Smart Motor Devices

Manual de usuario

CONTROLADORES DE MOTOR PASO A PASO

SMSD-1.5Modbus ver.3
SMSD-4.2Modbus
SMSD-8.0Modbus

2025

Precauciones y observaciones en el texto:

Atención
Diferentes tipos de peligro, que pueden resultar en daños a la propiedad o lesiones.

Información
Recomendaciones, consejos o referencia a otro documento.

Aspectos destacados y formato de fragmentos de texto:

- Etiqueta o marca de preferencia

1) Algunas acciones deben realizarse en orden secuencial

— Enumeraciones comunes

Información del fabricante

Smart Motor Devices se adhiere a la línea de desarrollo continuo y se reserva el derecho de realizar cambios y mejoras en el diseño y software del producto sin previo aviso.

La información contenida en este manual está sujeta a cambios en cualquier momento y sin previo aviso.

Contenido

1. EL PROPÓSITO DEL CONTROLADOR

El controlador está diseñado para operar con motores paso a paso. El dispositivo puede ser controlado por un PLC (vía RS-485 Modbus ASCII/RTU) o puede operar en modo independiente según el programa de ejecución preestablecido.

El controlador proporciona un funcionamiento de paso completo o micropasos de hasta 1/256. El controlador proporciona un movimiento suave con un bajo nivel de vibraciones y una alta precisión de posicionamiento.

El controlador proporciona los siguientes modos de control:

- Modo de control de programa: funcionamiento autónomo según el algoritmo de ejecución preestablecido o control en tiempo real desde una PC o PLC mediante comandos dados a través del protocolo Modbus.
- Control de velocidad analógico: la velocidad de rotación del motor se ajusta mediante el potenciómetro en la parte frontal del controlador.
- Modo de control STEP/DIR: controla la posición del motor mediante señales lógicas de pulso.

El controlador tiene 8 entradas lógicas y 10 salidas (2 de estas salidas son de alto voltaje). El estado de I/O se puede leer o configurar desde un programa de ejecución del usuario o directamente mediante comandos Modbus. Un programa de ejecución interno se puede descargar y cargar en el controlador a través de USB o RS-485.

Proporcionamos software para el ajuste, ensamblaje o edición de programas de ejecución del controlador y para la carga del programa de ejecución en la memoria del controlador.

El controlador tiene protección contra sobrecalentamiento.

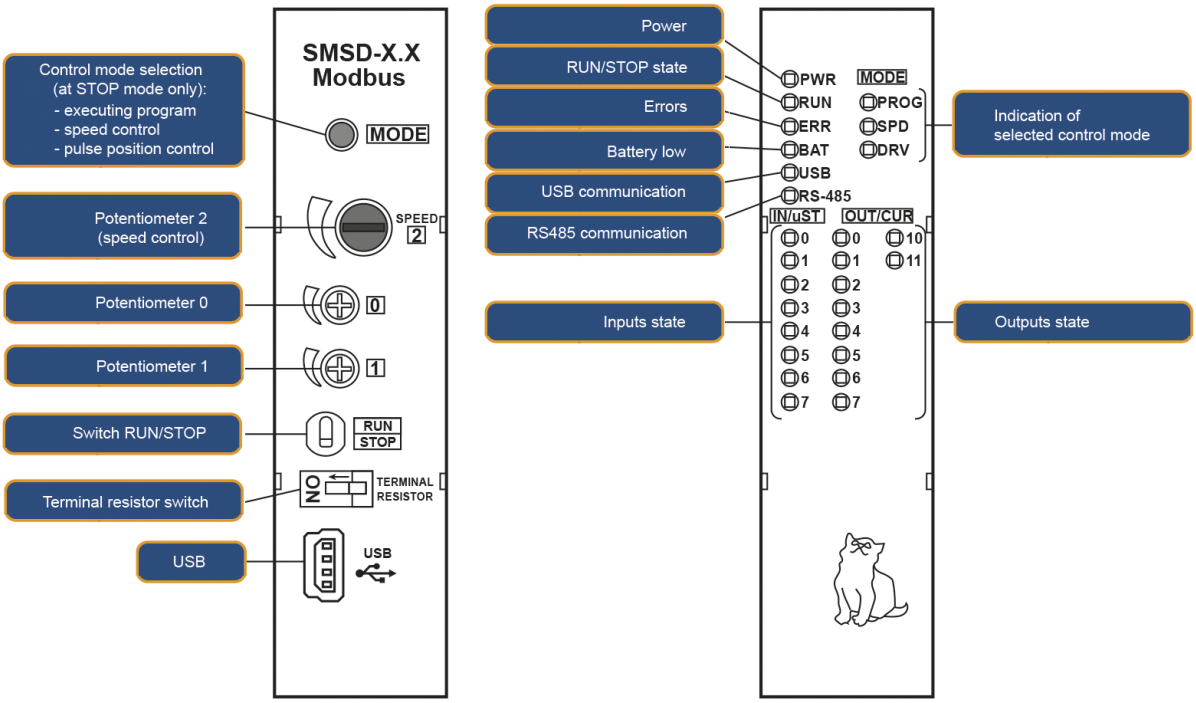


Fig. 1. Propósito de los elementos de control

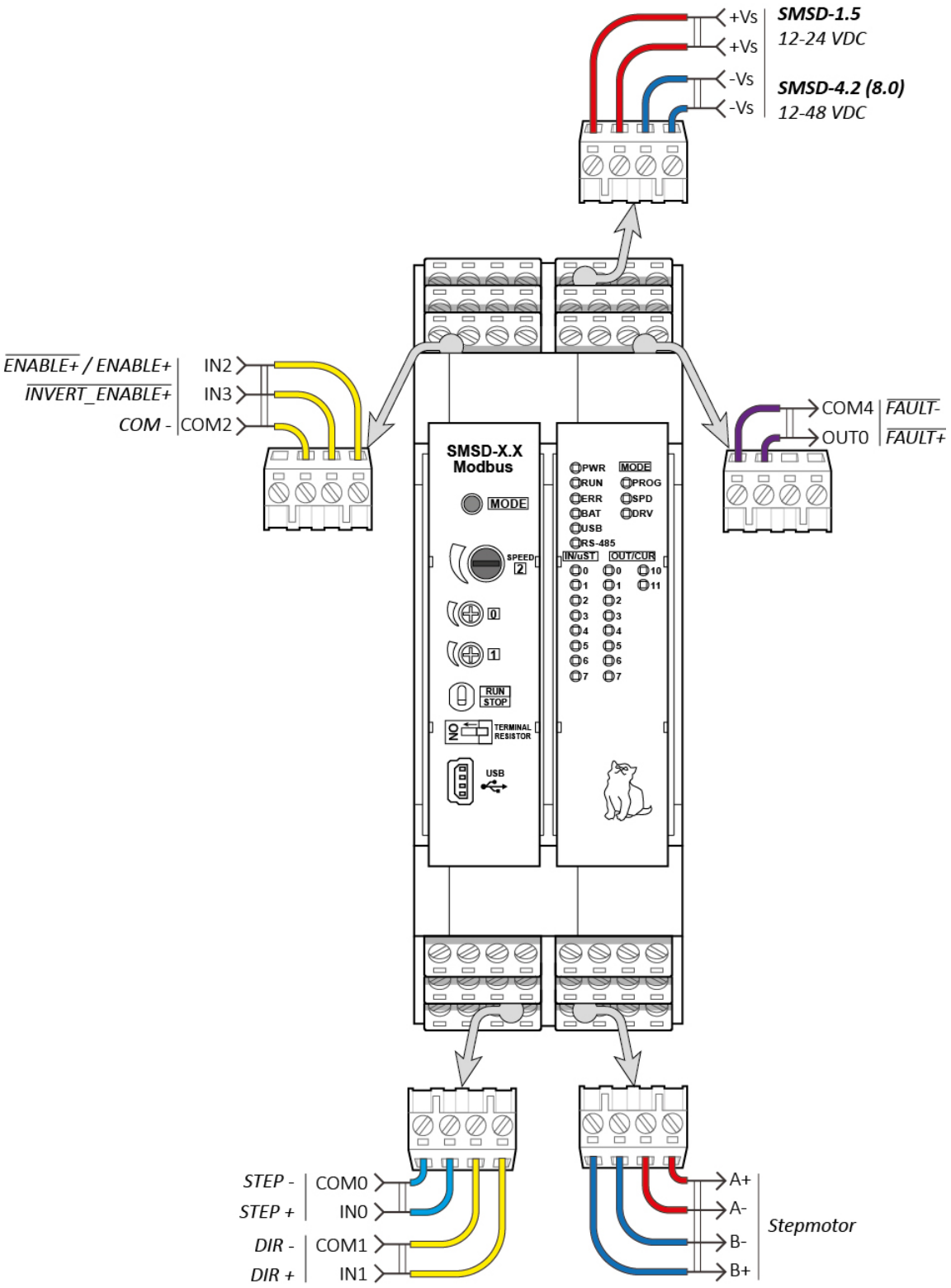


Fig. 2a. Asignación de terminales – modo de control del controlador

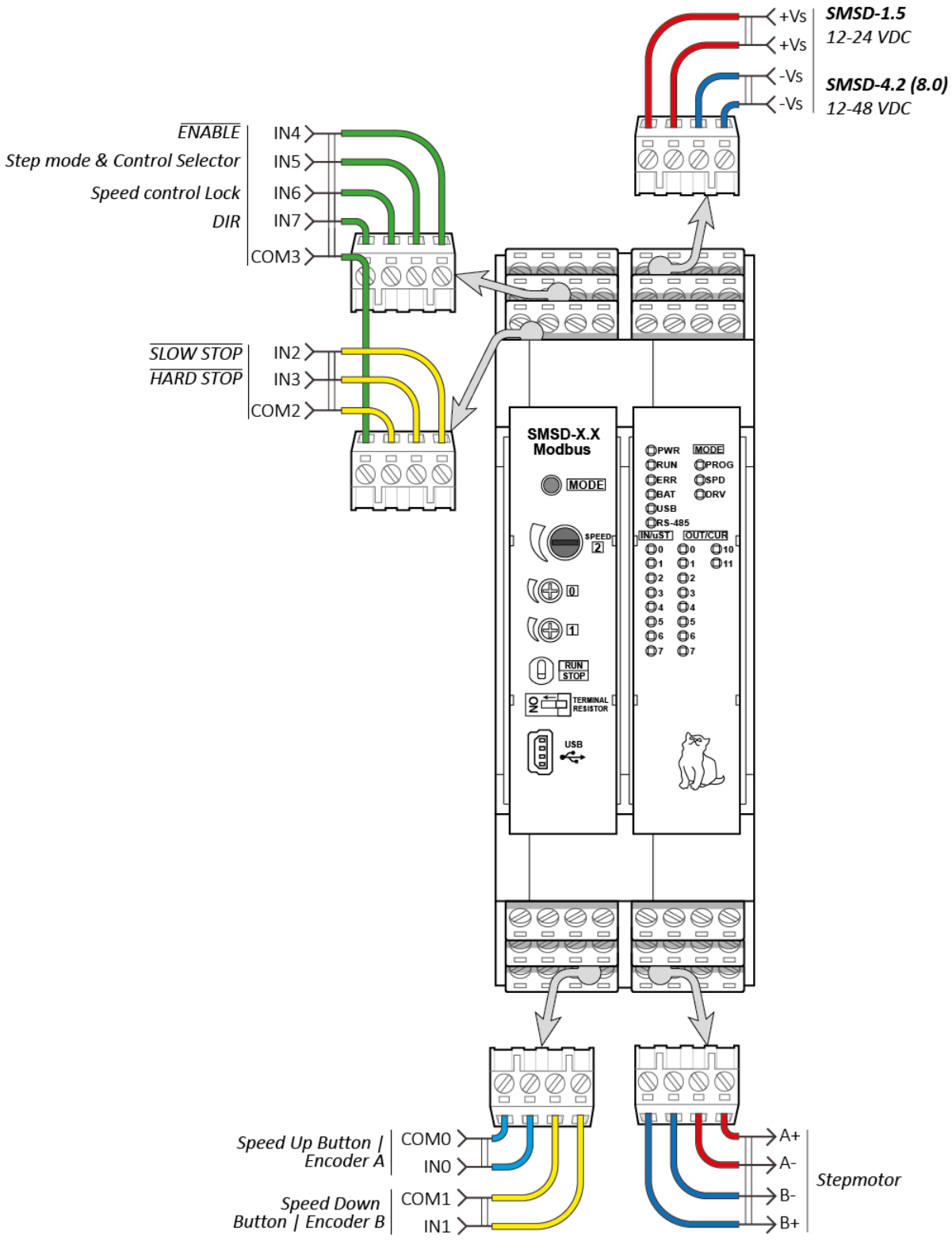


Fig. 3b. Asignación de terminales – modo de control de velocidad

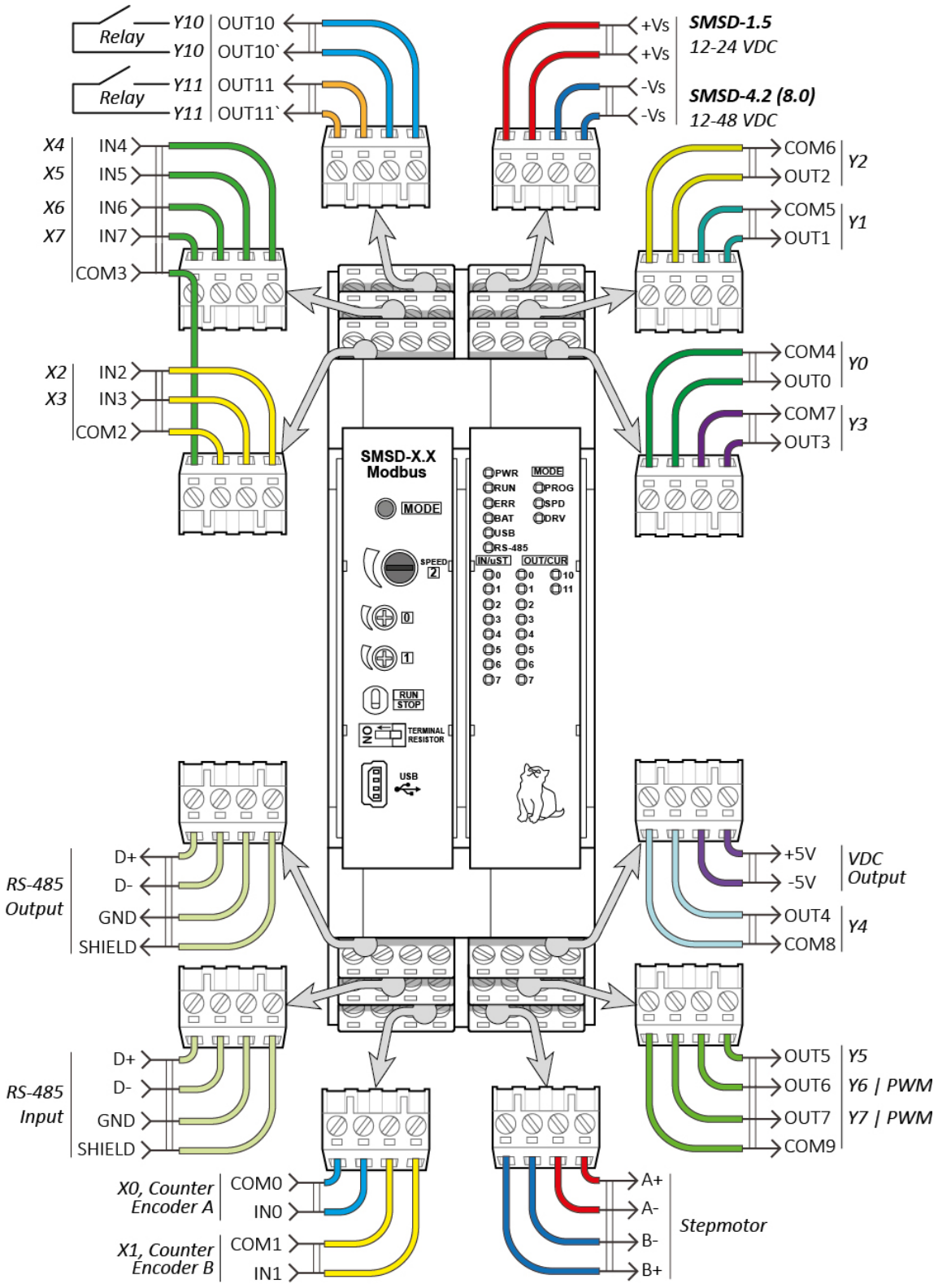


Fig. 4c. Asignación de terminales – modo de control del programa del usuario

La Fig. 1muestra el panel frontal del controlador con los elementos de control e indicación. El indicador **PWR** indica la presencia de voltaje de alimentación. **RUN** indica el estado actual del controlador (RUN o STOP). En el modo **PROG** (ejecución de un programa de usuario) y el modo **SPD** (control de velocidad), el estado activo del indicador **RUN** indica la ejecución del programa, el estado inactivo indica el estado de parada. En el modo **DRV** (modo Step/Dir), el estado inactivo de **RUN** indica la posibilidad de configurar los parámetros del controlador mediante los elementos de control del controlador. El estado activo de **RUN** indica la entrada al modo de funcionamiento; los cambios de parámetros están deshabilitados. La capacidad de cambiar entre los modos PROG / SPD / DRV está deshabilitada en el estado **RUN**. En el estado **STOP**, el cambio del modo de control se puede realizar mediante el botón **MODE**. **ERR** indica la existencia de errores. **BAT** indica batería baja dentro de la unidad. **USB** y **RS485** indican el proceso de una trama Modbus transmitida a través de USB y RS-485. En los modos **PROG** y **SPD**, los LEDs **IN0 ...IN7**indican la presencia de un nivel alto de una señal lógica en la entrada correspondiente. Los estados activos de los indicadores **OUT0 ...OUT11** indican el estado abierto de la salida del transistor (ver Fig. 5– Fig. 8). En el modo **DRV** (modo Paso/Dir) los LEDs de las entradas **IN0 ...IN7** indican la configuración de micropasos, las salidas **OUT0 ...OUT3** muestran la configuración de la corriente de funcionamiento, **OUT4 ...OUT7** muestran la configuración de la corriente de mantenimiento.

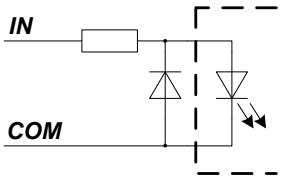


Fig. 5. Entradas IN0 e IN1

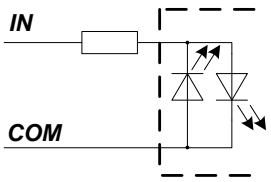


Fig. 6. Entradas IN2 – IN7

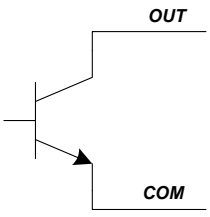


Fig. 7. Salidas OUT0 – OUT7

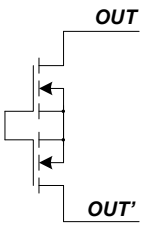


Fig. 8. Salidas OUT10 y OUT11

La posición de los potenciómetros **0, 1, 2 (SPEED)** se convierte en valores de 12 bits, que son accesibles para su uso posterior en un programa de ejecución y se pueden leer mediante un comando Modbus. En el modo **SPD** (control de velocidad), el potenciómetro **2 (SPEED)** se utiliza para ajustar la velocidad de rotación, el potenciómetro 0 (tasa de aceleración y deceleración, 1 (corriente

de trabajo del motor. En el modo **DRV** (Pulse/Dir), el potenciómetro 2 se utiliza para configurar los micropasos, 0 – corriente de trabajo del motor, 1 - corriente de mantenimiento.

El interruptor **RUN/STOP** se utiliza para iniciar y detener la ejecución del programa en los modos de control **PROG** y **SPD**. Se utiliza para cambiar entre la configuración de parámetros y el funcionamiento en el modo de control **DRV**.

La Fig. 2(a–c) muestra los terminales del controlador y su propósito dependiendo del modo de control.

2. ESPECIFICACIONES TÉCNICAS

Característica		Valor	
		mín	máx
Tensión de alimentación, VCC	SMSD-1.5Modbus ver.3	12	36
	SMSD-4.2Modbus y SMSD-8.0Modbus	12	49
Máx. corriente de salida por fase, A	SMSD-1.5Modbus ver.3	0.15	1.5
	SMSD-4.2Modbus	1.0	4.2
	SMSD-8.0Modbus	2.8	8.0
Alto nivel de entradas lógicas, VCC			2.4
Nivel bajo de entradas lógicas, VCC		0,7	
Tensión recomendada (mejor tiempo de respuesta) de las entradas lógicas, VCC			5
Máx. nivel de voltaje de las entradas lógicas, VCC			24(1)
Tensión de salida del transistor lógico, VCC			80
Corriente máxima de salida del transistor lógico, mA			50
Voltaje de la salida del relé lógico, VCA/VCC			350
Corriente máxima de salida de relé lógico CC (CA/CC), mA			250 (~120)
Corriente máxima de salida adicional +5VDC, mA			200
Duración del nivel de voltaje alto/bajo de las señales	Señal STEP en modo de control Paso/Dir (DRV), ns	250(2)	
	señales en las entradas IN0 e IN1, ns	70(2)	
	señales en las entradas IN2…IN7, µs	5(2)	
Frecuencia de generación de la señal PWM, Hz		0,3	5000
Tiempo de instrucción base, µs(3)		20	

(1) – Al utilizar una tensión de más de 8V para las entradas IN2.. .IN7 y más de 12V para las entradas IN0.. .IN1, es necesario instalar resistencias limitadoras de corriente.
– Resistencias recomendadas para las entradas IN2.. .IN7: no menos de 270 Ohmios cuando se utiliza para una fuente de señal de 12V y no menos de 1 kOhm cuando se utiliza para 24V.
– La resistencia recomendada para las entradas IN0 e IN1 es de 9.1 kOhm cuando se utiliza una fuente de señal lógica de 24V.

(2) en condición de nivel de alto voltaje de 5VDC

(3) - sin tener en cuenta el retorno a la línea cero, la configuración de las salidas y la lectura de las entradas

Información adicional

№	Característica	Valor
1	Posibles velocidades de transmisión para la transmisión de datos RS-485	1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, 256000
2	Posibles ajustes de micropasos	1/1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/128, 1/256
3	Protocolo de comunicación	Modbus RTU, Modbus ASCII
4	Lenguaje de programación	IL (Lista de Instrucciones), LD* (Diagrama de Escalera)

* *Se necesita un software especial para convertir LD a IL antes de escribir el programa en el controlador*

3. SECUENCIA DE OPERACIÓN

La secuencia de operación del controlador es la siguiente:

- lectura de dispositivos externos (entradas lógicas, bobinas Modbus);
- procesamiento del programa del usuario;
- configuración de nuevos estados de los dispositivos de salida (salidas lógicas, entradas discretas Modbus, ejecución de movimiento).

El programa del usuario consiste en una secuencia de instrucciones de control (comandos) que determinan la funcionalidad final. El controlador ejecuta los comandos secuencialmente, uno por uno. El paso total del programa se repite continuamente. El tiempo requerido para un paso del programa se llama tiempo de ciclo, y los pasos del programa se llaman escaneo cíclico.

Los controladores pueden operar en modo de tiempo real y se pueden utilizar tanto para construir nodos de automatización locales como sistemas de I/O distribuidos con la organización del intercambio de datos a través de la interfaz RS-485 con protocolo Modbus.

Ofrecemos software especial para ensamblar y depurar programas de usuario, que no requiere recursos informáticos significativos y es una herramienta sencilla para los usuarios. Se utilizan dos lenguajes de programación: LD (lógica de contactos en escalera o diagramas de escalera) e IL (lista de instrucciones).

3.1. El principio de funcionamiento de los circuitos de contactos de relé y diagramas de escalera en el controlador

El lenguaje de los diagramas de escalera es un derivado del diagrama de circuito de contacto de relé en una representación simplificada. Los circuitos de contacto de relé en el controlador tienen un conjunto de componentes básicos, tales como: contacto normalmente abierto, contacto normalmente cerrado, bobina (salida), temporizador, contador, etc., así como instrucciones aplicadas: funciones matemáticas, comandos de control de motor, procesamiento de datos y una gran cantidad de funciones y comandos especiales. Podemos asumir que el controlador son decenas o cientos de relés, contadores, temporizadores y memoria separados. Todos estos contadores, temporizadores, etc. físicamente no existen, sino que son modelados por el procesador y están diseñados para intercambiar datos entre funciones integradas, contadores, temporizadores, etc.

El lenguaje lógico de contacto de relé en el controlador es muy similar a los circuitos eléctricos básicos de contacto de relé si comparamos sus símbolos gráficos utilizados. Puede haber dos tipos de lógica en los circuitos de contacto de relé: combinada, es decir, circuitos que constan de fragmentos que son independientes entre sí, y lógica secuencial, donde todos los pasos del programa están interconectados y el circuito no se puede paralelizar.

Lógica combinada

El primer segmento del circuito consta de un contacto normalmente abierto X0 y una bobina Y0, que determina el estado de la salida Y0. Cuando el estado del contacto X0 está abierto (lógico "0"), el estado de la salida Y0 también está abierto (lógico "0"). Cuando el contacto X0 está cerrado, la salida Y0 también cambia su estado a cerrado (lógico "1").

El segundo segmento del circuito consta de un contacto normalmente cerrado X1 y una bobina Y1, que determina el estado de la salida Y1. En el estado normal del contacto X1, la salida Y1 estará cerrada (lógico "1"). Cuando el estado del contacto X1 cambia a abierto, la salida Y1 también cambia su estado a abierto.

En el tercer segmento del circuito, el estado de la salida Y2 depende de una combinación de los estados de los tres contactos de entrada X2, X3 y X4. La salida Y2 está cerrada cuando X2 está apagado y X4 está encendido, o cuando X3 y X4 están encendidos.

El esquema general es una combinación de tres segmentos, que operan independientemente uno del otro.

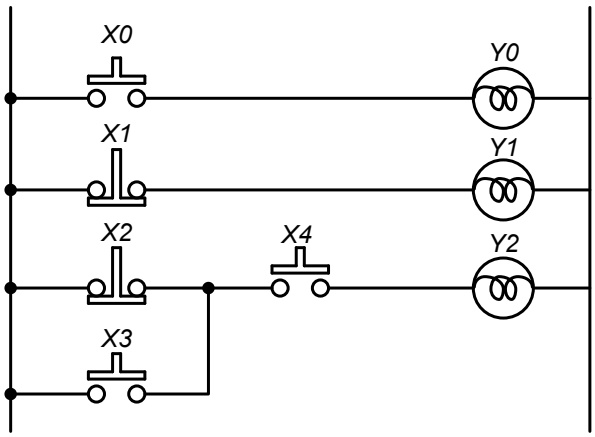


Fig. 9. Circuito de contacto de relé

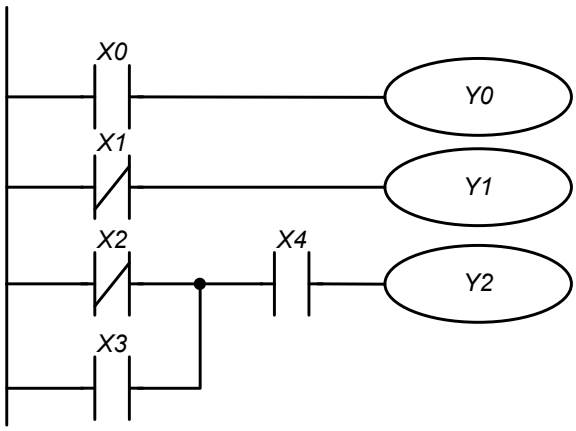


Fig. 10. Diagrama de escalera en el controlador

Lógica secuencial

En los circuitos lógicos secuenciales, el resultado de la ejecución en un paso anterior es una condición de entrada para el siguiente paso. En otras palabras, una salida en el paso anterior es una entrada en el paso siguiente.

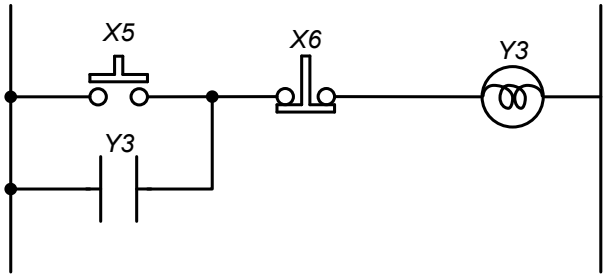


Fig. 11. Circuito de contacto de relé

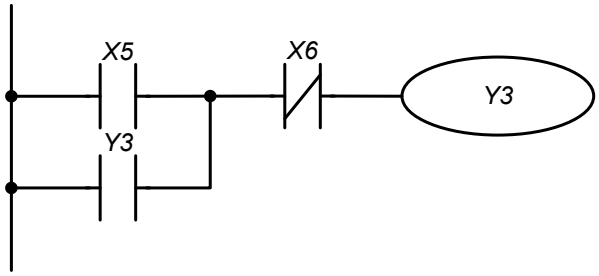


Fig. 12. Diagrama de escalera en el controlador

Cuando el contacto X5 está cerrado, la salida Y3 cambia su estado a cerrado. Sin embargo, cuando X5 se abre de nuevo, Y3 mantiene su estado cerrado hasta el momento en que X6 se abre. En este circuito, la salida Y3 se autobloquea.

3.2. Diferencias entre la lógica de los circuitos reales de contacto de relé y los diagramas de escalera en el controlador

Todos los procesos de control especificados se realizan simultáneamente (en paralelo) en los circuitos eléctricos convencionales de contacto de relé. Cada cambio en el estado de las señales de entrada afecta inmediatamente al estado de las señales de salida.

Un cambio en el estado de las señales de entrada que ocurrió durante el paso actual del programa en el controlador se reconoce solo en el siguiente ciclo del programa. Este comportamiento se suaviza debido al corto tiempo de ciclo.

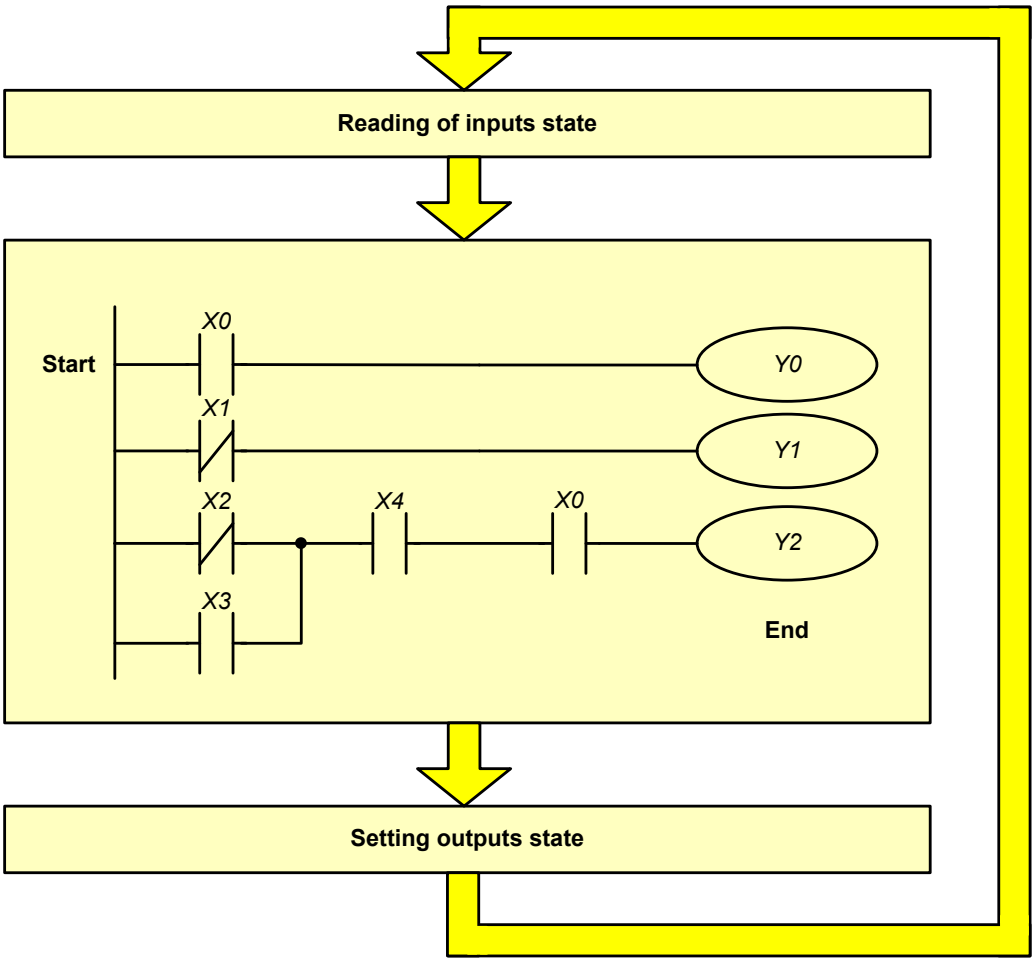


Fig. 13. Secuencia de operación en el controlador

Durante el funcionamiento, el controlador lee continuamente el estado actual de las entradas y cambia el estado de las salidas (encendido/apagado) según el programa del usuario.

La Fig. 13muestra un diagrama de flujo de un ciclo del programa.

En la primera etapa, el controlador lee el estado de las entradas físicas y virtuales (cuyos estados se establecen a través de las bobinas Modbus) y las almacena en búfer en la memoria interna del controlador.

En la segunda etapa, se procesa el estado de las entradas almacenadas en búfer y el estado de las salidas en la memoria del controlador cambia de acuerdo con un programa de usuario determinado. Por lo tanto, todas las modificaciones de las salidas se producen sin cambiar su condición física. Durante esta etapa, el estado de las entradas físicas y virtuales puede cambiar, pero el siguiente almacenamiento en búfer del estado actualizado ocurrirá en la primera etapa del siguiente ciclo del programa del usuario.

En la tercera etapa, el controlador cambia el estado de las salidas físicas y virtuales.

Otra diferencia entre la lógica de contacto de relé del controlador y los circuitos eléctricos convencionales de contacto de relé es que los programas de usuario se ejecutan en filas solo de izquierda a derecha y de arriba a abajo. Por ejemplo, un circuito con una dirección de corriente inversa (sección a-b en) resultará en un error durante la compilación en el controlador.

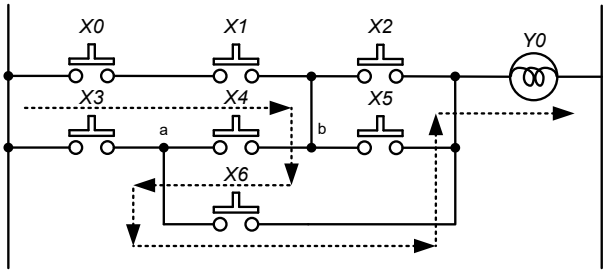


Fig. 14. Circuito de contacto de relé eléctrico

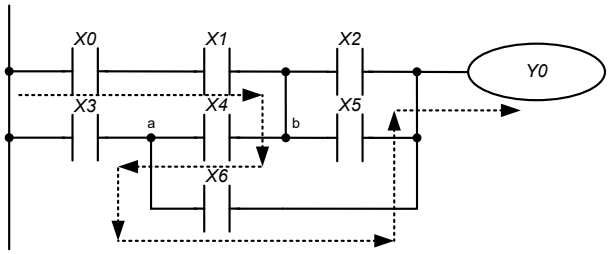


Fig. 15. Circuito de relé-contacto del controlador

Un error en la tercera fila

3.3. Operandos

Todos los objetos internos (dispositivos) del controlador – operandos – se dividen en unos pocos tipos diferentes y tienen direcciones. Cada tipo tiene su designación y formato, que determina qué espacio ocupa en la memoria del controlador. Así, por ejemplo, los relés de entrada se denominan "X" y tienen un formato de 1 bit, y los registros de datos de propósito general se denominan "D" y tienen un formato de 16 bits (1 palabra) o 32 bits (2 palabras).

Tipo y nombre del operando		Descripción
Entrada	X	Relés de entrada. Determinan el estado de los dispositivos de bit externos, que están conectados a los terminales de entrada del controlador, y el estado de las entradas virtuales, cuyo estado se puede configurar a través del protocolo Modbus. Estos operandos pueden tomar uno de dos valores posibles: 0 o 1. El direccionamiento es octal: X0, X1.. . X7, X10, X11, ...

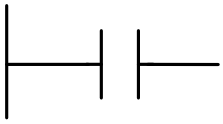
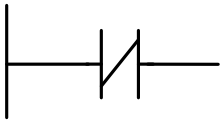
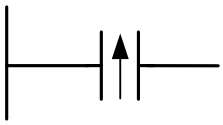
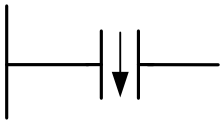
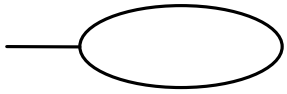
Salida	Y	Relés de salida. Determinan el estado de los terminales de salida del controlador, a los que se conecta la carga, y el estado de las salidas virtuales, cuyo estado se puede leer a través del protocolo Modbus. En el programa pueden ser contactos o bobinas. Estos operandos pueden tomar uno de dos valores posibles: 0 o 1. El direccionamiento es octal: Y0, Y1.. .Y7, Y10, Y11, ...
Marcador	M	Relés auxiliares. Es una memoria para resultados binarios intermedios. En el programa de usuario pueden ser contactos o bobinas. Estos operandos pueden tomar uno de dos valores posibles: 0 o 1. El direccionamiento es decimal: M0, M1.. .M7, M8, M9,...
Temporizador	T	Relé de tiempo. El programa se puede utilizar para el almacenamiento del valor actual del temporizador y tiene un formato de 16 bits. Además, estos operandos se pueden utilizar como contactos y toman uno de dos estados: 0 o 1. La direccionamiento es decimal: T0, T1 ...T64
Contador	C	Se utiliza un contador para implementar el conteo. El programa se puede utilizar para el almacenamiento del valor actual del contador y tiene un formato de 16 bits o 32 bits, y también se puede utilizar como un contacto y tomar uno de dos significados posibles: 0 o 1. La direccionamiento es decimal: C0, C1.. .C66
Constante decimal	K	Determina un número en decimal
Constante hexadecimal	H	Determina un número hexadecimalmente
Constante de punto flotante	F	Determina un número de punto flotante
Registro de datos	D	Almacenamiento de datos. Formato de 16 bits o 32 bits. El direccionamiento es decimal: D0, D1,..., D391. Para datos de 32 bits, un elemento ocupa dos registros. Como ejemplo, para leer datos de 32 bits del registro D10, los datos se leen de los registros D10 y D11.
Registro índice	A	

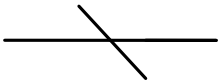
	B	Almacenamiento de datos para resultados intermedios e identificación de índice. Formato de 16 bits. Direccionamiento: A0 – A7, B0 – B7 decimal
Puntero	P	Una dirección para la llamada al subprograma. Decimal.
Puntero de interrupción	I	Dirección del manejo de interrupciones. Decimal.

3.4. Símbolos gráficos de las instrucciones de control en un diagrama de escalera

Un circuito de contacto de relé consta de una línea vertical a la izquierda y líneas horizontales que se extienden hacia la derecha. La línea vertical a la izquierda es una línea de bus, las líneas horizontales son líneas de comando = pasos. Hay símbolos de condiciones de entrada en las líneas de comando que conducen a comandos (instrucciones) ubicados a la derecha. Las combinaciones lógicas de estas condiciones de entrada determinan cuándo y cómo se ejecutan los comandos de la derecha.

