SIEMENS

SIMATIC

Esquema de contactos (KOP) para S7-300 y S7-400

Manual de referencia

Este manual forma parte del paquete de documentación con la referencia:
6ES7810-4CA08-8DW1

Edición 03/2006
A5E00706951-01

Consignas de seguridad

Este manual contiene las informaciones necesarias para la seguridad personal así como para la prevención de daños materiales. Las informaciones para su seguridad personal están resaltadas con un triángulo de advertencia; las informaciones para evitar únicamente daños materiales no llevan dicho triángulo. De acuerdo al grado de peligro las consignas se representan, de mayor a menor peligro, como sigue**:**

Peligro
Significa que, si no se adoptan las medidas preventivas adecuadas se producirá la muerte, o bien lesiones corporales graves.

Advertencia
Significa que, si no se adoptan las medidas preventivas adecuadas puede producirse la muerte o bien lesiones corporales graves.

Precaución
Con triángulo de advertencia significa que si no se adoptan las medidas preventivas adecuadas, pueden producirse lesiones corporales.

Precaución
Sin triángulo de advertencia significa que si no se adoptan las medidas preventivas adecuadas, pueden producirse daños materiales**.**

Atención
Significa que puede producirse un resultado o estado no deseado si no se respeta la consigna de seguridad correspondiente.

Si se dan varios niveles de peligro se usa siempre la consigna de seguridad más estricta en cada caso. Si en una consigna de seguridad con triángulo de advertencia se alarma de posibles daños personales, la misma consigna puede contener también una advertencia sobre posibles daños materiales.

Personal cualificado

El equipo/sistema correspondiente sólo deberá instalarse y operarse respetando lo especificado en este documento. Sólo está autorizado a intervenir en este equipo el personal cualificado. En el sentido del manual se trata de personas que disponen de los conocimientos técnicos necesarios para poner en funcionamiento, conectar a tierra y marcar los aparatos, sistemas y circuitos de acuerdo con las normas estándar de seguridad.

Uso conforme

Considere lo siguiente:

Advertencia

El equipo o los componentes del sistema sólo se podrán utilizar para los casos de aplicación previstos en el catálogo y en la descripción técnica, y sólo asociado a los equipos y componentes de Siemens y de tercera que han sido recomendados y homologados por Siemens.

El funcionamiento correcto y seguro del producto presupone un transporte, un almacenamiento, una instalación y un montaje conforme a las prácticas de la buena ingeniería, así como un manejo y un mantenimiento rigurosos.

Marcas registradas

Todos los nombres marcados con ® son marcas registradas de Siemens AG. Los restantes nombres y designaciones contenidos en el presente documento pueden ser marcas registradas cuya utilización por terceros para sus propios fines puede violar los derechos de sus titulares.

Exención de responsabilidad

Hemos comprobado la concordancia del contenido de esta publicación con el hardware y el software descritos. Sin embargo, como es imposible excluir desviaciones, no podemos hacernos responsable de la plena concordancia. El contenido de esta publicación se revisa periódicamente; si es necesario, las posibles correcciones se incluyen en la siguiente edición.

Siemens AG
Automation and Drives
Postfach 4848,
90437 NÜRNBERG
ALEMANIA

A5E00706951-01
03/2006

Prológo

Objetivo del manual

Este manual le servirá de ayuda al crear programas de usuario con el lenguaje de pogramación KOP.

Describe los elementos del lenguaje de programación KOP, así como su sintaxis y sus funciones.

Nociones básicas

Este manual está dirigido a programadores de programas S7, operadores y personal de mantenimiento que dispongan de conocimientos básicos sobre los autómatas programables.

Además es necesario estar familiarizado con el uso de ordenadores o equipos similares a un PC (p. ej. unidades de programación) bajo los sistemas operativos MS Windows 2000 Professional, MS Windows XP Profesional o MS Windows Server 2003.

Objeto del manual

El software en el que se basan las indicaciones del manual es STEP 7 V5.4.

Cumplimiento de la normativa IEC 1131-3

KOP sigue los principios del lenguaje ”Esquema de contactos” (en inglés Ladder Logic) fijados en la norma DIN EN-61131-3 (int. IEC 1131-3). En la tabla sobre cumplimiento de normas contenida en el archivo NORM_TAB.WRI de STEP 7 encontrará información más detallada sobre el cumplimiento de las normas.

Requisitos

Para entender correctamente el presente manual de KOP se requieren conocimientos teóricos acerca de los programas S7, que se pueden consultar en la Ayuda en pantalla de STEP 7. Como que los paquetes acerca de los lenguajes de programación se basan en el software estándar de STEP 7, debería conocerse ya mínimamente el uso del software y su documentación.

Este manual forma parte del paquete de documentación "STEP 7 Información de referencia“.

La tabla siguiente da una visión de conjunto de la documentación de STEP 7:

Manuales	**Tema**	**Referencia**
Información básica de STEP 7 compuesta por: • STEP 7 : Introducción y ejercicios prácticos • Programar con STEP 7 • Configurar el hardware y la comunicación con STEP 7 • De S5 a S7, Guía para facilitar la transición	Nociones básicas para el personal técnico. Describe cómo realizar soluciones de control con el software STEP 7 para los sistemas S7-300/400.	6ES7810-4CA08-8DW0
Información de referencia para STEP 7, compuesta por • Manuales KOP/FUP/AWL para S7-300/400 • Funciones estándar y funciones de sistema para S7-300/400 Tomo 1 y Tomo 2	Esta obra de consulta describe los lenguajes de programación KOP, FUP y AWL así como las funciones estándar y las funciones de sistema como complemento a la 'Información básica de STEP' .	6ES7810-4CA08-8DW1

Ayudas en pantalla	**Tema**	**Referencia**
Ayuda de STEP 7	Nociones básicas para diseñar programas y configurar el hardware con STEP 7. Disponible en forma de Ayuda en pantalla.	Componente del paquete de software STEP 7
Ayudas de referencia para AWL/KOP/FUP Ayudas de referencia para SFBs/SFCs Ayudas de referencia para los bloques de organización	Información de referencia sensible al contexto	Componente del paquete de software STEP 7

Ayuda en pantalla

Como complemento al manual puede recurrir a la Ayuda en pantalla integrada en el software.

A la Ayuda que está integrada en el software se accede de distinta manera:

- La Ayuda sensible al contexto ofrece información sobre el contexto actual, p. ej. sobre el cuadro de diálogo que esté abierto o sobre la ventana activa. Para acceder a esta ayuda pulse el botón de comando ”Ayuda” o bien la tecla F1.
- El menú **Ayuda** ofrece varios comandos de menú: **Temas de Ayuda** abre el índice de la Ayuda de STEP 7.
- A través de "Glosario" se accede al glosario para todas las aplicaciones de STEP 7.

Este manual es un extracto de la Ayuda de KOP. Debido a que la estructura del manual se corresponde a grandes rasgos con la de la Ayuda en pantalla puede alternar la lectura del manual con la de la Ayuda en pantalla.

Asistencia adicional

Si tiene preguntas relacionadas con el uso de los productos descritos en el manual a las que no encuentre respuesta, diríjase a la sucursal o al representante más próximo de Siemens, en donde le pondrán en contacto con el especialista.

Encontrará a su persona de contacto en la página de Internet:

http://www.siemens.com/automation/partner

Encontrará una guía sobre el conjunto de la información técnica correspondiente a los distintos productos y sistemas SIMATIC en la página de Internet:

http://www.siemens.com/simatic-tech-doku-portal

Encontrará el catálogo y el sistema de pedidos on-line en:

http://mall.automation.siemens.com/

Centro de formación SIMATIC

Para ofrecer a nuestros clientes un fácil aprendizaje de los sistemas de automatización SIMATIC S7, les ofrecemos distintos cursillos de formación. Diríjase a su centro de formación regional o a la central en D 90327 Nuernberg.

Teléfono: +49 (911) 895-3200.

Internet: http://www.sitrain.com

Technical Support

Podrá acceder al Technical Support de todos los productos de A&D

- a través del formulario de Internet para el Support Request
  http://www.siemens.com/automation/support-request
- Teléfono: + 49 180 5050 222
- Fax: + 49 180 5050 223

Encontrará más información sobre nuestro Technical Support en la página de Internet
http://www.siemens.com/automation/service

Service & Support en Internet

Además de nuestra documentación, en Internet le ponemos a su disposición todo nuestro know-how.

http://www.siemens.com/automation/service&support

En esta página encontrará:

- "Newsletter" que le mantendrán siempre al día ofreciéndole informaciones de última hora,
- La rúbrica "Servicios online" con un buscador que le permitirá acceder a la información que necesita,
- El "Foro" en el que podrá intercambiar sus experiencias con cientos de expertos en todo el mundo,
- El especialista o experto de Automation & Drives de su región,
- Bajo la rúbrica "Servicios" encontrará información sobre el servicio técnico más próximo, sobre reparaciones, repuestos etc.

