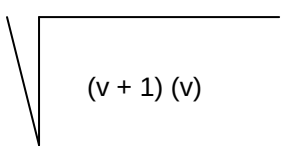
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	ESP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	'	A	B	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	Q	R	s	t	u	v	w	x	y	z	{	|	}	~	///

Aclaraciones para interpretar la tabla:
- "ESP" significa espacio;
-El código de "@" es 40 (hexa);
-El código de "!" es 21 (hexa); etc.

FUN4: Raíz cuadrada FUN4 v 0 0

Al ejecutarse esta función se extrae la raíz cuadrada del contenido de las variables (v) y (v + 1). Considérese que la raíz de un número de 16 bits puede expresarse si es necesario en 8 bits.

EJEMPLO



Acc (parte baja = raíz en 8 bits)
502 (parte baja = raíz en 8 bits)
503 (parte alta = 0)

FUN5: Mensaje

Para incluir mensajes en una aplicación, el usuario debe estructurar el programa del PLC para que cada estado tenga su correspondiente mensaje.
La asignación de un mensaje de proceso a un estado se realiza mediante la instrucción FUN5, que determina la salida de un mensaje a un renglón específico del display. El formato con el que se debe utilizar la FUN5 es el siguiente:

```
FUN5: [Número_de_mensaje] [Número_de_renglón] 0
```

Número_de_mensaje: Es el número que asocia el editor a cada mensaje del estado de variables de proceso.
Número_de_renglón: Selecciona uno de los 2 renglones con los que cuenta el display para presentar los mensajes. Este puede tomar 2 valores: 1 y 2.

Una vez tenido en cuenta lo anterior se utiliza el ALFA Edit para editar cada mensaje. Este entorno integrado editor / cargador de mensajes, corre en cualquier computadora personal del tipo IBM PC o Compatible, y permite editar, enviar y recibir los mensajes del SCD.

FUN6:

El tclcd posee 8 teclas de función, las cuales son tenidas en cuenta por el programa usuario del PLC, mediante la utilización de la FUN6. Esta instrucción devuelve un 1 en el acumulador cuando detecta que una de ellas fue apretada y nos ubica en la dirección 502 el número de tecla activada por el ordenador (de 1 a 8). La función se utiliza con el siguiente formato:

```
FUN6     0     0     0
```

Como se verá no precisa parámetros; lo que hace es invocar a las rutinas del PLC, para que indiquen si fue apretada una tecla y cuál de ellas.

PI: PI nro v y vmax

Permite establecer controles PI sobre variables transferidas a la posición correspondiente de IO. v es la primera de las variables de control e y es la primera variable de la tabla de IO, las cuales poseen los siguientes campos para cada PI definido:

v:	y:
Set Point	Valor de entrada
Kp	Valor de salida
Ki	nro especifica el número de controles PI, y por cada uno de estos se define el grupo de campos anterior uno a continuación del otro.

NEMÓNICO

```
Con   v   <0...501>
          y   <0...501>
          nro   <1...12>
          vmax   <0...1024>
          Acc = 0   <sin acción> (cualquier otro valor ejecuta PI ídem Acc = 1)
          Acc = 2   <Reset PID>
```

SCALE: SCALE v1 v2 v3

Realiza el escalamiento de la variable v1 a partir de los valores v2 y v3.

EJEMPLO

$$V1 = \frac{ent\ (v1 * v3)}{v2} \qquad 502 - 503 = \frac{resto\ (v1 * v3)}{v2}$$

Si bien no hay limitación en el resultado de (v1 * v3), el resultado de la última división debe estar en 16 bits.
El tiempo de ejecución puede llegar a ser de 4 mseg, de acuerdo a los valores v1, v2 y v3.

NEMÓNICO

```
Con     v1  <0...511> variable de 16 bits
           v2  <0...511> variable de 16 bits
           v3  <0...511> variable de 16 bits
```

MACROS

Si durante la edición de un programa se considera que será frecuente el uso de dos o más instrucciones consecutivas siempre en el mismo orden, puede definirse un macro, esto es, darle un nombre a una sucesión de instrucciones. Cada vez que se invoque el nombre así definido, como si fueran unas instrucciones especificadas en la definición.
La definición de un macro puede dividirse en 4 partes, que se escriben una a continuación de otra:

1) Una línea que consta del nombre del macro comenzando desde el margen izquierdo, seguido, separado por lo menos por un espacio, de la sentencia “@MACRO” (comienza la definición de un macro).