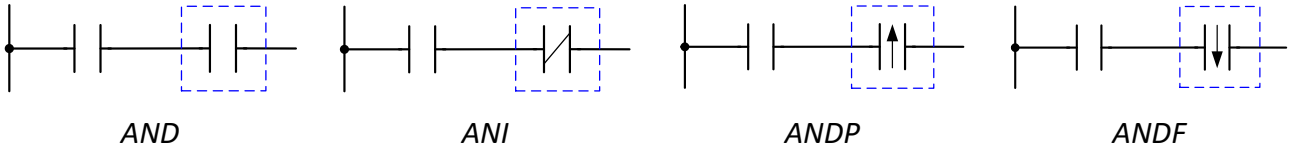
Los siguientes símbolos se utilizan en los circuitos de contactos de relé:

Símbolo	Descripción	Comando	Operandos
	Contacto de entrada – normalmente abierto	LD	X, Y, M, T, C
	Contacto de entrada – normalmente cerrado	LDI	X, Y, M, T, C
	Contacto de pulso de entrada – flanco ascendente	LDP	X, Y, M, T, C
	Contacto de pulso de entrada – flanco descendente	LDF	X, Y, M, T, C
	Señal de salida (bobina)	OUT	Y, M

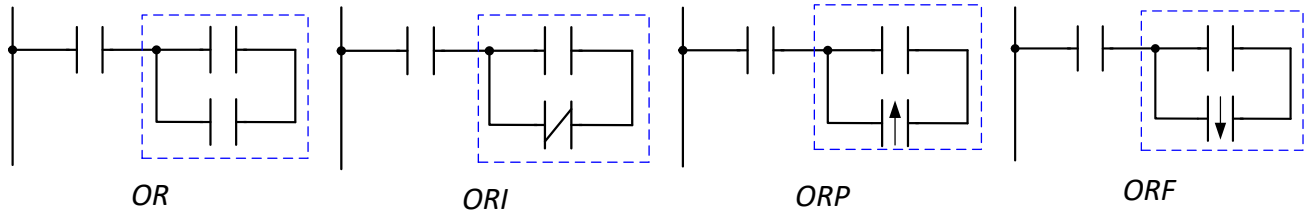
	Instrucciones básicas y de aplicación	Consulte el capítulo 7. Instrucciones de aplicación	Consulte el capítulo 7.Instrucciones de aplicación
	Inversión lógica	INV	-

Los contactos de entrada se pueden combinar en bloques en serie y en paralelo:

Conexiones seriales:



Conexiones en paralelo:



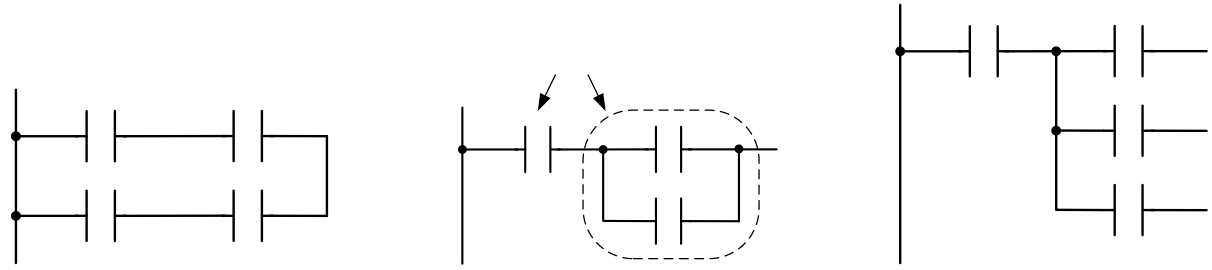


Fig. 16. Símbolos gráficos de las instrucciones de control

El escaneo del programa comienza en la esquina superior izquierda del diagrama y termina en la esquina inferior derecha. El siguiente ejemplo ilustra la secuencia de un programa:

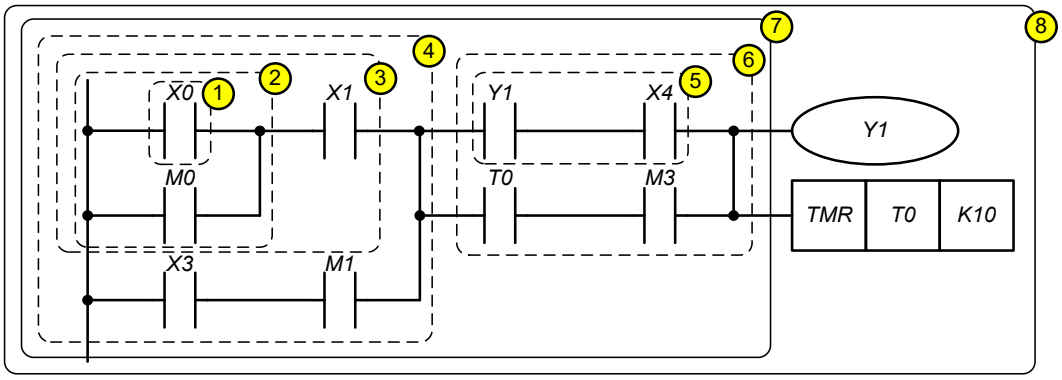
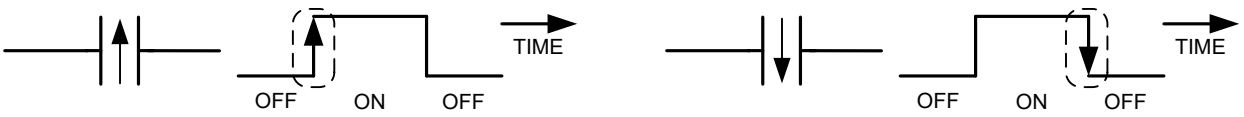


Fig. 17. Secuencia de un programa

1	LD	X0	
2	OR	M0	
3	AND	X1	
4	LD	X3	
	AND	M3	
	ORB		
5	LD	Y1	
	AND	X4	
6	LD	T0	
	AND	M3	
	ORB		
7	ANB		
8	OUT	Y1	
	TMR	T0	K10

Los símbolos de las señales de entrada con flanco ascendente (cuando una señal cambia de 0 a 1) y con flanco descendente (cuando una señal cambia de 1 a 0) se explican a continuación:



Contacto de flanco ascendente Contacto de flanco de bajada

Fig. 18. Filtrado de flancos

Los comandos de bloque lógico ANB y ORB no corresponden a condiciones específicas en el circuito de contactos de relé, sino que describen la relación entre los bloques. El comando ANB realiza la operación **LÓGICA AND** en las condiciones de ejecución producidas por dos bloques lógicos.

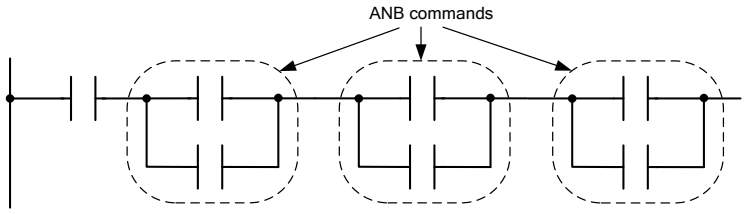


Fig. 19. Instrucción ANB

El comando ORB realiza una operación **LÓGICA OR** en las condiciones de ejecución producidas por dos bloques lógicos.

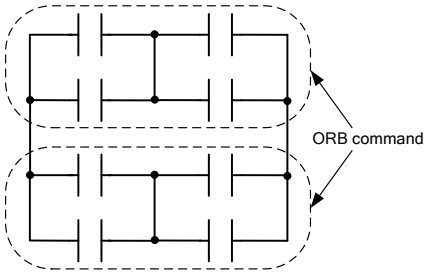


Fig. 20. Instrucción ORB

3.5. Convertir circuitos de contactos de relé (LD) a código mnemotécnico (IL)

La siguiente figura muestra un programa presentado en forma de símbolos de contacto de relé (LD) y una lista de instrucciones - código mnemotécnico (IL). La figura muestra la secuencia de conversión del diagrama de escalera (LD) en el código ejecutado por el controlador (IL).

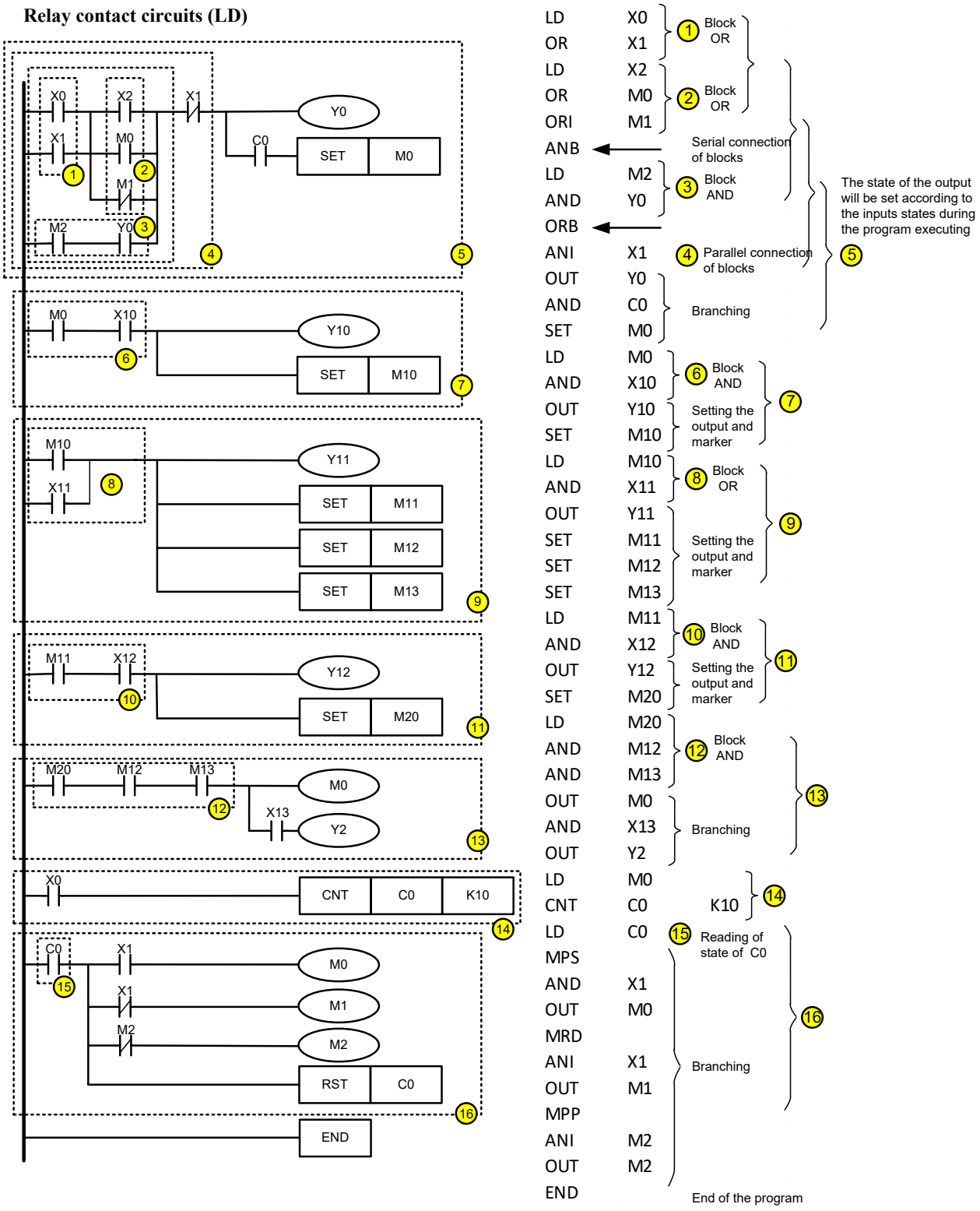


Fig. 21. Conversión de LD a IL

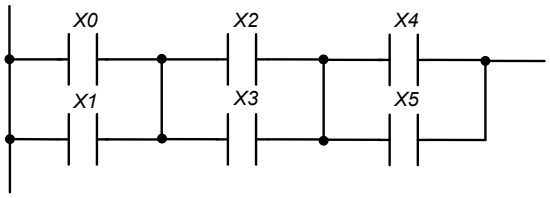
El procesamiento del circuito de contactos de relé comienza en la esquina superior izquierda y termina en la inferior derecha, sin embargo, puede haber excepciones y varias opciones para convertir a código mnemotécnico, como se muestra en los siguientes ejemplos:

Ejemplo 1

El siguiente diagrama de escalera se puede convertir en una lista de instrucciones de dos maneras diferentes, pero el resultado será idéntico (Fig. 22).

El primer método de codificación es el más preferible ya que el número de bloques lógicos es ilimitado.

El segundo método está limitado por el número máximo de bloques lógicos (el número máximo de bloques es 8).

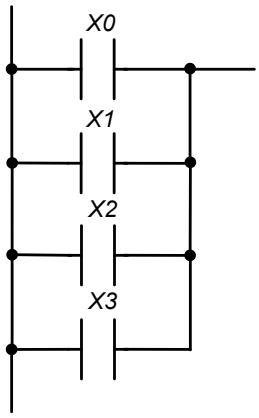


Método 1		Método 2	
LD	X0	LD	X0
OR	X1	OR	X1
LD	X2	LD	X2
OR	X3	OR	X3
ANB		LD	X4
LD	X4	OR	X5
OR	X5	ANB	
ANB		ANB	

Fig. 22. Diferentes métodos de uso de las instrucciones ANB

Ejemplo 2

A continuación se muestran diferentes métodos de codificación de contactos conectados en paralelo (Fig. 23).



Método 1		Método 2	
LD	X0	LD	X0
OR	X1	LD	X1
OR	X2	LD	X2
OR	X3	LD	X3
		ORB	
		ORB	
		ORB	

Fig. 23. Diferentes métodos de uso de las instrucciones ORB

El primer método de conversión del diagrama de escalera en una lista de instrucciones es el más preferible desde el punto de vista del uso de la RAM del controlador.

4. Funcionalidad del controlador

4.1. Descripción general de los operandos

Tipo	Operando			Rango de direcciones		Función
Relés (memoria de 1 bit)	**X**	Relés de entrada externos	Entradas físicas	X0…X7	Máx. 128 puntos	Entradas del controlador
			Entradas virtuales (Bobina Modbus)	X10…X177		
	Y	Relés de salida externos	Salidas físicas	Y0…Y7	Máx. 128 puntos	Salidas del controlador
			Salidas virtuales (Entradas Discretas Modbus)	Y10…Y177		
	M	Relés internos (marcadores)	Uso general	M0…M99, M111…M127	Máx. 128 puntos	Memoria binaria intermedia. Corresponde a los relés intermedios en los circuitos eléctricos.
			Propósito especial	M100…M110		
	T	Temporizadores	Resolución 100 ms	T0…T47 (T46, T47 – acumulativo)	Máx. 64 puntos	Usados como contactos (T), que se cierran cuando el temporizador correspondiente alcanza su valor establecido (comando TMR)
			Resolución 10 ms	T48…T63 (T62, T63 – acumulativo)		
	C	Contadores	Incremental de propósito general	C0…C63	Máx. 66 puntos	Usados como contactos (C), que se cierran cuando el contador correspondiente alcanza su valor establecido (comando CNT)
			Pulsos externos	C64, C65		
Registros	**T**	Valor actual del temporizador		64 puntos (T0…T63)		Registros para el almacenamiento de los

						valores actuales del temporizador
	C	Valor actual del contador		66 – Contadores de 32 bits		Registros para el almacenamiento de los valores actuales del contador
	D	Registros de datos	Uso general	D0…D319, D385…D391	Máx. 384 puntos	Se utiliza para almacenar datos. Los registros especiales configuran el controlador y muestran su estado.
			No volátil(1)	D320…D351		
			Especial	D352…D384		
	A	Registros índice	Menores de 16 bits	A0…A7	Máx. 16 puntos	Puede usarse para indicación de índice
	B		16 bits principales	B0…B7		
Punteros	**P**	Punteros para las instrucciones CALL, CJ		32 puntos (P0…P31)		Etiquetas para instrucciones de transiciones y subprogramas
	I	Interrupciones	Comunicación	I0	Máx. 15 puntos	Etiquetas para el subprograma para el procesamiento de interrupciones
			Temporizado	I1…I100 (Máx. 4 puntos)		
			Externo	I1000…I1007		
			Del controlador	I2000, I2001		
Constantes	**K**	Constantes decimales		K-32768 ...K32767 (funciones de 16-bit) K-2147483648 ...K2147483647 (funciones de 32-bit)		
	H	Constantes hexadecimales		H0000...HFFFF (funciones de 16 bits) H00000000...HFFFFFFFF (funciones de 32-bit)		
	F	Constante de punto flotante		F±1.175494351 E-38… 3.402823466 E+38 (Funciones de 32 bits solamente)		

(1) – El almacenamiento de datos es proporcionado por la fuente de alimentación interna CR2032.

4.2. Direccionamiento y funciones de las entradas [X] y salidas [Y]

Las entradas y salidas en el programa de usuario se representan mediante operandos. Al especificar la dirección del operando, es posible referirse a las entradas y salidas físicas y virtuales del controlador durante la programación.

Las entradas/salidas discretas se direccionan en el sistema octal, lo que significa que los números 8 y 9 no se utilizan para entradas y salidas.

Función de los relés de entrada X

Los relés de entrada X leen el estado de los dispositivos físicos externos (botones, interruptores, contactos de relé, etc.) conectados directamente a los terminales de entrada del controlador. Cada entrada X se puede utilizar en el programa un número ilimitado de veces.

Función de los relés de salida Y

Los relés de salida Y controlan el estado de los contactos de salida físicos del controlador y, por lo tanto, los dispositivos de carga (lámparas, bobinas de relé, etc.) conectados directamente a los terminales de salida del controlador.

Cada salida Y se puede utilizar en el programa un número ilimitado de veces, pero se recomienda utilizar la bobina de salida Y en el programa no más de una vez, porque cuando la bobina Y se utiliza varias veces, el estado de salida está determinado por la última Y en el escaneo.

El estado de las señales de I/O se puede leer en el programa mediante diferentes instrucciones.

El proceso de manejo de señales de I/O en el controlador:

Entradas:

1. El controlador lee el estado de los dispositivos de entrada externos y lo almacena al comienzo de cada ciclo de escaneo.
2. Los cambios en el estado de entrada durante el ciclo no se aceptarán si el pulso de entrada es muy corto (menor que el tiempo de un escaneo).

Programa:

3. El controlador ejecuta el programa comenzando desde la línea 0 y almacena el estado de todos los operandos en la memoria del objeto.

Salidas:

4. Después de ejecutar la instrucción END, el estado del relé de salida Y se escribe en la memoria de las salidas y los estados de los contactos de salida cambiarán.

4.3. Direccionamiento y función de los relés internos [M]

Para almacenar los resultados binarios de las uniones lógicas (estados de señal "0" o "1"), se utiliza una memoria intermedia (relé interno) dentro del programa. Corresponden a relés intermedios en sistemas de control basados en lógica de relés.

Se utilizan dos tipos de relés internos en el controlador:

1. De propósito general, que no se guardan cuando se apaga la alimentación;
2. De propósito especial, que proporciona al usuario una funcionalidad adicional.

Los relés internos se programan como salidas. Se pueden utilizar en el programa un número ilimitado de veces. El direccionamiento de los relés internos es en formato decimal.

Designación de marcadores especiales:

Marcador	Función
M100…M107	Estos relés auxiliares se utilizan solo en conjunto con las interrupciones de las entradas externas I1000 ... I1007, respectivamente. El valor del marcador corresponde al estado de la entrada física (IN0 ... IN7) en el momento en que se procesó la interrupción (I1000 ... I1007). Los valores X0 ... X7 se actualizan solo al comienzo del siguiente escaneo del programa de usuario. Por ejemplo, después de entrar en la rutina de interrupción I1004, es posible determinar el estado de la entrada IN4 solicitando el estado de M104 (LD M104) (el valor de X4 no es relevante en este caso).
M108	El flanco ascendente de este relé auxiliar indica la finalización de la inicialización de los periféricos del controlador. Durante el trabajo posterior, el marcador mantiene un valor alto. Reiniciar el marcador reinicializa el controlador. Por ejemplo, después de redefinir las salidas por los generadores de señal PWM y las entradas por los contadores de pulsos, se requiere una reinicialización. En este caso, es necesario reiniciar M108.
M109	Ajustar el marcador activa la indicación “ERR” en el panel frontal del controlador, reiniciarlo la desactiva.
M110	Configurar este marcador y luego reiniciar M108 resultará en un reinicio completo del controlador.

4.4. Direccionamiento y función de los temporizadores [T]

Algunos procesos de control requieren un relé de tiempo. Muchos sistemas controlados por relés utilizan relés de tiempo que se activan con retardo. El controlador utiliza elementos de memoria interna para estos fines, llamados temporizadores. Las características de los temporizadores se pueden determinar en el programa.

El direccionamiento de los temporizadores es decimal.

T	Temporizadores	Resolución 100 ms	T0…T47 (T46, T47 – acumulativos)	Máx. 64 puntos
		Resolución 10 ms	T48…T63 (T62, T63 – acumulativos)	

El ajuste de tiempo requerido se determina mediante una constante decimal K, que indica el número de pasos de tiempo contados (discretos).

Ejemplo: un temporizador de resolución de 100 ms configurado como K5, el valor real de la configuración será 5 x 100 = 500 ms.

El temporizador funciona con un retardo de conexión. Se activa con el estado del contacto = 1. Después de contar el valor de tiempo establecido, el temporizador establece el contacto de entrada T correspondiente en el estado "1". El temporizador vuelve al estado de apagado y restablece su valor actual cuando su contacto de entrada se establece en "0".

El ajuste del tiempo también se puede realizar indirectamente mediante un número decimal registrado previamente en el registro de datos D.

En los controladores, el temporizador comienza a contar inmediatamente cuando ejecuta el comando TMR.

Explicación del funcionamiento de dos tipos de temporizadores:

Temporizador de propósito general

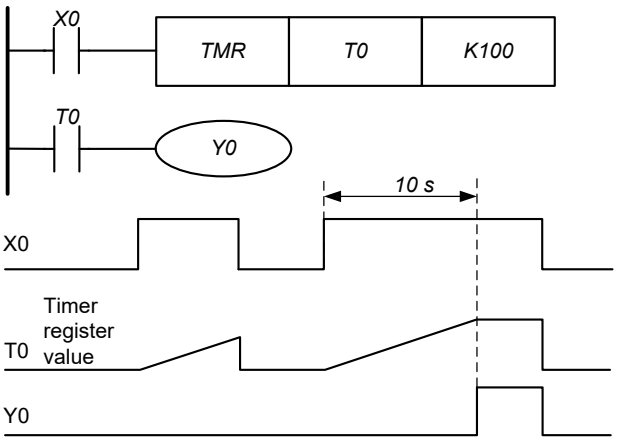


Fig. 24. Principio de funcionamiento del temporizador de propósito general

Cuando la entrada X0 toma el estado "1", comienza el conteo del tiempo establecido. Después de alcanzar los 10 segundos programados, la salida Y0 toma el estado "1". El temporizador se apaga y el registro T0 se restablece a cero tan pronto como la entrada X0 toma el estado "0".

Temporizador acumulativo

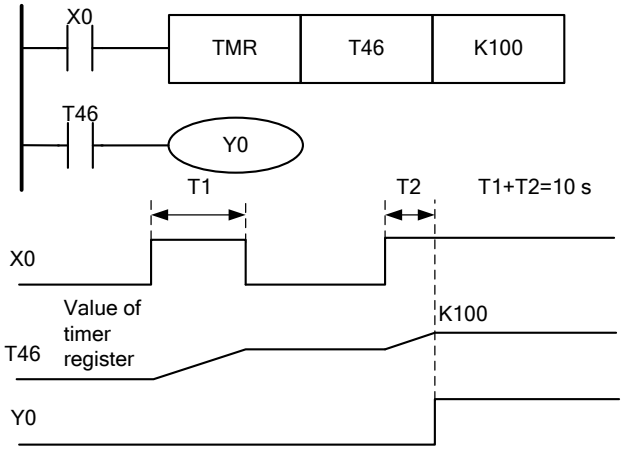


Fig. 25. Principio de funcionamiento del temporizador acumulativo

Además de los temporizadores de propósito general, el controlador cuenta con temporizadores acumulativos, que, después de deshabilitar la conexión lógica de control, guardan el valor de tiempo acumulado.

4.5. Direccionamiento y función de los contadores [C]

Es necesario contar impulsos (sumar o restar) en algunos procesos de control. Muchos sistemas controlados por relés utilizan contadores de pulsos para este propósito. El controlador utiliza dos tipos de elementos de memoria interna (contadores).

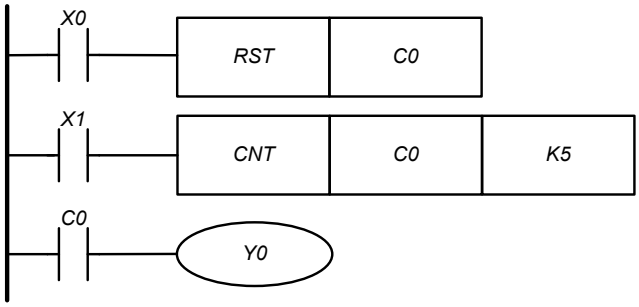
El direccionamiento de los temporizadores es decimal.

C	Contadores	Incremental de propósito general	C0…C63	Máx. 66 puntos
		Pulsos externos (hardware)	C64, C65	

Función de los contadores:

Cuando la señal de entrada del contador cambia su estado de 0 a 1, el valor actual del contador C se incrementa en uno. Cuando se vuelve igual al valor establecido (punto de ajuste), el contacto de trabajo del contador se activa.

```
LD      X0
RST     C0
LD      X1
CNT     C0      K5
LD      C0
OUT     Y0
```



El contador se reinicia cuando X0=1: el valor actual del registro C0 = 0, el contacto C0 está abierto.

Después de cambiar X1 de 0 a 1, el valor de C) se incrementa en uno.

Cuando el valor del registro C0 = 5, los contactos C0 y Y0 se cierran y todos los pulsos siguientes en la entrada no se cuentan.

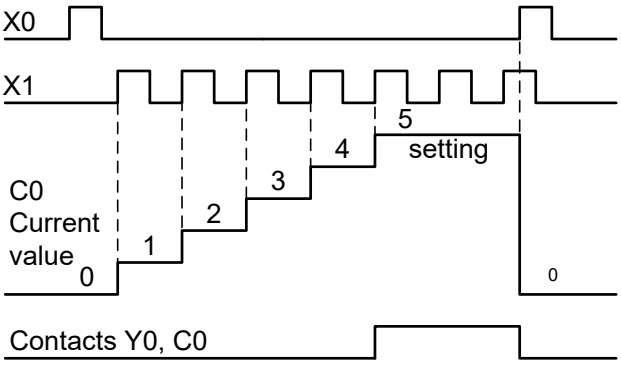
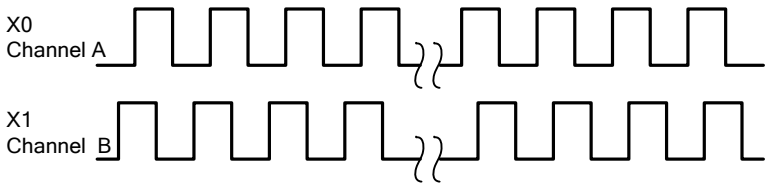


Fig. 26. Principio de funcionamiento del contador

Los contadores de propósito general no cuentan por encima de un umbral, a diferencia de los contadores de hardware, que no cuentan con la señal de entrada, sino solo con el estado físico de la entrada discreta a la que se refieren. Dependiendo del valor del registro de configuración D355, los contadores se pueden configurar de la siguiente manera:

Valor del registro D355	Configuración
0	El valor predeterminado. X0 y X1 (IN0 e IN1) operan como entradas discretas.
1	La entrada discreta X0 se refiere al contador C64, que cuenta los flancos de subida de los pulsos. X1 funciona como una entrada discreta.
2	La entrada discreta X0 se refiere al contador C64, que cuenta los flancos de bajada de los pulsos. X1 opera como una entrada discreta.
3	La entrada discreta X0 se refiere al contador C64, que cuenta tanto los flancos de subida como los flancos de bajada de los pulsos. X1 opera como una entrada discreta.
4	X0 opera como una entrada discreta. La entrada discreta X1 se refiere al contador C65, que cuenta los flancos de subida de los pulsos.
5	X0 opera como una entrada discreta. La entrada discreta X1 se refiere al contador C65, que cuenta los flancos de bajada de los pulsos.
6	Las entradas discretas X0 y X1 se refieren a los contadores C64 y C65, respectivamente. Los contadores cuentan los flancos de subida de los pulsos.
7	Las entradas discretas X0 y X1 se refieren a los contadores C64 y C65, respectivamente. C64 cuenta los flancos de bajada de los pulsos, C65 cuenta los flancos de subida de los pulsos.
8	Las entradas discretas X0 y X1 se refieren a los contadores C64 y C65, respectivamente. C64 cuenta tanto los flancos de subida como los flancos de bajada de los pulsos, C65 cuenta los flancos de subida de los pulsos.
9	Las entradas discretas X0 y X1 se refieren a los contadores C64 y C65. C64 cuenta los flancos de subida de los pulsos, C65 cuenta los flancos de bajada de los pulsos.
10	Las entradas discretas X0 y X1 se refieren a los contadores C64 y C65, respectivamente. Los contadores cuentan los flancos de bajada de los pulsos.
11	Las entradas discretas X0 y X1 se refieren a los contadores C64 y C65, respectivamente. C64 cuenta tanto los flancos de subida como los flancos de bajada de los pulsos, C65 cuenta los flancos de bajada de los pulsos.

12	Las entradas discretas X0 y X1 se refieren al contador C64 y operan como un codificador. Se aplica una señal de cuadratura a las entradas. 

4.6. Direccionamiento y función de los registros [D], [A], [B]

Registros de datos [D]

Los registros representan la memoria de datos dentro del controlador. Los registros pueden almacenar valores numéricos e información binaria uno tras otro.

Los datos se almacenan en un registro de 16 bits (D0, etc.), que puede almacenar un número de -32768 a +32767. La unión de dos registros de 16 bits da un "registro doble" de 32 bits (D0, D1, etc.), que puede almacenar un número de -2147483648 a +2147483647.

El direccionamiento de los registros de datos es decimal. Para los registros dobles (32 bits), el direccionamiento comienza con el registro inferior de 16 bits.

D	Registros de datos	Uso general	D0…D319, D385…D391	Máx. 384 puntos
		No volátil	D320…D351	
		Especial	D352…D384	

Existen los siguientes tipos de registros de datos:

Registros de datos de propósito general:

Estos registros se utilizan durante la ejecución del programa del usuario, los datos no se guardan cuando se apaga la alimentación.

Registros de datos no volátiles:

Los datos de estos registros se guardan en la memoria del controlador cuando se apaga la alimentación. La alimentación de la memoria la proporciona una fuente interna, CR2032.

Registros de índice:

Este registro se utiliza para almacenar resultados intermedios y para indicar operandos.

Registros especiales:

Estos registros se utilizan para configurar el controlador y para acceder a alguna funcionalidad especial. Los números de los registros especiales se indican en la tabla siguiente:

Registro	Función	Valores
D352	El registro contiene datos sobre la posición del potenciómetro “0” en el panel frontal del controlador.	0…4095
D353	El registro contiene datos sobre la posición del potenciómetro “1” en el panel frontal del controlador.	0…4095
D354	El registro contiene datos sobre la posición del potenciómetro “2”, “Speed”.	0…4095
D355	El registro configura los tipos de entrada IN0 e IN1, para más detalles, consulte la sección 4.5	0…12

<table>
<tr>
<td>D356</td>
<td>El registro configura los tipos de salidas OUT6 y OUT7 para la instrucción PWM (véase 7. Instrucciones de aplicación, instrucción PWM).
<table>
<tr><th rowspan="2">Valor</th><th rowspan="2">Tiempo de discretización, mks</th><th colspan="2">Función de salida</th></tr>
<tr><th>SAL6</th><th>SAL7</th></tr>
<tr><td>0</td><td>–</td><td>salida</td><td>salida</td></tr>
<tr><td>1</td><td>100</td><td>Generador PWM</td><td>salida</td></tr>
<tr><td>2</td><td>10</td><td>Generador PWM</td><td>salida</td></tr>
<tr><td>3</td><td>100</td><td>salida</td><td>Generador PWM</td></tr>
<tr><td>4</td><td>10</td><td>salida</td><td>Generador PWM</td></tr>
<tr><td>5</td><td>100</td><td>Generador PWM</td><td>Generador PWM</td></tr>
<tr><td>6</td><td>10</td><td>Generador PWM</td><td>Generador PWM</td></tr>
</table>
Consultar la sección 7. Instrucciones de aplicación(instrucción PWM) para obtener información detallada sobre la generación de la señal PWM.</td>
<td>0…6</td>
</tr>
<tr>
<td>D357…D384</td>
<td>Registros de estado y configuración de modo del controlador del motor paso a paso. Consulte la sección 8. «Instrucciones para el control del controlador de motor paso a paso» para obtener más detalles.</td>
<td>–</td>
</tr>
</table>

4.7. Registros índice [A], [B]

Los registros de índice se utilizan para indexar las direcciones de los operandos y cambiar los valores constantes.

Los registros de índice son registros de 16 bits.

En las instrucciones de 32 bits, los registros de índice A y B se utilizan en combinación. A contiene 16 bits de orden inferior y B contiene 16 bits de orden superior. El registro de índice A se utiliza como dirección de destino.

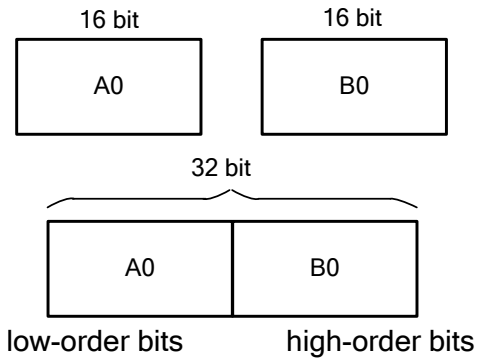


Fig. 27. Estructura del registro de índice

Ejemplo de transferencia de datos del registro de datos D5A0 al registro de datos D10B0:

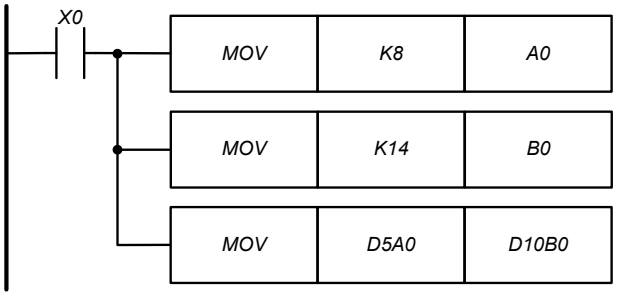


Fig. 28. Transferencia de datos usando registros de índice

Cuando X0 = 1: A0 = 8, B = 14

- La dirección de origen de la transferencia es D5A0 = 5 + 8 = D13
- La dirección de destino es D10B0 = 10 + 14 = 24.
- De esta manera, los datos se transfieren del registro D13 al registro de datos D24

Los registros de índice se pueden utilizar para operaciones de transferencia de datos y comparación junto con operandos de bytes y operandos de bits.

También es posible indexar constantes de la misma manera. Al indexar constantes, es necesario utilizar el símbolo "@". Por ejemplo: MOV K10 @ A0 D0B0.

4.8. Punteros [P], [I].

P	Punteros de instrucción CALL, CJ		32 puntos (P0…P31)		Etiquetas o marcas para comandos de transición o llamada de subprogramas
I	Interrupciones	Comunicación	I0	Máx. 15 puntos	Etiquetas para el subprograma para el procesamiento de interrupciones
		Temporizado	I1…I100 (Máx. 4 puntos)		
		Externo	I1000…I1007		
		Controlador	I2000, I2001		

Los punteros (P)se utilizan en combinación con las instrucciones CJ (transiciones) o CALL (subprogramas). Estos punteros son direcciones de ubicaciones de lugares o subprogramas, que fueron marcados.

Un ejemplo de ejecución de una instrucción de salto CJ:

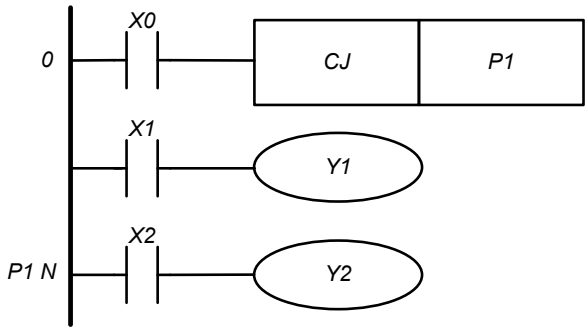
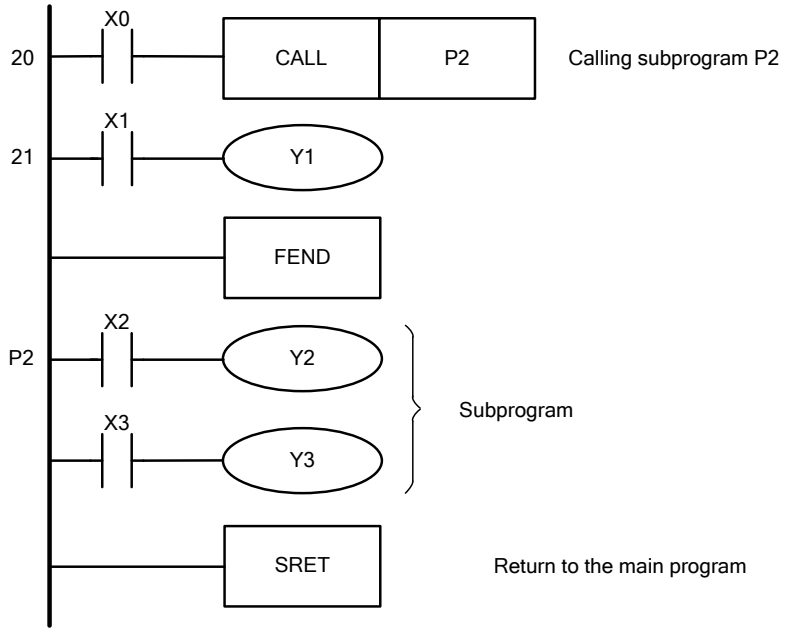


Fig. 29. Implementación de la instrucción CJ

Cuando X0 = 1, después de la ejecución de la línea 0, el programa va inmediatamente a la línea con el puntero P1, y las líneas ubicadas entre ellas no se ejecutan.

Si X0 = 0, el programa se ejecuta normalmente paso a paso.

Un ejemplo del uso de subprogramas:



Cuando X0 = 1 en la línea 20, la ejecución del programa va directamente a la línea marcada como P2, el subprograma se ejecuta y, después del comando SRET, la ejecución del programa regresa a la línea 21.

Fig. 30. Implementación de la instrucción CALL

Los punteros de interrupción (I) se utilizan con las instrucciones EI, DI e IRET para interrumpir la ejecución del programa principal. Existen los siguientes tipos de interrupciones:

1. Interrupción de comunicación: si el controlador recibe una trama de difusión a través del protocolo Modbus, inmediatamente (independientemente del ciclo de escaneo) va al subprograma de procesamiento de interrupciones, que está marcado con el puntero I0. Vuelve al programa principal después de que se ejecute la instrucción IRET.
2. Interrupción temporizada: el subprograma de procesamiento de interrupciones se ejecuta automáticamente a intervalos de tiempo especificados de 10 a 1000 ms en incrementos de 10 ms. En total, es posible tener hasta 4 interrupciones temporizadas. Como ejemplo, las interrupciones con punteros I10, I50, I80 e I100 se ejecutarán una vez cada 100 ms, 500 ms, 800 ms y 1 s, respectivamente. La ejecución regresa al programa principal después de la instrucción IRET.
3. Interrupciones externas: cuando la señal en la entrada IN0 ...IN7 cambia de 0 a 1 o de 1 a 0, el controlador inmediatamente pasa a la ejecución del subprograma de procesamiento de interrupciones con el puntero I correspondiente (IN0 → I1000, IN1 → I1001, etc.). El retorno al programa principal ocurre después de que se ejecuta la instrucción IRET.

4. Interrupción del controlador: cuando ocurre un error durante la conmutación de las fases del motor, lo cual se representa mediante el registro especial D381 (ERROR_CODE, más detalles en la sección 8. “Instrucciones para el control del controlador del motor paso a paso”), el controlador pasa a la ejecución del subprograma con el puntero I2000. Cuando cambia el estado del controlador del motor paso a paso, registro D371 (MOTOR_STATUS, para obtener más detalles, consulte la sección 8. “Instrucciones para el control del controlador del motor paso a paso”), el controlador pasa a la ejecución del subprograma con el puntero I2001. El controlador regresa al programa principal después de que se ejecuta la instrucción IRET.

5. Códigos de error

Si el LED "ERR" está encendido después de cargar y ejecutar el programa del usuario, esto significa que el programa del usuario contiene un error: un error gramatical o un error de operando incorrecto. Cada error que ocurre en el controlador se registra en un registro especial (se registran el número de paso y el código de error). Esta información se puede leer utilizando una PC o PLC. La tabla a continuación contiene una lista de códigos de error y descripciones.

Dirección	Tipo	Tamaño	Descripción
E004	Registros de Entrada	16 bits	Código de error durante la ejecución de un programa de usuario.
E084	Registros de Entrada	16 bits	Línea del programa de usuario, donde se detectó el error.
E004	Bobinas		Marca de error durante la ejecución del programa de usuario.

Código de error	Descripción
2012h	Comando desconocido
1007h	Error interno, no se ha identificado el tipo de colisión de señal.
1005h	Error interno, no se identifica el tipo de señal en caso de colisión de nivel.
1006h	Error interno, el tipo de señal cuando la colisión de nivel invertido no se identifica.
2002h	Instrucción LD, desbordamiento de pila.
2001h	No se ha identificado el tipo de señal principal.
2000h	Procesamiento de comandos tipo LD, el código de instrucción ha cambiado.
1001h	Error interno, tipo desconocido de colisión única.
1000h	Error interno, colisión única en el valor actual. El tipo de señal del operando es desconocido.
200Bh	Procesamiento del comando de tipo AND al eliminar la señal de la pila de salida. Tipo de colisión desconocido.
1004h	Error interno de colisión de grupo. Tipo de colisión desconocido.