Índice

1 Operaciones lógicas con bits

1.1 Lista de operaciones lógicas con bits

Descripción

Las operaciones lógicas con bits operan con dos dígitos, 1 y 0. Estos dos dígitos consituyen la base de un sistema numérico denominado sistema binario. Los dos dígitos 1 y 0 se denominan dígitos binarios o bits. En el ámbito de los contactos y bobinas, un 1 significa activado ("conductor") y un 0 significa desactivado ("no conductor").

Las operaciones lógicas con bits interpretan los estados de señal 1 y 0, y los combinan de acuerdo con la lógica de Boole. Estas combinaciones producen un 1 ó un 0 como resultado y se denominan "resultado lógico" (RLO). Las operaciones lógicas con bits permiten ejecutar las más diversas funciones.

Se dispone de las operaciones lógicas con bits siguientes:

- ---| |--- Contacto normalmente abierto
- ---| / |--- Contacto normalmente cerrado
- ---(SAVE) Cargar resultado lógico (RLO) en registro RB
- XOR O-exclusiva
- ---( ) Bobina de relé, salida
- ---(#)--- Conector
- ---|NOT|--- Invertir resultado lógico (RLO)

Las siguientes operaciones reaccionan ante un RLO de 1:

- ---(S) Activar salida
- ---(R) Desactivar salida
- SR Desactivar flip-flop de activación
- RS Activar flip-flop de desactivación

Otras operaciones reaccionan ante un cambio de flanco positivo o negativo para ejecutar las siguientes funciones:

- ---(N)--- Detectar flanco 1 --> 0
- ---(P)--- Detectar flanco 0 --> 1
- NEG Detectar flanco de señal negativo (1 --> 0)
- POS Detectar flanco de señal positivo (0 --> 1)

- Leer directamente de periferia
- Escribir directamente en periferia

1.2 ---| |--- Contacto normalmente abierto

Símbolo

<Operando>

---| |---

Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando>	BOOL	E, A, M, L, D, T, Z	Bit consultado

Descripción de la operación

---| |--- (Contacto normalmente abierto) se cierra si el valor del bit consultado, que se almacena en el **<operando>** indicado, es "1". Si el contacto está cerrado, la corriente fluye a través del contacto y el resultado lógico (RLO) es "1".

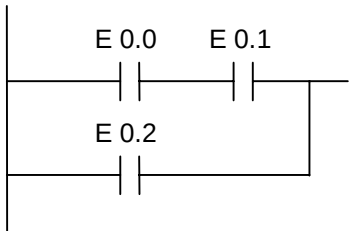
De lo contrario, si el estado de señal en el **<operando>** indicado es "0", el contacto está abierto. Si el contacto está abierto no hay flujo de corriente y el resultado lógico de la operación (RLO) es "0".

En las conexiones en serie, el contacto ---| |--- se combina bit a bit por medio de una Y lógica con el RLO. Cuando las conexiones se realizan en paralelo, el contacto se combina con el RLO por medio de una O lógica.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo



La corriente puede fluir si:

el estado en las entradas E 0.0 Y E 0.1 es "1" **O** el estado en la entrada E 0.2 es "1".

1.3 ---| / |--- Contacto normalmente cerrado

Símbolo

<Operando>

---| / |---

Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando>	BOOL	E, A, M, L, D, T, Z	Bit consultado

Descripción de la operación

---| / |--- (Contacto normalmente cerrado) se abre si el valor del bit consultado, que se almacena en el **<operando>** indicado, es "0". Si el contacto está cerrado, la corriente fluye a través del contacto y el resultado lógico (RLO) es "1".

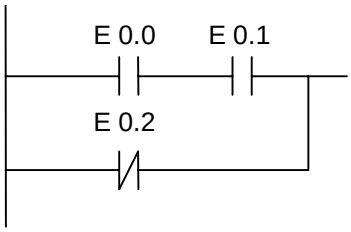
De lo contrario, si el estado de señal en el **<operando>** indicado es "1" , el contacto está abierto. Si el contacto está abierto no hay flujo de corriente y el resultado lógico de la operación (RLO) es "0".

Cuando se realizan conexiones en serie, el contacto ---| / |--- se combina bit a bit por medio de una Y lógica con el RLO. Si las conexiones se efectúan en paralelo, el contacto se combina con el RLO por medio de una O lógica.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo



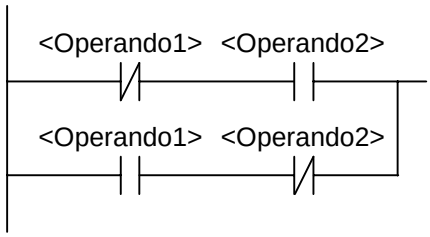
La corriente puede fluir si:

el estado en las entradas E 0.0 Y E 0.1 es "1" **O** el estado en la entrada E 0.2 es "1".

1.4 XOR O-exclusiva

Símbolos

Para la función XOR es necesario crear un segmento de contactos normalmente abiertos y normalmente cerrados (tal como se representa abajo).

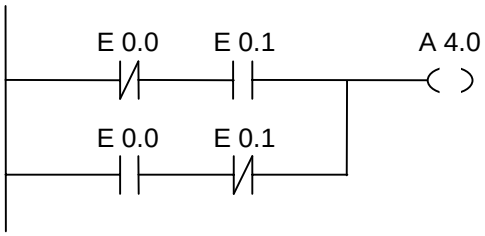


Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando1>	BOOL	E, A, M, L, D, T, Z	Bit que se ha consultado
<Operando2>	BOOL	E, A, M, L, D, T, Z	Bit que se ha consultado

Descripción de la operación

XOR (O-exclusiva) genera un RLO de "1" si el estado de señal de los dos bits indicados es distinto.

Ejemplo



La salida A 4.0 es "1" si (E 0.0 es 0 **Y** E 0.1 es 1) **O** (E 0.0 es 1 **Y** E 0.1 es 0).

1.5 ---|NOT|--- Invertir resultado lógico (RLO)

Símbolo

---|NOT|---

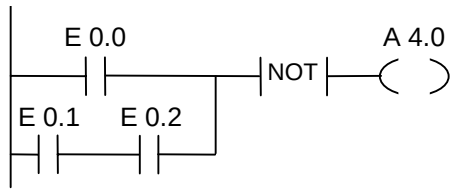
Descripción de la operación

---|NOT|--- (invertir resultado lógico) invierte el bit RLO.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	-	1	x	-

Ejemplo



La salida A 4.0 es "0" si:

El estado en la entrada E 0.0 es "1" **O** el estado en E 0.1. **Y** E 0.2 es "1".

1.6 ---( ) Bobina de relé, salida

Símbolo

<Operando>

---( )

Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando>	BOOL	E, A, M, L, D	Bit asignado

Descripción de la operación

---( ) (Bobina de relé (salida)) opera como una bobina en un esquema de circuitos. Si la corriente fluye hasta la bobina (RLO = 1), el bit en el **<operando>** se pone a "1". Si no fluye corriente hasta la bobina (RLO = 0), el bit en el **<operando>** se pone a "0". Una bobina de salida sólo puede colocarse dentro de un esquema de contactos en el extremo derecho de un circuito. Como máximo puede haber 16 salidas múltiples (v. ejemplos). Se puede crear una salida negada anteponiendo a la bobina de salida la operación ---|NOT|--- (invertir el resultado lógico).

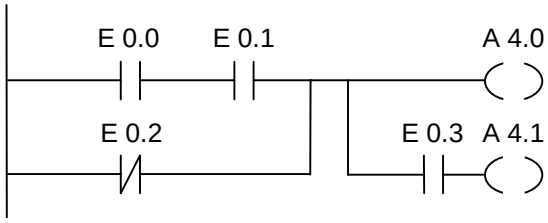
Dependencia con respecto al MCR (Master Control Relay)

La dependencia con respecto al MCR solamente se activa cuando una bobina de salida se encuentra dentro de un área MCR activa. Si el MCR está conectado y la corriente fluye a una bobina de salida, el bit direccionado toma el estado de señal actual del flujo de corriente. Si el MCR está desconectado se escribe un "0" en el operando indicado, independientemente del estado del flujo de corriente.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	-	0

Ejemplo



La salida A 4.0 es "1" si:

(el estado de la entrada E 0.0 Y E 0.1 es "1") **O** el estado de la entrada E 0.2 es "0".