2) Las definiciones de las variables y etiquetas (labels) a usar; deben estar separadas del margen izquierdo, debiéndose considerar 3 casos que se mostrarán con ejemplos:

a) @INT XX1 => XX1 será una variable entera (usada para hacer referencia a una variable interna o a una constante).

b) @FLOAT XX2 => XX2 será una variable con punto decimal (usada para hacer referencia a un tiempo).

c) @LABEL XX3 => XX3 será una etiqueta (label).

3) La secuencia de instrucciones a las que equivaldrá el macro; en esta secuencia se hará referencia a las variables y labels recién definidas.

4) La sentencia “@ENDM”, que indica que terminó la definición del macro.

Cuando se llame al macro, deberán colocarse a continuación del nombre del mismo las variables y/o etiquetas que adoptarán las funciones de las que fueron definidas, respetando el orden impuesto en la definición.
Se tratará de aclarar el uso de los macros con el siguiente ejemplo: Supongamos que, para el programa que se está desarrollando, sería cómodo poseer una instrucción que salte a una subrutina si se da que el contenido de una variable es mayor o igual que cierto valor.
La forma explícita de lograrlo sería:

```
CPI NIVEL 1000
OR 506
JP1 REBALSA   (si NIVEL es mayor o igual que 1000 salta a ejecutar REBALSA)
```

Podemos definir el siguiente macro, llamado JPMIG (JP si Mayor o Igual):

```
JPMIG @MACRO
      @INT VARIA
```

```
@INT LIMIT
@LABEL MAYIG
CPI VARIA LIMIT
OR 506
JP1 MAYIG
@ENDM
```

Con lo cual, las 3 instrucciones se podrán reemplazar por:

JPMIG NIVEL 1000 REBALSA

Así mismo, si se desea que, por ejemplo, si TEMP es mayor o igual que 200 se salte a ejecutar ALTAT se escribirá simplemente:

```
JPMIG TEMP 200 ALTAT
```

Obsérvese que la utilidad de definir un macro se evidencia la estructura del programa requiere que muchas veces se escriba la misma secuencia de instrucciones, sean o no, las variables y/o etiquetas utilizadas, las mismas.

TCLCD CAIPE: TECLADO DISCPLAY ALFANUMERICO

El Tclcd Caipe es un periférico de la familia de productos Caipe que le permitirá contar en sus automatizaciones con un teclado alfanumérico de 32 teclas y un display de cristal líquido de 2 líneas de 40 caracteres cada una, con conexión directa a la CPU, sin perder la compatibilidad con los anteriores miembros de la familia.
Este display amplia la potencia de nuestros equipos permitiendo entre otras:

- **Informarle al operador, mediante una serie de mensajes de texto, cual es el estado actual del proceso mientras**
- **se monitorean variables del Plc.**
- **Presentar mensajes de inicio de operaciones, alarmas, confirmaciones, etc.**
- **Ingreso y monitoreo de set-point involucrados con el proceso.**
- **Ingreso de códigos alfanuméricos.**

La tarea de programación del teclado y display desde el lenguaje SCD 80, se ve simplificada en la edición de mensajes utilizando el Alfaedit, un nuevo editor con que la firma acompaña el producto.

ESTADOS DEL TCLCD

El Tclcd tiene los siguientes estados de funcionamiento:
- Variables de proceso
- Setos
- Códigos

MEMORIA DE VARIABLES

El estado de variables de proceso es en el que se encontrará el dispositivo a menos que se haya forzado el ingreso a uno de los otros dos pulsando una de las teclas habilitadas para tal fin. De estos estados el Tclcd saldrá cuando transcurran unos segundos y no se haya pulsado ninguna tecla o con ESC.
Aquí es donde se encuentran los mensajes relacionados con el estado del proceso que se esta controlando. Este estado además permite el monitoreo permanente de cualquier variable de proceso involucrada con la automatización.
El: **"Total metros/hora producidos = 1200"**
"Proceso finalizado"
El mensaje que se desea mostrar en un instante determinado lo especifica el programador desde el lenguaje usuario del Plc, mediante la *FUN5*.

Variables de proceso posee un seudo-estado relacionado con él, denominado **Monitoreo** donde se presentarán en la pantalla todos los mensajes contenidos dentro del **Buffer de mensajes del Plc**. Este estado debe ser forzado utilizando uno de los cursores, y luego ingresar la **clave de acceso** con que se cargaron los mensajes. Una vez ingresada correctamente la clave, se presentará el primer mensaje de variables de proceso. Aquí por un periodo corto de tiempo si no se pulsa una tecla se vuelve al estado-Variables de proceso automáticamente o pulsando **ESC** también se sale del estado. La idea de este estado es el monitoreo por parte del instalador o del personal autorizado, de todos los mensajes en cualquier momento, y de ciertas variables ocultas que no se quieren ver en pantalla, es decir que no estén programadas desde el programa usuario.