Código de error	Descripción
1002h	Error interno de colisión de grupo. El tipo de señal operando para la colisión de nivel no está definido.
1003h	Error interno de colisión de grupo. El tipo de señal de operando para la colisión de nivel inverso no está definido.
200Ch	Procesamiento del comando de tipo AND, el código de instrucción ha cambiado.
200Dh	Procesamiento del comando de tipo OR, el código de instrucción ha cambiado.
2010h	No hay entradas en la pila principal cuando se aplica la instrucción ANB.
200Fh	Error al aplicar la instrucción ANB. No hay entradas en la pila de salida y solo hay una entrada en la pila principal.
200Eh	Señal desconocida en la pila de salida con el comando ANB.
2011h	La ausencia de al menos dos elementos en la pila principal para aplicar la instrucción ORB.
2013h	Desbordamiento de pila por bifurcación, instrucción MPS.
2016h	La pila de bifurcación está vacía, instrucciones MRD, MPP.
2015h	Desbordamiento de pila principal, instrucciones MRD, MPP.
2014h	El tipo de señal no se reconoce al asignar el selector, las instrucciones MRD y MPP.
2017h	Desbordamiento de pila, instrucción NEXT.
3012h	Preescaneo – el índice P está fuera de rango.
3014h	Preexploración – el índice I está fuera de rango.
3013h	Preescaneo. No se pudo crear una nueva interrupción temporizada, se excedió el límite de cantidad.
201Dh	Tipo de operando incorrecto, instrucciones CJ/CJP.
201Ch	El operando está fuera de rango, instrucciones CJ/CJP.
2024h	Tipo de operando incorrecto, instrucciones CALL/CALLP.
2023h	El operando está fuera de rango, instrucciones CALL/CALLP.
2025h	No hay puntos de retorno en la pila, instrucción SRET.

Código de error	Descripción
2004h	Se recibió un comando END/FEND durante el procesamiento de la interrupción.
202Ah	Se recibió un comando IRET en el programa principal.
2056h	Instrucción END, la pila principal no está vacía.
2057h	Instrucción END, la pila de bifurcaciones no está vacía.
2058h	Instrucción END, la pila de ciclos no está vacía.
2059h	Instrucción END, la pila de subprogramas no está vacía.
2026h	Instrucción IRET, la pila principal no está vacía.
2027h	Instrucción IRET, la pila de bifurcación no está vacía.
2028h	Instrucción IRET, la pila de ciclos no está vacía.
2029h	Instrucción IRET, la pila de subprogramas no está vacía.
2020h	Instrucción CALL/CALLP, operando de índice desconocido.
201Eh	Instrucción CALL/CALLP, el operando índice A está fuera de rango.
201Fh	Instrucción CALL/CALLP, el operando índice B está fuera de rango.
201Ah	Instrucción CJ/CJP, operando de índice desconocido.
2018h	Instrucción CJ/CJP, el operando índice A está fuera de rango.
2019h	Instrucción CJ/CJP, el operando índice B está fuera de rango.
201Bh	Instrucción CJ/CJP, el puntero solicitado no existe.
2022h	Instrucción CALL/CALLP, la marca solicitada no existe.
2021h	Instrucción CALL/CALLP, desbordamiento de pila.
2003h	Operando incorrecto, instrucción OUT.
200Ah	Operando incorrecto, instrucción SET/RST.
2005h	El conjunto de instrucciones no se puede aplicar al operando C.
2006h	El conjunto de instrucciones no se puede aplicar al operando T.
2007h	El conjunto de instrucciones no se puede aplicar al operando D.

Código de error	Descripción
2008h	El conjunto de instrucciones no se puede aplicar al operando A.
2009h	El conjunto de instrucciones no se puede aplicar al operando B.
202Dh	Instrucción INV, tipo de señal desconocido.
202Bh	En la instrucción TMR, el primer argumento no es típico.
202Ch	La instrucción CNT, el primer argumento no es típico.
202Eh	Instrucción INC/DEC, operando incorrecto.
2037h	Instrucción ADD/SUB/MUL/DIV/WAND/WOR/WXOR, el tipo de operando 3d es incorrecto.
2038h	Instrucción NEG/ABS, tipo de operando incorrecto.
2030h	Instrucción CMP, el tipo de operando 3d es incorrecto.
2031h	Instrucción ZCP, el tipo de operando 3d es incorrecto.
202Fh	Instrucción MOV/BMOV/FMOV, tipo incorrecto de operando de destino.
2039h	Instrucción XCH, el tipo de dato del 1eroperando es incorrecto.
203Ah	Instrucción XCH, el tipo de dato del segundo operando es incorrecto. .
203Bh	Instrucción ROR/ROL, el tipo de datos del 1eroperando es incorrecto.
2033h	Instrucción ZRST, los operandos no son del mismo tipo.
2032h	Instrucción ZRST, el tipo de operando es incorrecto.
2036h	Instrucción DIV, división entre cero de un número entero.
2046h	Instrucción DECO, el tipo de operando 2d es incorrecto.
2047h	Instrucción ENCO, el tipo de operando 2d es incorrecto.
2048h	Instrucción SUM, el tipo de operando 2d es incorrecto.
2049h	Instrucción BON, el tipo de operando 2d es incorrecto.
204Bh	Instrucción SQR, el tipo de operando 2d es incorrecto.
204Ah	Instrucción SQR, valor negativo.
204Ch	Instrucción POW, el tipo de operando 3d es incorrecto.

Código de error	Descripción
203Ch	Instrucción FLT, el tipo de operando 2d es incorrecto.
203Dh	Instrucción INT, el tipo de operando 2d es incorrecto.
203Eh	En la instrucción PWM, el tercer operando no es aplicable para la salida de la señal PWM.
203Fh	Instrucción PWM, el tipo de operando 3d es incorrecto.
2041h	Instrucción DECMP, el tipo del primer operando es incorrecto.
2040h	Instrucción DECMP, el tipo de operando 2d es incorrecto.
2042h	Instrucción DECMP, el tipo de operando 3d es incorrecto.
2045h	Instrucción DEZCP, el tipo de operando 3d es incorrecto.
2044h	Instrucción DEZCP, el tipo del primer operando es incorrecto.
2043h	Instrucción DEZCP, el tipo de operando 2d es incorrecto.
2050h	Instrucción DEADD/DESUB/DEMUL/DEDIV/DEPOW, el tipo de operando 3d es incorrecto.
204Fh	Instrucción DEADD/DESUB/DEMUL/DEDIV/DEPOW, el tipo del primer operando es incorrecto.
204Eh	Instrucción DEADD/DESUB/DEMUL/DEDIV/DEPOW, el tipo de operando 2d es incorrecto.
204Dh	Instrucción DEDIV, dividir por cero.
3015h	Error de pre-escaneo, comando desconocido detectado.
2053h	Instrucción DESQR, el tipo del primer operando es incorrecto.
2052h	Instrucción DESQR, el tipo de operando 2d es incorrecto.
2051h	Valor negativo de la instrucción DESQR.
2035h	Instrucción LD#, desbordamiento de pila.
2034h	Instrucción LD#, tipo de señal principal no reconocido.
4000h	Desbordamiento de la cola de interrupciones del conmutador.
4001h	Desbordamiento de la cola de interrupciones temporizadas TIM0.
4002h	Desbordamiento de la cola de interrupciones temporizadas TIM1.

Código de error	Descripción
4003h	Desbordamiento de la cola de interrupciones temporizadas TIM2.
4004h	Desbordamiento de la cola de interrupciones temporizadas TIM3.
4005h	Desbordamiento de la cola de interrupciones externas IN0.
4006h	Desbordamiento de la cola de interrupciones externas IN1.
4007h	Desbordamiento de la cola de interrupciones externas IN2.
4008h	Desbordamiento de la cola de interrupciones externas IN3.
4009h	Desbordamiento de la cola de interrupciones externas IN4.
400Ah	Desbordamiento de la cola de interrupciones externas IN5.
400Bh	Desbordamiento de la cola de interrupciones externas IN6.
400Ch	Desbordamiento de la cola de interrupciones externas IN7.
400Dh	Desbordamiento de la cola de interrupciones del controlador.
400Eh	Desbordamiento de la cola de interrupciones por cambio de estado del motor.
2054h	Instrucción TRD, tipo de operando incorrecto.
2055h	Instrucción TWR, tipo de operando incorrecto.
3000h	Las pilas están vacías, sin valor de señal.
3001h	La pila principal está vacía, sin valor de señal.
3003h	El valor del registro de índice está fuera de rango.
3004h	Índice del operando X fuera de rango.
3005h	Índice del operando Y fuera de rango.
3006h	El índice del operando M está fuera de rango.
3007h	El índice del operando C está fuera de rango.
3008h	El índice del operando T está fuera de rango.
3009h	El índice del operando A/B está fuera de rango.
300Ah	Índice del operando D fuera de rango.

Código de error	Descripción
300Bh	Índice del operando P fuera de rango.
300Ch	Índice del operando I fuera de rango.
300Dh	Tipo de operando desconocido
300Fh	Imposible obtener el valor del operando.
300Eh	El número FLOAT se usa con una instrucción de 16 bits.
3010h	Obteniendo el valor del operando - tipo de operando incorrecto.
3011h	Obteniendo token de operando - tipo de operando incorrecto.
5000h	Alimentación + 5V - cortocircuito
205Ah	Instrucción TWR, formato de tiempo incorrecto.
205Bh	Instrucción MOD, división entre cero.
205Ch	Comando DWR, formato de fecha inválido.
205Dh	Desbordamiento de la pila de instrucciones de tipo contacto (LD#*)
205Eh	Desbordamiento de pila de instrucciones de tipo contacto (AND#*)
205Fh	Desbordamiento de la pila de instrucciones de tipo contacto (OR#*)

6. Instrucciones básicas

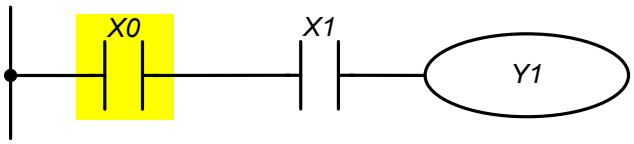
Instrucción	Función
LD	Contacto normalmente abierto

Operando	X	Y	M	T	C	A	B	D
	●	●	●	●	●			

Descripción:

La instrucción LD se utiliza como un contacto normalmente abierto para la programación del inicio de cadenas lógicas. Se ubica a la izquierda en el esquema de contactos y se conecta directamente a la línea del bus de alimentación.

Uso:



LD	**X0**
AND	X1
OUT	Y1

La instrucción LD X0 "contacto normalmente abierto X0" inicia la conexión lógica secuencial. Si en las entradas X0 y X1 hay simultáneamente una señal "1", entonces la salida Y1 se establecerá en el estado "1".

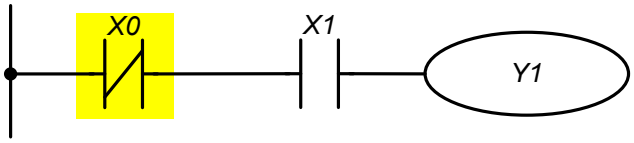
Instrucción	Función
LDI	Contacto normalmente cerrado

Operando	X	Y	M	T	C	A	B	D
	●	●	●	●	●			

Descripción:

La instrucción LDI se utiliza como un contacto normalmente cerrado para la programación del inicio de cadenas lógicas. Se ubica a la izquierda en el esquema de contactos y se conecta directamente a la línea del bus de alimentación.

Uso:



LDI	**X0**
AND	X1
OUT	Y1

La instrucción LDI X0 "contacto normalmente cerrado X0" inicia la conexión lógica secuencial. Si en las entradas X0 y X1 hay simultáneamente una señal "1", entonces la salida Y1 se establecerá en el estado "1".

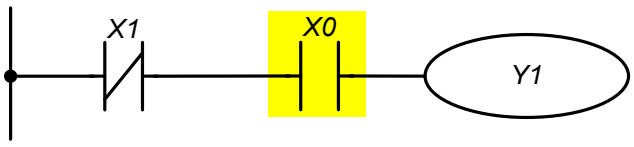
Instrucción	**Función**
AND	Conexión en serie - contacto normalmente abierto (AND lógico)

Operando	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
	•	•	•	•	•			

Descripción:

La instrucción AND se utiliza como un contacto normalmente abierto conectado en serie para la programación de la operación de multiplicación lógica (AND). La instrucción representa una operación lógica y, por lo tanto, no se puede programar al principio de la secuencia. Para las instrucciones de inicio de secuencia, se debe usar LD o LDI.

Uso:



LDI	X1
AND	**X0**
OU	Y1

La instrucción AND X0 "Conexión en serie - contacto normalmente abierto X0" crea una conexión lógica en serie con el contacto X1 y se utiliza para realizar la operación de multiplicación lógica. Si hay "0" en la entrada X1 y "1" en X0, entonces la salida Y1 cambia al estado "1".

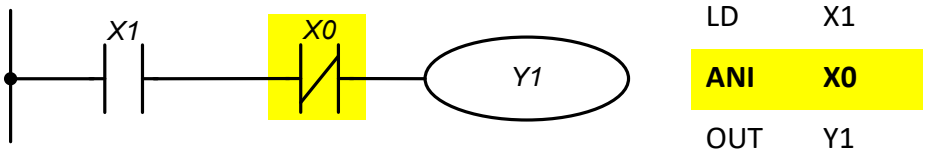
Instrucción	Función
ANI	Conexión en serie - contacto normalmente cerrado (NAND lógico)

Operando	X	Y	M	T	C	A	B	D
	●	●	●	●	●			

Descripción:

La instrucción ANI se utiliza como un contacto normalmente cerrado conectado en serie para la programación de la operación lógica NAND (AND NOT). La instrucción representa una operación lógica y, por lo tanto, no se puede programar al principio de la secuencia. Para las instrucciones de inicio de secuencia, se debe usar LD o LDI.

Uso:



La instrucción "Conexión en serie - contacto normalmente cerrado X0" crea una conexión lógica en serie con el contacto X1 y se utiliza para realizar la operación lógica NAND. Si hay "1" en la entrada X1 y "0" en X0, entonces la salida Y1 cambia al estado "1".

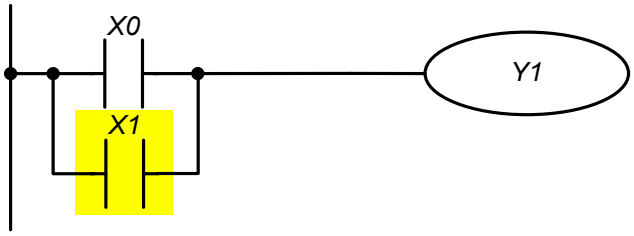
Instrucción	Función
OR	Conexión en paralelo – contacto normalmente abierto (OR lógico)

Operando	X	Y	M	T	C	A	B	D
	●	●	●	●	●			

Descripción:

La instrucción OR se utiliza como un contacto normalmente abierto conectado en paralelo para la programación de la suma lógica (OR). La instrucción representa una operación lógica y, por lo tanto, no se puede programar al principio de la secuencia. Para las instrucciones de inicio de secuencia, se debe usar LD o LDI.

Uso:



LD	X0
OR	**X1**
OUT	Y1

La instrucción "Conexión en paralelo - contacto normalmente abierto X1" crea una conexión lógica en paralelo con el contacto X0 y se utiliza para realizar la operación de suma lógica. Si al menos una de las entradas X0 o X1 es "1", entonces la salida Y1 cambia al estado "1".

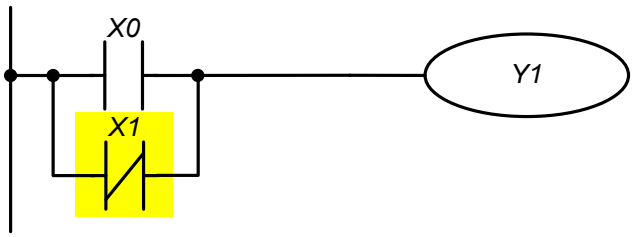
Instrucción	**Función**
ORI	Conexión en paralelo – contacto normalmente cerrado (NOR lógico)

Operando	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
	•	•	•	•	•			

Descripción:

La instrucción ORI se utiliza como un contacto normalmente cerrado conectado en paralelo para la programación de la operación lógica NOR (OR NOT). La instrucción representa una operación lógica y, por lo tanto, no se puede programar al principio de la secuencia. Para las instrucciones de inicio de secuencia, se debe usar LD o LDI.

Uso:



LD	X0
ORI	**X1**
OUT	Y1

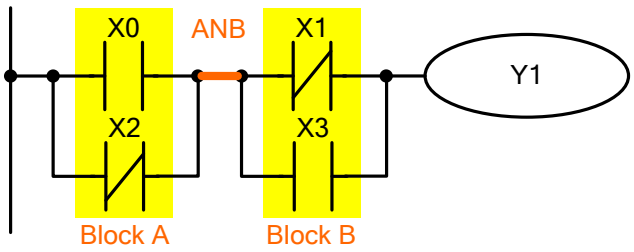
La instrucción "Conexión en paralelo - contacto normalmente cerrado X1" crea una conexión lógica en paralelo con el contacto X0 y se utiliza para realizar la operación de la instrucción lógica NOR (OR NOT). Si la entrada X0 es "1" o la entrada X1 es "0" (una o ambas condiciones al mismo tiempo), entonces la salida Y1 cambia al estado "1".

Instrucción	Función
ANB	Bloque «AND»: conexión en serie de bloques

Descripción:

- La instrucción ANB se utiliza para la conexión en serie de dos cadenas lógicas (bloques). Los bloques separados de elementos conectados en paralelo se ingresan en el programa por separado. Para conectar estos bloques en serie, se programa una instrucción ANB después de cada bloque.
- Inicio de ramificación programado utilizando las instrucciones LD o LDI.
- La instrucción ANB es independiente y no requiere ningún operando.
- La instrucción ANB dentro de todo el programa del usuario se puede utilizar un número ilimitado de veces.
- La instrucción ANB se muestra como una conexión en serie en un diagrama de contactos. La instrucción ANB en una lista de instrucciones del lenguaje IL se puede mostrar en un circuito de contactos como un puente.
- Si es necesario conectar algunos bloques separados uno tras otro, el número de instrucciones LD/LDI y también el número de instrucciones ANB debe limitarse a 8.

Uso:




```
LD      X0
ORI     X2
LDI     X1
OR      X3
ANB
OUT     Y1
```

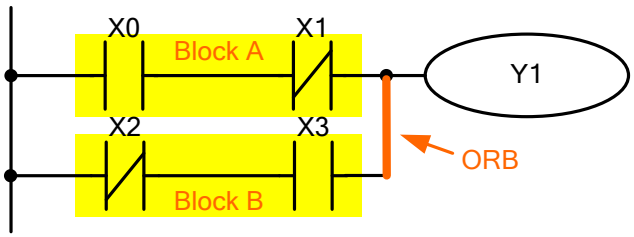
La instrucción ANB crea una serie de conexiones lógicas entre dos bloques lógicos (Bloque A y Bloque B).

Instrucción	Función
ORB	Bloque «OR»: conexión en paralelo de bloques

Descripción:

- La instrucción ORB se utiliza para la conexión en paralelo de dos o más contactos o bloques conectados en serie. Si varios bloques conectados en serie se conectan en paralelo, es necesario agregar una instrucción ORB después de cada bloque.
- El inicio de la ramificación se programa utilizando las instrucciones LD o LDI.
- La instrucción ORB es independiente y no requiere operandos.
- La instrucción ORB dentro del programa de usuario se puede utilizar un número ilimitado de veces.
- Si se programan varios bloques separados directamente uno tras otro, es necesario limitar el número de instrucciones LD y LDI y también el número de instrucciones ORB a 8.
- La instrucción ORB se muestra como una conexión en paralelo en un diagrama de contactos. La instrucción ORB en una lista de instrucciones de lenguaje IL se puede mostrar en un circuito de contactos como un puente.

Uso:




```
LD      X0
ANI     X1
LDI     X2
AND     X3
OR
OUT     Y1
```

La instrucción ORB crea una serie de conexiones lógicas entre dos bloques lógicos (Bloque A y Bloque B).

Instrucción	Función
MPS	Desplazamiento hacia abajo en la pila

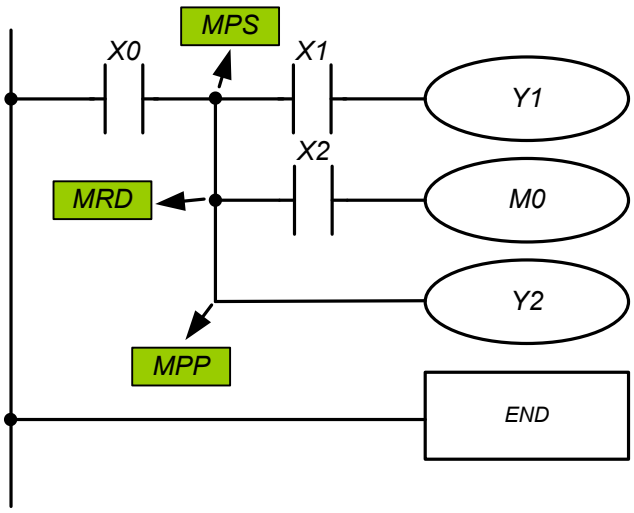
Instrucción	Función
MRD	Leer el valor de la pila

Instrucción	Función
MPP	Salir de la pila

Descripción:

- Las instrucciones MPS, MRD y MPP se utilizan para crear niveles de conexiones lógicas (por ejemplo, después de una expresión lógica inicial, crear varias expresiones lógicas en la salida, es decir, activar varias bobinas de salida).
- Utilizando la instrucción MPS, se almacena el resultado anterior de las conexiones lógicas (procesamiento de una expresión lógica).
- Utilizando la instrucción MRD, es posible crear varias ramas independientes entre el principio (MPS) y el final (MPP) de la rama. El resultado del procesamiento de una expresión lógica en el punto MPS se tiene en cuenta en cada rama.
- La última rama se crea mediante la instrucción MPP.
- La ramificación abierta con la instrucción MPS siempre debe cerrarse con la instrucción MPP.
- Las instrucciones MPS, MRD y MPP no necesitan operandos.
- Estas instrucciones no se muestran en el diagrama de contactos. Si la programación se realiza en un circuito de contactos, las ramas se utilizan como de costumbre. Al convertir un programa de usuario de un diagrama de escalera (LD) a una lista de instrucciones (IL), se deben agregar las instrucciones MPS, MRD y MPP a IL.

Uso:




```
LD      X0
MPS
AND     X1
OUT     Y1
MPD
AND     X2
OUT     M0
MPP
OUT     Y2
END
```

MPS

Un resultado intermedio (valor X0) en el primer nivel de conexiones lógicas se muestra en el primer lugar en la memoria de pila de conexiones intermedias. Se realiza la multiplicación lógica de X1 con X0 y se establece la salida Y1.

MRD

Antes de ejecutar la siguiente instrucción, se lee un resultado intermedio en la primera posición de la memoria de conexiones lógicas. Se realiza la multiplicación lógica de X2 con X0, y se establece la salida de M0.

MPP

Antes de ejecutar la siguiente instrucción, se lee un resultado intermedio en la primera posición de la memoria de conexiones lógicas. La salida Y2 está activada. La operación en el 1er nivel de resultados intermedios se completa y la memoria de conexiones lógicas se borra.

Instrucción	**Función**
OUT	Bobina de salida

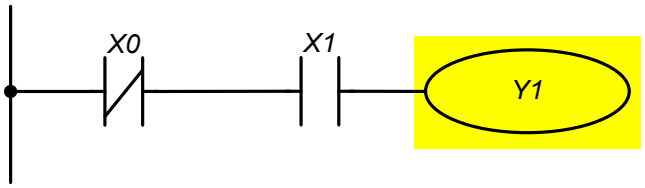
Operando	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
		●	●					

Descripción:

- La instrucción OUT se utiliza para establecer una bobina de salida en función del resultado de las conexiones lógicas (el resultado del procesamiento de la expresión lógica por parte del controlador).
- Utilizando la instrucción OUT, es posible finalizar la programación de una línea (expresión lógica).
- También es posible programar varias instrucciones OUT como resultado del procesamiento de una expresión lógica.
- El resultado de las conexiones lógicas representado por la instrucción OUT se puede aplicar en los siguientes pasos del programa como el estado de la señal de entrada, es decir, se puede leer muchas veces en muchas expresiones lógicas.

- El resultado de las conexiones lógicas representado por la instrucción OUT está activo (encendido) mientras las condiciones para su activación sean válidas.
- Al programar la doble grabación de las mismas salidas (sus direcciones), pueden surgir problemas durante la ejecución del programa. Evite la doble grabación de las salidas, ya que esto puede provocar interferencias al ejecutar el programa.

Uso:




```
LDI    X0
AND    X1
OUT    Y1
```

Si X0 = 0 y X1 = 1, la instrucción OUT Y1 establece el estado de la salida Y1 = "1".

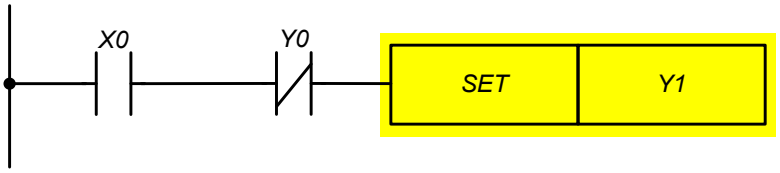
Instrucción	Función
SET	Activación de la salida retentiva

Operando	X	Y	M	T	C	A	B	D
		•	•					

Descripción:

- El estado del operando se puede establecer directamente mediante la instrucción SET.
- Los operandos Y y M se pueden activar mediante la instrucción SET.
- Tan pronto como se establece la condición de entrada para la instrucción SET (señal "1"), el operando correspondiente se activa.
- Si las condiciones de entrada para la instrucción SET ya no se cumplen, el operando correspondiente permanece activado.

Uso:




```
LD     X0
ANI    Y0
SET    Y1
```

La salida Y1 se activa cuando se cumplen las condiciones de entrada (X0, Y0). Después de eso, la salida Y1 no depende de las condiciones de entrada. La única forma de desactivar la salida Y1 es utilizar la instrucción RST o apagar la fuente de alimentación del controlador.

Instrucción	Función
RST	Reinicio del estado del operando

Operando	X	Y	M	T	C	A	B	D
		●	●	●	●	●	●	●

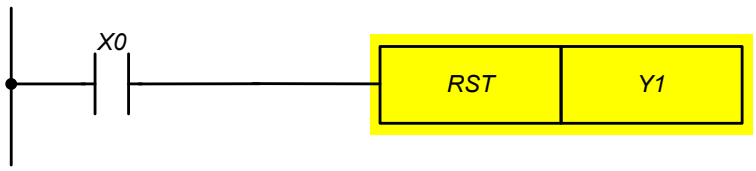
Descripción:

El estado de un operando se puede restablecer directamente.

La instrucción RST desactiva los operandos correspondientes. Significa:

- Las salidas Y, contactos M se apagan (estado de la señal "0").
- Los valores actuales de los temporizadores y contadores, los valores de los registros D, A y B se restablecen a "0".
- Tan pronto como se establece la condición de entrada para la instrucción RST (señal "1"), el operando correspondiente se apaga.
- Si las condiciones de entrada para la instrucción RST ya no se cumplen, el operando correspondiente permanece apagado.

Uso:




```
LD    X0
RST   Y1
```

La salida Y1 se apaga cuando se cumple la condición X1 y permanece apagada incluso cuando no se cumple la condición X0.

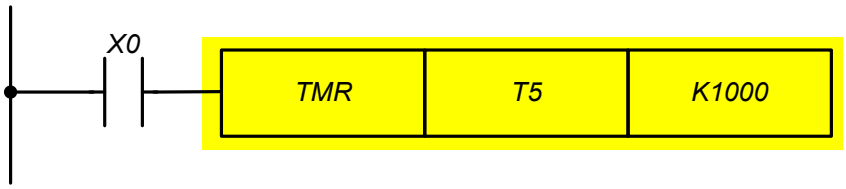
Instrucción	Función
TMR	Temporizador (16-bit)

Operandos	K	H	F	X	Y	M	T	C	A	B	D
S1							●				
S2	●	●					●	●	●	●	●

Descripción:

- La instrucción TMR se utiliza para establecer un estado de señal (encender/apagar) dependiendo del resultado de las conexiones lógicas después de un período de tiempo especificado en la instrucción.
- Utilizando la instrucción TMR, es posible finalizar la programación de una línea (expresión lógica).
- El resultado de las conexiones lógicas representadas por la instrucción TMR se puede utilizar en los siguientes pasos del programa como el estado de la señal de entrada, es decir, se puede leer muchas veces en muchas expresiones lógicas.
- El resultado de las conexiones lógicas representadas por la instrucción TMR está activo (encendido) mientras las condiciones de entrada sean válidas.

Uso:




```
LD    X0
TMR   T5   K1000
```

Cuando la condición X0 = 1, la instrucción TMR T5 cuenta hasta que el valor en el registro T5 alcanza el valor de K1000 (100 segundos). Si X0 = 0, la ejecución de la instrucción TMR se detendrá y T5 se restablecerá a "0".

Instrucción		Función
CNT	S1 S2	Contador (16-bit)
DCNT		Contador (32-bit)

Operandos	K	H	F	X	Y	M	T	C	A	B	D
S1								•			
S2	•	•					•	•	•	•	•

Información

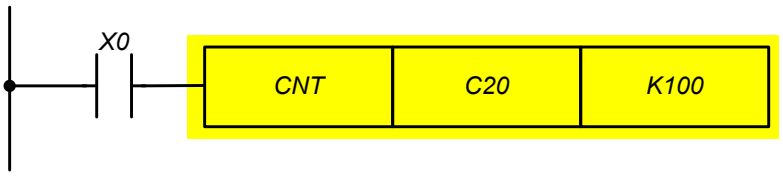
(D) Usualmente, para usar instrucciones de 32-bits, se añade el prefijo "D" al nombre de la instrucción. (D) – solo existe una versión de 32-bits de la instrucción.

P Para las instrucciones de impulso con una "vida útil" de un escaneo, se añade el sufijo "**P**". El concepto de *un solo escaneo* debe atribuirse al operando utilizado. Por ejemplo, el operando **M0** en la línea 7 del programa principal se estableció de "0" a "1". Ahora, para todas las instrucciones siguientes que están antes de FEND o END, el operando **M0** tiene un componente de pulso en el flanco ascendente (el resultado para LDP M0 será "1"), así como para las instrucciones que comienzan desde la línea 0ª hasta la 6ª línea durante el siguiente escaneo, el operando **M0** tendrá un componente de pulso. Al ir a la línea 7 (o inferior si se utiliza un comando CJ), **M0** tendrá un nivel de señal alto sin componentes de pulso. Por lo tanto, se realizó un círculo a lo largo del cuerpo del programa: un escaneo, desplazado a la línea de cambio de operando. En caso de que surjan interrupciones antes de llegar a la línea 7, el operando **M0** mantiene el componente de pulso hasta que regresa al programa principal. Si el operando **M0** se ha modificado en una interrupción o subprograma, entonces el lugar donde se cambia el operando se considera la línea desde la cual se realizó la transición al subprograma o la línea del programa principal, antes del procesamiento del cual se llamó al controlador de interrupciones.

Descripción:

- La instrucción CNT se utiliza para sumar el número de cierres del contacto de entrada y asignar el estado de la señal (activar la salida) cuando el valor actual del contador alcanza el valor establecido.
- Utilizando la instrucción CNT, es posible finalizar la programación de una línea (expresión lógica).
- - El resultado de las conexiones lógicas representadas por la instrucción CNT se puede aplicar en los siguientes pasos del programa como el estado de la señal de entrada, es decir, se puede leer muchas veces en muchas expresiones lógicas.
- Para restablecer el valor actual de un contador, utilice la instrucción RST.
- **Atención:** los contadores de hardware cuentan por encima del umbral y funcionan independientemente de la presencia de una señal de entrada.

Uso:




```
LD    X0
CNT   C20   K100
```

Cuando X0 cambia de "0" a "1", el valor del registro C20 aumenta en 1. Se repite hasta que el valor del registro C20 alcanza K100 (100 pulsos). Después de eso, el conteo se detiene y el contacto C20 se activa. Para restablecer el valor del registro C20, utilice la instrucción RST C20.

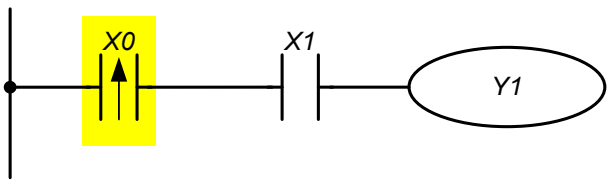
Instrucción	Función
LDP	Inicio de la expresión lógica con un sondeo de flanco ascendente (impulso)

Operando	X	Y	M	T	C	A	B	D
	●	●	●	●	●			

Descripción:

- La instrucción LDP se utiliza para programar el inicio de pulso de una conexión lógica.
- La instrucción LDP debe programarse al principio del circuito.
- La instrucción LDP también se utiliza junto con las instrucciones ANB y ORB para iniciar la ramificación.
- La instrucción LDP después de un flanco positivo se almacena durante la duración del ciclo del programa (escaneo).

Uso:



LDP	**X0**
AND	X1
OUT	Y1

La instrucción "LDP X0" inicia la conexión lógica en serie. Si la entrada X0 cambia de "0" a "1" (y X1 = 1), entonces la salida Y1 mantiene el estado "1" durante un escaneo.

Instrucción	Función
LDF	Inicio de una expresión lógica con un flanco de bajada (impulso)

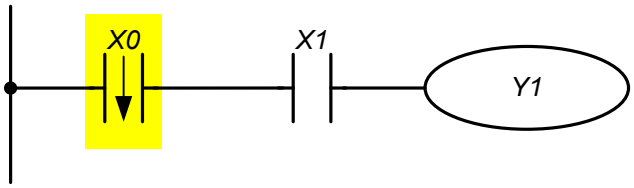
Operando	X	Y	M	T	C	A	B	D
	●	●	●	●	●			

Descripción:

- La instrucción LDF se utiliza para programar el inicio de pulso de una conexión lógica.

- La instrucción LDF debe programarse al principio del circuito.
- La instrucción LDF también se utiliza junto con las instrucciones ANB y ORB para iniciar la ramificación.
- La instrucción LDF después de un flanco negativo se almacena durante la duración del ciclo del programa (escaneo).

Uso:



LDF	X0	3
AND	X1	
OUT	Y1	

La instrucción "LDF X0" inicia la conexión lógica en serie. Si la entrada X0 cambia de "1" a "0" (y X1 = 1), entonces la salida Y1 mantiene el estado "1" durante un escaneo.

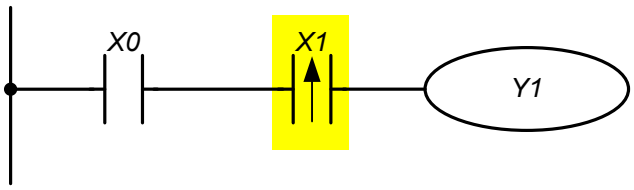
Instrucción	Función
ANDP	«AND» con sondeo de flanco ascendente (impulso)

Operando	X	Y	M	T	C	A	B	D
	●	●	●	●	●			

Descripción:

- La instrucción ANDP se utiliza para programar un contacto de pulso conectado en serie con sondeo de flanco ascendente (impulso).

Uso:



LD	X0
ANDP	**X1**
OUT	Y1

La instrucción "ANDP X1" crea una conexión lógica en serie. Si la entrada X1 cambia de "0" a "1" (y X0 = 1), entonces la salida Y1 mantiene el estado "1" durante un escaneo.

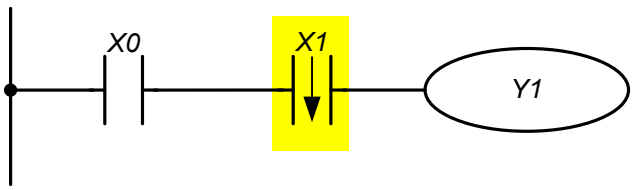
Instrucción	Función
ANDF	«AND» con sondeo en flanco de bajada (impulso)

Operando	X	Y	M	T	C	A	B	D
	●	●	●	●	●			

Descripción:

— La instrucción ANDF se utiliza para programar un contacto de pulso conectado en serie con sondeo de flanco descendente (impulso).

Uso:




```
LD      X0
ANDF    X1
OUT     Y1
```

La instrucción "ANDF X1" crea una conexión lógica en serie. Si la entrada X1 cambia de "1" a "0" (y X0 = 1), entonces la salida Y1 mantiene el estado "1" durante un escaneo.

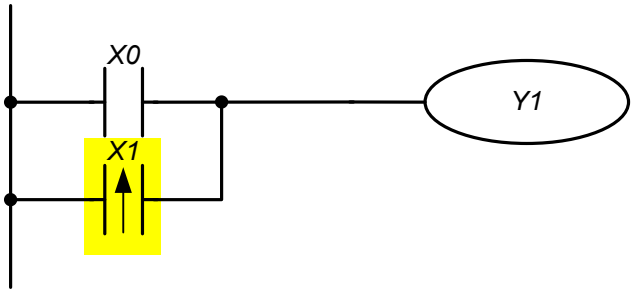
Instrucción	Función
ORP	«OR» con sondeo de flanco ascendente (impulso)

Operando	X	Y	M	T	C	A	B	D
	●	●	●	●	●			

Descripción:

— La instrucción ORP se utiliza para la programación de un contacto de pulso conectado en paralelo con sondeo de flanco ascendente (impulso).

Uso:




```
LD    X0
ORP   X1
OUT   Y1
```

La instrucción "ORP X1" crea una conexión lógica paralela. La salida Y1 mantendrá el estado "1" durante un escaneo si la entrada X1 cambia de "0" a "1" o X0 = 1.

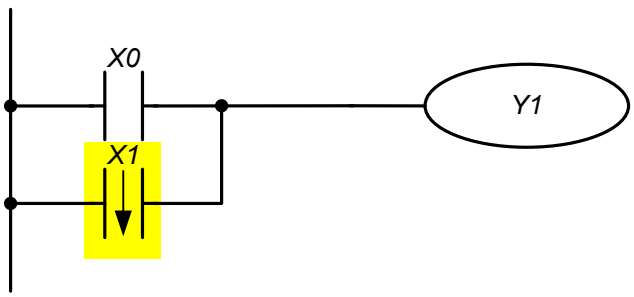
Instrucción	Función
ORF	«OR» con flanco de bajada (impulso)

Operando	X	Y	M	T	C	A	B	D
	●	●	●	●	●			

Descripción:

— La instrucción ORF se utiliza para programar un contacto de pulso conectado en paralelo con sondeo de flanco descendente (impulso).

Uso:




```
LD    X0
ORF   X1
OUT   Y1
```

La instrucción "ORF X1" crea una conexión lógica paralela. La salida Y1 mantendrá el estado "1" durante un escaneo si la entrada X1 cambia de "1" a "0" o X0 = 1.

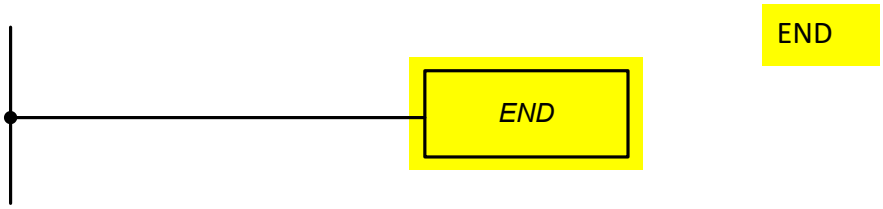
Instrucción	Función
END	**Fin del programa**

Descripción:

El final de un programa de usuario y la transición al principio del programa (paso 0).

- Cada programa del controlador debe terminar con una instrucción END.
- Si se está programando una instrucción END, en este punto, el procesamiento del programa finaliza. Las áreas posteriores del programa ya no se tienen en cuenta. Después del procesamiento de la instrucción END, se configuran las salidas y el programa se inicia (paso 0).

Uso:



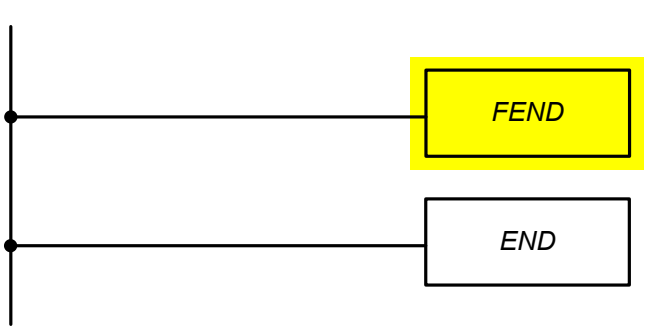
Instrucción	Función
FEND	**Fin del programa principal**

Descripción:

El final del programa de usuario principal y la transición al principio del programa (paso 0). Las principales diferencias con la instrucción END son las siguientes:

- El procesamiento no termina con el comando FEND. La instrucción FEND separa el programa principal de los subprogramas y los manejadores de interrupciones, que se encuentran en el área entre las instrucciones FEND y END y están enmarcados por P y SRET, I e IRET.
- Si no se utilizan subprogramas e interrupciones en un programa de usuario, la instrucción FEND no es necesaria.
- La instrucción FEND solo se puede utilizar una vez.

Uso:



FEND

END

Instrucción	Función
NOP	**Línea vacía en el programa**

Descripción:

Una línea vacía sin funciones lógicas se puede utilizar posteriormente para cualquier instrucción, por ejemplo, durante el ensamblaje de un programa o para la depuración.