La salida A 4.1 es "1"si:

(el estado de la entrada E 0.0 Y E 0.1 es "1" **O** el estado de la entrada E 0.2 es "0") **Y** el estado de la entrada E 0.3 es "1".

Si el circuito del ejemplo se encuentra en un área MCR activa:

Al estar conectado el MCR, las salidas A 4.0 y A 4.1 se ponen a 1 conforme al estado de señal del flujo de corriente, tal como se ha descrito más arriba.

Si el MCR está desconectado, las salidas A 4.0 y A 4.1 se ponen a "0", independientemente del estado de señal del flujo de corriente.

1.7 ---(#)--- Conector

Símbolo

<Operando>

---(#)---

Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando>	BOOL	E, A, M, L, D	Bit asignado

* Un operando de la pila de datos locales sólo puede utilizarse si ha sido declarado en la tabla de declaración de variables en el área TEMP de un bloque de código (FC, FB, OB).

Descripción de la operación

---(#)--- (Conector) es un elemento intercalado que cumple una función de asignación; el conector almacena el RLO actual (el estado de señal del flujo de corriente) en el **<operando>** que se haya especificado. Este elemento de asignación memoriza la combinación lógica de bits de la última rama abierta que esté antes que él. Si se conecta en serie con otros elementos, la operación ---(#)--- se inserta igual que un contacto. El elemento ---(#)--- nunca debe conectarse a una barra de alimentación, ni colocarse directamente detrás de una rama, y tampoco debe emplearse como final de una rama. Se puede crear la negación del elemento ---(#)--- anteponiéndole el elemento ---|NOT|--- (invertir el resultado lógico).

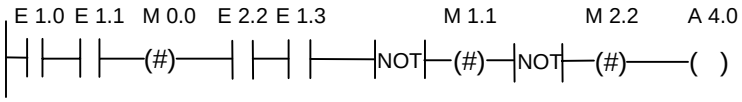
Dependencia con respecto al MCR (Master Control Relay)

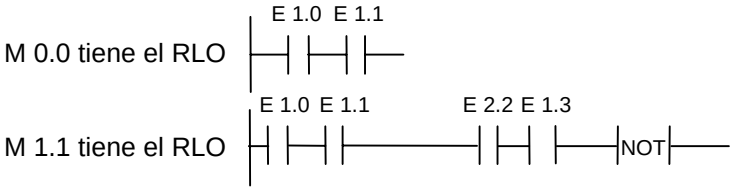
La dependencia con MCR solamente se activa cuando un conector se encuentra dentro de un área de MCR activa. Si el MCR está conectado y la corriente fluye a un conector, el bit direccionado toma el estado de señal actual del flujo de corriente. Si el MCR está desconectado se escribe un "0" en el operando indicado, independientemente del estado del flujo de corriente.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	-	1

Ejemplo


E 1.0 E 1.1 M 0.0 E 2.2 E 1.3 M 1.1 M 2.2 A 4.0
(#)
NOT
(#)
NOT
(#)
( )


E 1.0 E 1.1
M 0.0 tiene el RLO
E 1.0 E 1.1 E 2.2 E 1.3
M 1.1 tiene el RLO
NOT
M 2.2 tiene el RLO de toda combinación de bits

1.8 ---(R) Desactivar salida

Símbolo

<Operando>

---(R)

Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando>	BOOL	E, A, M, L, D, T, Z	Bit desactivado

Descripción de la operación

---(R) (Desactivar salida) sólo se ejecuta si el RLO de las operaciones anteriores es "1" (flujo de corriente en la bobina). Si fluye corriente a la bobina (RLO es "1"), el **<operando>** indicado del elemento se pone a "0". Un RLO de "0" (= no hay flujo de corriente en la bobina) no tiene efecto alguno, de forma que el estado de señal del operando indicado del elemento no varía. El **<operando>** también puede ser un temporizador (N.° de T) cuyo valor de temporización se pone a "0", o un contador (N.° de Z) cuyo valor de contaje se pone a "0".

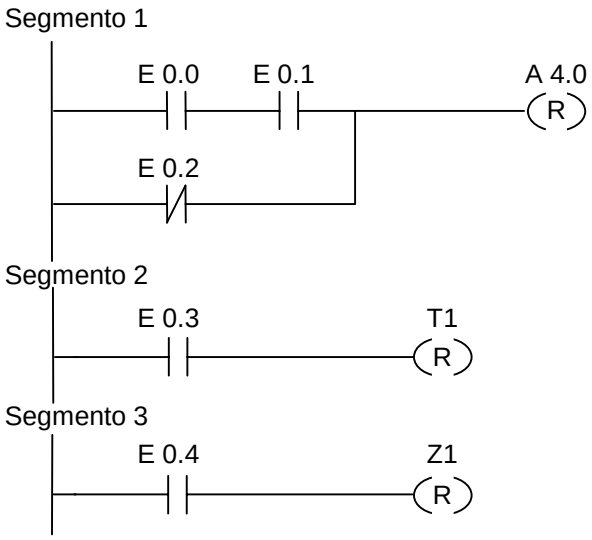
Dependencia con respecto al MCR (Master Control Relay)

La dependencia con respecto al MCR solamente se activa cuando una bobina se encuentra dentro de un área MCR activa. Si el MCR está conectado y la corriente fluye a una bobina, el bit direccionado se pone a "0". Si el MCR está desconectado el estado de señal del operando indicado del elemento no varía, independientemente del estado del flujo de corriente.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	-	0

Ejemplo



La salida A 4.0 sólo se pone a "0" si:

(el estado en la entrada E 0.0 **Y** en la entrada E 0.1 es "1") **O** el estado en la entrada E 0.2 es "0".

El temporizador T1 sólo sE pone a 0 si:

el estado de señal en la entrada E 0.3 es "1".

El contador Z1 sólo se pone a 0 si:

el estado de señal en la entrada E 0.3 es "1".

Si el circuito del ejemplo se encuentra en un área MCR:

Al estar conectado el MCR, A 4.0, T1 y SZ1 se ponen a 0, tal como se ha descrito más arriba.

Si el MCR está desconectado, A 4.0, T1 y Z1 no se modifican, independientemente del estado de señal del RLO (estado de señal del flujo de corriente).

1.9 ---(S) Activar salida

Símbolo

<Operando>

---(S)

Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando>	BOOL	E, A, M, L, D	Bit activado

Descripción de la operación

---(S) (Activar bobina) sólo se ejecuta si el RLO de las operaciones anteriores es "1" (flujo de corriente en la bobina). Si el RLO es "1", el **<operando>** indicado del elemento se pone a "1".

Un RLO = 0 no tiene efecto alguno, de forma que el estado de señal actual del operando indicado del elemento no se altera.

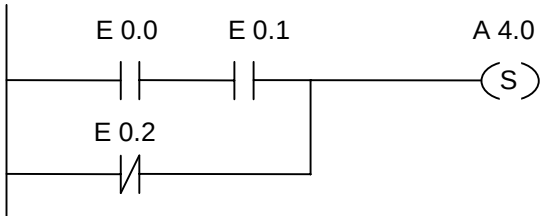
Dependencia con respecto al MCR (Master Control Relay)

La dependencia con respecto al MCR solamente se activa cuando una bobina se encuentra dentro de un área MCR activa. Si el MCR está conectado y la corriente fluye a una bobina, el bit direccionado toma el estado de señal actual del flujo de corriente. Si el MCR está desconectado se escribe un "0" en el operando indicado del elemento, independientemente del estado del flujo de corriente.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	-	0

Ejemplo



La salida A 4.0 sólo se pone a "1" si:

(el estado en la entrada E 0.0 **Y** en E 0.1 es "1") **O** el estado en la entrada E 0.2 es "1".

Si el RLO es "0", el estado de señal de la salida A 4.0 no varía.

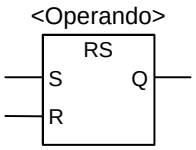
Si el circuito del ejemplo se encuentra en un área MCR:

Al estar conectado el MCR, la salida A 4.0 se pone a 1, tal como se ha descrito más arriba.