ESTADO DE SETEO

Este estado destinado al ingreso de los set-point relacionados con el proceso. existen dos niveles de seteos:

- **Seteos de bajo nivel:** están destinados a que el operador a cargo de la máquina, los modifique. A estos se ingresa pulsando la tecla SET seguida de la tecla MENOS.
- **Seteos restringidos:** Los cuales poseen una clave que restringe el acceso a personal no autorizado con esta tarea.

Para ingresar en el, una vez presionada la tecla SET, se debe ingresar en un corto tiempo la clave de acceso, compuesta por tres dígitos numéricos. Sólo si la clave fue ingresada correctamente en el tiempo prefijado se accede a este estado.

CONSIDERACIONES GENERALES:

Se recuerda que debe conectarse a una buena masa todo borne del sistema en que se indique. La conexión se realizará con cable suficientemente grueso, o bien a través de barras.

Se recomienda además que se utilicen cablecanales distintos para los circuitos de entradas y salidas.

Así mismo, debe tenerse especial cuidado con el cableado de señales analógicas de entrada, ya que la inducción sobre las mismas de espúreos provenientes de picos de conmutación de salidas, o bien de otros sistemas, puede provocar lecturas erróneas de datos que afecten al sistema de control. Especialmente para el caso de entradas analógicas de tensión, se recomienda el uso de cable blindado.

Es importante la convergencia de las masas a un punto con cables de suficiente sección (4mm²) evitando formar lazos cerrados.

ALIMENTACIÓN DEL EQUIPO, ENTRADAS Y ACCESORIOS

El SCD-Serie 80 puede solicitarse para ser alimentado con 220Vca (Versiones “A”), o con corriente continua 12/24/48/110 Vcc (Versiones “B”).

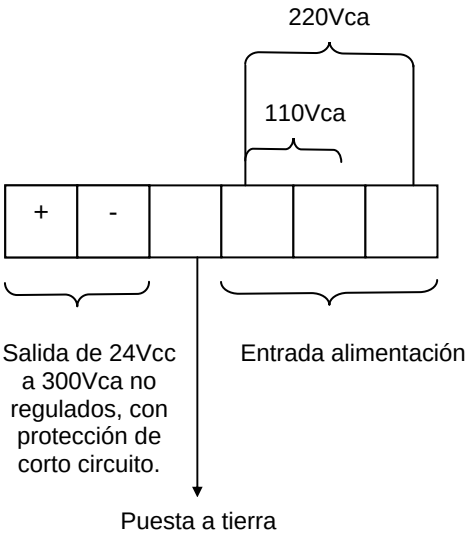
Posee una fuente para alimentar los circuitos externos de entradas digitales. Esta fuente es de 24Vcc aislada galvánicamente de la anterior y 300mA protegida contra cortocircuitos con un fusistor (se repone solo al eliminar el cortocircuito).

Las características y aplicaciones de cada una se verán a continuación.

Las borneras de las fuentes del SCD 80 son como las siguientes:

.

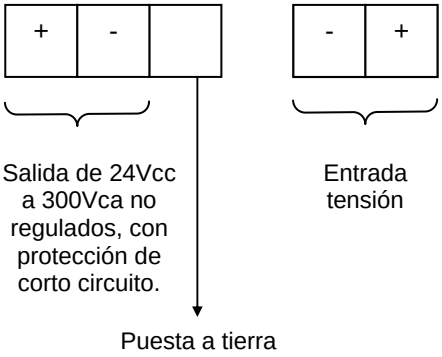
Fuente de corriente alterna:



Fuente de corriente continua:

Puede pedirse para tensiones de alimentación de:

12Vcc
24Vcc
48Vcc
110Vcc
} +/- 30%



Gracias, por confiar en los productos y la calidad de **CAIPE Automatización**. Esperamos cumplir con sus expectativas eficazmente, si tiene alguna opinión constructiva, no dude en darla a conocer en www.caipe.com/foro.

EN ARGENTINA

CAIPE Automatización (Bellplast SRL)
Tel./Fax: (++54) 11 4218-1841 // 4115-1603
Av. Hipólito Irigoyen 2164 1º piso (B1869BUR)
Avellaneda – Buenos Aires – Argentina

EN URUGUAY

CAIPE Ingeniería
Tel./Fax: (++598) 2309 5905
Dr. Marcelino Díaz y García 145 (11900)
Montevideo – Uruguay