- Después de ensamblar correctamente un programa, las instrucciones NOP deben eliminarse; de lo contrario, extienden innecesariamente el tiempo del ciclo del programa.
- El número de instrucciones NOP en un programa no está limitado.

Uso:

```
LD      X0
NOP
OUT     Y0
```

Las instrucciones NOP no se muestran en los diagramas de contactos.

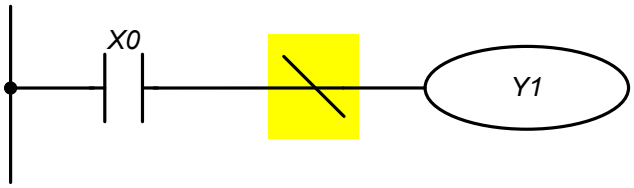
Instrucción	Función
INV	**Inversión: reemplazar el resultado de las conexiones lógicas con el opuesto**

Descripción:

- La instrucción INV invierte el estado de la señal de resultado de las instrucciones colocadas antes.
- El resultado de las conexiones lógicas “1” antes de la instrucción INV se convierte en “0” después de ella.

- El resultado de las conexiones lógicas “0” antes de la instrucción INV se convierte en “1” después de ella.
- La instrucción INV se puede aplicar como una instrucción AND o ANI.
- La instrucción INV se puede utilizar para invertir la señal de resultado de un circuito complejo.
- La instrucción INV se puede utilizar para invertir el resultado de la señal de las instrucciones de pulso LDP, LDF, ANP, etc.

Uso:




```
LD      X0
INV
OUT     Y1
```

Si la entrada X0 = 0, la salida Y1 = 1. Si la entrada X0 = 1, la salida Y1 = 0.

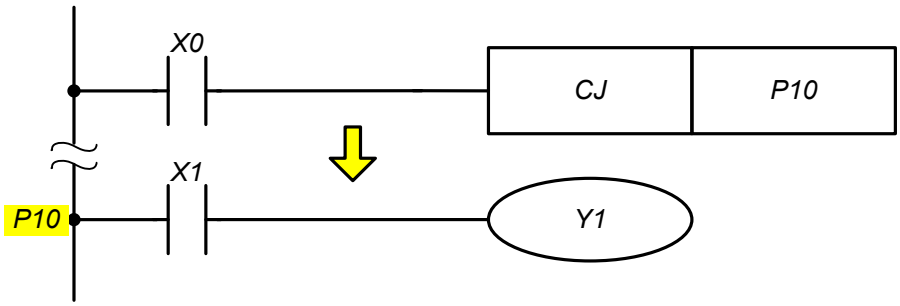
Instrucción	Función
P	Direccionamiento de un punto de salto en un programa o subprograma

Operando	0…31

Descripción:

- La instrucción P se utiliza para indicar un punto de transición para las instrucciones CJ, CALL.
- El número de punto en el programa no debe repetirse.

Uso:




```
LD      X0
CJ      P10
.
.
P       10
LD      X1
OUT     Y1
```

El punto P10 indica la dirección de transición para ejecutar la instrucción CJ P10.

Instrucción	Función
I	Abordando un punto de interrupción

Operando	0…100, 1000…1007, 2000, 2001

Descripción:

La instrucción I se utiliza para indicar el punto de transición al manejador de interrupciones. Globalmente, las interrupciones se habilitan mediante la instrucción EN y se deshabilitan mediante la instrucción DS.

En total, el controlador puede tener 15 interrupciones.

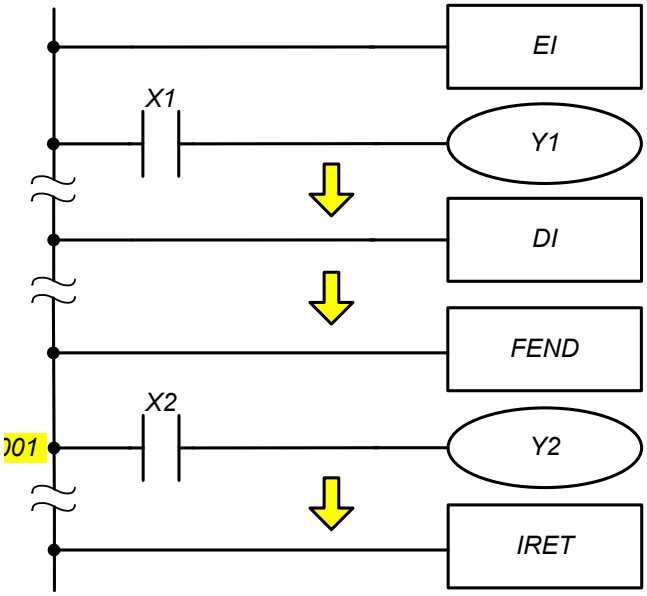
Una interrupción que se produce cuando se recibe una trama Modbus (difusión o dirigida al controlador) a través de RS-485 se marca como I0.

Es posible organizar hasta 4 interrupciones temporizadas en el controlador In, donde n es el período de llamada del manejador de interrupciones de 10 ms y puede tener un valor de 1 a 100. Por lo tanto, $n = \frac{T}{10} ms$, donde T es el período deseado de llamada del manejador (medido en ms).

8 interrupciones externas **I1000**…**I1007** corresponden a las entradas discretas 0..7. La interrupción surge cuando se cambia el nivel de la señal de entrada.

2 interrupciones del controlador: **I2000**– surge en caso de error e **I2001**– surge cuando cambia el estado del motor (consulte la sección 8. “Instrucciones para el control del controlador del motor paso a paso” para obtener más detalles).

Uso:



EI		Habilitación de interrupciones
LD	X1	Contacto NO X1
OUT	Y1	Salida Y1
…		
DI		Desactivación de interrupciones
…		
FEND		Fin del programa principal
I	**1001**	Punto de entrada para el controlador de interrupciones.
LD	X2	Contacto NO X2
OUT	Y2	Salida Y2
…		
IRET		Fin del manejador de interrupciones

7. Instrucciones de aplicación

Instrucción	Operandos	Variantes asociadas	Función
CJ	S	P	Salto condicional

S	Se utilizan punteros P como operandos. Los operandos pueden ser indexados (A, B)

Descripción:

Utilizando la instrucción CJ, se puede omitir una parte del programa. Al aplicar esta instrucción, se puede reducir el tiempo de ejecución del programa. Por ejemplo, omitir una sección del programa asignada para inicializar los periféricos del controlador, activar interrupciones, etc. (consulte la sección 4.8para obtener más detalles).

CALL	S	P	Llamando subprograma

S	Se utilizan punteros P como operandos. Los operandos pueden ser indexados (A, B)

Descripción:

La instrucción CALL se utiliza para llamar a un subprograma.

- Un subprograma se marca con puntos P y se puede llamar mediante una instrucción CALL.
- La instrucción SRET debe colocarse al final del subprograma.
- Un subprograma debe colocarse después de la instrucción FEND y antes de la instrucción END.
- Cuando se ejecuta la instrucción CALL, el controlador va al punto marcado. Después de ejecutar la instrucción SRET, el controlador regresa al programa principal a la instrucción que sigue a CALL.
- Los puntos se pueden usar con un número ilimitado de instrucciones CALL.
- Los subprogramas se pueden llamar desde otros subprogramas. Máx. 8 niveles de anidamiento posibles.

SRET			Fin del subprograma

Descripción:

La instrucción SRET define el final de un subprograma (consulte la sección 4.8para obtener más detalles).

- Cada subprograma debe terminar con la instrucción SRET.
- El programa regresa a la instrucción que sigue a la instrucción CALL después de procesar el SRET.
- La instrucción SRET solo se puede usar junto con la instrucción CALL.

Nota:esta instrucción no requiere una condición de entrada (no se necesitan contactos).

IRET			Fin del manejador de interrupciones

Descripción:

La instrucción IRET define el final del procesamiento de interrupciones (consulte la sección 4.8 para obtener más detalles).

Nota: esta instrucción no requiere una condición de entrada (no se necesitan contactos).

EI			Habilitación de interrupciones globales

Descripción:

La instrucción EI habilita el procesamiento de interrupciones (consulte la sección 4.8 para obtener más detalles).

Nota: esta instrucción no requiere una condición de entrada (no se necesitan contactos).

DI			Desactivación de interrupciones globales

Descripción:

La instrucción DI deshabilita el procesamiento de interrupciones (consulte la sección 4.8 para obtener más detalles).

Nota: esta instrucción no requiere una condición de entrada (no se necesitan contactos).

Llamada de un subprograma de manejo de interrupciones

- Al procesar una interrupción, se realiza una transición del programa principal al controlador de interrupciones.
- Después de la interrupción, se completa el procesamiento y el controlador regresa al programa principal.
- El inicio del subprograma de interrupción se determina configurando la marca (punto de interrupción).
- El final del subprograma de interrupción está determinado por la instrucción IRET.
- El subprograma de interrupción debe programarse al final del programa de usuario después de la instrucción FEND y antes de la instrucción END.

Nota: Si no se programa ninguna de las dos instrucciones EI o DI, el modo de interrupción no se activa, es decir, no se procesará ninguna de las señales de interrupción.

Ejecución de un subprograma de interrupción

Varios subprogramas de interrupción que van uno tras otro se procesan en secuencia de su llamada.

Si se llaman varios subprogramas de interrupción al mismo tiempo, el programa de interrupción con la dirección de punto más baja se procesa primero.

FOR	S		Inicio de un bucle FOR-NEXT

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•

Nota:esta instrucción no requiere una condición de entrada (no se necesitan contactos).

NEXT			Fin de un bucle FOR-NEXT

Nota:esta instrucción no requiere una condición de entrada (no se necesitan contactos).

Ciclos

Las instrucciones FOR/NEXT se utilizan para programar repeticiones cíclicas de partes del programa (bucle del programa).

Descripción:

- La parte del programa entre las instrucciones FOR y NEXT se repite "n" veces. Después de completar la instrucción FOR, el programa continúa con el paso del programa después de la instrucción NEXT.
- El valor "n" puede estar en un rango de +1 a +32 767.Si se establece algún valor del rango de 0 a -32 768, el bucle FOR-NEXT se ejecuta solo una vez.
- Son posibles hasta 8 niveles de anidamiento de bucles FOR-NEXT.
- Las instrucciones FOR y NEXT solo se pueden usar en pares. Cada instrucción FOR debe coincidir con la instrucción NEXT.

Fuente de errores

Aparecen errores en el programa en los siguientes casos:

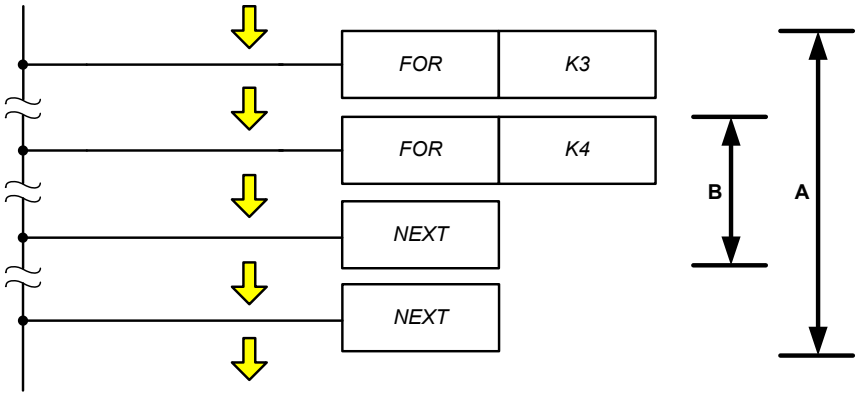
- La siguiente instrucción va antes de la instrucción FOR.
- El número de instrucciones NEXT difiere del número de instrucciones FOR.
- Una gran cantidad de repeticiones de "n" puede aumentar significativamente el tiempo de ejecución de un programa.

Un ejemplo de uso de instrucciones FOR/NEXT:

El siguiente ejemplo muestra dos bucles FOR-NEXT, uno dentro del otro.

La parte del programa A se ejecuta 3 veces (K3 significa número decimal 3).

La parte del programa B se ejecuta 4 veces dentro de cada repetición de la parte A (K4 significa número decimal 4).



CMP	S1 S2 D	D P	Comparación de datos numéricos

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D					•	•					

Nota: el operando D toma 3 direcciones.

Descripción:

Comparación de dos valores de datos numéricos (mayor, menor, igual).

- Se comparan los datos en ambas fuentes (S1) y (S2).
- El resultado de la comparación (mayor, menor, igual) se muestra (indica) activando el relé M o la salida Y. El resultado de la comparación determina cuál de los contactos en el operando de destino (D) está activo:

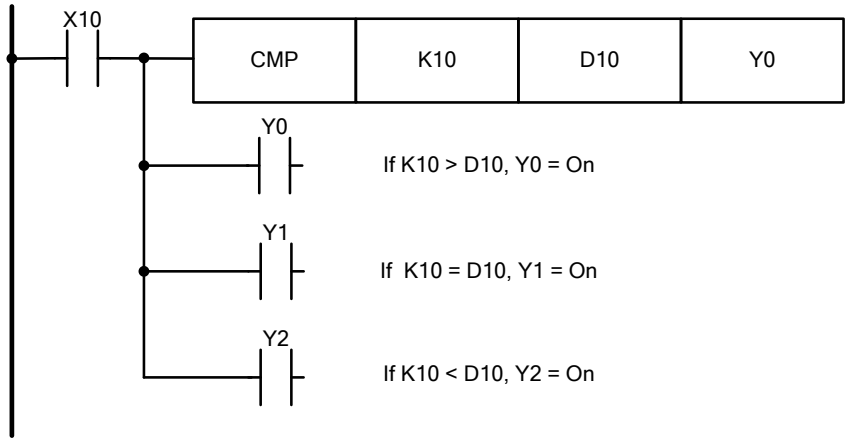
(S1) > (S2) → (D)

(S1) = (S2) → (D+1)

(S1) < (S2) → (D+2)

- Los datos en S1 y S2 se procesan como datos enteros con signo.

Ejemplo:



Y0: se activa si K10 > registro de datos D10, Y1 e Y2 se desactivan.

Y1: se activa si K10 = registro de datos D10, Y0 e Y2 se desactivan.

Y2: se activa si K10 < registro de datos D10, Y0 e Y1 se desactivan.

Y0, Y1, Y2 no se modifican si la condición de entrada X10=0.

Para reiniciar los resultados de la comparación, utilice las instrucciones RST, ZRST.

ZCP	S1 S2 S D	D P	Comparación de zonas de datos numéricos

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
S	•	•					•	•	•	•	•
D					•	•					

Nota:

- El operando D ocupa 3 direcciones.
- El operando S1 debe ser menor que S2.

Descripción:

- Comparación de valores de datos numéricos con áreas de datos numéricos (mayor, menor, igual)
- Los datos en la fuente (S) se comparan con los datos en ambas fuentes (S1) y (S2)
- El resultado de la comparación (mayor, menor, igual) se muestra (indica) activando el relé M o la salida Y. El resultado de la comparación determina cuál de los contactos en el operando de destino (D) está activo.

(S) < (S1) → (D)

(S1) ≤ (S) ≤ (S2) → (D + 1)

(S) > (S2) → (D + 2)

- Si el valor en (S1) es mayor que el valor en (S2), todos los contactos en el operando (D) se reinician.

  Para reiniciar los resultados de la comparación, utilice las instrucciones RST, ZRST.

MOV	S D	D P	Transferencia de datos

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•	•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•

Descripción:

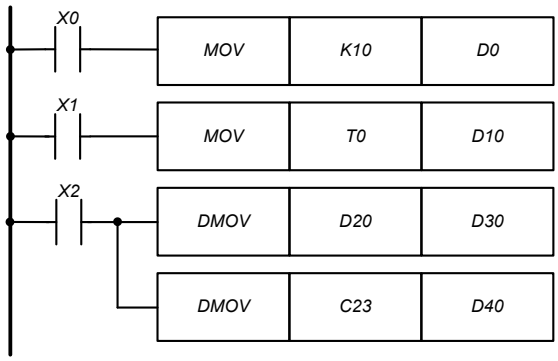
- La instrucción MOV se utiliza para transferir datos desde una fuente de datos (S) a un destino (D). El valor de la fuente (S) no cambia.
- Los datos en la fuente de datos (S) se leen como valores binarios al ejecutar la instrucción MOV.
- Los operandos de bit ocupan la cantidad de direcciones correspondientes al tipo de instrucción: 16 o 32 direcciones. En este caso, es posible combinar los tipos de operandos para la fuente y el destino. Por ejemplo, como resultado de ejecutar el comando MOV D3 M0, los relés M0 ...M15 muestran el valor del registro D3 en forma binaria.

Ejemplo:

Si la condición de entrada X0 está activada, el valor del registro D0 = 10. Si X0 está desactivada, el valor del registro D0 no se modifica.

Si la condición de entrada X1 está activada, el valor actual del temporizador T0 se transfiere al registro de datos D10. Si X1 está desactivada, el valor del registro D10 no se modifica.

Si la condición de entrada X2 está activada, el valor de los registros D20 y D21 se transfiere a los registros de datos D30 y D31; el valor actual del contador C23 se transfiere a los registros de datos D40 y D41.



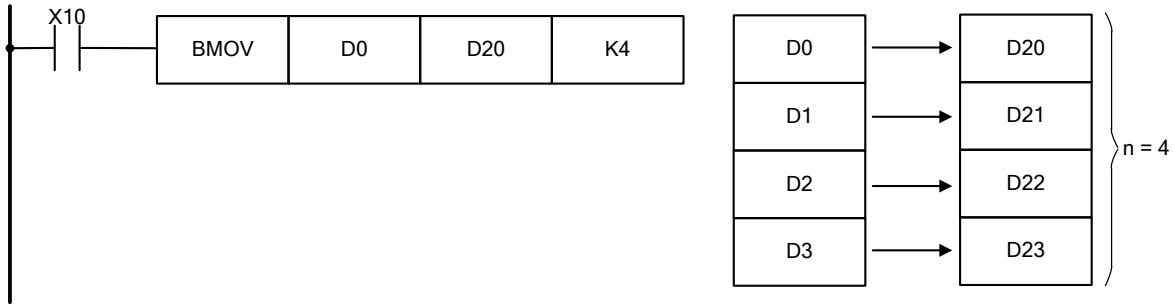
BMOV	S D n	D P	Transferencia de datos en bloque

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●	●	●	●	●	●	●	●	●	●
D					●	●	●	●	●	●	●
n	●	●					●	●	●	●	●

Descripción:

Copiar el paquete de datos. El desplazamiento durante la operación se realiza tanto para el operando de origen (S) como para el operando de destino (D) a (n) elementos de bloque, según la instrucción (16 bits o 32 bits).

Ejemplo:



Si X10 está activado, los valores de los registros D0 – D3 se transfieren a los registros D20 – D23.

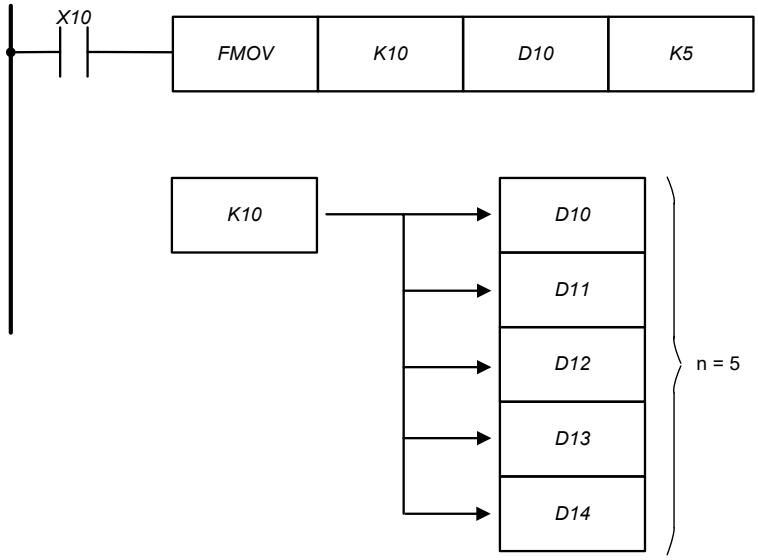
FMOV	S D n	D P	Transferencia de datos a múltiples direcciones

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●	●	●	●	●	●	●	●	●	●
D					●	●	●	●	●	●	●
n	●	●					●	●	●	●	●

Descripción:

El valor del operando de origen (S) se copia a (n) operandos de destino (D) del mismo tipo.

Ejemplo:



La instrucción FMOV copia el valor "10" a los registros de datos D10...D14.

XCH	D1 D2	D P	Intercambio de datos

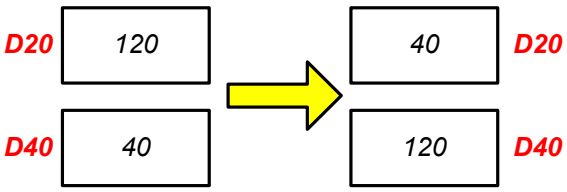
	K	H	F	X	Y	M	T	C	A	B	D
D1							●	●	●	●	●
D2							●	●	●	●	●

Descripción:

Los valores de los operandos (D1) y (D2) se intercambian.

Ejemplo:

Si X0 = 1, se realiza el intercambio de datos:



ADD	S1 S2 D	D P	Suma de datos numéricos

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●					●	●	●	●	●
S2	●	●					●	●	●	●	●
D							●	●	●	●	●

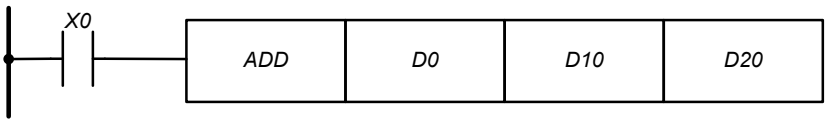
Descripción:

- Los datos binarios en los operandos de origen (S1) y (S2) se suman. El resultado de la suma se almacena en el operando de destino (D). La operación se realiza en tipos de datos enteros con signo.

  (S1) + (S2) = (D)

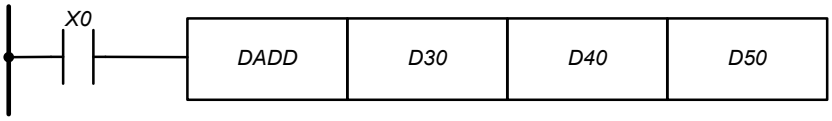
- El bit más significativo contiene el signo del resultado: 0 – signo de un número positivo, 1 – signo de un número negativo.
- Al ejecutar una instrucción de 32 bits, los 16 bits inferiores deben indicarse en el operando. El siguiente registro de datos contiene los 16 bits superiores.

Ejemplos:



(D0) + (D10) = (D20)

Si X0 está activado, los valores de los registros de datos D0 y D10 se suman, el resultado se guarda en el registro de datos D20.



(D31, D30) + (D41, D40) = (D51, D50)

Si X0 está activado, el resultado de sumar los valores de los registros (D31, D30) y (D41, D40) se guarda en los registros de datos (D51, D50).

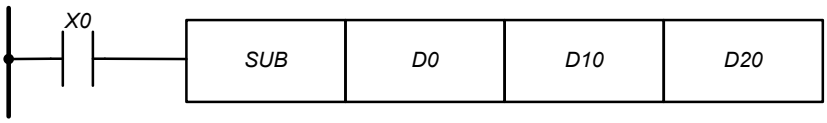
SUB	S1 S2 D	D P	Resta de datos numéricos

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Descripción:

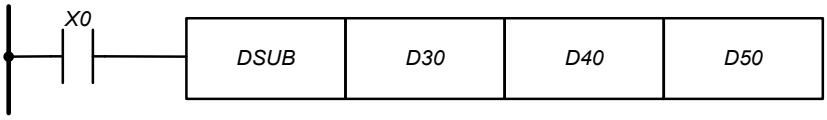
- El valor de datos en (S2) se resta del valor de datos (S1). El resultado de la resta se almacena en el operando de destino (D). La operación se realiza en tipos de datos enteros con signo.
  (S1) – (S2) = (D)
- El bit más significativo contiene el signo del resultado: 0 – signo de un número positivo, 1 – signo de un número negativo.
- Al ejecutar una instrucción de 32 bits, los 16 bits inferiores deben indicarse en el operando. El siguiente registro de datos contiene los 16 bits superiores.

Ejemplos:



(D0) - (D10) = (D20)

Si X0 está activado, se calcula la diferencia entre los valores de datos en los registros D0 y D10. El resultado se guarda en el registro de datos D20.



(D31, D30) - (D41, D40) = (D51, D50)

Si X0 está activado, se calcula la diferencia entre los valores de datos en los registros (D31, D30) y (D41, D40). El resultado se guarda en los registros de datos (D51, D50).

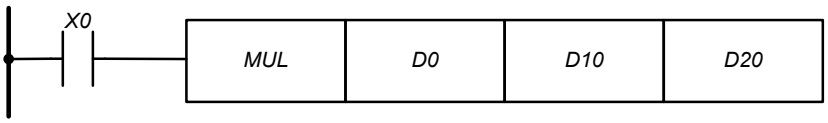
MUL	S1 S2 D	D P	Multiplicación de datos numéricos

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Descripción:

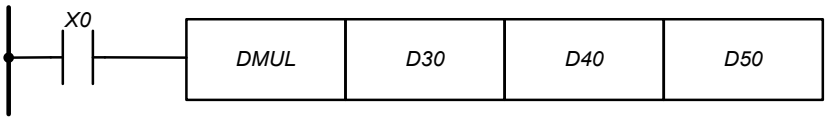
- Los datos en los operandos (S1) y (S2) se multiplican juntos. El resultado se almacena en el operando de destino (D). La operación se realiza en tipos de datos enteros con signo.
  (S1) x (S2) = (D)
- El bit más significativo contiene el signo del resultado: 0 – signo de un número positivo, 1 – signo de un número negativo.
- Al ejecutar una instrucción de 32 bits, los 16 bits inferiores deben indicarse en el operando. El siguiente registro de datos contiene los 16 bits superiores.

Ejemplos:



(D0) * (D10) = (D20)

Si X0 está activado, los valores en los registros de datos D0 y D10 se multiplican entre sí. El resultado se guarda en el registro de datos D20.



(D31, D30) * (D41, D40) = (D51, D50)

Si X0 está activado, los valores en los registros (D31, D30) y (D41, D40) se multiplican entre sí. El resultado se guarda en los registros de datos (D51, D50).

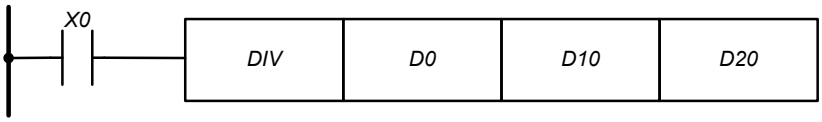
DIV	S1 S2 D	D P	División de datos numéricos

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●					●	●	●	●	●
S2	●	●					●	●	●	●	●
D							●	●	●	●	●

Descripción:

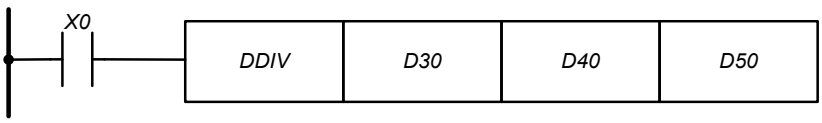
- El valor del operando de origen (S1) se divide por el valor de los datos del operando de origen (S2). Toda la parte del resultado de la división se almacena en el operando de destino (D). La operación se realiza en tipos de datos enteros con signo.
  (S1) / (S2) = (D)
- El bit de orden superior contiene el signo del resultado: 0 – signo de un número positivo, 1 – signo de un número negativo.
- Al ejecutar una instrucción de 32 bits, los 16 bits inferiores deben indicarse en el operando. El siguiente registro de datos contiene los 16 bits superiores.
- La división por ceroproduce un error.

Ejemplos:



(D0) / (D10) = (D20)

Si X0 está activado, se realiza la división de los valores de datos en los registros de datosD0 y D10. El resultado se guarda en el registro de datosD20.



(D31, D30) / (D41, D40) = (D51, D50)

Si X0 está activado, se realiza la división de los valores de datos en los registros (D31, D30) y (D41, D40). El resultado se guarda en los registros de datos (D51, D50).

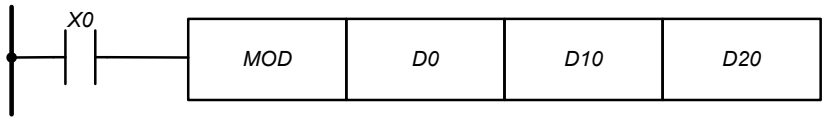
MOD	S1 S2 D	D P	Cálculo del resto de la división

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●					●	●	●	●	●
S2	●	●					●	●	●	●	●
D							●	●	●	●	●

Descripción:

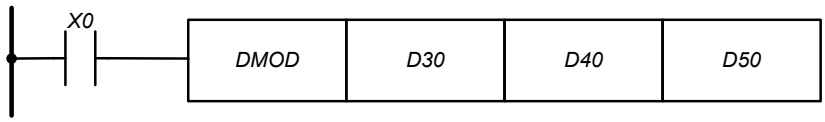
- El valor del operando de origen (S1) se divide por el valor de los datos del operando de origen (S2). El resto de la división se almacena en el operando de destino (D). La operación se realiza en tipos de datos enteros con signo.
  (S1) % (S2) = (D)
- El bit de orden superior contiene el signo del resultado: 0 – signo de un número positivo, 1 – signo de un número negativo.
- Al ejecutar una instrucción de 32 bits, los 16 bits inferiores deben indicarse en el operando. El siguiente registro de datos contiene los 16 bits superiores.
- La división por ceroproduce un error

Ejemplos:



(D0) % (D10) = (D20)

Si X0 está activado, se realiza la división de los valores de datos en los registros de datos D0 y D10. El resultado (resto de la división) se guarda en el registro de datos D20.



(D31, D30) % (D41, D40) = (D51, D50)

Si X0 está activado, se realiza la división de los valores de datos en los registros (D31, D30) y (D41, D40). El resultado (resto en la división) se guarda en los registros de datos (D51, D50).

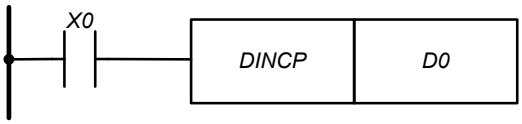
INC	D	D P	Incremento de datos numéricos

	K	H	F	X	Y	M	T	C	A	B	D
D							●	●	●	●	●

Descripción:

El valor en el operando(D) se incrementa en 1.

Ejemplo:



El valor en los registros de datos (D1, D0) se incrementa en 1 si la condición de entrada X0 está activada. La instrucción se activa debido a la función de pulso conectada para que el proceso de suma no se realice en cada ciclo de programa.

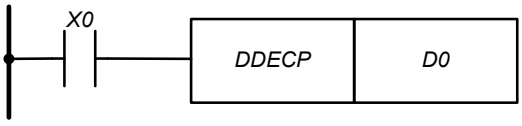
DEC	D	D P	Decremento de datos numéricos

	K	H	F	X	Y	M	T	C	A	B	D
D							●	●	●	●	●

Descripción:

El valor en el operando(D) se decrementa en 1.

Ejemplo:



El valor en los registros de datos (D1, D0) se decrementa en 1 si la condición de entradaX0 está activada. La instrucción se activa debido a la función de pulso conectada para que la decrementación no se realice en cada ciclo de programa.

WAND	S1 S2 D	D P	multiplicación lógica de datos numéricos (operación "AND")

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●					●	●	●	●	●
S2	●	●					●	●	●	●	●
D							●	●	●	●	●

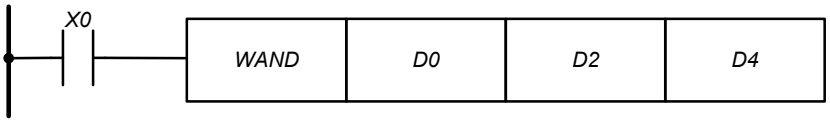
Nota: WAND es una instrucción de 16 bits, DAND es una instrucción de 32 bits.

Descripción:

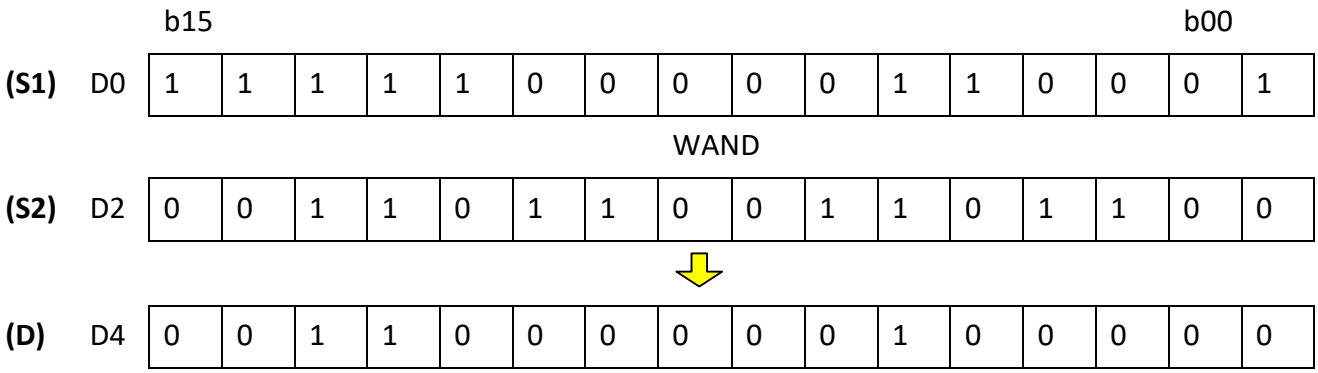
- La operación "multiplicación lógica AND" para datos numéricos es una operación a nivel de bit (realizada bit por bit).
- Los valores de los operandos fuente (S1) y (S2) se multiplican bit por bit. El resultado se almacena en el operando de destino (D).
- La tabla de verdad de la multiplicación lógica:

(S1)	(S2)	(D)
1	1	1
1	0	0
0	1	0
0	0	0

Ejemplo:



Si X0 = 1, se realiza la multiplicación lógica de los valores de los registros de datos D0 y D2. El resultado se guarda en el registro de datos D4.



WOR	S1 S2 D	D P	suma lógica de datos numéricos (operación OR)

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

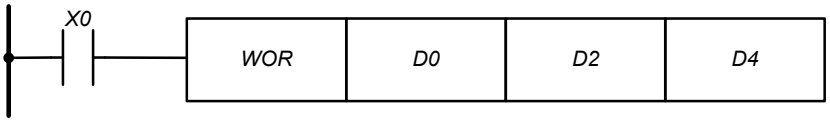
Nota: WOR es una instrucción de 16 bits, DOR es una instrucción de 32 bits.

Descripción:

- La operación "suma lógica OR" para datos numéricos es una operación a nivel de bit (realizada bit por bit).
- Los valores de los operandos fuente (S1) y (S2) se suman bit por bit. El resultado se almacena en el operando de destino (D).
- La tabla de verdad de la suma lógica:

(S1)	(S2)	(D)
1	1	1
1	0	1
0	1	1
0	0	0

Ejemplo:



Si X0 = 1, se realiza la assuma lógicade los valores de los registros de datos D0 y D2. El resultado se guarda en el registro de datos D4.

		b15															b00
(S1)	D0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
									WOR								
(S2)	D2	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
									⇩								
(D)	D4	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1

WXOR	S1 S2 D	D P	Operación lógica "OR exclusivo"

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

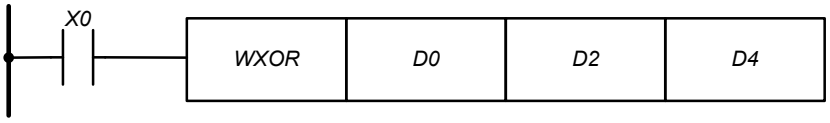
Nota: WXOR es una instrucción de 16 bits, DXOR es una instrucción de 32 bits.

Descripción:

- La operación "o exclusivo OR" para datos numéricos es una operación a nivel de bit (realizada bit por bit)
- Los valores de los operandos fuente (S1) y (S2) se procesan bit por bit. El resultado se almacena en el operando de destino (D).
- La tabla de verdad de la o exclusivo OR:

(S1)	(S2)	(D)
1	1	0
1	0	1
0	1	1
0	0	0

Ejemplo:



Si X0 = 1, la operación “o exclusivo OR” se realiza con valores en los registros de datos D0 y D2. El resultado de la operación se guarda en el registro de datos D4.

		b15														b00
(S1)	D0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
									WXOR							
(S2)	D2	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
									⇩							
(D)	D4	1	1	0	0	1	1	1	0	0	1	0	1	1	1	0	1

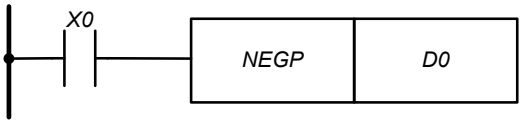
NEG	D	D P	Negación lógica

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Descripción:

Operación de negación lógica (inversión de todos los bits en forma binaria y suma con 1) para datos numéricos.

Ejemplo:



Si X0 = 1, la operación de negación lógicay modificación se realiza con el valor en el operandoD0.

13931 → -13931 + 1 = -13931

		b15														b00
(D)	D0	0	0	1	1	0	1	1	0	0	1	1	0	1	0	1	1
(D)	D0	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	1

-13931 → 13930 + 1 = 13931

		b15														b00
(D)	D0	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	1
(D)	D0	0	0	1	1	0	1	1	0	0	1	1	0	1	0	1	1

ROL	D n	D P	Desplazamiento cíclico a la izquierda

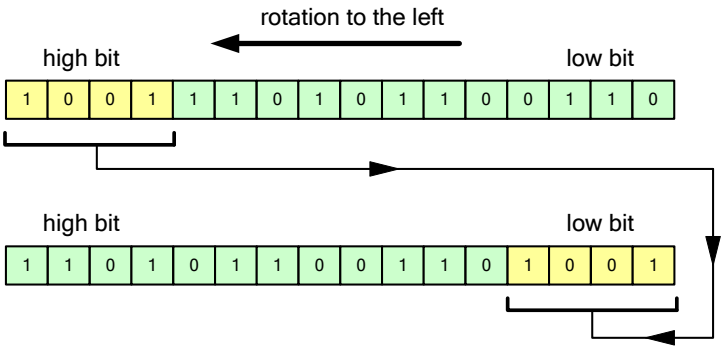
	K	H	F	X	Y	M	T	C	A	B	D
D							●	●	●	●	●
n	●	●					●	●	●	●	●

Descripción:

Rotación de bitsen (n) posiciones a la izquierda.

Ejemplo:

Si X0 = 1, los bits del valor en el registro de datosD10 rotan en 4 bits a la izquierda y el valor se modifica.



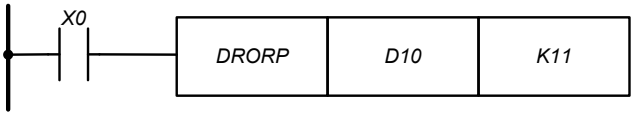
ROR	D n	P	Desplazamiento cíclico a la derecha

	K	H	F	X	Y	M	T	C	A	B	D
D							●	●	●	●	●
n	●	●					●	●	●	●	●

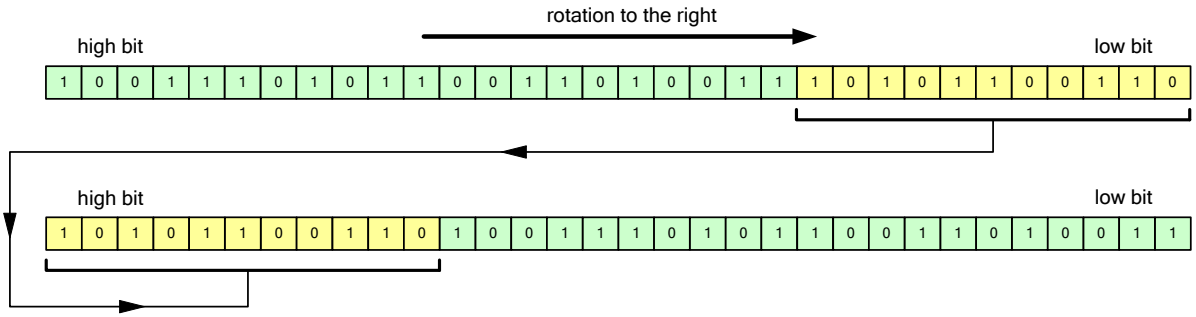
Descripción:

Rotación de bitsen (n) posiciones a la derecha.

Ejemplo:



Si X0 = 1, los bits del valor en el registro de datosD10 rotan en 11 bits a la derecha y el valor se modifica.