Si el MCR está desconectado, la salida A 4.0 no se modifica, independientemente del estado de señal del RLO (estado de señal del flujo de corriente).

1.10 RS Activar flip-flop de desactivación

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando>	BOOL	E, A, M, L, D	Bit activado o desactivado
S	BOOL	E, A, M, L, D	Activación habilitada
R	BOOL	E, A, M, L, D	Desactivación habilitada
Q	BOOL	E, A, M, L, D	Estado de señal de <operando>

Descripción de la operación

RS (Activar flip-flop de desactivación) se desactiva si el estado en la entrada R es "1" y si el estado en la entrada S es "0". De no ser así, cuando el estado en la entrada R es "0" y el estado en la entrada S es "1", se activa el flip-flop. Si el RLO es "1" en ambas entradas, la operación Desactivar flip-flop de activación ejecuta en el **<operando>** indicado primero la operación Desactivar y seguidamente la operación Activar, de modo que la dirección permanece activada para el resto del ciclo de programa.

Las operaciones S (Activar) y R (Desactivar) sólo se ejecutan si el RLO es 1. Si el RLO es 0 estas operaciones no se ven afectadas y el operando indicado no varía.

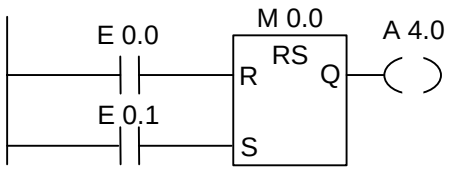
Dependencia con respecto al MCR (Master Control Relay)

La dependencia con respecto al MCR solamente se activa si la operación Activar flip-flop de desactivación se encuentra dentro de un área MCR activa. Si el MCR está conectado, el bit direccionado se pone a "1" (se activa) ó a "0" (se desactiva), tal como se ha descrito más arriba. Si el MCR está desconectado, el estado actual del operando indicado no se altera, independientemente de cuál sea es estado de las entradas.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo



Si el estado en la entrada E 0.0 es "1" y en la entrada E 0.1 es "0", se activa la marca M 0.0 y la salida A 4.0 es "0". De no ser así, cuando el estado de señal en la entrada E 0.0 es 0 y en E 0.1 es 1, se activa la marca M 0.0 y la salida A 4.0 es "1". Si ambos estados de señal son "0", no cambia nada. Si ambos estados de señal son "1" domina la operación Activar, debido al orden en que están dipuestas las operaciones. M 0.0 se activa y la salida A 4.0 es "1".

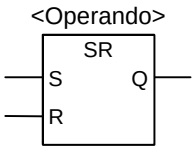
Si el esquema del ejemplo anterior se encuentra dentro de un área MCR activa

Cuando el MCR está conectado, la salida A 4.0 se pone a 1 ó a 0, tal como se ha descrito arriba.

Si el MCR está desconectado, la salida A 4.0 no se modifica, independientemente cuál sea el estado de señal de las entradas.

1.11 SR Desactivar flip-flop de activación

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando>	BOOL	E, A, M, L, D	Bit activado o desactivado
S	BOOL	E, A, M, L, D	Activación habilitada
R	BOOL	E, A, M, L, D	Desactivación habilitada
Q	BOOL	E, A, M, L, D	Estado de señal de <operando>

Descripción de la operación

SR (Desactivar flip-flop de activación) se activa si el estado en la entrada S es "1" y si el estado de la entrada R es "0". De no ser así, cuando el estado en la entrada S es "0" y el estado de la entrada R es "1", se desactiva el flip-flop. Si el RLO es "1" en ambas entradas, la operación Desactivar flip-flop de activación ejecuta en el <**operando**> indicado primero la operación Activar y seguidamente la operación Desactivar, de modo que la dirección permanece desactivada para el resto del ciclo de programa.

Las operaciones S (Activar) y R (Desactivar) sólo se ejecutan si el RLO es 1. Si el RLO es 0, estas operaciones no se ven afectadas y el operando indicado no varía.

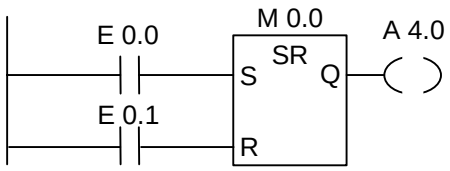
Dependencia con respecto al MCR (Master Control Relay)

La dependencia con respecto al MCR solamente se activa si la operación Desactivar flip-flop de activación se encuentra dentro de un área MCR activa. Si el MCR está conectado, el bit direccionado se pone a "1" (se activa) ó a "0" (se desactiva), tal como se ha descrito más arriba. Si el MCR está desconectado, el estado actual del operando indicado no se altera, independientemente de cuál sea es estado de las entradas.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo



Si el estado en la entrada E 0.0 es "1" y en la entrada E 0.1 es el estado es "0", se activa la marca M 0.0, y la salida A 4.0 es "1". De no ser así, cuando el estado de señal en la entrada E 0.0 es 0 y en E 0.1 es 1, se desactiva la marca M 0.0 y la salida A 4.0 es "0". Si ambos estados de señal son "0", no cambia nada. Si ambos estados de señal son "1", domina la operación Desactivar debido al orden en que están dispuestas las operaciones. M 0.0 se desactiva y la salida A 4.0 es "0".

Si el esquema del ejemplo anterior se encuentra dentro de un área MCR activa:

Cuando el MCR está conectado, A 4.0 se pone a 1 ó a 0, tal como se ha descrito más arriba.

Si el MCR está desconectado, A4.0 no varía, independientemente del estado de señal de las entradas.

1.12 ---(N)--- Detectar flanco decreciente (1 --> 0)

Símbolo

<Operando>

---(N)---

Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando>	BOOL	A, M, D	Marca de flancos que almacena el estado de señal anterior del RLO

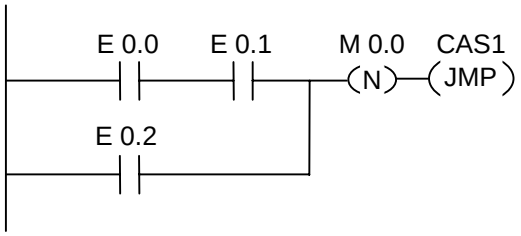
Descripción de la operación

---(N)--- (Detectar flanco decreciente (1 --> 0)) detecta un cambio del estado de señal en el operando de "1" a "0", e indica este cambio tras la operación con RLO = 1. El estado de señal del RLO se compara con el estado de señal del operando, es decir, con la marca de flancos. Si el estado de señal del operando es "1" y el RLO anterior a la operación es "0", el RLO posterior a la operación será "1" (impulso); en todos los otros casos será "0". El RLO anterior a la operación se almacena en el operando.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	x	1

Ejemplo



La marca de flancos M 0.0 almacena el estado de señal del RLO de la combinación de bits en su conjunto. Si el estado de señal del RLO cambia de "1" a "0" se ejecuta el salto a la meta CAS1.

1.13 ---(P)--- Detectar flanco creciente RLO (0 --> 1)

Símbolo

<Operando>

---(P)---

Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando>	BOOL	E, A, M, L, D	Marca de flancos que almacena el estado de señal anterior del RLO

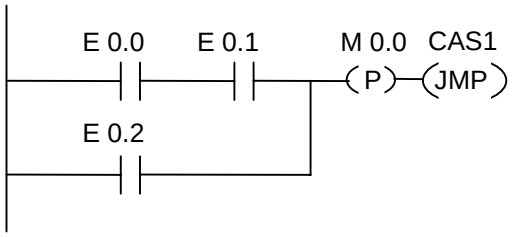
Descripción de la operación

---(P)--- (Detectar flanco creciente RLO (0 --> 1)) detecta un cambio del estado de señal en el operando, de "0" a "1", e indica este cambio tras la operación mediante RLO = 1. El estado de señal actual del RLO se compara con el estado de señal del operando, es decir, con la marca de flancos. Si el estado de señal del operando es "0" y el RLO anterior a la operación es "1", el RLO detrás de la operación será "1" (impulso); en todos los demás casos será "0". El RLO anterior a la operación se almacena en el operando.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	x	x	1

Ejemplo



La marca de flancos M 0.0 almacena el estado del RLO de toda la combinación de bits. Si el estado de señal del RLO cambia de "0" a "1", se ejecuta el salto a la meta CAS1.