ZRST	D1 D2	D P	Reinicio de grupo de operandos

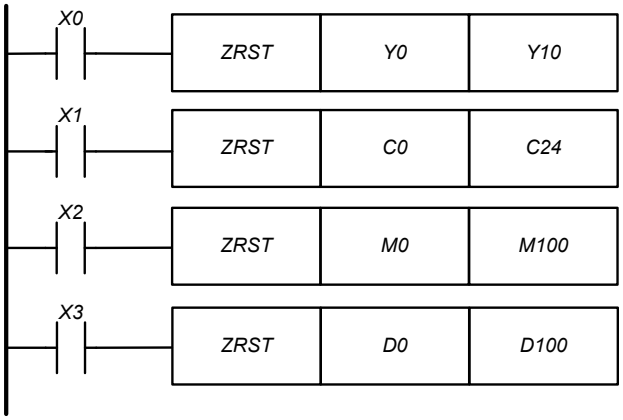
	K	H	F	X	Y	M	T	C	A	B	D
D1					●	●	●	●	●	●	●
D2					●	●	●	●	●	●	●

Descripción:

Los valores de varios operandos que se siguen uno tras otro (región de operandos) se pueden restablecer mediante la instrucción ZRST. Los contactos de bitse desactivan, los registros se establecen en un valor de "0".

- Los operandos(D1) y (D2) determinan la región que se va a restablecer.
- Los operandos en (D1) y (D2) deben ser del mismo tipo.
- (D1) – la primera dirección de la región, (D2) – la última dirección de la región.
- (D1) debe ser menor que (D2).

Ejemplo:



Cuando se cumplen las condiciones de entrada, los operandos de bitY0 ... Y10, M0 ... M100 se desactivan (pasan al estado "0"). Los operandos numéricos C0 ... C24, D0 ... D100 se establecen en el valor real "0". Las bobinasy los contactos correspondientes se desconectan.

DECO	S D n	P	Decodificador 8 → 256 bits

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●		●	●	●	●	●	●	●	●
D					●	●	●	●	●	●	●
n	●	●					●	●	●	●	●

Nota: Si (D) es un operando de bit: (n) = 1…8. Si (D) es un operando: (n) = 1…4. Si (n) está fuera del rango posible, la instrucción se ejecuta con el (n) máximo posible dependiendo de (D).

Descripción:

Decodificación de datos. Los datos en (n) operandos se decodifican comenzando desde la dirección de inicio, que se especifica en (S). El operando (D) determina la dirección inicial del destino (donde se escribe el resultado del descifrado).

(n) es el número de operandos cuyos datos deben ser decodificados. Al especificar el operando de biten (D), se debe observar lo siguiente: 1 ≤ (n) ≤ 8. Al especificar el operando numérico en (D), se debe observar lo siguiente: 1 ≤ (n) ≤ 4.

(S) es la dirección de inicio de los operandos cuyos datos deben ser decodificados.

2^n– número de operandos a ser decodificados.

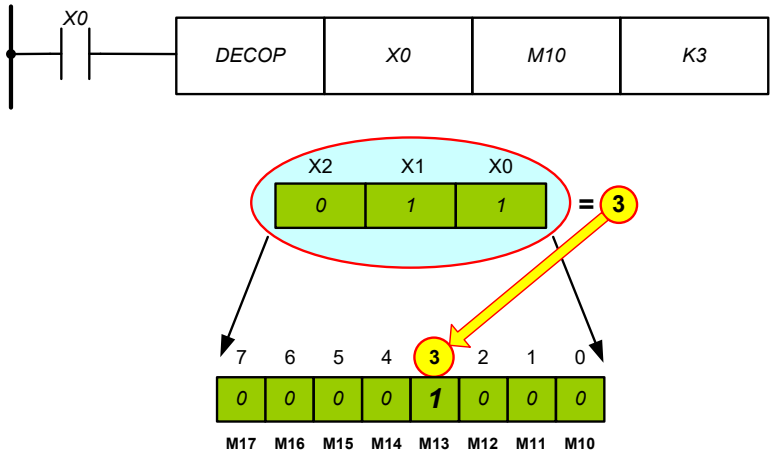
(D) es una dirección de inicio al operando de destino.

¡Atención! La instrucción no se ejecuta si (n) = 0.

En consecuencia, la salida permanece activa si las condiciones de entrada al final de la acción se desactivan de nuevo.

Ejemplo:

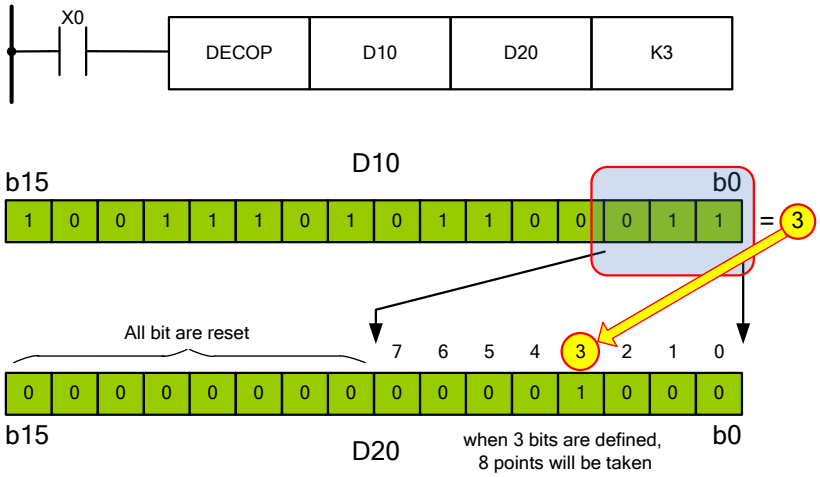
Uso de una instrucción DECO con operandos de biten (D).



Si (n) = 3, los operandos de entrada X0, X1 y X2 son procesados. $2^n = 2^3 = 8$ direcciones son usadas como un destino M10…M17.

El valor de los operandos de entrada es 1 + 2 = 3. Así que la tercera dirección del destino, i.e., relé M13, es activada. Si el valor del operando de entrada "0" es procesado, entonces el relé M10 es activado.

Uso de una instrucción DECO con operandos numéricos en (D).



Los 3 bits de orden inferior del registro de datos D10 son decodificados. El resultado de la decodificación 1 + 2 = 3 es transferido al registro de datos D20. El tercer bites activado en este registro de datos.

Si el valor para (n) <3, entonces todos los bits innecesarios de un número mayor en las direcciones de destino son puestos a cero.

ENCO	S D n	P	Codificador 256 → 8 bits

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●		●	●	●	●	●	●	●	●
D					●	●	●	●	●	●	●
n	●	●					●	●	●	●	●

Nota: Si (D) es un operando de bit: (n) = 1…8. Si (D) es un operando: (n) = 1…4. Si (n) está fuera del rango posible, la instrucción se ejecuta con el (n) máximo posible dependiendo de (D).

Descripción:

Codificación de datos. Los datos en 2^noperandos se codifican comenzando desde la dirección de inicio, que se especifica en (S). El operando (D) determina la dirección inicial del destino (donde se escribe el resultado del descifrado).

2^nes el número de operandos cuyos datos deben ser codificados.

(n) – el número de operandos de destino.

Al especificar el operando de biten (S), se debe observar lo siguiente: 1 ≤ (n) ≤ 8. Al especificar el operando numérico en (S), se debe observar lo siguiente: 1 ≤ (n) ≤ 4.

(S) es la dirección de inicio de los operandos cuyos datos deben ser codificados.

(D) es la dirección de inicio del operando de destino.

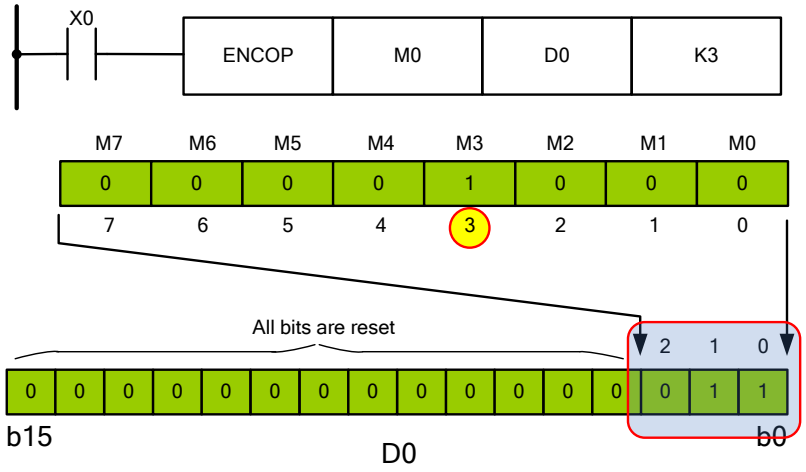
Si varios operandos especificados en (S) tienen el valor “1”, entonces solo el bit de orden inferiores procesado.

¡Atención! La instrucción no se ejecuta si (n) = 0.

En consecuencia, la salida permanece activa si las condiciones de entrada al final de la acción se desactivan de nuevo.

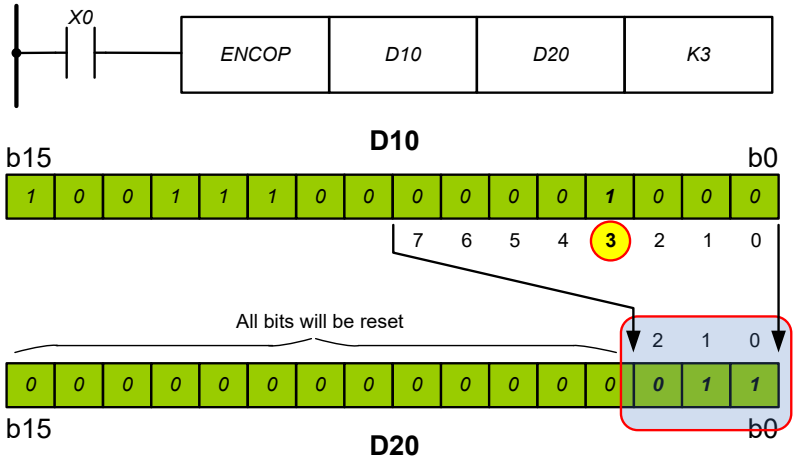
Ejemplo:

Uso de una instrucción ENCO con operandos de biten (S).



Si 2n = 23 = 8, entonces las direcciones del relé de salida son M0.. .M7. A medida que la salida 3d se activa (es decir, M3 = 1), el valor 3 se escribe en el registro de datosD0.

Uso de una instrucción ENCO con operandos numéricos en (S).



El tercer bit está activado en el registro de datosD10. Así que el valor 3 es codificado y guardado en el registro de datosD20.

SUM	S D	D P	Suma de bits individuales

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●					●	●	●	●	●
D							●	●	●	●	●

Descripción:

- Determinando el número de bits activos en una palabra de datos. La instrucción cuenta los bits activados en (S).
- El valor del resultado es escrito en (D).

Si una instrucción de 32 bises procesada, entonces los 16 bits superiores (D + 1) de los operandos de destino(D) son puestos a cero, ya que el número máximo de bits activados en (S) es 32.

BON	S D n	D P	Comprobar el estado del bit

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●					●	●	●	●	●
D					●	●					
n	●	●					●	●	●	●	●

Note: La condición necesaria: (n) = 0…15 (16 bit), (n) = 0…31 (32 bit).

Descripción:

Un solo bites verificado dentro de la palabra de datos. Si el bit(n) es activado en (S), entonces el bit correspondiente en (D) es activado.

SQR	S D	D P	Cálculo de la raíz cuadrada

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●					●	●	●	●	●
D											●

Descripción:

Cálculo de la raíz cuadrada$(D) = \sqrt{(S)}$

Se calcula la raíz cuadrada del valor en el operando(S), el resultado se redondea a un entero y se escribe en el operando(D). La operación se realiza en tipos de datos enteros con signo.

¡Atención! La raíz cuadrada de un número negativo siempre lleva a un error.

FLT	S D	D P	Convertir entero a punto flotante

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●					●	●	●	●	●
D											●

Descripción:

La instrucción FLT convierte un número entero con signo a formato de punto flotante.

- El entero en el operando (S) se convierte a un número de punto flotante. El resultado se guarda en el operando(D).
- El resultado de la conversión es siempre un número de 32 bits.

PWM	S1 S2 D		PWM salida de pulso

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●					●	●	●	●	●
S2	●	●					●	●	●	●	●
D					●						

Note: El valor del operando(S1) debe ser menor o igual al valor del operando(S2).

Descripción:

(S1) – ancho de pulso, t.

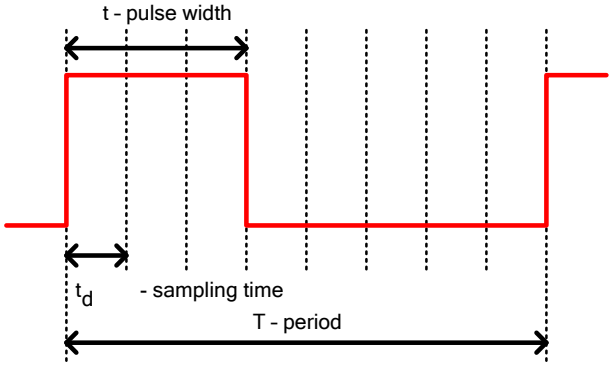
(S2) – duración del período, T.

(D) – dirección de salida, Y6 o Y7.

Los valores válidos para (S1) y (S2) son de 1 a 32767. (S1) y (S2) son el número de intervalos de muestreo. El período de muestreo es establecido simultáneamente para ambos canales, 100 µs o 10 µs, por el registro especialD356 (ver sección 4.6para más detalles).

Una señal PWM está presente en la salida mientras la señal en la entrada de la instrucción PWM esté activa.

Ejemplo:



Sea el período de muestreo de 100 µs. Cuando la condición de entrada X0 = 1 se cumple, la señal PWM con un período de 8 x 100 µs = 0.8 ms y una duración de pulso de 3 x 100 µs = 0.3 ms aparece en la salida Y6.

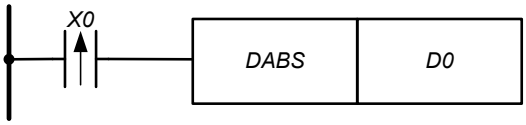
ABS	D	D P	valor absoluto

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Descripción:

La instrucción ABS escribe el valor absoluto de un número en el operando (D). Si el valor en (D) es negativo, entonces después de ejecutar la instrucción ABS, el signo "-" se descarta y el número se vuelve positivo. Si (D) tiene un valor positivo, entonces no se producen cambios.

Ejemplo:



Cuando se cumple la condición de entrada, se determina el módulo del número en el registro (D1, D0).

POW	S1 S2 D	D P	elevación a una potencia

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●					●	●	●	●	●
S2	●	●					●	●	●	●	●
D											●

Descripción:

Elevación a una potencia: $(D) = (S1)^{(S2)}$.

La instrucción POW eleva el valor en el operando(S1) a la potencia (S2). El resultado se guarda en el operando (D). La operación se realiza en tipos de datos enteros con signo.

DECMP	S1 S2 D	D P	comparación de números de punto flotante

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●	●								●
S2	●	●	●								●
D					●	●					

Nota:

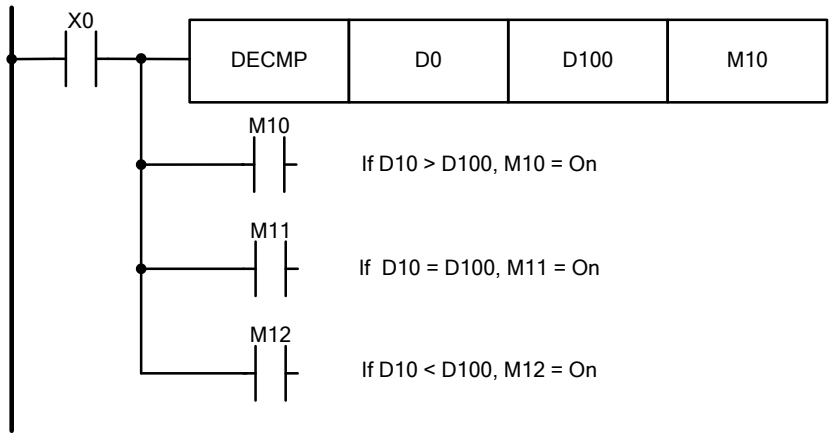
- Instrucción de 32 bits solamente.
- El operando(D) toma 3 direcciones consecutivas.
- Los tipos K y H no se convierten a F, sino que se proyectan en un área de memoria. Para convertir tipos de datos enterosa datos de punto flotante, use la instrucción **FLT.**

Descripción:

La instrucción DECMP realiza la comparación de dos números binarios de punto flotante y emite el resultado de la comparación.

- La instrucción DECMP compara el número de punto flotante en (S1) con el número de punto flotante en (S2).
- El resultado de la comparación se almacena en 3 direcciones sucesivas.
- Si el número en (S2) es menor que el número en (S1), entonces el operando de bit (D) se activa.
- Si el número en (S2) es igual al número en (S1), entonces el operando de bit ((D) +1) se activa.
- Si el número en (S2) es mayor que el número en (S1), entonces el operando de bit ((D) +2) se activa.
- Los operandos de salida sondeados permanecen activados después de deshabilitar las condiciones de entrada de la instrucción DECMP.

Ejemplo:



Cuando el contacto X0 se activa, el número de punto flotante especificado en D100 (S2) se compara con el número de punto flotante especificado en D0 (S1). Si el número en D100 es menor que el número D0, entonces el relé M10 se activa. Si el número en D100 es igual al número D0, entonces el relé M11 se activa. Si el número en D100 es mayor que el número D0, entonces el relé M12 se activa.

Para obtener los resultados de la comparación en la forma: ≤ ≥ ≠, puede utilizar combinaciones de contactos en paralelo M10 - M12. Para restablecer el resultado, puede utilizar las instrucciones RST, ZRST.

DEZCP	S1 S2 S D	D P	comparación de punto flotante por zona

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
S	•	•	•								•
D					•	•					

Nota:

- Instrucción de 32 bits solamente.
- El operando(D) toma 3 direcciones consecutivas.
- Los tipos K y H no se convierten a F, sino que se proyectan en un área de memoria. Para convertir tipos de datos enterosa datos de punto flotante, use la instrucción **FLT.**

Descripción:

Comparación del número de punto flotante con el área seleccionada (indicada) y la salida del resultado de la comparación.

- La instrucción DEZCP compara el número de punto flotante en el operando (S1) con el área entre (S1) y (S2).
- El resultado de la comparación se almacena en 3 direcciones sucesivas.
- Si el número en el operando (S) es menor que los números en los operandos (S1) y (S2), entonces el operando de bit (D) se activa.
-
- Si el número en el operando (S) es igual a los números entre (S1) y (S2), entonces el operando de bit ((D) +1) se activa.
- Si el número en el operando (S) es mayor que los números entre (S1) y (S2), entonces el operando de bit ((D) +2) se activa.
- Los operandos de salida sondeados permanecen activados después de que las condiciones de entrada de la instrucción DEZCP se deshabilitan.
- Si el número en el operando (S1) es mayor que el número en el operando (S2), entonces todos los bits en (D), (D+1) y (D+2) se restablecerán.

DEADD	S1 S2 D	D P	suma de números de punto flotante

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Nota:

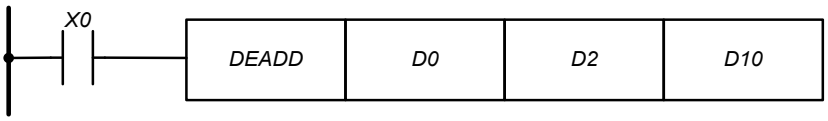
- Instrucción de 32 bits solamente.
- Los tipos K y H no se convierten a F, sino que se proyectan en un área de memoria. Para convertir tipos de datos enterosa datos de punto flotante, use la instrucción **FLT.**

Descripción:

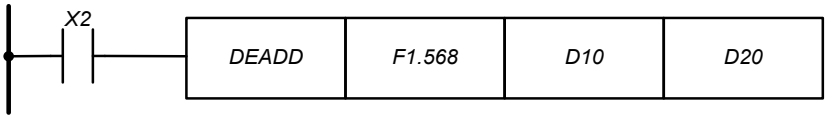
Cálculo de la suma de dos números en formato binario de punto flotante.

- Los números de punto flotante especificados en (S1) y (S2) se suman. El resultado de la suma se almacena en el operando de destino (D).
- Se utilizan dos registros consecutivos para cada operando.
- El mismo operando puede ser usado para la fuente y para el destino. En este caso, el resultado calculado se almacena nuevamente en el operando de origeny se puede utilizar para el siguiente cálculo. Este proceso se repite en cada ciclo de programa.

Ejemplos:



Cuando la entrada X0 se activa, el número de punto flotante en (D3, D2) se suma al número de punto flotante en (D1, D0). El resultado se guarda en (D11, D10).



Cuando la entrada X2 se activa, el número de punto flotante en (D11, D10) se suma a la constante F1.568. El resultado se guarda en (D21, D20).

DESUB	S1 S2 D	D P	resta de números de punto flotante

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Nota:

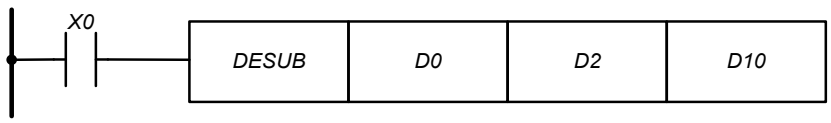
- Instrucción de 32 bits solamente.
- Los tipos K y H no se convierten a F, sino que se proyectan en un área de memoria. Para convertir tipos de datos enterosa datos de punto flotante, use la instrucción **FLT.**

Descripción:

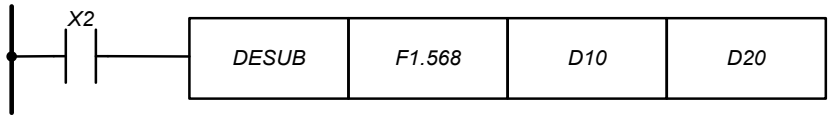
La instrucción DESUB calcula la diferencia de dos números en formato binario de punto flotante.

- El número de punto flotante especificado en (S2) se resta del número de punto flotante especificado en (S1). El resultado se guarda en (D).
- Se utilizan dos registros consecutivos para cada operando.
- El mismo operandopuede ser usado para la fuente y para el destino. En este caso, el resultado calculado se almacena nuevamente en el operando de origeny se puede utilizar para el siguiente cálculo. Este proceso se repite en cada ciclo de programa.

Ejemplos:



Cuando la entrada X0 se activa, el número de punto flotanteen (D3, D2) se resta del número de punto flotanteen (D1, D0). El resultado se guarda en (D11, D10).



Cuando la entrada X2 se activa, el número de punto flotante en (D11, D10) se resta de la constante F1.568. El resultado se guarda en (D21, D20).

DEMUL	S1 S2 D	D P	multiplicación de números de punto flotante

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Nota:

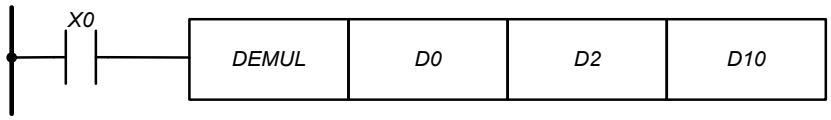
- Instrucción de 32 bits solamente.
- Los tipos K y H no se convierten a F, sino que se proyectan en un área de memoria. Para convertir tipos de datos enterosa datos de punto flotante, use la instrucción **FLT.**

Descripción:

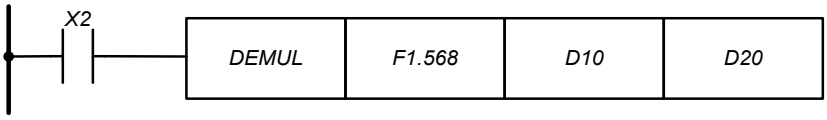
Multiplicación de dos números en formato binario de punto flotante.

- El número de coma flotante especificado en el operando(S1) se multiplica por el número de coma flotante en el operando(S2). El resultado se guarda en el operando(D).
- Se utilizan dos registros consecutivos para cada operando.
- El mismo operando puede ser usado para la fuente y para el destino. En este caso, el resultado calculado se almacena nuevamente en el operando de origeny se puede utilizar para el siguiente cálculo. Este proceso se repite en cada ciclo de programa.

Ejemplos:



Cuando la entrada X0 se activa, el número de punto flotante en (D1, D0) se multiplica por el número de punto flotante en (D3, D2). El resultado se guarda en (D11, D10).



Cuando la entrada X2 se activa, la constante F1.568 se multiplica por el número de punto flotanteen (D11, D10). El resultado se guarda en (D21, D20).

DEDIV	S1 S2 D	D P	división de números de punto flotante

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Nota:

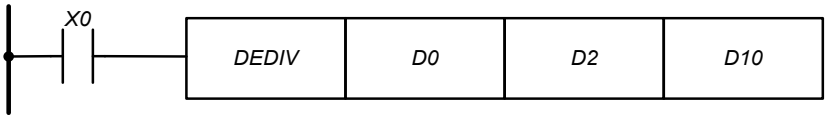
- Instrucción de 32 bits solamente.
- Los tipos K y H no se convierten a F, sino que se proyectan en un área de memoria. Para convertir tipos de datos enterosa datos de punto flotante, use la instrucción **FLT.**

Descripción:

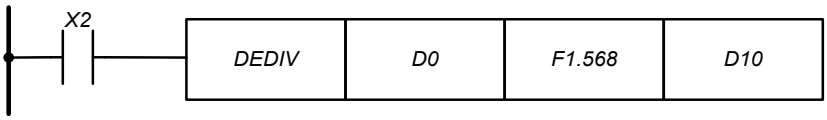
Cálculo del cociente de dividir dos números en formato binario de punto flotante.

- El número de punto flotante especificado en el operando (S1) se divide por el número de punto flotante especificado en el operando (S2). El resultado se guarda en (D).
- Se utilizan dos registros consecutivos para cada operando.
- El mismo operando puede ser usado para la fuente y para el destino. En este caso, el resultado calculado se almacena nuevamente en el operando de origeny se puede utilizar para el siguiente cálculo. Este proceso se repite en cada ciclo de programa.
- El operando (S2) no puede ser cero porque no se permite la división por cero.

Ejemplos:



Cuando la entrada X0 se activa, el número de punto flotanteen (D1, D0) se divide por el número de punto flotanteen (D3, D2). El resultado se guarda en (D11, D10).



Cuando la entrada X2 se activa, el número de punto flotante en (D1, D0) se divide por la constante F1.568. El resultado se guarda en (D11, D10).

DESQR	S D	D P	raíz cuadrada en formato de punto flotante

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•								•
D											•

Nota:

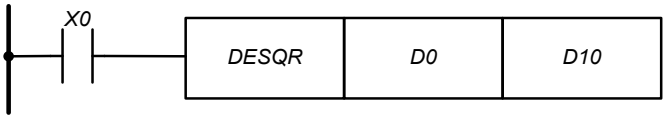
- Instrucción de 32 bits solamente.
- Los tipos K y H no se convierten a F, sino que se proyectan en un área de memoria. Para convertir tipos de datos enterosa datos de punto flotante, use la instrucción **FLT.**
- Condición necesaria: (S) ≥ 0

Descripción:

Calculando la raíz cuadradade un número binario en formato de punto flotante.

- La raíz cuadrada se calcula a partir del número de coma flotante especificado en el operando (S). El resultado se guarda en (D).
- Se utilizan dos registros consecutivos para cada operando.
- El mismo operando puede ser usado para la fuente y para el destino. En este caso, el resultado calculado se almacena nuevamente en el operando de origeny se puede utilizar para el siguiente cálculo. Este proceso se repite en cada ciclo de programa.

Ejemplos:



$\sqrt{(D1, D0)} \rightarrow (D11, D10)$

Cuando la entrada X0 está activada, se calcula la raíz cuadrada del número de punto flotante en (D1, D0). El resultado se guarda en (D11, D10).

Cuando la entrada X0 se activa, se calcula la raíz cuadrada de la constante F16. El resultado se guarda en (D11, D10).

DEPOW	S1 S2 D	D P	elevación a una potencia en formato de punto flotante

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Nota:

- Instrucción de 32 bits solamente.
- Los tipos K y H no se convierten a F, sino que se proyectan en un área de memoria. Para convertir tipos de datos enterosa datos de punto flotante, use la instrucción **FLT.**

Descripción:

Elevando un número a una potencia en formato de punto flotante binario.

- El número especificado en (S1) se eleva a la potencia (S2). El resultado se guarda en el operando(D).
  $(S1)^{(S2)} = (D)$.
- Se utilizan dos registros consecutivos para cada operando.
- El mismo operando puede ser usado para la fuente y para el destino. En este caso, el resultado calculado se almacena nuevamente en el operando de origeny se puede utilizar para el siguiente cálculo. Este proceso se repite en cada ciclo de programa.

INT	S D	D P	conversión de un número de punto flotante a un entero

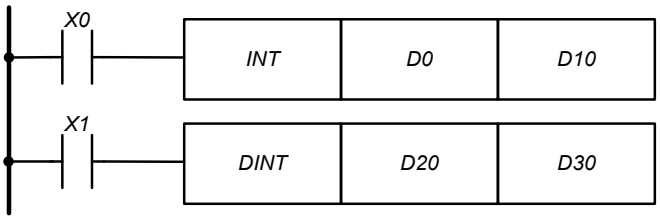
	K	H	F	X	Y	M	T	C	A	B	D
S	.										●
D							●	●	●	●	●

Descripción:

La instrucción INT convierte números de punto flotante a enteros, redondeado al entero más cercano.

- El número de punto flotante especificado en (S) se redondeado al entero más cercanoy se guarda en (D).
- El operando de origenes siempre una doble palabra.
- Cuando se utiliza la instrucción INT, el operando de palabraes el operandodel destino.
- Cuando se utiliza la instrucción DINT, el operando de destinoes un operando de doble palabra.
- La instrucción INT es la función inversade la instrucción FLT.

Ejemplo:



Cuando la entrada X0 está activada, el número de punto flotanteen (D0, D1) se redondea al valor entero inferior más cercano. El resultado se guarda en D10.

Cuando la entrada X1 está activada, el número de punto flotanteen (D20, D21) se redondea al valor entero inferior más cercano. El resultado se guarda en (D30, D31).

TRD	D	P	lectura de datos de tiempo

	K	H	F	X	Y	M	T	C	A	B	D
D											•

Nota: El operandoD toma 3 direcciones consecutivas.

Descripción:

Leyendo el valor actual del reloj de tiempo real.

- Usando la instrucción TRD, los datos de tiempo realse leen (horas, minutos, segundos).
- Estos datos se guardan en 3 direcciones de operando consecutivas (D).

Ejemplo:

Cuando la entrada X0 se activa, los datos de tiempo realse leen y se guardan en los registros D0 .... D2

registro	Función	Valor	Ejemplo	
D0	Segundos	0…59	20	12:36:20
D1	Minutos	0…59	36	
D2	Horas	0…23	12	

TWR	S	P	registro de datos de tiempo

	K	H	F	X	Y	M	T	C	A	B	D
S											•

Nota: El operando(S) toma 3 direcciones consecutivas.

Descripción:

La instrucción TWR se utiliza para cambiar los datos de tiempo real (horas, minutos, segundos).

Los datos se toman de 3 direcciones consecutivas, especificadas en (S).

Si los valores en (S) exceden el rango de valores permitido, surge un error.

Ejemplo:

Cuando se satisface la condición de entrada, el reloj de tiempo real del controlador se establece en los valores indicados en los registros D0 .... D2.

registro	Función	Valor	Ejemplo	
D0	Segundos	0…59	42	03:11:42
D1	Minutos	0…59	11	
D2	Horas	0…23	3	

DRD	D	P	lectura de datos de fecha

	K	H	F	X	Y	M	T	C	A	B	D
D											•

Nota: El operandoD toma 3 direcciones consecutivas.

Descripción:

Leyendo el valor de la fecha actual.

- Usando la instrucción DRD, la fecha actual se lee (día, mes, año).
- Estos datos se guardan en 3 direcciones de operando consecutivas (D).

Ejemplo:

Cuando la entrada X0 se activa, los datos de tiempo realse leen y se guardan en los registros D0 .... D2.

registro	Función	Valor	Ejemplo	
D0	Día	1…31	26	26.01.22
D1	Mes	1…12	1	
D2	Año	21…99	22	

DWR	S	P	registro de datos de fecha

	K	H	F	X	Y	M	T	C	A	B	D
S											•

Nota: El operando(S) toma 3 direcciones consecutivas.

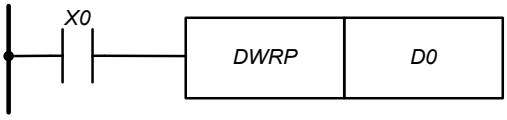
Descripción:

La instrucción DWR se utiliza para cambiar los datos de la fecha (día, mes, año).

Los datos se toman de 3 direcciones consecutivas, especificadas en (S).

Si los valores en (S) exceden el rango de valores permitido, surge un error.

Ejemplo:



Cuando se satisface la condición de entrada, el reloj de tiempo real del controlador se establece en los valores indicados en los registros D0 .... D2.

registro	Función	Valor	Ejemplo	
D0	Día	1…31	4	04.02.22
D1	Mes	1…12	2	
D2	Año	21…99	22	

LD#	S1 S2	D	operaciones lógicas de tipo contacto

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●		●	●	●	●	●	●	●	●
S2	●	●		●	●	●	●	●	●	●	●

Nota:

- El símbolo **#** es &, |, ^.
- Los operandos de bit se toman de 16 o 32, dependiendo del tipo de instrucción, y se convierten a un tipo de datos entero para su posterior procesamiento.

Descripción:

Realizando la operación lógica "AND", "OR" y "o exclusivo" en los operandos(S1) y (S2), y activando el contacto LD, dependiendo del resultado de la operación.

Las instrucciones LD# en el programa se encuentran a la izquierda y abren una conexión lógicao son condiciones para la ejecución de comandos a la derecha.

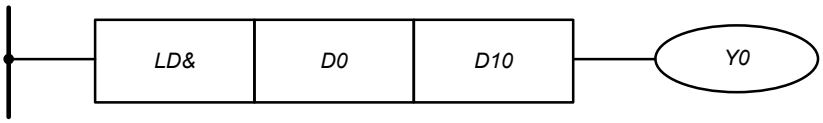
instrucciones de 16 bits	instrucciones de 32 bits	Contacto cerrado si:	Contacto abierto si:
LD&	DLD&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
LD|	DLD|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
LD^	DLD^	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: multiplicación lógica (AND)

|: assuma lógica (OR)

^: OR exclusive (XOR)

Ejemplo:



AND#	S1 S2	D	operaciones lógicas de tipo contacto conexión en serie

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●		●	●	●	●	●	●	●	●
S2	●	●		●	●	●	●	●	●	●	●

Nota:

- El símbolo **#** es &, |, ^.
- Los operandos de bit se toman de 16 o 32, dependiendo del tipo de instrucción, y se convierten a un tipo de datos entero para su posterior procesamiento.

Descripción:

Realizando la operación lógica "AND", "OR", "o exclusivo" en los operandos(S1) y (S2), y activando el contacto AND dependiendo del resultado de la operación.

Las instrucciones AND# en el programa se encuentran después de los comandos LD y crean una conexión lógica AND.

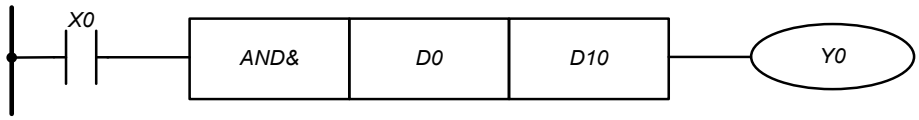
instrucciones de 16 bits	instrucciones de 32 bits	Contacto cerrado si:	Contacto abierto si:
AND&	DAND&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
AND|	DAND|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
AND^	DAND^	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: multiplicación lógica (AND)

|: assuma lógica (OR)

^: OR exclusive (XOR)

Ejemplo:



OR#	S1 S2	D	operaciones lógicas de tipo contacto conexión en paralelo

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●		●	●	●	●	●	●	●	●
S2	●	●		●	●	●	●	●	●	●	●

Nota:

- El símbolo **#** es &, |, ^.
- Los operandos de bit se toman de 16 o 32, dependiendo del tipo de instrucción, y se convierten a un tipo de datos entero para su posterior procesamiento.

Descripción:

Realizando la operación lógica "AND", "OR", "o exclusivo" en los operandos(S1) y (S2), y activando el contacto OR dependiendo del resultado de la operación.

Las instrucciones OR# en el programa se encuentran a la izquierda en paralelo a la instrucción LD y crean una conexión lógica OR.

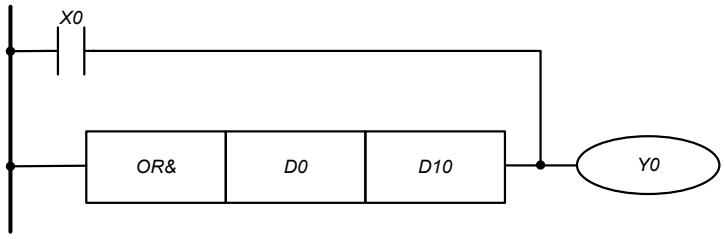
instrucciones de 16 bits	instrucciones de 32 bits	Contacto cerrado si:	Contacto abierto si:
OR&	DOR&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
OR|	DOR|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
OR^	DOR^	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: multiplicación lógica (AND)

|: assuma lógica (OR)

^: OR exclusive (XOR)

Ejemplo:



LD*	S1 S2	D	Operaciones de comparación de tipo de contacto

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●		●	●	●	●	●	●	●	●
S2	●	●		●	●	●	●	●	●	●	●

Nota:

- El símbolo **#** es =, >, <, <>, ≤, ≥.
- Los operandos de bit se toman de 16 o 32, dependiendo del tipo de instrucción, y se convierten a un tipo de datos entero para su posterior procesamiento.

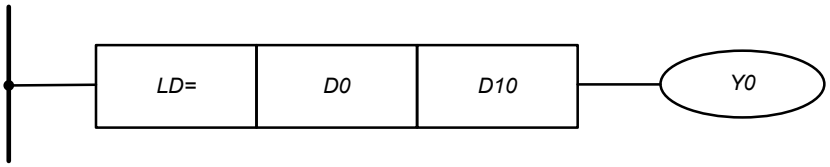
Descripción:

Comparación de los valores de los operandos(S1) y (S2), y activando un contacto LD, dependiendo del resultado de la operación.

- Las instrucciones LD* en el programa se encuentran a la izquierda y comienzan una conexión lógicao son condiciones para la ejecución de instrucciones a la derecha.
- Si el resultado de la comparación es verdadero, el contacto LDse activa.
- Si el resultado de la comparación es falso, el contacto LDse desactiva.

instrucciones de 16 bits	instrucciones de 32 bits	Contacto cerrado si:	Contacto abierto si:
LD=	DLD=	(S1) = (S2)	(S1) ≠ (S2)
LD>	DLD>	(S1) > (S2)	(S1) ≤ (S2)
LD<	DLD<	(S1) < (S2)	(S1) ≥ (S2)
LD<>	DLD<>	(S1) ≠ (S2)	(S1) = (S2)
LD<=	DLD<=	(S1) ≤ (S2)	(S1) > (S2)
LD>=	DLD>=	(S1) ≥ (S2)	(S1) < (S2)

Ejemplo:



AND*	S1 S2	D	Operaciones de comparación de tipo de contacto conexión en serie

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●		●	●	●	●	●	●	●	●
S2	●	●		●	●	●	●	●	●	●	●

Nota:

- El símbolo **#** es =, >, <, <>, ≤, ≥.
- Los operandos de bit se toman de 16 o 32, dependiendo del tipo de instrucción, y se convierten a un tipo de datos entero para su posterior procesamiento.

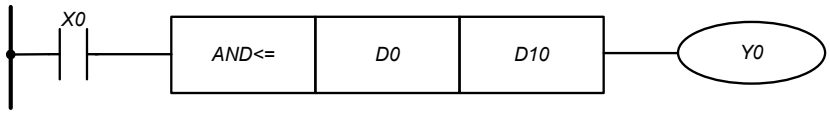
Descripción:

Comparación de los valores de los operandos S1 y S2, y activando el contacto and, dependiendo del resultado de la operación.

- Las instrucciones AND* en el programa se encuentran después de los comandos LD y crean una conexión lógica AND.
- Si el resultado de la comparación es verdadero, el contacto ANDse activa.
- Si el resultado de la comparación es falso, el contacto ANDse desactiva.

instrucciones de 16 bits	instrucciones de 32 bits	Contacto cerrado si:	Contacto abierto si:
AND=	DAND=	(S1) = (S2)	(S1) ≠ (S2)
AND>	DAND>	(S1) > (S2)	(S1) ≤ (S2)
AND<	DAND<	(S1) < (S2)	(S1) ≥ (S2)
AND<>	DAND<>	(S1) ≠ (S2)	(S1) = (S2)
AND<=	DAND<=	(S1) ≤ (S2)	(S1) > (S2)
AND>=	DAND>=	(S1) ≥ (S2)	(S1) < (S2)

Ejemplo:



OR*	S1 S2	D	Operaciones de comparación de tipo de contacto conexión en paralelo

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Nota:

- El símbolo **#** es =, >, <, <>, ≤, ≥.
- Los operandos de bit se toman de 16 o 32, dependiendo del tipo de instrucción, y se convierten a un tipo de datos entero para su posterior procesamiento.

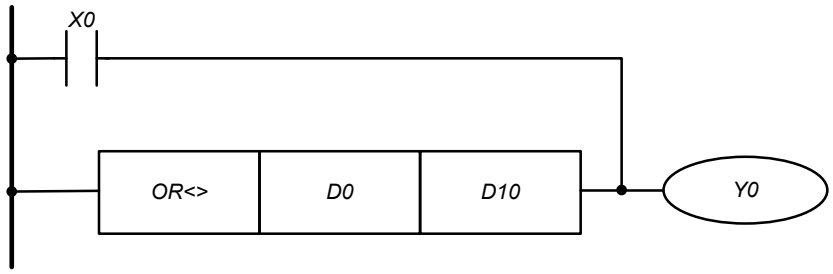
Descripción:

Comparación de los valores de los operandos S1 y S2, y activación del contacto or, dependiendo del resultado de la operación.

- Las instrucciones OR* en el programa se encuentran a la izquierda en paralelo a la instrucción LD y crean una conexión OR lógica.
- Si el resultado de la comparación es verdadero, el contacto OR se activa.
- Si el resultado de la comparación es falso, el contacto OR se desactiva.

instrucciones de 16 bits	instrucciones de 32 bits	Contacto cerrado si:	Contacto abierto si:
OR=	DOR=	(S1) = (S2)	(S1) ≠ (S2)
OR>	DOR>	(S1) > (S2)	(S1) ≤ (S2)
OR<	DOR<	(S1) < (S2)	(S1) ≥ (S2)
OR<>	DOR<>	(S1) ≠ (S2)	(S1) = (S2)
OR<=	DOR<=	(S1) ≤ (S2)	(S1) > (S2)
OR>=	DOR>=	(S1) ≥ (S2)	(S1) < (S2)

Ejemplo:



8. Instrucciones para el control del controlador de motor paso a paso

El controlador de motor paso a pasose controla mediante comandos que especifican los parámetros de rotación o movimiento. Todos los comandos se dividen en dos grupos: **RUN** y **MOVE**. El grupo RUN está diseñado para controlar la velocidad actual del accionamiento, y MOVE para controlar el movimiento. Para iniciar una rotación después de seleccionar un comando y configurar los parámetros del controlador, se llama a la instrucción **SPIN**.

SPIN		P	Realizar el movimiento especificado
Dirección	**Tipo de objeto**	**Descripción del objeto Modbus**	
5100h	**Bobinas**	Escribir “1” en este objeto (incluso si no se restablece) desencadena un movimiento parametrizado, el restablecimiento se ignora.	

Descripción:

La instrucción inicia una rotación parametrizada. Los parámetros de movimiento se establecen en los registros de servicio, que, al igual que las instrucciones del controlador de motor paso a paso, son accesibles a través del protocolo modbusen modo RUN. La instrucción SPIN tiene menor prioridad que xSTOP y xHIZ. Para evitar errores, se recomienda verificar los indicadores BUSY_MOVE y BUSY_RUN antes de llamar a la instrucción SPIN. A continuación, se muestra una descripción detallada de los registros de servicio del controlador.

Dirección	**Tipo de objeto**	**Registro de servicio**		**Tamaño (bit)**	**Descripción**
		Número	**Nombre**		
5000h	Registros de retención	D357	SPEED	32	La velocidad establecida (objetivo) del motor (en pulsos por segundo, pps) para los comandos del grupo RUN y la velocidad máxima para los comandos del grupo MOVE. El umbral inferior es de 8 pps, el límite superior es de 120000pps bajo la condición de restablecimiento de FS_SW_EN. Si FS_SW_EN está establecido, el límite superior es $SPEED_{max}=120000*2^{U\_STEP}$.

Dirección	Tipo de objeto	Registro de servicio		Tamaño (bit)	Descripción
		Número	Nombre		
5002h	Registros de retención	D359	MIN_SPEED	32	La velocidad de rotación mínima para los comandos del grupo RUN. Para el grupo MOVE, también es la velocidad mínima si el indicador CMIN_SPD_EN no está configurado. Si CMIN_SPD_EN está configurado, la velocidad mínima óptima se calcula automáticamente.
5004h	Registros de retención	D361	ACC	16	Aceleración, pps^2.
5005h	Registros de retención	D362	DEC	16	Deceleración, pps^2.
5006h	Registros de retención	D363	ABS	32	Posición actual. La unidad de valor es igual al desplazamiento por la cantidad de un micropaso.
5008h	Registros de entrada	D365	U_POS	16	La posición actual del micropaso está en cuatro pasos completos. Indica la posición del rotor del motor relativa a los polos del estator. El registro es de solo lectura.

<table>
<tr><th>Dirección</th><th>Tipo de objeto</th><th colspan="2">Registro de servicio</th><th>Tamaño (bit)</th><th>Descripción</th></tr>
<tr><th></th><th></th><th>Número</th><th>Nombre</th><th></th><th></th></tr>
<tr><td>5009h</td><td>Registros de retención</td><td>D366</td><td>U_STEP</td><td>16</td><td>Micropasos
<table>
<tr><th>Valor del registro</th><th>Micropasos</th></tr>
<tr><td>0</td><td>1/1</td></tr>
<tr><td>1</td><td>1/2</td></tr>
<tr><td>2</td><td>1/4</td></tr>
<tr><td>3</td><td>1/8</td></tr>
<tr><td>4</td><td>1/16</td></tr>
<tr><td>5</td><td>1/32</td></tr>
<tr><td>7</td><td>1/128</td></tr>
<tr><td>8</td><td>1/256</td></tr>
</table>
Un valor de “6” no es válido.</td></tr>
<tr><td>500Ah</td><td>Registros de retención</td><td>D374</td><td>DIR</td><td>16</td><td rowspan="2">dirección de rotación:
"1" – adelante,
"0" – atrás.</td></tr>
<tr><td>500Ah</td><td>Bobinas</td><td colspan="3">DIR</td></tr>
<tr><td>500Bh</td><td>Registros de retención</td><td>D377</td><td>FS_SPD_THR</td><td>32</td><td>El valor umbral para pasar de micropasos al modo de paso completo, medido en pasos completos por segundo.</td></tr>
</table>

Dirección	Tipo de objeto	Registro de servicio		Tamaño (bit)	Descripción
		Número	Nombre		
500Dh	Registros de retención	D379	FS_SW_EN	16	Ajustar el objeto a “1” activa la función de transformación: el controlador pasa al modo de paso completo después de alcanzar la velocidad especificada en FS_SPD_THR. Esta función permite obtener un mayor par motor a altas velocidades. Restablecer el objeto a “0” desactiva esta función (transformación/aumento de par).
500Dh	Bobinas	FS_SW_EN			
500Eh	Registros de retención	D372	TARGET_POS	32	La posición objetivo que se debe alcanzar. La unidad de valor es igual al desplazamiento de un micropaso.
5010h	Registros de retención	D376	CMD	16	Un comando de movimiento para el driver (consulte la tabla siguiente).

Dirección	Tipo de objeto	Registro de servicio		Tamaño (bit)	Descripción
		Número	Nombre		
5011h	Registros de retención	D375	SW_INPUT	16	El número de entrada para la conexión del sensor, para los comandos GOUNTIL_... y …RELEASE. Valores 0…7. Atención: La declaración de interrupción en el programa principal es necesaria para las entradas utilizadas, el manejador de interrupción puede dejarse vacío. Ejemplo: Un sensor está conectado a la entrada X3 (IN3), el programa de usuario debe incluir la siguiente parte: *FEND* *I 1003* *IRET* *END*
5012h	Registros de retención	D367	ACC_CUR	16	Corriente de aceleración, mA. Rango de valores válidos: SMSD-1.5Modbus ver.3 - 150…1500; SMSD-4.2Modbus – 1000…5000 SMSD-8.0Modbus – 2800…10000.
5013h	Registros de retención	D368	DEC_CUR	16	Corriente de deceleración, mA. Rango de valores válidos: SMSD-1.5Modbus ver.3 - 150…1500; SMSD-4.2Modbus – 1000…5000 SMSD-8.0Modbus – 2800…10000.

Dirección	Tipo de objeto	Registro de servicio		Tamaño (bit)	Descripción
		Número	Nombre		
5014h	Registros de retención	D369	STEADY_CUR	16	Corriente de velocidad constante, mA. Rango de valores válidos: SMSD-1.5Modbus ver.3 - 150…1500; SMSD-4.2Modbus – 1000…4200 SMSD-8.0Modbus – 2800…8000
5015h	Registros de retención	D370	HOLD_CUR	16	Corriente de retención, mA. Rango de valores válidos: SMSD-1.5Modbus ver.3 - 150…1500; SMSD-4.2Modbus – 1000…4200 SMSD-8.0Modbus – 2800…8000
5016h	Registros de retención	D382	CMIN_SPD_EN	16	"1" - usar el cálculo automático de la velocidad inicial y final de movimiento para los comandos del grupo MOVE. "0" - usar MIN_SPEED como la velocidad inicial y final.
5017h	Registros de retención	D380	ERROR_SET_HIZ	16	Los bits de este registro determinan qué errores del controlador deben conducir a la desenergización del motor (el eje gira libremente): el estado de alta impedancia (hiz).
5017h	Bobinas	TERMAL_OVER_CURRENT_ERROR_SET_HIZ			Bit 0 del registro D380. Si el bit está activado, un error TERMAL_ERROR (sobrecalentamiento del circuito del controlador – el registro D381) provoca la desenergización del motor (estado de alta impedancia (hiz)).

Dirección	Tipo de objeto	Registro de servicio		Tamaño (bit)	Descripción
		Número	Nombre		
5018h	Bobinas	SOFTWARE_ERROR_SET_HIZ			bit 1 del registro D380. Si el bit está activado, un error SOFTWARE_ERROR (el error interno del controlador – el registro D381) causa la desenergización del motor (estado de alta impedancia (hiz)).
5019h	Bobinas	CMD_ERROR_SET_HIZ			bit 2 del registro D380. Si el bit está activado, un error CMD_ERROR (imposibilidad de procesar un comando entrante – el registro D381) causa la desenergización del motor (estado de alta impedancia (hiz)).
501Ah	Bobinas	DATA_ERROR_SET_HIZ			bit 3 del registro D380. Si el bit está activado, un error DATA_ERROR (entrada de datos incorrecta en ACC, DEC, U_STEP – el registro D381) causa la desenergización del motor (estado de alta impedancia (hiz)).
501Bh	Bobinas	OUT_OF_LIM_MIN_SPD_ ERROR_SET_HIZ			bit 4 del registro D380. Si el bit está activado, un error OUT_OF_LIM_MIN_SPD_ERROR_SET_HIZ (velocidad establecida inferior a la mínima – el registro D381) causa la desenergización del motor (estado de alta impedancia (hiz)).

Dirección	Tipo de objeto	Registro de servicio		Tamaño (bit)	Descripción
		Número	Nombre		
501Ch	Bobinas	OUT_OF_LIM_MAX_SPD_ ERROR_SET_HIZ			bit 5 del registro D380. Si el bit está activado, un error OUT_OF_LIM_MAX_SPD_ERROR_SET_ HIZ (superación de la velocidad máxima posible – el registro D381) causa la desenergización del motor (estado de alta impedancia (hiz)).
501Dh	Bobinas	UNREACHABLE_FS_SPD_ ERROR_SET_HIZ			Bit 6 del registro D380. Si el bit está activado, un error UNREACHABLE_FS_SPD_ERROR_SET_HI Z (incapaz de alcanzar el umbral de velocidad de paso completo en el modo de aumento de par – el registro D381) provoca la desenergización del motor (estado de alta impedancia (hiz)).
501Eh	Bobinas	NOT_APP_FS_PARAM_ ERROR_SET_HIZ			Bit 7 del registro D380. Si el bit está activado, un error NOT_APP_FS_PARAM_ERROR_SET_HIZ (la transición desde el aumento de par no es posible mientras el motor está girando – el registro D381) provoca la desenergización del motor (estado de alta impedancia (hiz)).
5027h	Registros de retención	D381	ERROR_CODE	16	Código de error. Consulte a continuación la descripción de los bits del registro (errores).
5027h	Bobinas	TERMAL_OVER_CURRENT_ ERROR			Bit 0 del registro D381 TERMAL_OVER_ CURRENT_ERROR – sobrecalentamiento del circuito del controlador o exceso de corriente en los devanados del motor.

Dirección	Tipo de objeto	Registro de servicio		Tamaño (bit)	Descripción
		Número	Nombre		
5028h	Bobinas	SOFT_ERROR			Bit 1 del registro D381 SOFTWARE_ERROR – error interno del controlador.
5029h	Bobinas	CMD_ERROR			Bit 2 del registro D381 CMD_ERROR – No se puede procesar un comando entrante.
502Ah	Bobinas	DATA_ERROR			Bit 3 del registro D381 DATA_ERROR – Entrada de datos incorrecta en ACC, DEC, U_STEP
502Bh	Bobinas	OUT_OF_LIM_MIN_SPD_ERROR			Bit 4 del registro D381 OUT_OF_LIM_ MIN_SPD_ERROR – Lavelocidad establecida es inferior a la mínima.
502Ch	Bobinas	OUT_OF_LIM_MAX_SPD_ERROR			Bit 5 del registro D381 OUT_OF_LIM_ MAX_SPD_ERROR –Se excede la velocidad máxima posible.
502Dh	Bobinas	UNREACHABLE_FS_SPD_ERROR			Bit 6 del registro D381 UNREACHABLE _FS_SPD_ERROR –no se puede alcanzar el umbral de velocidad de paso completo en modo de aumento de par.
502Eh	Bobinas	NOT_APP_FS_PARAM_ERROR			Bit 7 del registro D381 . NOT_APP_FS_ PARAM_ERROR –la transición desde el aumento de par no es posible mientras el motor está girando.

Dirección	Tipo de objeto	Registro de servicio		Tamaño (bit)	Descripción
		Número	Nombre		
502F	Bobinas	OVLO/UVLO_INTERNAL_ PROTECTION_ERROR			Error: la tensión de los circuitos de alimentación internos está fuera del rango estándar.
5030	Bobinas	VS_OUT_OF_RANGE_ERROR			Error: la tensión de alimentación está fuera de rango.
5037h	Registros de entrada	D371	MOTOR_STATUS	16	El registro muestra el estado actual del motor y del sistema de control. La descripción de los bits del registro se encuentra a continuación.
5037h	Entradas discretas	HIZ			Bit 0 del registro D371. Estado de alta impedancia (hiz) del motor (las fases están desexcitadas).
5038h	Entradas discretas	STOP			Bit 1 del registro D371. Modo de retención.
5039h	Entradas discretas	ACCELERATING			Bit 2 del registro D371. Aceleración.
503Ah	Entradas discretas	DECELERATING			Bit 3 del registro D371. Deceleración.
503Bh	Entradas discretas	STEADY			Bit 4 del registro D371. Rotación a velocidad constante.
503Ch	Entradas discretas	BUSY_MOVE			Bit 5 del registro D371. Indicador de la imposibilidad de aplicar los comandos del grupo MOVE.
503Dh	Entradas discretas	BUSY_RUN			Bit 6 del registro D371. El indicador de la imposibilidad de usar los comandos del grupo RUN.

Dirección	Tipo de objeto	Registro de servicio		Tamaño (bit)	Descripción
		Número	Nombre		
5047h	Registros de entrada	D383	CURRENT_SPD	32	Velocidad actual, pps. (Se recomienda usar el indicador STEADY como un evento de alcance de una velocidad determinada).
5048h	Registros de retención	D385	EMERGENCY_DEC	32	Deceleración de emergencia, pps^2.
5100h	Bobinas	SPIN (APPLY_CMD)			Establecer el objeto en "1" o aplicar la instrucción SPIN activa la ejecución del comando especificado en el registro CMD (D376), con los parámetros especificados.
5101h	Bobinas	TORQUE (APPLY_CURRENT)			Establecer un objeto en "1" o aplicar una instrucción TORQUE aplica los valores de corriente ACC_CUR, DEC_CUR, RUN_CUR y HOLD_CUR para el motor.
5102h	Bobinas	HSTOP (HARD_STOP)			Establecer un objeto en "1" o aplicar la instrucción HSTOP detiene inmediatamente el motor y lo pone en modo de retención.
5103h	Bobinas	HHIZ (HARD_HIZ)			Establecer un objeto en "1" o aplicar la instrucción HHIZ pone inmediatamente el motor en estado de alta impedancia (hiz).
5104h	Bobinas	SSTOP (SLOW_STOP)			Establecer un objeto en "1" o aplicar la instrucción SSTOP detiene el motor según la DEC y, a continuación, lo pone en modo de retención.

Dirección	Tipo de objeto	Registro de servicio		Tamaño (bit)	Descripción
		Número	Nombre		
5105h	Bobinas	SHIZ (SLOW_HIZ)			Establecer un objeto en “1” o aplicar la instrucción SHIZ detiene el motor según la DEC y, a continuación, lo pone en estado de alta impedancia (hiz).

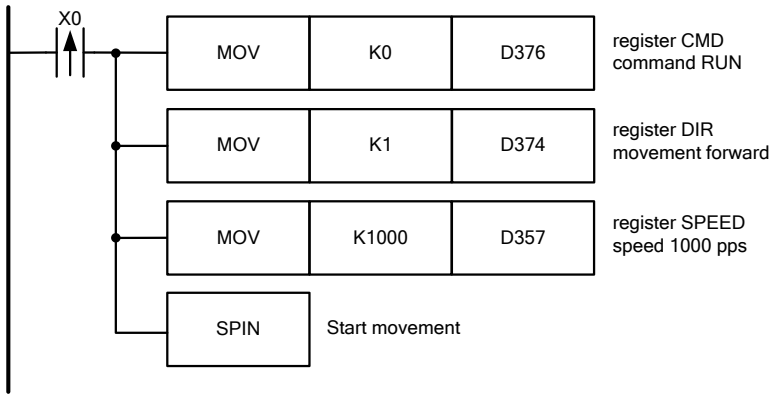
Comando de movimiento (registro-CMD)

Valor	Grupo	Nombre	Descripción
0	RUN	RUN	Rotación según la velocidad establecida SPEED, la aceleración ACC, la deceleración DEC y la dirección DIR.
1	MOVE	MOVE	Desplazamiento del número de pasos especificado TARGET_POS con los parámetros de movimiento dados SPEED, ACC, DEC y DIR.
2	MOVE	GOTO	Mover a la posición especificada TARGET_POS con los parámetros de movimiento dados SPEED, ACC, DEC. DIR depende de la posición actual, el valor proporcionado no se tiene en cuenta.
3	MOVE	GOTO_DIR	Desplazarse a la posición especificada TARGET_POS con los parámetros de movimiento dados SPEED, ACC, DEC y DIR.
4	MOVE	GOHOME	Desplazarse a la posición "0" con los parámetros de movimiento dados SPEED, ACC, DEC. El comando equivale a GOTO "0".

5	RUN	GOUNTIL_SLOWSTOP	Movimiento con una velocidad SPEED, aceleración ACC y dirección DIR establecidas hasta que el sensor SW_INPUT se active en un flanco ascendente, con una comprobación inicial del nivel de entrada, seguido de una deceleración y parada según el valor DEC establecido.
6	RUN	GOUNTIL_FRONT_SLOWSTOP	Movimiento con una velocidad SPEED, aceleración ACC y dirección DIR establecidas hasta que el sensor SW_INPUT se active en un flanco ascendente, con la consiguiente deceleración y parada según el valor DEC establecido.
7	RUN	GOUNTIL_HARDSTOP	Movimiento con una velocidad SPEED, aceleración ACC y dirección DIR establecidas hasta que el sensor SW_INPUT se active en un flanco ascendente, con una comprobación inicial del nivel de entrada, y luego pasa al modo de retención.
8	RUN	GOUNTIL_FRONT_HARDSTOP	Movimiento con una velocidad SPEED, aceleración ACC y dirección DIR establecidas hasta que el sensor SW_INPUT se active en un flanco ascendente, y luego pasa al modo de retención.

9	RUN	RELEASE	Movimiento con una velocidad SPEED, aceleración ACC y dirección DIR establecidas hasta que el sensor SW_INPUT se active en el flanco descendente, con una comprobación inicial del nivel de entrada, y luego pasa al modo de retención.
10	RUN	FRONT_RELEASE	Movimiento con una velocidad SPEED, aceleración ACC y dirección DIR establecidas hasta que el sensor SW_INPUT se active en el flanco descendente y luego pasa al modo de retención.

Ejemplo:



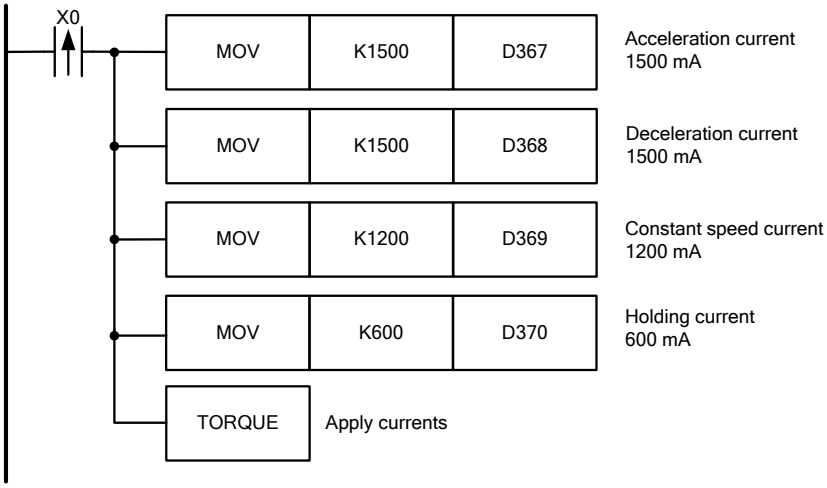
Cuando se cumple la condición de entrada, el comando de movimiento, la dirección y la velocidad de rotación se establecen en los registros de servicio. La siguiente instrucción SPIN inicia el movimiento.

TORQUE		P	Aplicar las corrientes configuradas al motor
5101h	**Bobinas**	Escribir “1” en un objeto (incluso si no se restablece) aplica al motor los valores de corriente, que están configurados en los registros de servicio. El restablecimiento (valor = “0”) se ignora.	

Descripción:

Aplicar esta instrucción establece las corrientes de operación del motor indicadas en los registros ACC_CUR (D367), DEC_CUR (D368), RUN_CUR (D369) y HOLD_CUR (D370).

Ejemplo:

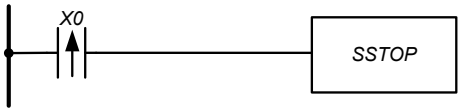


SSTOP		P	El motor se detiene según el parámetro DEC y luego entra en modo de retención.
5104h	**Bobinas**	Escribir “1” en un objeto (incluso si no se restablece) aplica el frenado del motor según DEC y luego pasa al modo de retención, el restablecimiento se ignora.	

Descripción:

Aplicar el frenado del motor de acuerdo con DEC y luego pasar al modo de retención. Esta instrucción anula la operación SPIN, tiene la misma prioridad que SHIZ, pero puede ser anulada por las instrucciones HSTOP y HHIZ.

Ejemplo:

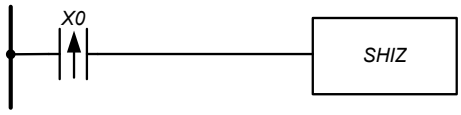


SHIZ		P	El motor se detiene según el parámetro DEC y luego pasa al modo HiZ.
5105h	**Bobinas**	Escribir “1” en un objeto (incluso si no se restablece) aplica el frenado del motor según DEC y luego pasa al modo HiZ, el restablecimiento se ignora.	

Descripción:

Aplicar el frenado del motor de acuerdo con DEC, seguido de la desenergización de los devanados. Esta instrucción anula la operación SPIN, tiene la misma prioridad que SSTOP, pero puede ser anulada por las instrucciones HSTOP y HHIZ.

Ejemplo:

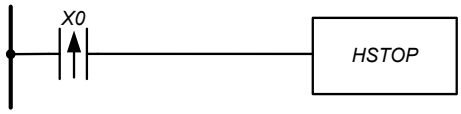


HSTOP		P	Detiene inmediatamente el motor y luego pasa al modo de retención.
5102h	**Bobinas**	Escribir “1” en un objeto (incluso si no se restablece) detiene inmediatamente el motor y luego pasa al modo de retención, el restablecimiento se ignora.	

Descripción:

Detiene inmediatamente el motor y luego entra en modo de retención. Esta instrucción anula SPIN, SSTOP, SHIZ y tiene la misma prioridad que HHIZ.

Ejemplo:

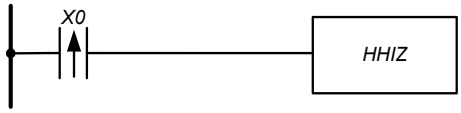


HHIZ		P	Detiene inmediatamente el motor y luego pasa al modo HiZ.
5103h	**Bobinas**	Escribir “1” en un objeto (incluso si no se restablece) detiene inmediatamente el motor y luego pasa al modo HiZ, el restablecimiento se ignora.	

Descripción:

Detiene inmediatamente el motor y luego entra en modo HiZ (el motor se desenergiza, el eje gira libremente). Esta instrucción anula SPIN, SSTOP, SHIZ y tiene la misma prioridad que HSTOP.

Ejemplo:



9. Parámetros de comunicación

El controladortiene interfaces USB y RS-485, ambas tienen el mismo acceso a registros y operandos de bit. La interfaz USB es un puerto com virtual (VCP), está destinada principalmente a configurar el controladory grabar programas de usuario, por lo tanto, tiene parámetros de comunicaciónfijos: Modbus ASCII, ID 1, 115200 baudios, 7, Par, 1. Las variaciones de parámetros para RS-485 se indican en Apéndice Aen la sección "Parámetros de comunicación de la interfaz RS-485". Parámetros de comunicaciónde fábrica para RS-485: Modbus RTU, ID 1, 9600 baudios, 8, par, 1.

9.1. Cambiar la configuración de comunicaciónpara RS-485

Establezca los parámetros de comunicación requeridos de acuerdo con la sección "Parámetros de comunicación de la interfaz RS-485" del Apéndice A. Para que los cambios surtan efecto, reiniciar el dispositivo. Esto se puede hacer apagando y encendiendo la alimentación o configurando el objeto Bobinas 8101h (Restablecer).

Ejemplo:

Es necesario cambiar los parámetros de comunicación a los siguientes: Modbus RTU, ID 100, 128000 baudios, 8, Impar, 1. Existen todas las combinaciones posibles de parámetros de comunicación en Apéndice A.

Secuencia de acción:

1) Escribir el valor 100d en los registros de retención 8103h – cambiar la dirección del dispositivo (ID) a 100.
2) Ajuste de bobinas 8100h – selección de protocolo RTU.
3) Escribir el valor 128000d en los registros de retención 8100h – ajuste de la velocidad de transferencia de datos 128000 baudios.
4) Escribir el valor 1d en los registros de retención 8102h – selección de tipo de paridad Impar.
5) Ajuste de bobinas 8101h – reiniciar el controlador.

9.2. Protocolo Modbus

Se recomienda encarecidamente leer la especificación del protocolo en el sitio. http://modbus.org/. Las funciones del protocolo compatibles se presentan en la tabla a continuación:

			Función	Código
Acceso a datos	Entradas discretas	1 bit	Leer Entradas Discretas	02(02h)
	Bobinas		Leer Bobinas	01(01h)
			Escribir Bobina Única	05(05h)
			Escribir Bobinas Múltiples	15(0Fh)
	Registros de entrada	16-bit	Leer Registro de Entrada	04(04h)
	Registros de retención		Leer Registros de Retención	03(03h)
			Escribir Registro Único	06(06h)
			Escribir Registros Múltiples	16(10h)
			Leer/Escribir Registros Múltiples	23(17h)
			Escribir Registro con Máscara	22(16h)

Los códigos de error del protocolo se presentan en la siguiente tabla:

Código de error	Descripción
01(01h)	*FUNCIÓN ILEGAL* El código de función no se puede procesar.
02(02h)	*DIRECCIÓN DE DATOS ILEGAL* La dirección del registro especificada en la solicitud no está disponible.
03(03h)	*VALOR DE DATOS ILEGAL* El valor contenido en el campo de datos de la solicitud no es válido.
04(04h)	*FALLO DEL DISPOSITIVO SERVIDOR* Se ha producido un error irrecuperable al realizar la acción solicitada.

Códigos de error registrados durante el procesamiento de paquetes de protocolo se presentan en las tablas siguientes.

Dirección	Tipo	Tamaño	Descripción
E003h	Registros de entrada	16-bit	Código de error al procesar la trama Modbus.
E003h	Bobinas	1 bit	Indicador de un error durante el intercambio a través del protocolo modbus.

Código de error	Descripción
0001h	Error de asignación de memoria.
0002h	Error de suma de comprobación.
0003h	Se produjo un error al recibir y procesar un paquete de difusión.
0004h	Desajuste del tamaño de la trama.
0005h	Error de función (0Fh). No todos los bits se han sobrescrito. .
0006h	Error de función (10h). No todos los registros se han sobrescrito. .
0007h	Error de función (17h). No todos los registros se han sobrescrito. .
0008h	Trama perdida debido a un error de DMA.
0009h	Trama perdida debido al desbordamiento de la pila de procesamiento de tramas.

Si el dispositivo está al final de la línea de comunicación RS-485, conecte una resistencia de terminación encendiendo el interruptor junto al conector RS-485, como se muestra en la Fig. 31.

Fig. 31. Conexión de la resistencia de terminación

10. Configuración del reloj de tiempo real

El controlador tiene un reloj de tiempo real, que se alimenta con una fuente interna (batería CR2032), lo que garantiza el funcionamiento del reloj mientras la alimentación principal está apagada. La misma batería se utiliza para el funcionamiento de los registros no volátilesy la configuración de seguridad de los parámetros de comunicación del controlador. El indicador **BAT** se ilumina en caso de ausencia o fallo inminente de la batería interna CR2032. El reloj de tiempo realse puede configurar a través del programa de usuarioutilizando la instrucción TWR o a través del protocolo modbus en el siguiente orden:

1) Desactive la sobreescritura automática de los registros de retención 8110h…8112h restableciendo las bobinas 8110h.
2) Registre el valor de tiempo actual en los registros de retención 8110h, 8111h, 8112h segundos, minutos, horas, respectivamente (consulte la sección «Configuración del reloj» en el «Apéndice A. Registros del controlador»).
3) Establezca un nuevo valor de tiempo configurando las bobinas 8111h.
4) Habilite la sobreescritura automática de los registros de retención 8110h…8112h configurando las bobinas 8110h.

11. Un programa de usuario- carga y lectura desde el controlador

11.1. Procedimiento de carga/descarga del programa de usuario

El controlador tiene dos áreas para descargar programas: general y especial.

El área de propósito general está destinada a cargar un programa de usuario con una longitud máxima de hasta 59752 líneas (el área está vacía de forma predeterminada). La longitud máxima del área especial es de 1926 líneas. Esta área contiene un programa para controlar la velocidad de un motor paso a pasoutilizando un potenciómetro, botones y un codificador. Si es necesario, esta área se puede sobrescribir.

A continuación, se muestra un ejemplo de un programa de usuario. La lista de los registros involucrados en estas operaciones se da en «Apéndice A. Registros del controladorApéndice A» en la sección «Trabajando con ROM».

Programa de usuario en forma LD:

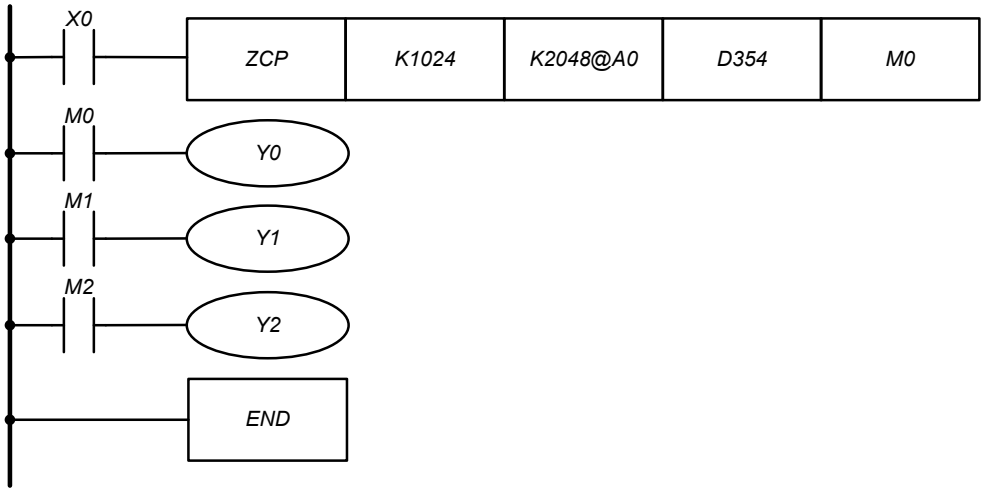


Fig. 32. programa de usuario

El programa de usuario convertido en IL:

```
LD      X0                                  ;condiciónde entrada para la operación de comparación de zona
ZCP     K1024   K2048@A0    D354    M0      ;comparación de zona, determinando la posición del; potenciómetro SPEED (2)
LD      M0                                  ;si el valor en el registro D354 es menor que 1024, entonces Y0 se activa, de lo contrario, se desactiva
OUT     Y0
LD      M1                                  ;si el valor en el registro D354 es mayor o igual que 1024 y menor o igual que la suma de 2048 y el valor de A0, entonces Y1 se activa, de lo contrario, se desactiva.
OUT     Y1
LD      M2                                  si el valor en el registro D354 es mayor que la suma de 2048 y el valor de A0, entonces Y2 se activa,
OUT     Y2
END                                         ;fin del programa
```

Todos los códigos de instrucción admitidos por el controlador se presentan en el «Apéndice B. Lista de instrucciones». Utilícelo al ensamblar un programa de usuario o utilice el software de PC suministrado para programar el controlador.

1) Asegúrese de que el controlador esté en modo stop. Es imposible cambiar el programa de usuario en el modo run. Para verificar el estado RUN/STOP del controlador, lea el valor de las entradas discretas F001h. Se restablece en el modo stop, se establece en el modo run.
2) Para leer el programa del controlador: compruebe si no está protegido contra lectura antes de leer un programa del controlador. Hay dos objetos de protección contra lectura en el controlador: bobinas F001h (para la protección de un programa de usuario) y F002h (para la protección de un programa de servicio). Es imposible leer el programa si la protección está configurada para el programa. Si la protección no está configurada, vaya al paso 5 para leer el programa del controlador.
3) Antes de escribir un nuevo programa en el controlador, es necesario borrar el anterior. Establezca las bobinas F003h para borrar el programa de usuario o las bobinasF004h para borrar un programa de servicio. En este ejemplo, se está escribiendo el programa principal, por lo que es necesario establecer las bobinas F003h.
4) Después de configurar las bobinas F003h (o F004h), espere hasta que las entradas discretas F000h se restablezcan, lo que indicará la finalización del procedimiento de borrado y la preparación de la ROM para un trabajo posterior.

5) Después de borrar el programa anterior, es necesario establecer el tipo de operación: lectura o escritura. Para escribir un nuevo programa, configure las bobinas F005h; para leer el programa desde el controlador, reinicie las bobinas F005h.
6) Seleccione el tipo de programa. Para el programa de usuario, reinicie las bobinas F006h; para el programa de servicio, configure las bobinas F006h.
7) Establezca el número de línea para escribir/leer el programa utilizando los registros de retenciónF100h. La numeración comienza desde 0. Para la operación de lectura, vaya al paso 10. Para escribir un nuevo programa, establezca su valor en 0.
8) Rellene el sector de descarga F300h ...F314 según el siguiente ejemplo:

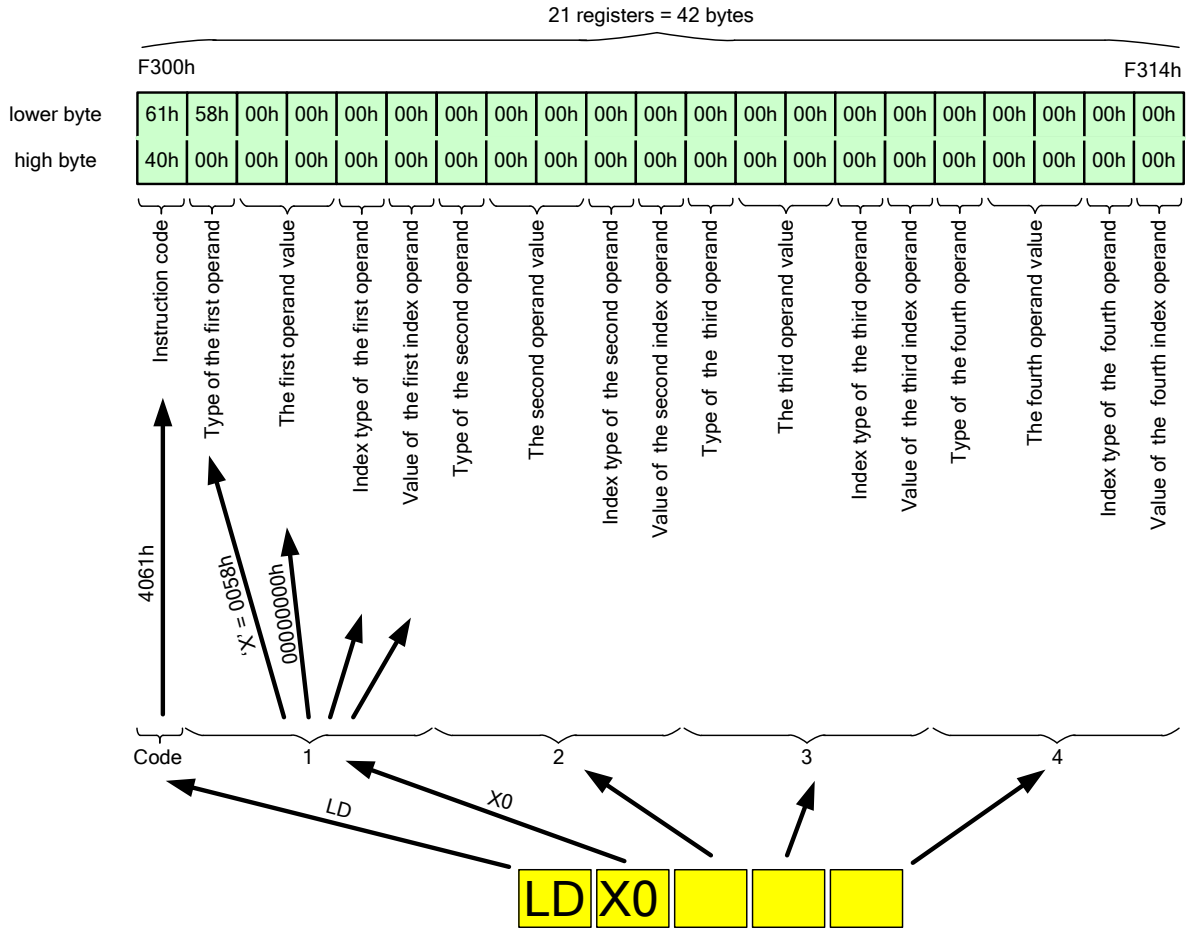


Fig. 33. Proyectando una instrucción con operandos en el espacio de direcciones del protocolo modbus

9) Configurar las bobinas F000h inicia la operación parametrizada en las bobinas F005h y F006h.En este ejemplo: escribir la primera línea en la ROM del controlador.Por lo tanto, repitiendo los pasos 7 a 9, moviéndose hacia abajo en el programa hasta el final, incrementando los registros de retenciónF100h, el programa se graba en el controlador.

Como ejemplo, a continuación se muestra la formación del sector de descargadesde la segunda línea del programa.