1.14 ---(SAVE) Cargar resultado lógico (RLO) en el registro RB

Símbolo

---(SAVE)

Descripción de la operación

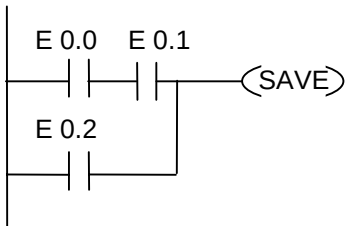
---(SAVE) (Cargar resultado lógico (RLO) en registro RB) almacena el RLO en el bit del resultado binario (RB) de la palabra de estado. Pero el bit de primera consulta /ER no se pone a cero.
Por esta razón, en una combinación lógica Y en el próximo segmento se combinará el estado del bit RB.

El uso de SAVE con una consulta del bit RB en el mismo bloque o en bloques subordinados no es recomendable, puesto que el bit RB puede ser modificado por numerosas operaciones intercaladas. La operación SAVE resulta especialmente útil antes de salir de un bloque, puesto que con ella la salida ENO (bit RB) se pone al valor del bit RLO, lo cual permite añadir un tratamiento de error a continuación del bloque.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	x	-	-	-	-	-	-	-	-

Ejemplo

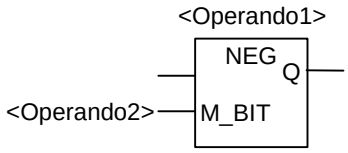


El estado del segmento (= RLO) se almacena en el bit RB.

RB Bit de resultado binario (Palabra de estado, bit 8)

1.15 NEG Detectar flanco de señal negativo (1 --> 0)

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando1>	BOOL	E, A, M, L, D	Señal consultada
<Operando2>	BOOL	A, M, D	Marca de flancos M_BIT; almacena el estado de señal anterior de <Operando1>
Q	BOOL	E, A, M, L, D	Detección de cambio de señal

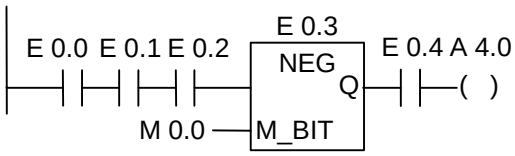
Descripción de la operación

NEG (Detectar flanco de señal (1 --> 0)) compara el estado de señal de **<Operando1>** con el estado de señal de la consulta anterior, que esta almacenada en el **<Operando2>** . Si el estado actual del RLO es "0" y el estado anterior era "1" (detección de un flanco decreciente), la salida Q después de esta función es "1", en todos los otros casos es "0".

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	1	x	1

Ejemplo

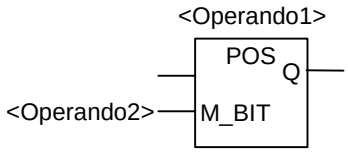


La salida A 4.0 es "1", si:

(el estado en E 0.0 **Y** en E 0.1 **Y** en E 0.2 es "1") **Y** E 0.3 tiene un flanco decreciente **Y** el estado en E 0.4 es "1".

1.16 POS Detectar flanco de señal positivo (0 --> 1)

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
<Operando1>	BOOL	E, A, M, L, D	Señal consultada
<Operando2>	BOOL	A, M, D	Marca de flancos M_BIT, almacena el estado de señal anterior de <Operando1>
Q	BOOL	E, A, M, L, D	Detección del cambio de señal

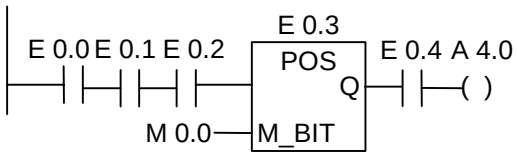
Descripción de la operación

POS (Detectar flanco de señal 0 --> 1) compara el estado de señal de **<Operando1>** con el estado de señal de la consulta anterior que está almacenado en **<Operando2>**. Si el estado actual del RLO es "1" y el estado anterior era "0" (Detección de un flanco creciente), la salida Q después de esta operación es "1"; en todos los otros casos es "0".

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	1	x	1

Ejemplo



La salida A 4.0 es "1", si:

(el estado en E 0.0 **Y** en E 0.1 **Y** en E 0.2 es "1") **Y** E 0.3 tiene un flanco creciente **Y** el estado en E 0.4 es "1".

1.17 Leer directamente de periferia

Descripción de la operación

Para la función **Leer directamente** de periferia hay que crear un segmento (tal como se representa abajo).

Puede suceder que en aplicaciones controladas por tiempo haya que leer el estado actual de una entrada digital con mayor frecuencia de lo normal (una vez por ciclo). La operación Leer directamente de periferia recibe el estado de la entrada digital inmediatamente desde el módulo de entrada en el instante en que se lee el circuito correspondiente. Si no se aplica esta función habrá que esperar hasta que finalice el ciclo principal (un ciclo completo de OB1), es decir, hasta que el área de memoria de las entradas haya sido actualizado con el estado del área de memoria de la periferia.

Si desea leer una o varias entradas directamente desde el módulo de entrada, utilice el área de memoria Entradas de Periferia (PE) en vez del área de memoria de las entradas (E). El área de memoria de la periferia se puede leer en formato de byte, palabra o doble palabra. Debido a esto no se puede leer una sola entrada digital a través de un contacto (bit).

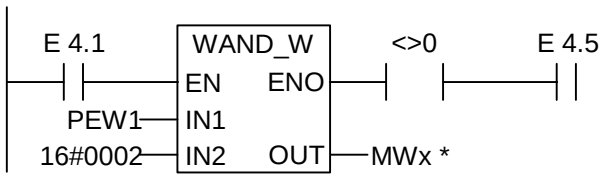
Transmisión condicional de tensión en función del estado de una entrada directa:

La CPU lee la palabra del área de memoria PE que contiene los datos relevantes.

1. La palabra del área de memoria PE se combina mediante una Y lógica con una constante que permite un resultado diferente de cero para el caso de que el bit de entrada esté activado ("1").
2. Se verifica que la condición ”diferente de cero” se cumple.

Ejemplo

Segmento KOP con la operación **Leer directamente** de periferia para la entrada E 1.1.



* Debe indicarse Mwx para poder almacenar el segmento. ”x” puede ser cualquier número permitido.

Descripción de la operación WAND_W:

PEW1 0000000000101010

W#16#0002 0000000000000010

Resultado 0000000000000010

En este ejemplo la entrada directa E 1.1 está conectada en serie con las entradas E 4.1 y E 4.5.

La palabra PEW1 contiene el estado directo de E 1.1. PEW1 se lógicamente con W#16#0002 mediante una Y lógica. El resultado es diferente de cero si E 1.1 (segundo bit)

es verdadero ("1") en PB1. El contacto A<>0 transmite la tensión si el resultado de la operación WAND_W es diferente de cero.

1.18 Escribir directamente en periferia

Descripción de la operación

Para aplicar la función **Escribir directamente** en periferia hay que crear un segmento (tal como se representa más abajo).

Puede suceder que en aplicaciones controladas por tiempo haya que transmitir el estado actual de una salida digital a un módulo de salida con mayor frecuencia de lo normal (una vez al finalizar el ciclo de OB1). La operación Escribir directamente en periferia actualiza el estado de una salida digital en el módulo de salida en el instante en que se escribe el circuito correspondiente. Si no se aplica esta función habrá que esperar hasta que finalice el ciclo principal (un ciclo completo de OB1), es decir, hasta que el área de memoria de la periferia haya sido actualizado con el estado del área de memoria de las salidas.

Si desea actualizar una o varias salidas directamente, utilice el área de memoria Salidas de Periferia (PA) en vez del área de memoria de las salidas (A). El área de memoria de las Salidas de Periferia se escribe en formato de byte, palabra o doble palabra. Por ello, no se puede actualizar una sola salida digital a través de una bobina (bit). Para escribir directamente el estado de una salida digital en un módulo de salida se busca el byte, la palabra o la doble palabra del área de memoria de las salidas A que contenga el bit en cuestión, y se copia en la memoria PA correspondiente (en los operandos del módulo de salidas).