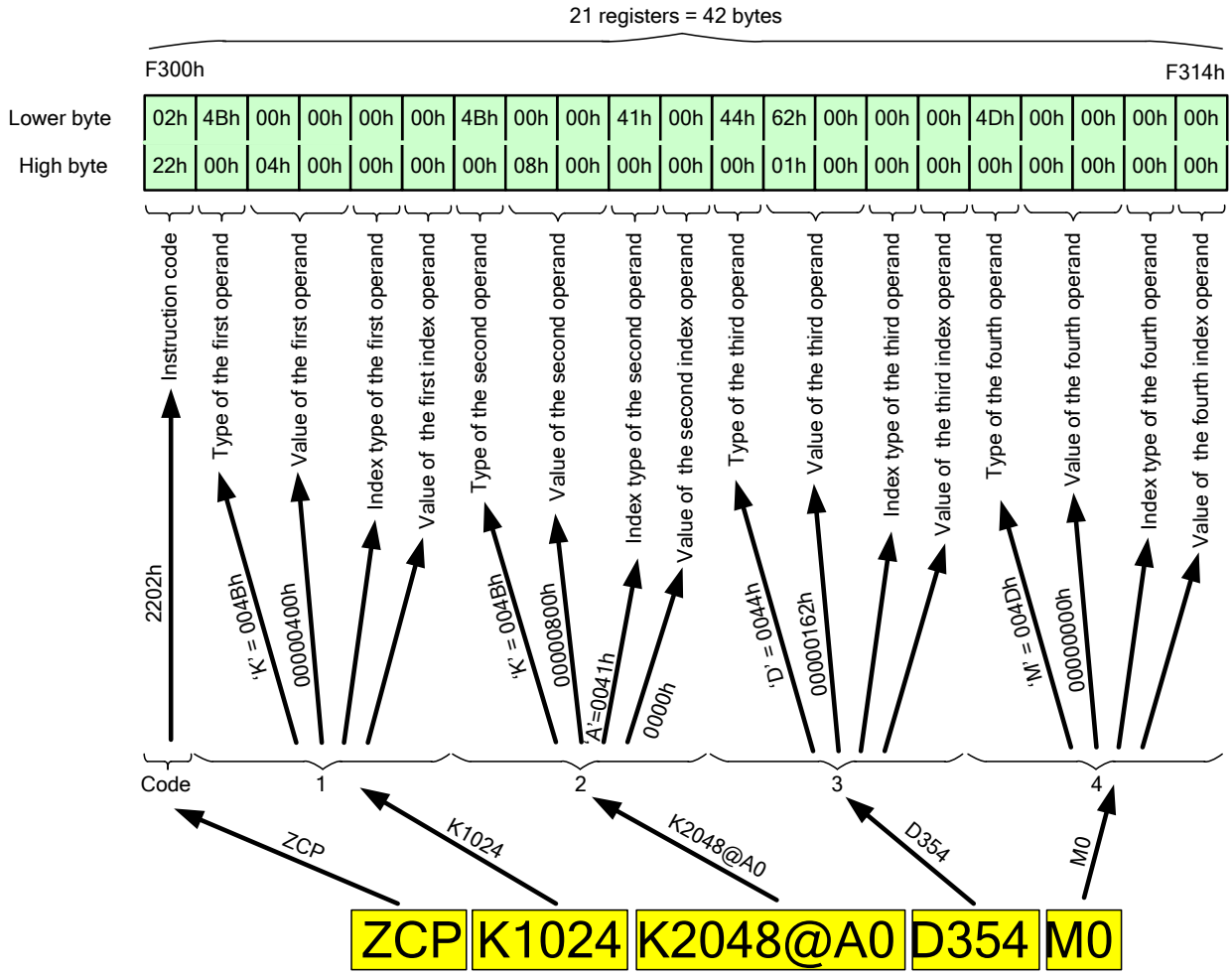


Fig. 34. Proyectando una instrucción con operandos en el espacio de direcciones del protocolo modbus

10) Para leer el programa, se necesita la operación opuesta. La diferencia es que el sector de cargatiene la dirección registros de entrada F200h ...F214h, y la operación parametrizada se realiza primero configurando las bobinas F000h, y luego leyendo el sector, y luego incrementando el número de línea hasta que llegue la instrucción END.

Los códigos de tipo de operandose muestran en la tabla a continuación:

operando	Código
K	4Bh
H	48h
F	46h
X	58h
Y	59h
M	4Dh
T	54h
C	43h
A	41h
B	42h
D	44h
P	50h
I	49h

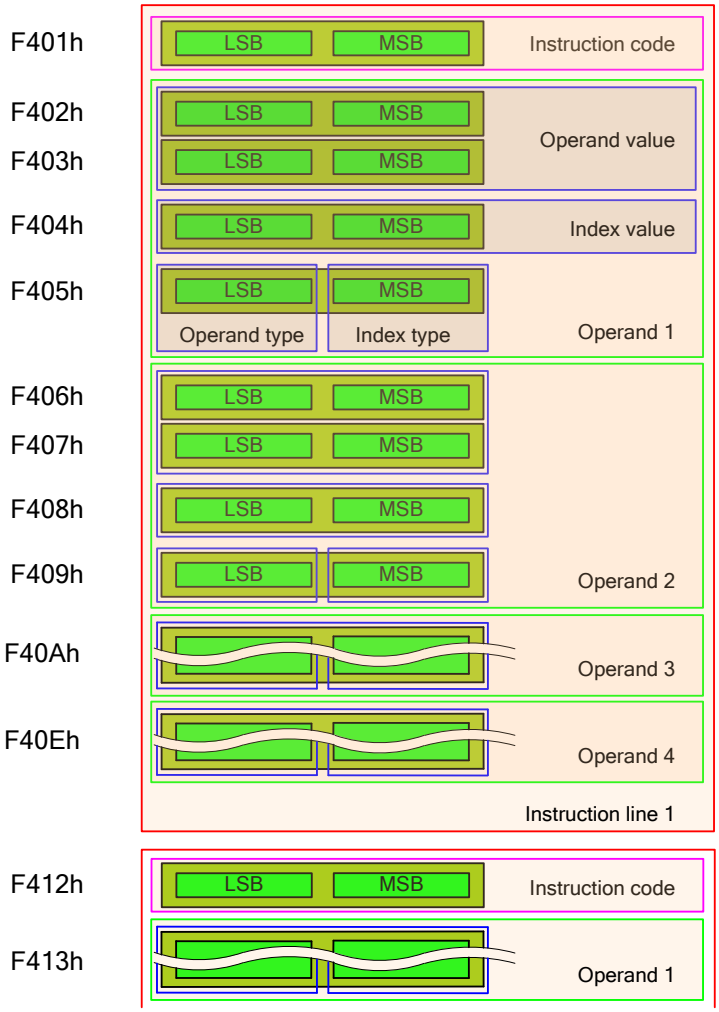
11.2. Carga/descarga en bloque de un programa de usuario

Un programa de usuariose puede leer y escribir más rápido si se utiliza la carga/descarga en bloque. El procedimiento de la carga/descarga en bloque de un programa de usuarioes similar a la sección «Procedimiento de carga/descarga del programa de usuario», pero con las siguientes diferencias:

Para escribir un programa de usuario	
Inicio del procedimiento	Utilice las bobinas F007h en lugar de F000h.
Sector de descarga	Las direcciones del sector de descarga son los Registros de retención F401h…F4FFh, el sector puede contener de 1 a 15 líneas de comando (instrucciones). El número de líneas a escribir se indica en los Registros de retención F400h.

Para leer un programa de usuario	
Inicio del procedimiento	Utilice las bobinas F007h en lugar de F000h.
Sector de carga	Las direcciones del sector de carga son los Registros de retención F401h…F4FFh, el sector puede contener de 1 a 15 líneas de comando (instrucciones). El número de líneas a leer se indica en los Registros de retención F400h. Cuando se completa el procedimiento, F400h mostrará el número real de líneas leídas. .
línea de comando	Se asignan 34 bytes para cada línea de comando en el sector de descarga/carga.

La estructura del sector de carga/descarga en bloquees la siguiente:



. .
. .
. .

11.3. Códigos de errorque se producen al trabajar con la ROM

Dirección	Tipo	Tamaño	Descripción
E002h	Registros de entrada	16 bits	Código de error al trabajar con la ROM
E002h	Bobinas		Indicador de un error en el proceso de trabajo con la ROM.

Código de error	Descripción
0001h	No se ha establecido la protección contra lectura para el programa principal.
0002h	No se ha establecido la protección contra lectura para el programa de servicio.
0003h	Error al borrar el sector del programa principal.
0004h	Error al borrar el sector del programa de servicio.
0005h	Error al escribir la instrucción en el programa principal.
0006h	Error al escribir la instrucción en el programa de servicio.

12. Modo de control de velocidad

Este modo está diseñado para controlar la velocidad de rotación de un motor paso a pasoutilizando el potenciómetro incorporado “SPEED” (2), los botones o un codificador.

Para entrar en el modo de control de velocidad, coloque el controlador en el estado STOP, luego use el botón de selección de modo para establecer el modo SPD. Ensamble y conecte al controlador el circuito que se muestra en la Fig. 35. – Conexión de elementos de control.

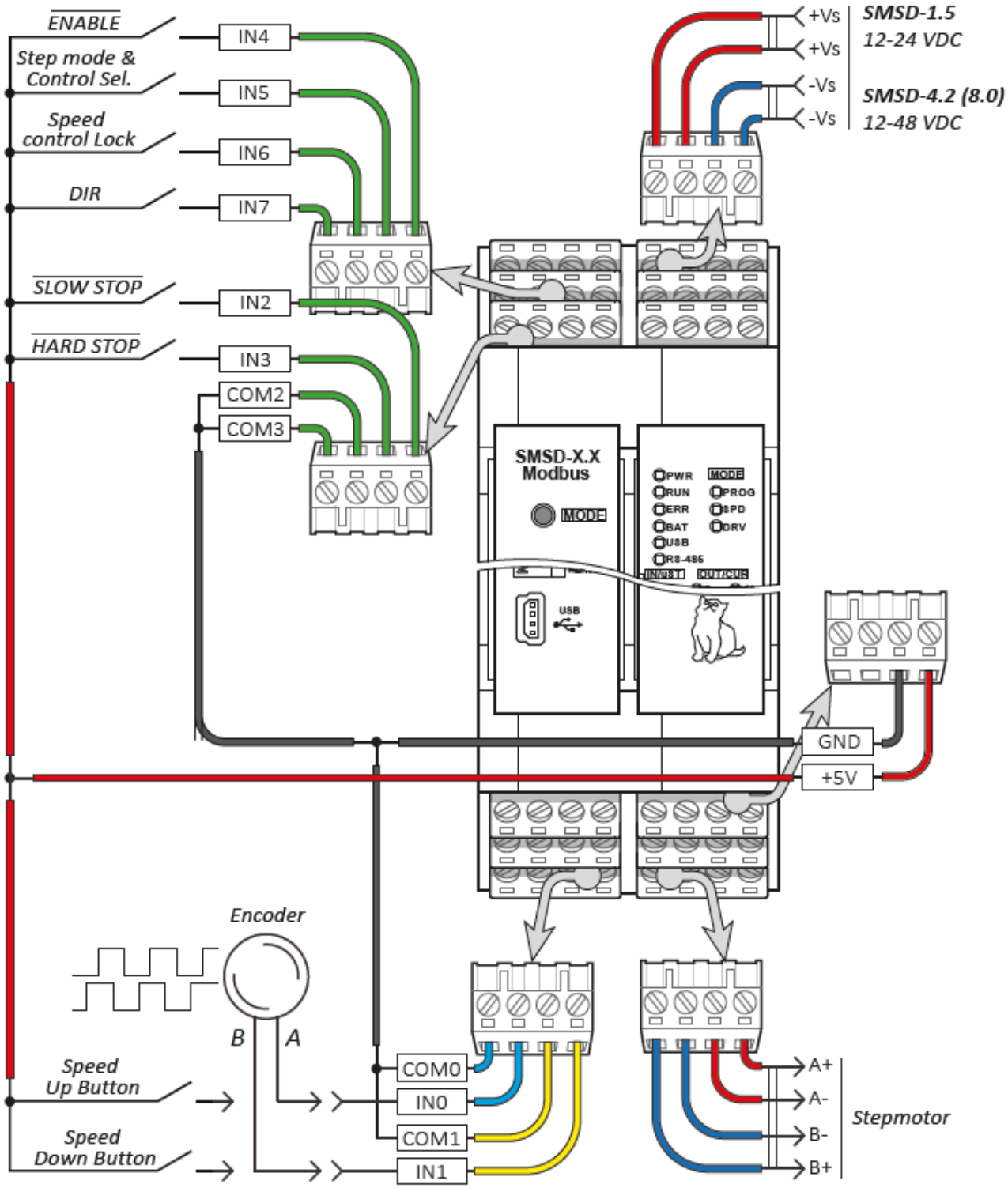


Fig. 35. Conexión de elementos de control

Atención

Al poner el controlador en el modo **RUN** mientras los interruptores de SLOW STOP y HARD STOP están cerrados y el interruptor de ENABLE está abierto, **causará la rotación del motor**. Para evitar una rotación incontrolada, gire el potenciómetro “SPEED” a la posición mínima o cambie la posición de cualquiera de los interruptores anteriores a la opuesta indicada en el diagrama.

En el estado **RUN**, seleccione los micropasos requeridos presionando el botón correspondiente. El método de control de velocidad se selecciona mediante la entrada IN5, más detalles en la tabla a continuación.

Indicación LED OUT0…7	micropasos
OUT0	1/1
OUT1	1/2
OUT2	1/4
OUT3	1/8
OUT4	1/16
OUT5	1/32
OUT6	1/128
OUT7	1/256

Indicación LED OUT10…11	Fuente de control de velocidad
Ambos están desactivados	La velocidad se ajusta con el potenciómetro «SPEED».
OUT10	La velocidad se ajusta con los botones «Aumentar» y «Disminuir» (IN0 e IN1). El incremento de cambio de velocidad se establece con el potenciómetro «SPEED».
OUT11	La velocidad se ajusta mediante un codificador, conectado a las entradas IN0 e IN1. El incremento de cambio de velocidad para un evento del codificador se establece mediante el potenciómetro «SPEED».

Cuando el bloqueo de cambio de velocidad está activado, el controlador deja de responder a los controles de velocidad. Esta opción está diseñada para evitar impactos mecánicos accidentales en el potenciómetro, el codificador y los botones.

DIR– cambia la dirección de rotación del motor.

ENABLE– controla la energización de las fases del motor paso a paso.

HARD STOP– la apertura del circuito detiene inmediatamente y pone el motor en el modo de retención. La corriente de retención es el 50% de la corriente de trabajo. El valor de la corriente de trabajo se establece mediante el potenciómetro (**1**) desde el valor mínimo hasta el valor máximo para el modelo.

SLOW STOP– la apertura del circuito hace que el motor se detenga de acuerdo con la desaceleración establecida por el potenciómetro (0) (el valor de la aceleración también se establece mediante el potenciómetro(0)).

El código del programa de servicio se proporciona en el Apéndice D. Código del programa de servicio “Control de Velocidad del Motor Paso a Paso”. El código se puede modificar para satisfacer requisitos específicos.

13. Modo de control de posición por pulsos Step/Dir

El controlador proporciona el modo de control de posición por pulsos mediante señales de paso por pulsos STEP (las entradas STEP +, STEP-) y la señal de dirección DIR (las entradas DIR +, DIR. La entrada ENABLE + controla la energización de las fases del motor. INVERT_ENABLE + invierte la señal ENABLE. La salida FAULT indica estados de alarma: sobre corriente y sobrecalentamiento, o pasos perdidos debido a estas dos razones (Fig. 2).

Para cambiar el controlador al modo de control de posición por pulsos, primero cámbielo al estado **STOP** y luego use el botón de selección de modo para cambiar al modo **DRV**. En este estado, las fases del motor están desenergizadas. Seleccione los micropasos necesarios, la corriente de trabajo y la corriente de retención. (La transición al modo se lleva a cabo después de un segundo después de la detección de la última señal Step en el flanco de subida del pulso).

Los micropasos se configuran mediante el potenciómetro SPEED, los valores de la corriente de trabajo – mediante el potenciómetro (0), la corriente de retención – mediante el potenciómetro (1). Los parámetros establecidos se muestran en el panel LED, más detalles en las tablas a continuación:

OUT0	OUT1	OUT2	OUT3	Corriente de funcionamiento		
OUT4	OUT5	OUT6	OUT7	corriente de retención		
				SMSD-1.5Modbus ver.3	SMSD-4.2Modbus	SMSD-8.0Modbus
●				150 мА	1000 мА	2800 мА
	●			245 мА	1285 мА	3310 мА
●	●			340 мА	1570 мА	3830 мА
		●		440 мА	1860 мА	4340 мА
●		●		535 мА	2140 мА	4860 мА
	●	●		630 мА	2430 мА	5370 мА
●	●	●		730 мА	2710 мА	5885 мА
			●	825 мА	3000 мА	6400 мА
●			●	920 мА	3285 мА	6910 мА
	●		●	1020 мА	3570 мА	7430 мА
●	●		●	1115 мА	3860 мА	7940 мА
		●	●	1210 мА	4140 мА	8460 мА
●		●	●	1310 мА	4430 мА	8970 мА
	●	●	●	1400 мА	4710 мА	9485 мА
●	●	●	●	1500 мА	5000 мА	10000 мА

Indicación LED IN0…7	micropasos
IN0	1/1
IN1	1/2
IN2	1/4
IN3	1/8
IN4	1/16
IN5	1/32
IN6	1/128
IN7	1/256

Cuando el controlador está en el estado **RUN**, los parámetros anteriores se fijan y se guardan después de apagar la alimentación. Utilice las entradas y salidas del controlador de acuerdo con la tabla de asignación de pines (Fig. 2).

14. Modo de control del programa de usuario

El controlador proporciona el modo de control de acuerdo con un programa de usuario y se controla mediante comandos Modbus. El controlador indica este modo de control mediante el indicador LED **PROG**. Un programa de usuario se puede enviar a la memoria del controlador cuando el controlador está en el estado **STOP**. Después de cambiar al estado **RUN**, el controlador comienza a ejecutar el programa de usuario. También es posible controlar el estado del controlador, el programa de usuario, las salidas físicas, el controlador de motor paso a paso y monitorear el estado de las entradas físicas a través de la interfaz rs-485 utilizando el protocolo modbus.

Ejemplos de programas de usuario que demuestran la funcionalidad básica del controlador se describen en el Apéndice C. Ejemplos de programas de usuario.

Apéndice A. Registros del controlador

Dirección	Tipo	Tamaño	Descripción
Parámetros de comunicación de la interfaz rs-485			
0x8100	Bobinas	-	Selección del protocolo de comunicación Restablecer — Modbus ASCII. Establecer — Modbus RTU. Los cambios se aplican después del reinicio del controlador.
0x8100	Registros de retención	32 bits	Velocidad en baudios. Valores permitidos: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, 256000. Los cambios se aplican después del reinicio del controlador. .
0x8102	Registros de retención	16 bits	Paridad: 0 – NINGUNA 1 – IMPAR 2 – PAR
			*Están disponibles los siguientes parámetros de transferencia de datos. :* *Modbus RTU:* ■ *8 bits / PAR / 1 bit de parada* ■ *8 bits / IMPAR / 1 bit de parada* ■ *8 bits / NINGUNA / 2 bits de parada* *Modbus ASCII:* ■ *7 bits / PAR / 1 bit de parada* ■ *7 bits / IMPAR / 1 bit de parada* *Los parámetros del bit de parada se establecen automáticamente.*
0x8103	Registros de retención	16 bits	ID del controlador (dirección del dispositivo). Valores permitidos: 1.. .247.
Ajuste del reloj			
0x8110	Registros de retención	16 bits	Segundos. Valores permitidos: 0.. .59.
0x8111	Registros de retención	16 bits	Minutos. Valores válidos: 0.. .59.

Dirección	Tipo	Tamaño	Descripción
0x8112	Registros de retención	16 bits	Horas. Valores válidos: 0.. .23.
0x8110	Bobinas	-	Actualización automática del registro. Establecer — en los registros 0x8110 - 0x8112, el valor actual de la hora. Restablecer — la actualización automática de datos está deshabilitada, se permite el registro de valores de usuario.
0x8111	Bobinas	-	El ajuste del objeto establece la hora desde los registros 0x8110 - 0x8112. Se permite activar de nuevo la actualización automática después de ajustar el objeto.
Ajuste de la fecha			
0x8113	Registros Registros	16 bits	Día. Valores válidos: 1…31
0x8114	Registros Registros	16 bits	Mes Valores válidos: 1…12
0x8115	Registros Registros	16 bits	Año Valores válidos: 21…99
		La configuración de la fecha es similar a la configuración de la hora con un restablecimiento preliminar de las bobinas 8110h (consulte la sección 10).	
0x8112	Bobinas	-	Establecer el objeto establece la fecha desde los registros 0x8110 - 0x8112. Se permite activar de nuevo la actualización automática después de ajustar el objeto.
Adicional			
0x8101	Bobinas	-	Establecer el objeto provoca un reinicio del controlador.

Dirección	Tipo	Tamaño	Descripción
0xF001	Registros de retención	16 bits	Modo de operación del controlador: programa de usuario (PROG), programa de servicio (SPD), modo de controlador (DRV). El cambio del modo de operación del controlador solo es posible en el estado STOP. 0 - Programa de usuario. 1 - Programa de servicio. 2 - Modo de controlador.
Trabajar con la ROM.			
0xF001	Entradas discretas	-	Estado del interruptor de palanca RUN/STOP. Las operaciones de la ROM no son posibles cuando el controlador está en el estado RUN. Restablecer — estado STOP. Establecer — estado RUN.
0xF000	Entradas discretas	-	Indicación del estado de la ROM. Restablecer — la ROM está lista para funcionar. Establecer — la ROM está ocupada.
0xF000	Bobinas	-	Objeto de control para la lectura/escritura línea por línea de un programa de usuario. El establecimiento del objeto inicia la operación parametrizada en los objetos 0xF005 y 0xF006.
0xF001	Bobinas	-	Protección contra lectura del programa principal (de usuario). Establecer — no protegido. Restablecer — la protección contra lectura está establecida. El intento de establecer el objeto provoca el borrado del programa principal.
0xF002	Bobinas	-	Protección contra lectura del programa de servicio. Establecer — no protegido. Restablecer — la protección contra lectura está establecida. El intento de establecer el objeto provoca el borrado del programa de servicio.
0xF003	Bobinas	-	Borrado del programa principal. Establecer el objeto inicia el procedimiento de borrado del programa principal. El restablecimiento del objeto se ignora.

Dirección	Tipo	Tamaño	Descripción
0xF004	Bobinas	-	Borrado del programa de servicio. Establecer el objeto inicia el procedimiento de borrado del programa de servicio. El restablecimiento del objeto se ignora.
0xF005	Bobinas	-	Selección del tipo de operación. Restablecimiento — lectura. Establecimiento — escritura.
0xF006	Bobinas	-	Selección de programa (principal/de servicio). Restablecimiento — principal. Establecimiento — de servicio.
0xF007	Bobinas	-	Objeto de control para la lectura/escritura en bloque de un programa de usuario (consulte la sección 11.2). El establecimiento del objeto inicia la operación parametrizada en los objetos 0xF005 y 0xF006.
0xF100	Registros de retención	16 bits	El número de línea en el programa que se va a leer o sobrescribir (0 ... 59753 - para el programa principal, 0 ... 1927 - para el programa de servicio).
Sector de lectura de rom línea por línea.			
0xF200	Registros de entrada	16 bits	Código de instrucción
0xF201	Registros de entrada	16 bits	Tipo del primer operando
0xF202	Registros de entrada	32 bits	Valor del primer operando
0xF204	Registros de entrada	16 bits	Tipo de índice del primer operando
0xF205	Registros de entrada	16 bits	Valor del primer operando de índice
0xF206	Registros de entrada	16 bits	Tipo del segundo operando
0xF207	Registros de entrada	32 bits	Valor del segundo operando
0xF209	Registros de entrada	16 bits	Tipo de índice del segundo operando
0xF20A	Registros de entrada	16 bits	Valor del segundo operando de índice
0xF20B	Registros de entrada	16 bits	Tipo del tercer operando
0xF20C	Registros de entrada	32 bits	Valor del tercer operando
0xF20E	Registros de entrada	16 bits	Tipo de índice del tercer operando
0xF20F	Registros de entrada	16 bits	Valor del tercer operando de índice

Dirección	Tipo	Tamaño	Descripción
0xF210	Registros de entrada	16 bits	Tipo del cuarto operando
0xF211	Registros de entrada	32 bits	Valor del cuarto operando
0xF213	Registros de entrada	16 bits	Tipo de índice del cuarto operando
0xF214	Registros de entrada	16 bits	Valor del cuarto operando de índice
Sector de escritura de rom línea por línea			
0xF300	Registros de retención	16 bits	Código de instrucción
0xF301	Registros de retención	16 bits	Tipo del primer operando
0xF302	Registros de retención	32 bits	Valor del primer operando
0xF304	Registros de retención	16 bits	Tipo de índice del primer operando
0xF305	Registros de retención	16 bits	Valor del primer operando de índice
0xF306	Registros de retención	16 bits	Tipo del segundo operando
0xF307	Registros de retención	32 bits	Valor del segundo operando
0xF309	Registros de retención	16 bits	Tipo de índice del segundo operando
0xF30A	Registros de retención	16 bits	Valor del segundo operando de índice
0xF30B	Registros de retención	16 bits	Tipo del tercer operando
0xF30C	Registros de retención	32 bits	Valor del tercer operando
0xF30E	Registros de retención	16 bits	Tipo de índice del tercer operando
0xF30F	Registros de retención	16 bits	Valor del tercer operando de índice
0xF310	Registros de retención	16 bits	Tipo del cuarto operando
0xF311	Registros de retención	32 bits	Valor del cuarto operando
0xF313	Registros de retención	16 bits	Tipo de índice del cuarto operando
0xF314	Registros de retención	16 bits	Valor del cuarto operando de índice
Sector de lectura/escritura para la carga/descarga en bloque de un programa de usuario (15 líneas de comando, 17 registros por línea)			
Línea 1 del sector de lectura/escritura			
0xF401	Registros de retención	16 bits	Código de instrucción de la línea 1 del sector de lectura/escritura

Dirección	Tipo	Tamaño	Descripción
Operando 1 de la línea 1 del sector de lectura/escritura			
0xF402	Registros de retención	32 bits	Valor del operando 1 de la línea 1 del sector de lectura/escritura
0xF404	Registros de retención	16 bits	Valor del índice del operando 1 de la línea 1 del sector de lectura/escritura
0xF405	Registros de retención	16 bits	8 bits LSB – Tipo de operando 1 – línea 1 del sector de lectura/escritura 8 bits MSB – Tipo de índice del operando 1 del sector de lectura/escritura
. . .	. . .	. . .	. . .
Operando 4 de la línea 1 del sector de lectura/escritura			
0xF40E	Registros de retención	32 bits	Valor del operando 4 de la línea 1 del sector de lectura/escritura
0xF404	Registros de retención	16 bits	Valor del índice del operando 4 de la línea 1 del sector de lectura/escritura
0xF405	Registros de retención	16 bits	8 bits LSB – Tipo de operando 4 – línea 1 del sector de lectura/escritura 8 bits MSB – Tipo de índice del operando 4 del sector de lectura/escritura
. . .	. . .	. . .	. . .
Línea 15 del sector de lectura/escritura			
0xF4EF	Registros de retención	16 bits	Código de instrucción de la línea 15 del sector de lectura/escritura
Operando 1 de la línea 15 del sector de lectura/escritura			
0xF4F0	Registros de retención	32 bits	Valor del operando 1 de la línea 15 del sector de lectura/escritura
0xF4F2	Registros de retención	16 bits	Valor del índice del operando 1 de la línea 15 del sector de lectura/escritura

Dirección	Tipo	Tamaño	Descripción
0xF4F3	Registros de retención	16 bits	8 bits LSB – Tipo de operando 1 – línea 15 del sector de lectura/escritura 8 bits MSB – Tipo de índice del operando 1 del sector de lectura/escritura
. . .	. . .	. . .	. . .
Operando 4 de la línea 15 del sector de lectura/escritura			
0xF4FC	Registros de retención	32 bits	Valor del operando 4 de la línea 15 del sector de lectura/escritura
0xF4FE	Registros de retención	16 bits	Valor del índice del operando 4 de la línea 15 del sector de lectura/escritura
0xF4FF	Registros de retención	16 bits	8 bits LSB – Tipo de operando 4 – línea 15 del sector de lectura/escritura 8 bits MSB – Tipo de índice del operando 4 del sector de lectura/escritura.
Errores			
0xE000	Bobinas	-	Establecer el objeto restablece todos los tipos de errores que son válidos para el estado actual del controlador.
0xE000	Entradas discretas	-	Error general. Se establece siempre cuando aparece al menos uno de los tipos de errores de 0xE001 a 0xE004.
0xE001	Entradas discretas	-	El estado establecido del objeto indica una batería CR2032 descargada dentro del controlador. Se requiere su reemplazo.
0xE002	Entradas discretas	-	El estado establecido del objeto indica un error durante la operación de rom. El código de error se especifica en los registros de entrada 0xE002.
0xE003	Entradas discretas	-	El estado establecido del objeto indica un error durante el proceso de intercambio utilizando el protocolo modbus. El código de error se especifica en los registros de entrada 0xE003.

Dirección	Tipo	Tamaño	Descripción
0xE004	Entradas discretas	-	El estado establecido del objeto indica un error durante la ejecución del programa de usuario. El código de error se especifica en los registros de entrada 0xE004. La línea del programa que causó el error se indica en 0xE084.
0xE002	Registros de entrada	16 bits	código de error de operación de rom.
0xE003	Registros de entrada	16 bits	código de error del protocolo modbus.
0xE004	Registros de entrada	16 bits	código de error del programa de usuario.
0xE084	Registros de entrada	16 bits	El número de una línea que causó error en el programa de usuario (la numeración comienza desde 0, consulte la descripción de las bobinas 0xF100 más arriba).
Acceso a los operandos del programa			
Salidas discretas			
0x1000	Entradas discretas	-	Salida discreta Y0
0x1001	Entradas discretas	-	Salida discreta Y1
. . .	. . .	. . .	. . .
0x107F	Entradas discretas	-	Salida discreta Y177
Estado de las entradas físicas discretas			
0x2000	Entradas discretas	-	Entrada discreta X0
. . .	. . .	. . .	. . .
0x2007	Entradas discretas	-	Entrada discreta X7

Dirección	Tipo	Tamaño	Descripción
Entradas discretas			
0x2008	Bobinas	-	Entrada discreta X10
0x2009	Bobinas	-	Entrada discreta X11
. . .	. . .	. . .	. . .
0x207F	Bobinas	-	Entrada discreta X177
registros de datos de propósito general D192.. .D255			
0x3000	Registros de entrada	16 bits	Registro D192
0x3001	Registros de entrada	16 bits	Registro D193
. . .	. . .	. . .	. . .
0x303F	Registros de entrada	16 bits	Registro D255
registros de datos de propósito general D256.. .D319			
0x4000	Registros de retención	16 bits	Registro D256
0x4001	Registros de retención	16 bits	Registro D257
. . .	. . .	. . .	. . .
0x403F	Registros de retención	16 bits	Registro D319
registros de datos no volátiles D320.. .D327			
0x3100	Registros de entrada	16 bits	Registro D320
. . .	. . .	. . .	. . .
0x3107	Registros de entrada	16 bits	Registro D327

Dirección	Tipo	Tamaño	Descripción
	registros de datos no volátiles D328.. .335		
0x4100	Registros de retención	16 bits	Registro D328
. . .	. . .	. . .	. . .
0x4107	Registros de retención	16 bits	Registro D335
	Contadores de hardware		
0x4200	Registros de retención	32 bits	contador C64
0x4202	Registros de retención	32 bits	Contador C65
	Convertidores analógico-digitales		
0x3200	Registros de entrada	16 bits	Registro D352, datos del potenciómetro «0»
0x3201	Registros de entrada	16 bits	Registro D353, datos del potenciómetro «1»
0x3202	Registros de entrada	16 bits	Registro D354, datos del potenciómetro «SPEED»
	Versiones de hardware y software		
0x8001	Registros de entrada	16 bits	Versión principal de hardware
0x8002	Registros de entrada	16 bits	Versión menor de hardware
0x8003	Registros de entrada	16 bits	Versión principal de software
0x8004	Registros de entrada	16 bits	Versión menor de software
0x8005	Registros de entrada	16 bits	Versión principal del cargador de arranque
0x8006	Registros de entrada	16 bits	Versión menor del cargador de arranque
	Control de motor paso a paso		
0x5000	Registros de retención	32 bits	Registro D357 – SPEED.
0x5002	Registros de retención	32 bits	Registro D359 – MIN_SPEED.
0x5004	Registros de retención	16 bits	Registro D361 – ACC.
0x5005	Registros de retención	16 bits	Registro D362 – DEC.
0x5006	Registros de retención	32 bits	Registro D363 – ABS.
0x5008	Registros de entrada	16 bits	Registro D365 – U_POS.

Dirección	Tipo	Tamaño	Descripción
0x5009	Registros de retención	16 bits	Registro D366 – U_STEP.
0x500A	Registros de retención	16 bits	Registro D374 – DIR.
0x500A	Bobinas	-	Registro D374 – DIR.
0x500B	Registros de retención	32 bits	Registro D377 – FS_SPD_THR.
0x500D	Registros de retención	16 bits	Registro D379 – FS_SW_EN.
0x500D	Bobinas	-	Registro D379 – FS_SW_EN.
0x500E	Registros de retención	32 bits	Registro D372 – TARGET_POS.
0x5010	Registros de retención	16 bits	Registro D376 – CMD.
0x5011	Registros de retención	16 bits	Registro D375 – SW_INPUT.
0x5012	Registros de retención	16 bits	Registro D367 – ACC_CUR.
0x5013	Registros de retención	16 bits	Registro D368 – DEC_CUR.
0x5014	Registros de retención	16 bits	Registro D369 – RUN_CUR.
0x5015	Registros de retención	16 bits	Registro D370 – HOLD_CUR.
0x5016	Registros de retención	16 bits	Registro D382 – CMIN_SPD_EN.
0x5017	Registros de retención	16 bits	Registro D380 – ERROR_SET_HIZ.
0x5017	Bobinas	-	TERMAL_ERROR_SET_HIZ
0x5018	Bobinas	-	SOFTWARE_ERROR_SET_HIZ
0x5019	Bobinas	-	CMD_ERROR_SET_HIZ
0x501A	Bobinas	-	DATA_ERROR_SET_HIZ
0x5027	Registros de retención	16 bits	Registro D381 – ERROR_CODE.
0x5027	Bobinas	-	TERMAL_ERROR_OVER_CURRENT
0x5028	Bobinas	-	SOFT_ERROR
0x5029	Bobinas	-	CMD_ERROR
0x502A	Bobinas	-	DATA_ERROR
0x502F	Bobinas	-	OVLO/UVLO_INTERNAL_PROTECTION_ERROR (SMSD-4.2Modbus y SMSD-8.0Modbus)
0x5030	Bobinas	-	VS_OUT_OF_RANGE_ERROR
0x5037	Registros de entrada	16 bits	Registro D371 – MOTOR_STATUS.
0x5037	Entradas discretas	-	HIZ

Dirección	Tipo	Tamaño	Descripción
0x5038	Entradas discretas	-	STOP
0x5039	Entradas discretas	-	ACCELERATING
0x503A	Entradas discretas	-	DECELERATING
0x503B	Entradas discretas	-	STEADY
0x503C	Entradas discretas	-	BUSY_MOVE
0x503D	Entradas discretas	-	BUSY_RUN
0x5048	Registros Registros	32 bits	Registro D385 – EMERGENCY_DEC.
0x5100	Bobinas	-	Instrucción SPIN – APPLY_CMD.
0x5101	Bobinas	-	Instrucción TORQUE – APPLY_CURRENT.
0x5102	Bobinas	-	Instrucción HSTOP – HARD_STOP.
0x5103	Bobinas	-	Instrucción HHIZ – HARD_HIZ.
0x5104	Bobinas	-	Instrucción SSTOP – SLOW_STOP.
0x5105	Bobinas	-	Instrucción SHIZ – SLOW_HIZ.

Apéndice B. Lista de instrucciones

Instrucción		Descripción
API	Código	
Instrucciones básicas		
LD	0x4061	Contacto normalmente abierto
LDI	0x4001	Contacto normalmente cerrado
AND	0x4065	Conexión en serie - contacto normalmente abierto (and lógico)
ANI	0x4005	Conexión en serie - contacto normalmente cerrado (nand lógico)
OR	0x4066	Conexión en paralelo – contacto normalmente abierto (or lógico)
ORI	0x4046	Conexión en paralelo – contacto normalmente cerrado (nor lógico)
LDP	0x4821	Inicio de expresión lógica con sondeo por flanco ascendente (impulso)
LDF	0x4841	Inicio de una expresión lógica con sondeo por flanco descendente (impulso)
ANDP	0x4825	«AND» con un sondeo por flanco ascendente (impulso)
ANDF	0x4845	«AND» con un sondeo por flanco descendente (impulso)
ORP	0x4806	«OR» con sondeo por flanco ascendente (impulso)
ORF	0x4826	«OR» con sondeo por flanco descendente (impulso)
TMR	0x2014	Temporizador (16 bits)
CNT	0x2015	Contador (16 bits)
DCNT	0x3015	Contador (32 bits)
INV	0x4016	Inversión: sustituir el resultado de las conexiones lógicas por el opuesto
ANB	0x4007	Bloque «AND»: conexión en serie de bloques
ORB	0x4008	Bloque «OR»: conexión en paralelo de bloques
MPS	0x4009	Desplazamiento hacia abajo en la pila
MRD	0x402A	Leer valor de la pila
MPP	0x400A	Salir de la pila
SET	0x2024	Activación de la salida con enclavamiento (estableciendo el “1” lógico)

RST	0x2004	Reinicio del estado del operando
OUT	0x2002	Bobina de salida: asignación a la salida del resultado de una expresión lógica
FEND	0x6003	Fin del programa principal
NOP	0x8011	Línea vacía en el programa
P	0x6051	Direccionamiento de un punto de salto en un programa o subprograma
I	0x6031	Direccionamiento de un punto de interrupción
END	0x6023	Fin de programa
Instrucciones para bucles, transiciones, subprograma		
CJ	0x200D	Salto condicional - ir a la línea de programa especificada
CJP	0x280D	Salto condicional - ir a la línea de programa especificada con sondeo por flanco ascendente (impulso)
CALL	0x200E	Llamada a subprograma
CALLP	0x280E	Llamada a subprograma con sondeo por flanco ascendente (impulso)
SRET	0x600F	Fin de subprograma
FOR	0x400B	Inicio de un bucle FOR-NEXT
NEXT	0x400C	Fin de un bucle FOR-NEXT
Interrupciones		
IRET	0x6010	Fin del manejador de interrupción
EI	0x8012	Habilitación de interrupciones globales
DI	0x8013	Deshabilitación de interrupciones globales
Transferencia de datos y comparación		
CMP	0x2201	Comparación de datos numéricos
CMPP	0x2A01	Comparación de datos numéricos con sondeo por flanco ascendente (impulso)
DCMP	0x3201	Comparación de datos numéricos, instrucción de 32 bits

DCMPP	0x3A01	Comparación de datos numéricos, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
ZCP	0x2202	Comparación de zona de datos numéricos
ZCPP	0x2A02	Comparación de zona de datos numéricos con sondeo por flanco ascendente (impulso)
DZCP	0x3202	Comparación de zona de datos numéricos, instrucción de 32 bits
DZCPP	0x3A02	Comparación de zona de datos numéricos, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
MOV	0x2018	Transferencia de datos
MOVP	0x2818	transferencia de datos con sondeo por flanco ascendente (impulso)
DMOV	0x3018	transferencia de datos, instrucción de 32 bits
DMOVP	0x3818	transferencia de datos, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
BMOV	0x2038	Transferencia de datos en bloque
BMOVP	0x2838	transferencia de datos en bloque con sondeo por flanco ascendente (impulso)
DBMOV	0x3038	transferencia de datos en bloque, instrucción de 32 bits
DBMOVP	0x3838	Transferencia de datos en bloque, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
FMOV	0x2058	Transferencia de datos a múltiples direcciones
FMOVP	0x2858	Transferencia de datos a múltiples direcciones con sondeo por flanco ascendente (impulso)
DFMOV	0x3058	Transferencia de datos a múltiples direcciones, instrucción de 32 bits
DFMOVP	0x3858	Transferencia de datos a múltiples direcciones, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
XCH	0x220A	Intercambio de datos
XCHP	0x2A0A	Intercambio de datos con sondeo por flanco ascendente (impulso)
DXCH	0x320A	Intercambio de datos, instrucción de 32 bits

DXCHP	0x3A0A	Intercambio de datos, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
Operaciones aritméticas (enteros)		
ADD	0x2208	Suma de datos numéricos
ADDP	0x2A08	Suma de datos numéricos con sondeo por flanco ascendente (impulso)
DADD	0x3208	Suma de datos numéricos, instrucción de 32 bits
DADDP	0x3A08	Suma de datos numéricos, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
SUB	0x2228	Resta de datos numéricos
SUBP	0x2A28	Resta de datos numéricos con sondeo por flanco ascendente (impulso)
DSUB	0x3228	Resta de datos numéricos, instrucción de 32 bits
DSUBP	0x3A28	Resta de datos numéricos, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
MUL	0x2248	Multiplicación de datos numéricos
MULP	0x2A48	Multiplicación de datos numéricos con sondeo por flanco ascendente (impulso)
DMUL	0x3248	Multiplicación de datos numéricos, instrucción de 32 bits
DMULP	0x3A48	multiplicación de datos numéricos, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
DIV	0x2268	División de datos numéricos
DIVP	0x2A68	división de datos numéricos con sondeo por flanco ascendente (impulso)
DDIV	0x3268	división de datos numéricos, instrucción de 32 bits
DDIVP	0x3A68	división de datos numéricos, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
MOD	0x22E8	Resto de la división
MODP	0x2AE8	resto de la división con sondeo por flanco ascendente (impulso)
DMOD	0x32E8	Resto de la división, instrucción de 32 bits