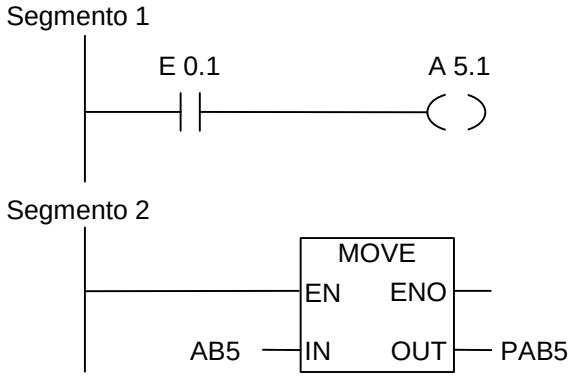
Advertencias

- Puesto que en el módulo de salidas se escribe el byte completo del área de memoria A, cuando se ejecuta la operación también cambian todos los bits de salida del byte que se actualiza.
- Si un bit de salida tiene estados intermedios (1/0) durante el programa que no deben transmitirse a los módulos de salida, la operación Escribir directamente en periferia puede originar estados que son peligrosos (impulsos de transición a las salidas).
- Una regla de aplicación general en cuanto a la configuración es que, en un programa, un módulo de salida externo sólo puede direccionarse una única vez como bobina. Cumpliendo esta regla se evitará la mayoría de problemas que pudiera provocar la operación Escribir directamente en periferia.

Ejemplo

Segmento KOP con la operación **Escribir directamente** en periferia y con el módulo de salidas digitales 5, canal 1.

Los estados de los bits pertenecientes al byte de salida direccionado (AB5), o bien se actualizan, o bien no cambian. A la salida A5 se le asigna el estado de señal de E 0.1. AB5 se copia en el área de memoria directa correspondiente de las Salidas de Periferia (PAB5).



En este ejemplo, A 5.1 es el bit de salida solicitado.

El byte PAB5 contiene el estado del bit de salida A 5.1.

Los demás bits del byte de salida PAB5 también se actualizan al copiar usando la operación **MOVE**.

2 Operaciones de comparación

2.1 Lista de operaciones de comparación

Descripción

Las operaciones comparan las entradas IN1 e IN2 según los tipos de comparación siguientes:

== IN1 es igual A IN2
<> IN1 es diferente A IN2
> IN1 es mayor quE IN2
< IN1 es menor quE IN2
>= IN1 es mayor o igual A IN2
<= IN1 es menor o igual A IN2

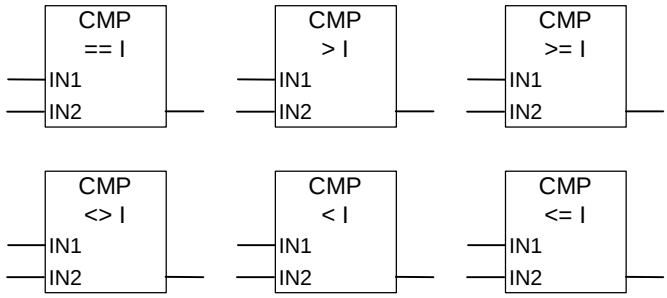
Si la comparación es verdadera, el RLO de la operación es "1". El RLO se combina mediante una Y lógica con el RLO del circuito completo siempre que el elemento de comparación esté conectado en serie, y mediante una O lógica si el cuadro está conectado en paralelo.

Se dispone de las operaciones de comparación siguientes:

- CMP ? I Comparar enteros (16 Bit)
- CMP ? D Comparar enteros dobles (32 Bit)
- CMP ? R Comparar números en coma flotante

2.2 CMP ? I Comparar enteros

Símbolos



Parámetro	Tipo de datos	Area de memoria	Descripción
Entrada de cuadro	BOOL	E, A, M, L, D	Resultado de la última combinación
Salida de cuadro	BOOL	E, A, M, L, D	Resultado de la comparación; sólo se continuará a procesar si RLO en la entrada de cuadro = 1.
IN1	INT	E, A, M, L, D o constante	Primer valor a comparar
IN2	INT	E, A, M, L, D o constante	Segundo valor a comparar

Descripción de la operación

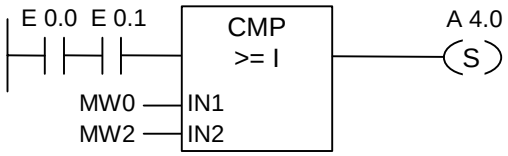
CMP ? I (Comparar enteros) puede utilizarse como un contacto normal. El cuadro puede colocarse en las mismas posiciones que puede tomar un contacto normal. Las entradas IN1 y IN2 son comparadas atendiendo al criterio de comparación que se haya seleccionado.

Si la comparación es verdadera, el RLO de la operación es "1". El RLO se combina mediante una Y lógica con el RLO del circuito completo siempre que el elemento de comparación esté conectado en serie, y mediante una O lógica si el cuadro está conectado en paralelo.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	0	-	0	x	x	1

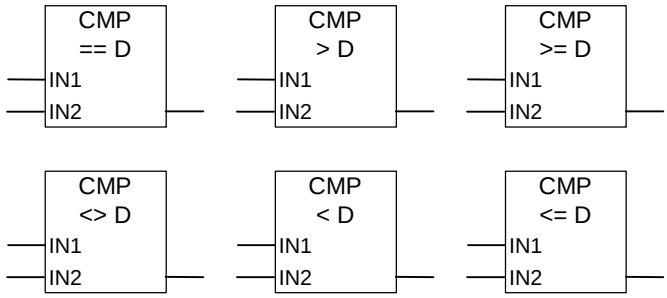
Ejemplo



La salida A 4.0 se activa si E 0.0 **Y** E 0.1 son 1 **Y** si MW0 >= MW2.

2.3 CMP ? D Comparar enteros dobles

Símbolos



Parámetro	Tipo de datos	Area de memoria	Descripción
Entrada de cuadro	BOOL	E, A, M, L, D	Resultado de la última combinación
Salida de cuadro	BOOL	E, A, M, L, D	Resultado de la comparación; sólo se continuará a procesar si RLO en la entrada de cuadro = 1.
IN1	DINT	E, A, M, L, D o constante	Primer valor a comparar
IN2	DINT	E, A, M, L, D o constante	Segundo valor a comparar

Descripción de la operación

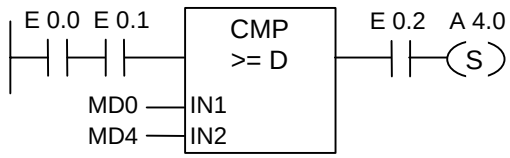
CMP ? D (Comparar enteros dobles) puede utilizarse como un contacto normal. El cuadro puede colocarse en las mismas posiciones que puede tener un contacto normal. Las entradas IN1 y IN2 son comparadas atendiendo al criterio de comparación que se haya seleccionado.

Si la comparación es verdadera, el RLO de la operación es "1". El RLO se combina mediante una Y lógica con el RLO de un circuito siempre que el elemento de comparación esté conectado en serie, y mediante una O lógica si el cuadro está conectado en paralelo.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	0	-	0	x	x	1

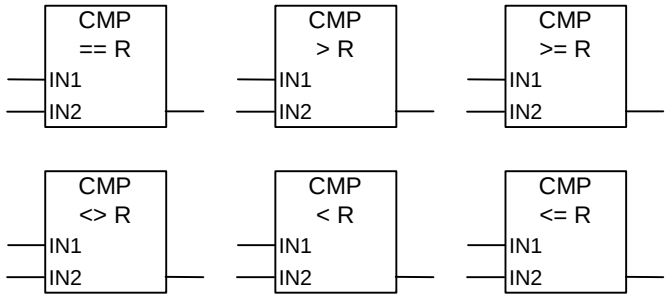
Ejemplo



La salida A 4.0 se activa si E 0.0 **Y** E 0.1 tienen el estado de señal 1 **Y** si MD0 >= MD4 **Y** si E 0.2 tiene el estado de señal 1.

2.4 CMP ? R Comparar números de coma flotante

Símbolos



Parámetro	Tipo de datos	Area de memoria	Descripción
Entrada de cuadro	BOOL	E, A, M, L, D	Resultado de la última combinación
Salida de cuadro	BOOL	E, A, M, L, D	Resultado de la comparación; sólo se continuará a procesar si RLO en la entrada de cuadro = 1.
IN1	REAL	E, A, M, L, D o constante	Primer valor a comparar
IN2	REAL	E, A, M, L, D o constante	Segundo valor a comparar

Descripción de la operación

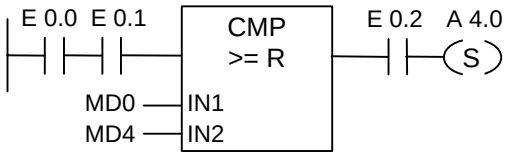
CMP ? R (Comparar números en coma flotante) puede utilizarse como un contacto normal. El cuadro puede colocarse en las mismas posiciones que puede tomar un contacto normal. Las entradas IN1 y IN2 son comparadas atendiendo al criterio de comparación que se haya seleccionado.

Si la comparación es verdadera, el RLO de la operación es "1". El RLO se combina mediante una Y lógica con el RLO del circuito completo siempre que el elemento de comparación esté conectado en serie, y mediante una O lógica si el cuadro está conectado en paralelo.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

Ejemplo



La salida A 4.0 se activa si E 0.0 **Y** E 0.1 son 1 **Y** si MD0 >= MD4 **Y** si E 0.2 es 1.

3 Operaciones de conversión

3.1 Lista de operaciones de conversión

Descripción

Las operaciones de conversión leen el contenido del parámetro IN y lo convierten o le cambian el signo. El resultado se puede recoger en el parámetro OUT.

Se dispone de las operaciones de conversión siguientes:

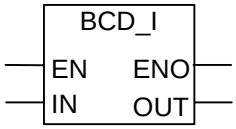
- BCD_I Convertir BCD en entero
- I_BCD Convertir entero en BCD
- BCD_DI BCD-Zahl in 32-Bit-Ganzzahl wandeln
- I_DI Convertir entero en entero doble
- DI_BCD Convertir entero doble en BCD
- DI_R Convertir entero doble en real

- INV_I Complemento a 1 de un entero
- INV_DI Complemento a 1 de un entero doble
- NEG_I Complemento a 2 de un entero
- NEG_DI Complemento a 2 de un entero doble

- NEG_R Invertir el signo de un número real
- ROUND Redondear a entero
- TRUNC Truncar entero
- CEIL Redondear número real a entero superior
- FLOOR Redondear número real a entero inferior

3.2 BCD_I Convertir BCD en entero

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	WORD	E, A, M, L, D	Número BCD
OUT	INT	E, A, M, L, D	Valor entero (16 bits) de un número BCD

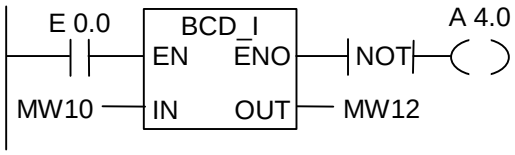
Descripción de la operación

BCD_I (Convertir BCD en entero) lee el contenido del parámetro IN como número en formato decimal codificado en binario de tres dígitos (+/- 999) y convierte este número en un valor entero (de 16 bits). El parámetro OUT contiene el resultado en formato de número entero. ENO siempre tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	-	-	-	-	0	1	1	1

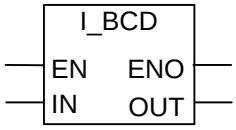
Ejemplo



Si la entrada E 0.0 es 1, el contenido de MW10 se lee como número de tres dígitos en formato decimal codificado en binario y se convierte en número entero (de 16 bits). El resultado se deposita en MW12. La salida A 4.0 será "1" si no se lleva a cabo la conversión (ENO = EN = 0).

3.3 I_BCD Convertir entero en BCD

Símbolo



Formato

Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	INT	E, A, M, L, D	Entero (de 16 bits)
OUT	WORD	E, A, M, L, D	Valor BCD del entero (16 bits)

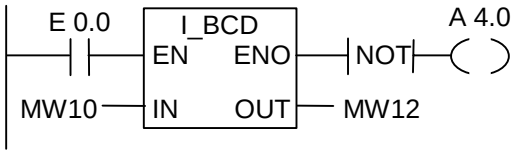
Descripción de la operación

I_BCD (Convertir entero en BCD) lee el contenido del parámetro IN como valor entero (16 bits) y convierte este valor en un número de tres dígitos en formato decimal codificado en binario (+/- 999). El parámetro OUT contiene el resultado. Si se produce un desbordamiento, ENO = 0.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	-	-	x	x	0	x	x	1

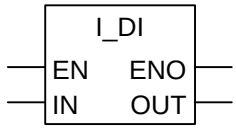
Ejemplo



Si la entrada E 0.0 es 1, el contenido de MW10 se lee como número entero (16 bits) y se convierte en un número de tres dígitos en formato decimal codificado en binario. El resultado se deposita en MW12. La salida A 4.0 será "1" si se produce un desbordamiento o si no se procesa la instrucción (E0.0 = 0).

3.4 I_DI Convertir entero en entero doble

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	INT	E, A, M, L, D	Valor entero (de 16 bits) a convertir
OUT	DINT	E, A, M, L, D	Resultado: entero doble (de 32 bits)

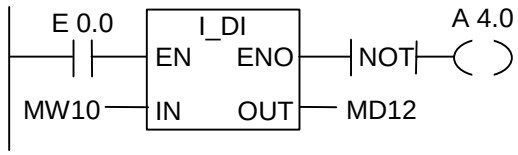
Descripción de la operación

I_DI (Convertir entero en entero doble) lee el contenido del parámetro IN como entero (16 bits) y convierte este número en entero doble (32 bits). El parámetro OUT contiene el resultado. ENO siempre tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
escribe:	1	-	-	-	-	0	1	1	1

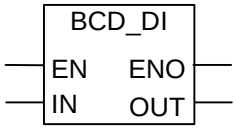
Ejemplo



Si la entrada E 0.0 es 1, el contenido de MW10 se lee como entero (de 16 bits) y se convierte en un entero doble (de 32 bits). El resultado se deposita en MD12. La salida A 4.0 será "1" si no se ejecuta la conversión (ENO = EN = 0).

3.5 BCD_DI Convertir BCD en entero doble

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	DWORD	E, A, M, L, D	Número BCD
OUT	DINT	E, A, M, L, D	Valor entero (de 32 bits) del número BCD

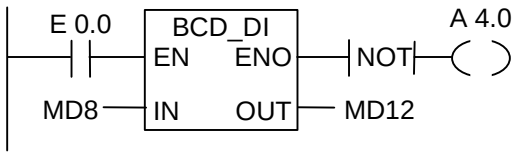
Descripción de la operación

BCD_DI (Convertir BCD en entero doble) lee el contenido del parámetro IN como número en formato decimal codificado en binario de siete dígitos (+/- 9999999), y convierte este número en un valor entero (de 32 bits). El parámetro OUT contiene el resultado en forma de número entero. ENO siempre tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	-	-	-	-	0	1	1	1

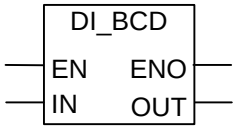
Ejemplo



Si la entrada E 0.0 es 1, el contenido de MD8 se lee como número de siete dígitos en formato decimal codificado en binario, y se convierte en número entero (de 32 bits). El resultado se deposita en MD12. La salida A 4.0 será "1" si no se lleva a cabo la conversión (ENO = EN = 0).

3.6 DI_BCD Convertir entero doble en BCD

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	DINT	E, A, M, L, D	Entero (de 32 bits)
OUT	DWORD	E, A, M, L, D	Valor BCD del entero (32 bits)

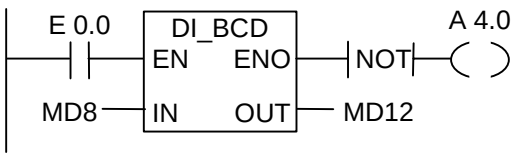
Descripción de la operación

DI_BCD (Convertir entero doble en BCD) lee el contenido del parámetro IN como valor entero (de 32 bits) y convierte este número en un número de siete dígitos en formato decimal codificado en binario (+/- 9999999). El parámetro OUT contiene el resultado. Si se produce un desbordamiento, ENO = 0.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	-	-	x	x	0	x	x	1

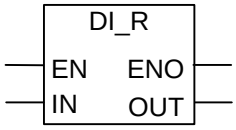
Ejemplo



Si la entrada E 0.0 es 1, el contenido de MD8 se lee como número entero (32 bits) y se convierte en un número de siete dígitos en formato decimal codificado en binario. El resultado se deposita en MD12. La salida A 4.0 será "1" si se produce un desbordamiento o en caso de que no se procese la instrucción (E0.0 = 0).