DMODP	0x3AE8	Resto de la división, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
INC	0x2037	Incrementar datos numéricos (aumentar en 1)
INCP	0x2837	Incrementar datos numéricos (aumentar en 1) con sondeo por flanco ascendente (impulso)
DINC	0x3037	Incrementar datos numéricos (aumentar en 1), instrucción de 32 bits
DINCP	0x3837	Incrementar datos numéricos (aumentar en 1), instrucción de 32 bits con sondeo por flanco ascendente (impulso)
DEC	0x2017	Decrementar datos numéricos (disminuir en 1)
DECP	0x2817	decrementar datos numéricos (disminuir en 1) con sondeo por flanco ascendente (impulso)
DDEC	0x3017	decrementar datos numéricos (disminuir en 1), instrucción de 32 bits
DDECP	0x3817	decrementar datos numéricos (disminuir en 1), instrucción de 32 bits con sondeo por flanco ascendente (impulso)
WAND	0x2288	Multiplicación lógica de datos numéricos (operación "AND")
WANDP	0x2A88	multiplicación lógica de datos numéricos (operación "and") con sondeo por flanco ascendente (impulso)
DAND	0x3288	multiplicación lógica de datos numéricos (operación "and"), instrucción de 32 bits
DANDP	0x3A88	multiplicación lógica de datos numéricos (operación "and"), instrucción de 32 bits con sondeo por flanco ascendente (impulso)
WOR	0x22A8	Suma lógica de datos numéricos (operación "OR")
WORP	0x2AA8	suma lógica de datos numéricos (operación "OR") con sondeo por flanco ascendente (impulso)
DOR	0x32A8	suma lógica de datos numéricos (operación "OR"), instrucción de 32 bits
DORP	0x3AA8	suma lógica de datos numéricos (operación " OR "), instrucción de 32 bits con sondeo por flanco ascendente (impulso)
WXOR	0x22C8	Operación lógica "OR exclusivo"
WXORP	0x2AC8	operación lógica "OR exclusivo" con sondeo por flanco ascendente (impulso)

DXOR	0x32C8	operación lógica "OR exclusivo", instrucción de 32 bits
DXORP	0x3AC8	operación lógica "OR exclusivo", instrucción de 32 bits con sondeo por flanco ascendente (impulso)
NEG	0x2209	Negación lógica
NEGP	0x2A09	negación lógica con sondeo por flanco ascendente (impulso)
DNEG	0x3209	negación lógica, instrucción de 32 bits
DNEGP	0x3A09	negación lógica, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
ABS	0x2229	Valor absoluto
ABSP	0x2A29	valor absoluto con sondeo por flanco ascendente (impulso)
DABS	0x3229	Valor absoluto, instrucción de 32 bits
DABSP	0x3A29	Valor absoluto, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
SQR	0x2215	Cálculo de raíz cuadrada
SQRP	0x2A15	Cálculo de raíz cuadrada con sondeo por flanco ascendente (impulso)
DSQR	0x3215	Cálculo de raíz cuadrada, instrucción de 32 bits
DSQRP	0x3A15	Cálculo de raíz cuadrada, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
POW	0x2216	Elevación a una potencia
POWP	0x2A16	Elevación a una potencia con sondeo por flanco ascendente (impulso)
DPOW	0x3216	elevación a una potencia, instrucción de 32 bits
DPOWP	0x3A16	elevación a una potencia, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
Operaciones de desplazamiento		
ROR	0x220B	Desplazamiento cíclico a la derecha
RORP	0x2A0B	desplazamiento cíclico a la derecha con sondeo por flanco ascendente (impulso)
DROR	0x320B	desplazamiento cíclico a la derecha, instrucción de 32 bits

DRORP	0x3A0B	desplazamiento cíclico a la derecha, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
ROL	0x222B	Desplazamiento cíclico a la izquierda
ROLP	0x2A2B	Desplazamiento cíclico a la izquierda con sondeo por flanco ascendente (impulso)
DROL	0x322B	Desplazamiento cíclico a la izquierda, instrucción de 32 bits
DROLP	0x3A2B	Desplazamiento cíclico a la izquierda, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
Procesamiento de datos		
ZRST	0x2203	Reinicio de grupo de operandos en un rango dado
ZRSTP	0x2A03	Reinicio de grupo de operandos en un rango dado con sondeo por flanco ascendente (impulso)
DECO	0x2211	Decodificador 8 → 256-bit
DECOP	0x2A11	Decodificador 8 → 256-bit con sondeo por flanco ascendente (impulso)
ENCO	0x2212	Codificador 256 → 8 bits
ENCOP	0x2A12	Codificador 256 → 8 bits con sondeo por flanco ascendente (impulso)
SUM	0x2213	Suma de bits individuales en el registro
SUMP	0x2A13	Suma de bits individuales en el registro con sondeo por flanco ascendente (impulso)
DSUM	0x3213	Suma de bits individuales en el registro, instrucción de 32 bits
DSUMP	0x3A13	Suma de bits individuales en el registro, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
BON	0x2214	Comprobar el estado de un bit con ajuste de una salida
BONP	0x2A14	comprobar el estado de un bit con ajuste de una salida con sondeo por flanco ascendente (impulso)
DBON	0x3214	comprobar el estado de un bit con ajuste de una salida, instrucción de 32 bits
DBONP	0x3A14	comprobar el estado de un bit con ajuste de una salida, instrucción de 32 bits con sondeo por flanco ascendente (impulso)

FLT	0x220C	Convertir entero a punto flotante
FLTP	0x2A0C	convertir entero a punto flotante con sondeo por flanco ascendente (impulso)
DFLT	0x320C	convertir entero a punto flotante, instrucción de 32 bits
DFLTP	0x3A0C	convertir entero a punto flotante, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
Operaciones de punto flotante		
DECMP	0x220F	Comparación de números de punto flotante
DECMPP	0x2A0F	comparación de números de punto flotante con sondeo por flanco ascendente (impulso)
DEZCP	0x2210	Comparación de zona de punto flotante
DEZCPP	0x2A10	comparación de zona de punto flotante con sondeo por flanco ascendente (impulso)
DEADD	0x2217	Suma de números de punto flotante
DEADDP	0x2A17	suma de números de punto flotante con sondeo por flanco ascendente (impulso)
DESUB	0x2237	Resta de números de punto flotante
DESUBP	0x2A37	resta de números de punto flotante con sondeo por flanco ascendente (impulso)
DEMUL	0x2257	Multiplicación de números de punto flotante
DEMULP	0x2A57	multiplicación de números de punto flotante con sondeo por flanco ascendente (impulso)
DEDIV	0x2277	División de números de punto flotante
DEDIVP	0x2A77	división de números de punto flotante con sondeo por flanco ascendente (impulso)
DESQR	0x2218	Raíz cuadrada en formato de punto flotante
DESQRP	0x2A18	raíz cuadrada en formato de punto flotante con sondeo por flanco ascendente (impulso)
DEPOW	0x2297	Elevación a una potencia en formato de punto flotante

DEPOWP	0x2A97	elevación a una potencia en formato de punto flotante con sondeo por flanco ascendente (impulso)
INT	0x220D	Convertir un número de punto flotante a un entero
INTP	0x2A0D	convertir un número de punto flotante a un entero con sondeo por flanco ascendente (impulso)
DINT	0x320D	convertir un número de punto flotante a un entero, instrucción de 32 bits
DINTP	0x3A0D	convertir un número de punto flotante a un entero, instrucción de 32 bits con sondeo por flanco ascendente (impulso)
Tiempo y PWM		
TRD	0x2219	Lectura del valor actual del reloj de tiempo real
TRDP	0x2A19	lectura del valor actual del reloj de tiempo real con sondeo por flanco ascendente (impulso)
TWR	0x221A	Cambio del valor de un reloj de tiempo real
TWRP	0x2A1A	cambio del valor de un reloj de tiempo real con sondeo por flanco ascendente (impulso)
PWM	0x220E	Salida de modulación por ancho de pulso (PWM)
Fecha		
DRD	0x2239	Lectura del valor de la fecha actual
DRDP	0x2A39	lectura del valor de la fecha actual con sondeo por flanco ascendente (impulso)
DWR	0x223A	Cambiar el valor de la fecha
DWRP	0x2A3A	cambiar el valor de la fecha con sondeo por flanco ascendente (impulso)
Operaciones lógicas de tipo contacto		
LD&	0x4204	el contacto está cerrado si S1 & S2 ≠ 0
DLD&	0x5204	El contacto está cerrado si S1 & S2 ≠ 0, instrucción de 32 bits
LD|	0x4224	El contacto está cerrado si S1 | S2 ≠ 0
DLD|	0x5224	El contacto está cerrado si S1 | S2 ≠ 0, instrucción de 32 bits

LD^	0x4244	El contacto está cerrado si S1 ^ S2 ≠ 0
DLD^	0x5244	El contacto está cerrado si S1 ^ S2 ≠ 0, instrucción de 32 bits
AND&	0x4205	Contacto en serie cerrado si S1 & S2 ≠ 0
DAND&	0x5205	contacto en serie cerrado si S1 & S2 ≠ 0, instrucción de 32 bits
AND|	0x4225	contacto en serie cerrado si S1 | S2 ≠ 0
DAND|	0x5225	contacto en serie cerrado si S1 | S2 ≠ 0, instrucción de 32 bits
AND^	0x4245	contacto en serie cerrado si S1 ^ S2 ≠ 0
DAND^	0x5245	contacto en serie cerrado si S1 ^ S2 ≠ 0, instrucción de 32 bits
OR&	0x4206	contacto en paralelo cerrado si S1 & S2 ≠ 0
DOR&	0x5206	contacto en paralelo cerrado si S1 & S2 ≠ 0, instrucción de 32 bits
OR|	0x4226	contacto en paralelo cerrado si S1 | S2 ≠ 0
DOR|	0x5226	contacto en paralelo cerrado si S1 | S2 ≠ 0, instrucción de 32 bits
OR^	0x4246	contacto en paralelo cerrado si S1 ^ S2 ≠ 0
DOR^	0x5246	contacto en paralelo cerrado si S1 ^ S2 ≠ 0, instrucción de 32 bits
Operaciones de comparación de tipo contacto		
LD=	0x4264	el contacto está cerrado si S1 = S2
DLD=	0x5264	el contacto está cerrado si S1 = S2, instrucción de 32 bits
LD>	0x4284	el contacto está cerrado si S1 > S2
DLD>	0x5284	el contacto está cerrado si S1 > S2, instrucción de 32 bits
LD<	0x42A4	el contacto está cerrado si S1 < S2
DLD<	0x52A4	el contacto está cerrado si S1 < S2, instrucción de 32 bits
LD<>	0x42C4	el contacto está cerrado si S1 ≠ S2
DLD<>	0x52C4	el contacto está cerrado si S1 ≠ S2, instrucción de 32 bits
LD<=	0x42E4	El contacto está cerrado si S1 ≤ S2
DLD<=	0x52E4	El contacto está cerrado si S1 ≤ S2, instrucción de 32 bits
LD>=	0x4304	El contacto está cerrado si S1 ≥ S2
DLD>=	0x5304	El contacto está cerrado si S1 ≥ S2, instrucción de 32 bits

AND=	0x4265	Contacto en serie cerrado si S1 = S2
DAND=	0x5265	Contacto en serie cerrado si S1 = S2, instrucción de 32 bits
AND>	0x4285	Contacto en serie cerrado si S1 > S2
DAND>	0x5285	Contacto en serie cerrado si S1 > S2, instrucción de 32 bits
AND<	0x42A5	Contacto en serie cerrado si S1 < S2
DAND<	0x52A5	Contacto en serie cerrado si S1 < S2, instrucción de 32 bits
AND<>	0x42C5	Contacto en serie cerrado si S1 ≠ S2
DAND<>	0x52C5	Contacto en serie cerrado si S1 ≠ S2, instrucción de 32 bits
AND<=	0x42E5	contacto en serie cerrado si S1 ≤ S2
DAND<=	0x52E5	contacto en serie cerrado si S1 ≤ S2, instrucción de 32 bits
AND>=	0x4305	contacto en serie cerrado si S1 ≥ S2
DAND>=	0x5305	contacto en serie cerrado si S1 ≥ S2, instrucción de 32 bits
OR=	0x4266	contacto en paralelo cerrado si S1 = S2
DOR=	0x5266	contacto en paralelo cerrado si S1 = S2, instrucción de 32 bits
OR>	0x4286	contacto en paralelo cerrado si S1 > S2
DOR>	0x5286	contacto en paralelo cerrado si S1 > S2, instrucción de 32 bits
OR<	0x42A6	contacto en paralelo cerrado si S1 < S2
DOR<	0x52A6	contacto en paralelo cerrado si contacto en paralelo cerrado si S1 < S2, instrucción de 32 bits
OR<>	0x42C6	contacto en paralelo cerrado si S1 ≠ S2
DOR<>	0x52C6	contacto en paralelo cerrado si S1 ≠ S2, instrucción de 32 bits
OR<=	0x42E6	contacto en paralelo cerrado si S1 ≤ S2
DOR<=	0x52E6	contacto en paralelo cerrado si S1 ≤ S2, instrucción de 32 bits
OR>=	0x4306	contacto en paralelo cerrado si S1 ≥ S2
DOR>=	0x5306	contacto en paralelo cerrado si S1 ≥ S2, instrucción de 32 bits
Control de motor paso a paso		
SPIN	0x2207	Iniciar movimiento preestablecido

SPINP	0x2A07	iniciar movimiento preestablecido con sondeo por flanco ascendente (impulso)
TORQUE	0x2227	Aplicar las corrientes establecidas al motor
TORQUEP	0x2A27	Aplicar las corrientes establecidas al motor con sondeo por flanco ascendente (impulso)
HSTOP	0x2247	Cambiar a modo de retención inmediatamente
HSTOPP	0x2A47	Cambiar a modo de retención inmediatamente con sondeo por flanco ascendente (impulso)
HHIZ	0x2267	Desenergizar las fases del motor inmediatamente (el eje gira libremente)
HHIZP	0x2A67	Desenergizar las fases del motor inmediatamente (el eje gira libremente) con sondeo por flanco ascendente (impulso)
SSTOP	0x2287	Desacelerar hasta la parada completa y cambiar al modo de retención
SSTOPP	0x2A87	Desacelerar hasta la parada completa y cambiar al modo de retención con sondeo por flanco ascendente (impulso)
SHIZ	0x22A7	Desacelerar hasta la parada completa y desenergizar las fases del motor (el eje gira libremente)
SHIZP	0x2AA7	Desacelerar hasta la parada completa y desenergizar las fases del motor (el eje gira libremente) con sondeo por flanco ascendente (impulso)

Apéndice C. Ejemplos de programas de usuario

Ejemplo 1. Uso del comando RUN

LDP	X0			***;capturar el frente del pulso en la entrada X0 (botón)***
DMOV	K8	D359		*;establecer velocidad mínima 8 pps*
DMOV	K120000	D357		*;establecer velocidad máxima 120000 pps*
FMOV	K30000	D361	K2	*;establecer aceleración y desaceleración 30000 pps²*
MOV	K3	D366		*;micropasos 1/8 (consulte la descripción de la instrucción SPIN)*
MOV	K1	D374		*;dirección – avance*
DMOV	K6000	D377		*;establecer velocidad de paso completo 6000 pps/s*
MOV	K1	D379		*;habilitar el cambio al modo de paso completo al alcanzar la velocidad de paso completo*
MOV	K0	D376		*;comando RUN*
FMOV	K1500	D367	K2	*;corrientes de aceleración y desaceleración 1500 mA*
MOV	K1200	D369		*;corriente de velocidad constante 1200 mA*
MOV	K600	D370		*;corriente de retención 600 mA*
TORQUE				*;aplicar valores de corriente*
FMOV	K0	D380	K3	*;sin respuesta de error, errores restablecidos, usar ;MIN_SPEED*
SPIN				*;iniciar movimiento*
LDP	X1			*;capturar el flanco de subida del pulso en la entrada X1 (botón)*
SSTOP				*;detener según la DEC preestablecida y pasar al modo de retención*
LDP	X2			*;capturar el flanco de subida del pulso en la entrada X2 (botón)*
SHIZ				*;detener según la DEC preestablecida y pasar al modo HiZ*
LDP	X3			*;capturar el flanco de subida del pulso en la entrada X3 (botón)*
HSTOP				*;pasar inmediatamente al modo de retención*
LDP	X4			*;capturar el flanco de subida del pulso en la entrada X4 (botón)*
HHIZ				*;pasar inmediatamente al modo HiZ*
END				*;fin del programa*

Ejemplo 2. Uso de los comandos MOVE, GOTO, GOHOME

LD	M0			***;para omitir la sección de inicialización, verificar la condición M0***
CJ	P1			*;y saltar a la línea marcada P1*
LDP	M108			*;M108 flanco de subida solo después de la inicialización*
DMOV	K120000	D357		*;establecer la velocidad máxima 120000 pps*
FMOV	K30000	D361	K2	*;establecer la aceleración y desaceleración 30000pps²*
MOV	K3	D366		*;micropasos 1/8 (consulte la descripción de la instrucción SPIN)*
DMOV	K6000	D377		*;establecer velocidad de paso completo 6000 pps/s*
MOV	K1	D379		*;habilitar el cambio al modo de paso completo al alcanzar la velocidad de paso completo*
FMOV	K1500	D367	K2	*;corrientes de aceleración y desaceleración 1500 mA*
MOV	K1200	D369		*;corriente de velocidad constante 1200 mA*
MOV	K600	D370		*;corriente de retención 600 mA*
TORQUE				*;aplicar valores de corriente*
FMOV	K0	D380	K2	*;sin respuesta de error, errores restablecidos*
MOV	K1	D382		*;usar el cálculo automático de la velocidad inicial y final*
DMOV	K0	D363		*;poner a cero la posición actual*
SET	M0			*;activar la condición de omisión de inicialización del controlador*
P	1			*;marca de transición*
LDP	X0			*;capturar el frente del pulso en la entrada X0 (botón)*
AND&	D371	K3		*;solo si el motor está en modo HiZ o Hold*
DMOV	K10000	D372		*;mover10000 micropasos*
MOV	K1	D374		*;en la dirección de avance*
MOV	K1	D376		*;serealiza mediante el comando MOVE*
SPIN				*;iniciar movimiento*
LDP	X1			*;capturar el flanco de subida del pulso en la entrada X1 (botón)*
AND&	D371	K3		*;solo si el motor está en modo HiZ o Hold*
DMOV	K100000	D372		*;moviéndose a una posición con coordenada 100000*
MOV	K2	D376		*;serealiza mediante el comando GOTO*
SPIN				*;iniciar movimiento*
LDP	X2			*;capturar el flanco de subida del pulso en la entrada X2 (botón)*

AND&	D371	K3	*;solo si el motor está en modo HiZ o Hold*
MOV	K0	D374	*;movimiento a la posición "0" en la dirección de retroceso*
MOV	K4	D376	*;serealiza mediante el comando GOHOME*
SPIN			*;iniciar movimiento*
LDP	X3		*;capturar el flanco de subida del pulso en la entrada X3 (botón)*
SSTOP			*;detener según la DEC preestablecida y pasar al modo de retención*
LDP	X4		*;capturar el flanco de subida del pulso en la entrada X4 (botón)*
SHIZ			*;detener según la DEC preestablecida y pasar al modo HiZ*
LDP	X5		*;capturar el flanco ascendente del pulso en la entrada X5 (botón)*
HSTOP			*;pasar inmediatamente al modo de retención*
LDP	X6		*;capturar el flanco ascendente del pulso en la entrada X6 (botón)*
HHIZ			*;pasar inmediatamente al modo HiZ*
END			*;fin del programa*

Ejemplo 3. Uso de los comandos GOUNTIL_SLOWSTOP y RELEASE

Usando los comandos GOUNTIL_SLOWSTOP y RELEASE como un ejemplo de movimiento a la posición de origen a lo largo del interruptor de límite positivo (ver Fig. 36).

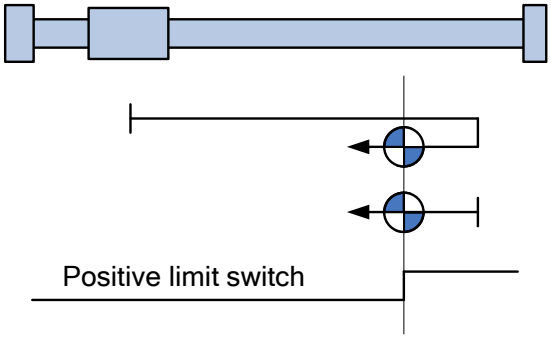


Fig. 36. Mover al origen

LD	M0			***;para omitir la sección de inicialización, verificar la condición M0***
CJ	P1			*;y saltar a la línea marcada P1*
LDP	M108			*;M108 flanco de subida solo después de la inicialización*
FMOV	K30000	D361	K2	*;establecer la aceleración y desaceleración 30000pps²*
MOV	K3	D366		*;micropasos 1/8 (consulte la descripción de la instrucción SPIN)*
DMOV	K6000	D377		*;establecer la velocidad de paso completo 6000 pps/seg*
MOV	K1	D379		*;habilitar el cambio al modo de paso completo al alcanzar la velocidad de paso completo*
FMOV	K1500	D367	K2	*;corrientes de aceleración y desaceleración 1500 mA*
MOV	K1200	D369		*;corriente de velocidad constante 1200 mA*
MOV	K600	D370		*;corriente de retención 600 mA*
TORQUE				*;aplicar valores de corriente*
FMOV	K0	D380	K2	*;sin respuesta de error, errores restablecidos*
MOV	K1	D382		*;usar el cálculo automático de la velocidad inicial y final*
MOV	K7	D375		*;el final de carrera está conectado a la entrada IN7*
SET	M0			*;activar la condición de omisión de inicialización del controlador*
P	1			*;marca de transición*
LDP	X0			*;capturar el frente del pulso en la entrada X0 (botón)*
AND&	D371	K3		*;solo si el motor está en modo HiZ o Hold*
DMOV	K20000	D357		*;establecer la velocidad máxima 20000 pps*

MOV	K5	D376	*;comando GOUNTIL_SLOWSTOP*
MOV	K1	D374	*;dirección – avance*
SPIN			*;iniciar movimiento*
SET	M1		*;establecer el indicador para iniciar la primera etapa*
LD	M1		*;esperar cuando se active el interruptor de fin de carrera en la primera etapa*
AND&	D371	K2	*;y el motor se detenga*
RST	M1		*;restablecer el indicador de la primera etapa*
DMOV	K1000	D357	*;disminuir la velocidad*
MOV	K9	D376	*;movimiento en la dirección opuesta hasta que el interruptor de fin de carrera*
MOV	K0	D374	*;se abra*
SPIN			*;iniciar movimiento*
SET	M2		*;y pasar a la segunda etapa*
LD	M2		*;esperando a que el interruptor de fin de carrera se abra y el motor se detenga*
AND&	D371	K2	*;en la segunda etapa*
RST	M2		*;restablecer el indicador de la segunda etapa*
DMOV	K0	D363	*;y restablecer la posición actual, que ahora se convierte en el origen*
LDP	X1		*;capturar el flanco de subida del pulso en la entrada X1 (botón)*
SSTOP			*;detener según la DEC preestablecida y pasar al modo de retención*
LDP	X2		*;capturar el flanco de subida del pulso en la entrada X2 (botón)*
SHIZ			*;detener según la DEC preestablecida y pasar al modo HiZ*
LDP	X3		*;capturar el flanco de subida del pulso en la entrada X3 (botón)*
HSTOP			*;pasar inmediatamente al modo de retención*
LDP	X4		*;capturar el flanco de subida del pulso en la entrada X4 (botón)*
HHIZ			*;pasar inmediatamente al modo HiZ*
FEND			*;fin del programa principal*

I	1007	*;manejador de interrupción para la entrada IN7 (requerido para los comandos GOUNTIL _... y ... RELEASE, puede dejarse vacío)*
IRET		*;volver al programa principal*
END		*;fin del programa*

Apéndice D. Código del programa de servicio "Control de Velocidad del Motor Paso a Paso"

LD	M0			***;condición de omisión de la inicialización***
CJ	P1			
LDP	M108			*;parte de inicialización*
MPS				
LD=	D320	K6		*;D320 almacena el valor de micropasos*
OR>	D320	K8		
OR<	D320	K0		
ANB				
MOV	K0	D320		
MRD				
MOV	D320	D366		
MOV	D320	D0		*;D0: el registro de servicio para la visualización de micropasos*
MOV	K0	D2		
MRD				
AND>	D0	K6		*;como el controlador no admite micropasos 1/64,*
DEC	D0			*;omitir este valor*
MRD				
DECO	D0	Y0	K3	*;visualización de micropasos en la escala de salidas*
MOV	K0	D379		*;configuración inicial del controlador de motor paso a paso*
MOV	K0	D376		
MOV	K250	D359		
MOV	K0	D380		
MRD				
LD<	D321	K0		*;D321 almacena los datos del método de control:*
OR>	D321	K2		*;potenciómetro/botones/codificador*
ANB				
MOV	K0	D321		
MRD				
SET	M0			
MOV	D321	Y10		*;visualización del método de control*
MRD				
AND<>	A0	D321		
MPS				
AND=	D321	K2		*;si se selecciona un **codificador**, entonces los periféricos*

MOV	K12	D355		*;del controlador deben configurarse en consecuencia*
MOV	D321	A0		
RST	M108			*;y reiniciar el programa*
MPP				
AND<>	D321	K2		
MOV	K0	D355		
MOV	D321	A0		
RST	M108			
MRD				
MUL	D353	K10	D4	*;datos del potenciómetro **1***
DIV	D4	K27	D4	
FMOV	D4	D367	K3	*;establecer la corriente de aceleración, desaceleración y velocidad constante*
DIV	D4	K2	D370	*;corriente de retención – 50% de la corriente de trabajo*
TORQUE				
MRD				
AND	X7			
MOV	K1	D374		
MRD				
ANI	X7			
MOV	K0	D374		
MRD				
DLD<	D322	K250		*;D322, D323 velocidad establecida por botones*
DOR>	D322	K120000		
ANB				
DMOV	K250	D322		
MRD				
DMOV	D322	D5		
MRD				
DLD<	D324	K250		*;D324, D325 velocidad establecida por el codificador*
DOR>	D324	K120000		
ANB				
DMOV	K250	D324		
MRD				
DMOV	D324	D15		
MRD				

DMOV	K0	C64	
DMOV	C64	D7	
MPP			
ANI	X4		
HSTOP			
LD	X2		
OUT	M102		
EI			;habilitar *interrupciones, el final del bloque de inicialización*
P	1		
LD	X4		
AND&	D371	HFE	
HHIZ			
LDI	X4		
MPS			
AND	M102		
AND	X3		
AND&	D371	K3	
CALL	P10A0		
SPIN			
MPP			
LDI	X3		
ANB			
AND&	D371	HFD	
HSTOP			
LDI	M102		
AND	X3		
ANI	X4		
AND&	D371	H15	
SSTOP			
LD	M109		*;si hubo un error y el indicador ERR estaba encendido -*
TMR	T0	K10	*;iniciar el temporizador para apagarlo*
AND	T0		
RST	M109		
LD<>	A0	K1	
CJ	P19		
LD=	A0	K1	

MPS			
ANDP	X0		
DADD	D5	D354	D5
MPS			
DAND>	D5	K120000	
DMOV	K120000	D5	
MPP			
DMOV	D5	D322	
MPP			
ANDP	X1		
DSUB	D5	D354	D5
MPS			
DAND<	D5	K250	
DMOV	K250	D5	
MPP			
DMOV	D5	D322	
P	19		
LD<>	A0	K2	
CJ	P20		
LD=	A0	K2	
MPS			
DSUB	C64	D7	D9
DADD	D7	D9	D7
DCMP	D9	K0	M1
MRD			
AND	M1		
MOV	D354	D1	
DMUL	D9	D1	D11
DADD	D11	D15	D15
MPP			
AND	M3		
MOV	D354	D1	
ABS	D9		
DMUL	D9	D1	D11
DADD	D15	D11	D15
LD	M1		

OR	M3			
MPS				
DAND<	D15	K250		
DMOV	K250	D15		
MRD				
DAND>	D15	K120000		
DMOV	K120000	D15		
MPP				
DMOV	D15	D324		
P	20			
FEND				*;fin del programa principal*
P	10			*;subprograma para configurar la velocidad con el potenciómetro*
LD	M108			
MOV	K0	D2		
MOV	D354	D1		
MPS				
AND=	D1	K0		
MOV	K1	D1		
MRD				
DMUL	D1	K29	D13	
MRD				
DAND<	D13	K250		
DMOV	K250	D13		
MPP				
DMOV	D13	D357		
CALL	P0			
SRET				
P	11			*; subprograma para configurar la velocidad con los botones*
LD	M108			
DMOV	D322	D357		
CALL	P0			
SRET				
P	12			*; subprograma para configurar la velocidad con el codificador*
LD	M108			

DMOV	D324	D357		
CALL	P0			
SRET				
P	0			*;subprograma para actualizar la aceleración y la desaceleración*
LD	M108			
MOV	D352	D3		
MPS				
AND=	D3	K0		
MOV	K1	D3		
MPP				
MUL	D3	K14	D361	
MOV	D361	D362		
SRET				
I	50			*;interrupción de 500 ms para actualizar las corrientes*
LD	M108			
MUL	D353	K10	D4	
DIV	D4	K27	D4	
FMOV	D4	D367	K3	
DIV	D4	K2	D370	
TORQUE				
IRET				
I	10			*;interrupción con un período de 100 ms para actualizar la velocidad*
LDI	X6			
AND	X2			
AND	M102			
AND	X3			
ANI	X4			
BON	D371	M60	K6	
ANI	M60			
CALL	P10A0			
SPIN				
IRET				
I	1005			*;interrupción desde la entrada IN5*
LD	M105			

INC	D320			
MPS				
AND=	D320	K6		
INC	D320			
MRD				
AND=	D320	K9		
MOV	K0	D320		
INC	D321			
MPS				
AND=	D321	K3		
MOV	K0	D321		
MRD				
MOV	D321	Y10		
MOV	D321	A0		
MRD				
AND=	D321	K2		
MOV	K12	D355		
RST	M108			
MPP				
AND<>	D321	K2		
MOV	K0	D355		
RST	M108			
MRD				
MOV	D320	D0		
MOV	D320	D366		
MRD				
AND>	D0	K6		
DEC	D0			
MPP				
DECO	D0	Y0	K3	
IRET				
I	1004			*;interrupción desde la entrada IN4*
LD	M104			
HHIZ				
LDI	M104			
MPS				

AND	X2		
AND	X3		
AND&	D371	K3	
CALL	P10A0		
SPIN			
MPP			
LDI	X2		
ORI	X3		
ANB			
HSTOP			
IRET			
I	1003		*;interrupción desde la entrada IN3*
LDI	M103		
ANI	X4		
HSTOP			
LD	M103		
AND	X2		
ANI	X4		
AND&	D371	K3	
CALL	P10A0		
SPIN			
IRET			
I	1002		*;interrupción desde la entrada IN2*
LDI	M102		
AND	X3		
ANI	X4		
SSTOP			
LD	M102		
AND	X3		
ANI	X4		
AND&	D371	K3	
CALL	P10A0		
SPIN			
IRET			
I	1007		*;interrupción desde la entrada IN7*
LD&	D371	H1C	

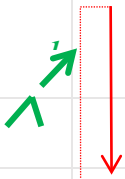
MPS			
AND	M107		
AND=	D374	K0	
SSTOP			
MPP			
ANI	M107		
AND=	D374	K1	
SSTOP			
LD	M107		
MOV	K1	D374	
AND	X2		
AND	X3		
ANI	X4		
AND&	D371	K3	
CALL	P10A0		
SPIN			
LDI	M107		
MOV	K0	D374	
AND	X2		
AND	X3		
ANI	X4		
AND&	D371	K3	
CALL	P10A0		
SPIN			
IRET			
I	2000		*;interrupción cuando ocurre un error del controlador*
LD&	D381	K1	
MOV	K0	D381	
SET	M109		
MOV	K0	T0	
IRET			
END			

Apéndice E. La vida útil de los flancos de los operandos M e Y

Toda la información a continuación es válida para los operandos M e Y, tanto para los flancos de subida como para los de bajada.

Ejemplo 1

Considere la vida útil del flanco de subida del operando M0, con el paso garantizado del punto de inicio de la vida en el próximo ciclo de scan.

Línea	Instrucción	Operandos, número de orden		Vida útil del frente de M0, ciclo de scan		Explicación
		1	2	1	2	
1	LDP	M108				El flanco ascendente de M108 solo existe en el primer ciclo de scan del programa
2	ZRST	D0	D2			Puesta a cero de D0…D2
3	LD	M108				
4	OUT	M0				El inicio de la vida del flanco de subida del operando M0 está en el primer ciclo de scan, y el final está en el segundo
5	P	1				
6	LD	M0				
7	CALL	P0				El estado actual M0 se guarda en cada salto al subprograma P0.
8	LDP	M0				Funcionará solo una vez, ya que se cumple la condición de volver a pasar por el punto de inicio del flanco de subida del operando M0 en el siguiente ciclo de scan del programa.

9	INC	D0				El resultado del programa es D0 = 1.
10	FEND					
11	I	0				La primera interrupción ocurre después de la línea 2 en el primer ciclo de scan. En este momento, el flanco de subida de M0 está ausente. La segunda interrupción se procesa después de la línea 6. La presencia de un flanco de subida en M0 se transmite al procesamiento de la interrupción, por lo que el resultado del trabajo es D2 = 1.
13	LDP	M0				
14	INC	D2				
15	IRET					
16	P	0				
17	LDP	M0				La condición se cumple en el primer ciclo de scan. La condición no se cumple en el segundo ciclo de scan.
18	INC	D1				El resultado es D1 = 1.
19	SRET					
20	END					

- interrupción

- existencia de un flanco de subida del operando

Ejemplo 2

Considere la vida útil del flanco de subida del operando M0, en ausencia de pasar por el punto de inicio de la vida en el próximo ciclo de scan.

Línea	Instrucción	Operandos, número de orden	Vida útil del frente de M0, ciclo de scan	Explicación

		1	2	1	2	3	
1	LDP	M0					
2	CJ	P1					Omitiendo el punto de inicio de la vida del flanco de subida.
3	LDP	M108					El flanco de subida del M108 solo existe en el primer ciclo de scan del programa.
4	ZRST	D0	D2				Poniendo a cero D0…D2.
5	LD	M108					
6	OUT	M0					El inicio de la vida del flanco de subida del operando M0 en el primer ciclo de scan. El siguiente paso por este punto será solo en el tercer ciclo de scan.
7	P	1					
8	LDP	M0					Funcionará dos veces, ya que el punto de inicio de la vida del flanco de subida fue omitido por el manejador de comandos en el segundo ciclo de scan, y el tiempo de vida se incrementó hasta que se recibió el comando END / FEND.
9	INC	D0					El resultado del programa es D0 = 2.
10	END						La vida útil del flanco de subida del operando M0 se extiende hasta el final del ciclo de scan debido a la ausencia del paso del punto de inicio de la vida útil del flanco de subida M0.

↓ - existencia de un flanco de subida del operando

Si se requiere un solo paso entre los puntos 7 ... 9, entonces, por ejemplo, se puede usar el contacto de relé opcional M1. El programa se cambiará como se muestra a continuación:

```
1   LDP     M0
2   CJ      P1
3   LDP     M108
4   ZRST    D0          D2
5   ZRST    M1                  ;Inicialización M1
6   LD      M108
7   OUT     M0
8   P       1
9   LDP     M0
10  ANI     M1                  ;Condición adicional
11  INC     D0                  ;El resultado del programa esD0 = 1
12  SET     M1                  ;Enclavamiento
13  END
```

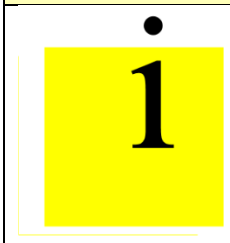
Apéndice F. Depuración del programa de usuario

El modo de depuración permite al usuario:

- establecer cuatro puntos de interrupciónpara la ejecución del programa de usuario(punto de interrupción),
- ver y editar operandos,
- pausar y reanudar la ejecución del programa de usuario.

A continuación, se muestra la lista de los registros del depurador:

Dirección	Tipo	Tamaño	Descripción
Control de la ejecución del programa de usuario			
0x6100	Registros de entrada	16 bits	Índice actual (número de línea de comando) del programa de usuario
0x6100	Bobinas	-	Establecer el objeto activa el modo de depuración, restablecerlo lo desactiva. Además, el modo de depuración se desactiva cuando el interruptor de palanca RUN/STOP se coloca en el estado STOP.
0x6100	Entradas discretas	-	Indicación del modo de depuración. Establecido: el controlador está en el modo de depuración Restablecido: el controlador no está en el modo de depuración
0x6101	Bobinas	-	Establecer el objeto suspende la ejecución del programa de usuario. Establecer el registro suspenderá la ejecución del programa de usuario en el índice actual. Restablecer: reanuda la ejecución.
0x6101	Entradas discretas	-	Indicación de suspensión de un programa de usuario. Establecido: el programa de usuario está suspendido Restablecido: el programa de usuario se ejecuta
0x6102	Bobinas	-	Establecer el objeto activa la depuración paso a paso. Al intentar reanudar el programa de usuario restableciendo las bobinas 6101h, la ejecución se abortará automáticamente en el siguiente índice.

Dirección	Tipo	Tamaño	Descripción
	Puntos de interrupción		
			*Además de la depuración paso a paso, cuando un programa de usuario en ejecución se suspende en cada siguiente línea de comando, es posible especificar cuatro puntos de interrupción en los que se suspenderá la ejecución del programa.*
	Punto de interrupción 1		
0x6200	Bobinas	-	Establecer el objeto activa el punto de interrupción 1. La ejecución del programa de usuario se suspenderá en el índice que se especifica en los registros de retención 6200h.
0x6200	Registros de retención	16 bits	Índice (número de línea de comando) del punto de interrupción 1.
	Punto de interrupción 2		
0x6201	Bobinas	-	Establecer el objeto activa el punto de interrupción 2. La ejecución del programa de usuario se suspenderá en el índice que se especifica en los registros de retención 6201h.
0x6201	Registros de retención	16 bits	Índice (número de línea de comando) del punto de interrupción 2.
	Punto de interrupción 3		
0x6202	Bobinas	-	Establecer el objeto activa el punto de interrupción 3. La ejecución del programa de usuario se suspenderá en el índice que se especifica en los registros de retención 6202h.
0x6202	Registros de retención	16 bits	Índice (número de línea de comando) del punto de interrupción 3.
	Punto de interrupción 4		
0x6203	Bobinas	-	La configuración del objeto activa el punto de interrupción 4. La ejecución del programa de usuario se suspenderá en el índice, que se especifica en los registros de retención 6203h.
0x6203	Registros Registros	16 bits	Índice (número de línea de comando) del punto de interrupción 4.

Dirección	Tipo	Tamaño	Descripción
Monitorización y edición de operandos			
0x6000	Bobinas	-	Solicitud para leer datos mediante la configuración del registro. El reinicio se produce automáticamente. La respuesta a la solicitud indica la disponibilidad de los datos solicitados sobre el operando.
0x6001	Bobinas	-	Solicitud para escribir datos mediante la configuración del registro. El reinicio se produce automáticamente. La respuesta a la solicitud indica que los datos se han escrito en el operando.
0x6002	Bobinas	-	Tamaño de los datos del registro de lectura/escritura Reinicio – 16 bits Configuración – 32 bits.
0x6003	Bobinas	-	La configuración del registro cancela la edición del valor del operando cuando se establecen las bobinas 6001h, para los operandos “C” y “T”.
0x6004	Bobinas	-	La configuración del registro cancela la edición de la señal del operando cuando se establecen las bobinas 6001h, para los operandos “C” y “T”.
0x6000	Entradas discretas	-	El objeto se establece cuando falla una operación de lectura o escritura de un operando. El reinicio se realiza automáticamente cuando se solicita una operación de lectura o escritura.
Parametrización de operandos			
0x6000	Registros de retención	16 bits	Tipo de operando. X (0x58), Y (0x59), M (0x4D), T (0x54), C (0x43), A (0x41), B (0x42), D (0x44).
0x6001	Registros de retención	16 bits	Índice de operando.

Dirección	Tipo	Tamaño	Descripción
Monitorización de operandos			
0x6002	Registros de entrada	32 bits	Este registro contiene el valor del operando (si está disponible) parametrizado para la lectura. El tamaño se establece mediante las bobinas 6002h.
0x6004	Registros de entrada	16 bits	Este registro contiene la señal del operando (si está disponible), parametrizada para la lectura. Valores posibles: 0x00 – nivel bajo, 0x03 – nivel alto, con un flanco de subida 0x02 – nivel alto, 0x04 – nivel bajo, con un flanco de bajada.
Edición de operandos			
0x6002	Registros de retención	32 bits	Este registro contiene el valor del operando (si está disponible) parametrizado la escritura. El tamaño se establece mediante las bobinas 6002h.
0x6004	Registros de retención	16 bits	Este registro contiene la señal del operando (si está disponible) parametrizada para la escritura. Valores posibles: 0x00 – nivel bajo, 0x03 – nivel alto, con un flanco de subida 0x02 – nivel alto, 0x04 – nivel bajo, con un flanco de bajada.

Dirección del departamento de ventas del fabricante:
Smart Motor Devices OÜ
Akadeemia tee 21/6, 12618, Tallinn, Estonia
Teléfono: + 372 6559914
Correo electrónico: mail@smd.ee
URL: https://smd.ee

Última modificación: 07.2025