3.7 DI_R Convertir entero doble en real

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	DINT	E, A, M, L, D	Entero doble
OUT	REAL	E, A, M, L, D	Número real

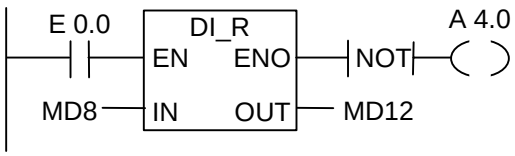
Descripción de la operación

DI_R (Convertir entero doble en real) lee el contenido del parámetro IN como valor entero (de 32 bits) y convierte este valor en número real. El parámetro OUT contiene el resultado. ENO siempre tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	-	-	-	-	0	1	1	1

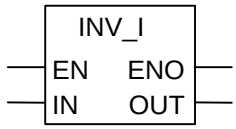
Ejemplo



Si la entrada E 0.0 es 1, el contenido de MD8 se lee como número entero doble y se convierte en un número real. El resultado se deposita en MD12. La salida A 4.0 será "1" si no lleva a cabo la conversión (ENO = EN = 0).

3.8 INV_I Complemento a 1 de un entero

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	INT	E, A, M, L, D	Valor entero (de 16 bits) de entrada
OUT	INT	E, A, M, L, D	Complemento a 1 del entero (de 16 bits) de IN

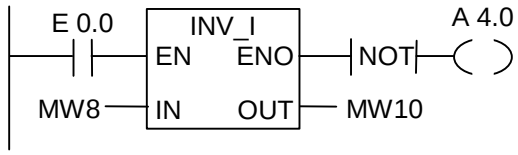
Descripción de la operación

INV_I (Complemento a 1 de un entero) lee el contenido del parámetro IN y combina el valor con la plantilla hexadecimal W#16#FFFF mediante una operación lógica O-EXCLUSIVA. Esta operación invierte el estado de cada bit. La salida de habilitación ENO siempre tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	-	-	-	-	0	1	1	1

Ejemplo



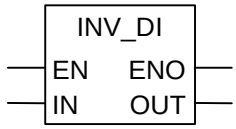
Si la entrada E 0.0 es 1 se invierte el estado de cada bit de MW8.

Por ejemplo:

MW8 = 00000000 00000000 se convierte en MW10 = 11111111 11111111. La salida A 4.0 será "1" si no se ejecuta la conversión (ENO = EN = 0).

3.9 INV_DI Complemento a 1 de un entero doble

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	DINT	E, A, M, L, D	Valor entero doble de entrada
OUT	DINT	E, A, M, L, D	Complemento a 1 del entero doble de IN

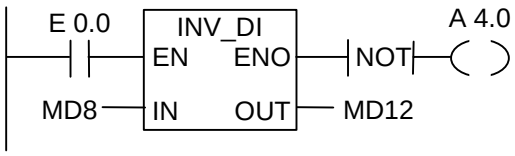
Descripción de la operación

INV_DI (Complemento a 1 de un entero doble) lee el contenido del parámetro IN y combina el valor con la plantilla hexadecimal W#16#FFFF FFFF mediante una operación lógica O-EXCLUSIVA. Esta operación invierte el estado de cada bit. La salida de habilitación ENO siempre tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	-	-	-	-	0	1	1	1

Ejemplo



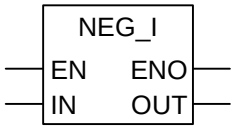
Si la entrada E 0.0 es 1 se invierte el estado de cada bit de MD8.

Por ejemplo:

MD8 = F0FF FFF0 se convierte en MD12 = 0F00 000F. La salida A 4.0 será "1" si no se ejecuta la conversión (ENO = EN = 0).

3.10 NEG_I Complemento a 2 de un entero

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	INT	E, A, M, L, D	Valor entero de entrada
OUT	INT	E, A, M, L, D	Complemento a 2 del entero de IN

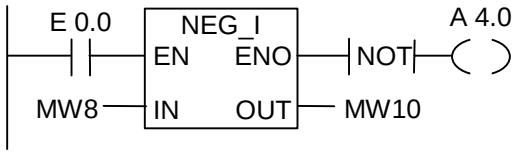
Descripción de la operación

NEG_I (Complemento a 2 de un entero) lee el contenido del parámetro IN y ejecuta la operación Complemento a 2. La operación invierte el signo (ejemplo: de un valor positivo a un valor negativo). La salida de habilitación ENO siempre tiene el mismo estado de señal que EN, exceptuando el siguiente caso: Si el estado de señal de EN es 1 y se produce un desbordamiento, el estado de señal de ENO será 0.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

Ejemplo



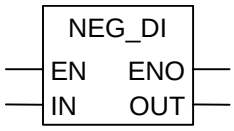
Si la entrada E 0.0 es 1, el parámetro OUT deposita el valor de MW 8 con el signo opuesto en MW10.

De MW8 = + 10 resulta MW10 = - 10. La salida A 4.0 será "1" si no se ejecuta la conversión (ENO = EN = 0).

Si el estado de señal de EN es 1 y se produce un desbordamiento, el estado de señal de ENO será 0.

3.11 NEG_DI Complemento a 2 de un entero doble

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	DINT	E, A, M, L, D	Valor entero doble de entrada
OUT	DINT	E, A, M, L, D	Complemento a 2 del entero doble de IN

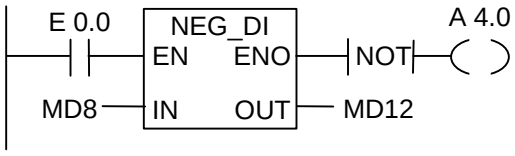
Descripción de la operación

NEG_DI (Complemento a 2 de un entero doble) lee el contenido del parámetro IN y ejecuta la operación Complemento a 2. La operación invierte el signo (ejemplo: de un valor positivo a un valor negativo). La salida de habilitación ENO siempre tiene el mismo estado de señal que EN, exceptuando el siguiente caso: si el estado de señal de EN es 1 y se produce un desbordamiento, el estado de señal de ENO será 0.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

Ejemplo



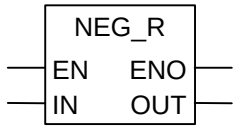
Si la entrada E 0.0 es 1, el parámetro OUT deposita en MD12 el valor de MD 8 con el signo opuesto.

De MD8 = + 1000 resulta MD12 = - 1000. La salida A 4.0 será "1" si no se ejecuta la conversión (ENO = EN = 0).

Si el estado de señal de EN es 1 y se produce un desbordamiento, el estado de señal de ENO será 0.

3.12 NEG_R Invertir signo de un número real

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: número real
OUT	REAL	E, A, M, L, D	Número real IN con signo invertido

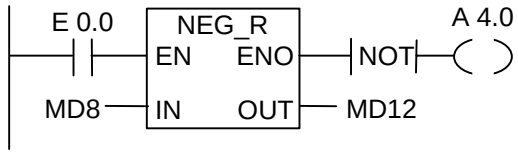
Descripción de la operación

NEG_R (Invertir signo de un número real) lee el contenido del parámetro IN e invierte su signo. Esta operación equivale a una multiplicación por (-1). La operación invierte el signo (ejemplo: de un valor positivo a un valo negativo). La salida de habilitación ENO siempre tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	-	-	-	-	0	x	x	1

Ejemplo

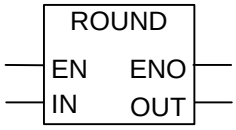


Si la entrada E 0.0 es 1, el parámetro OUT deposita en MD12 el valor de MD8 con el signo opuesto.

De MD8 = + 6,234 resulta MD12 = - 6,234. La salida A 4.0 será "1" si no se ejecuta la conversión (ENO = EN = 0).

3.13 ROUND Redondear a entero doble

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor a redondear
OUT	DINT	E, A, M, L, D	IN, redondeado al próximo entero

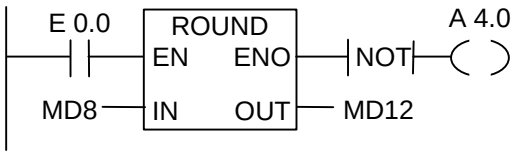
Descripción de la operación

ROUND (Redondear a entero doble) lee el contenido del parámetro IN como número real y convierte este número en un entero de 32 bits. El resultado es el número entero más próximo ("redondeo por arriba/abajo"). Si el número real se encuentra justo en el medio de dos números enteros se proporciona el número par. El resultado se deposita en el parámetro OUT. Si se produce un desbordamiento, ENO es 0.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	-	-	x	x	0	x	x	1

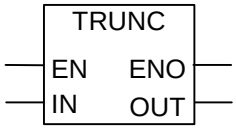
Ejemplo



Si la entrada E 0.0 es 1, el contenido de MD8 se lee como número real y se convierte en el número entero (32 bits) más próximo. El resultado de esta función "Redondear" se deposita en MD12. La salida A 4.0 será "1" si se produce un desbordamiento o en caso de que no se procese la instrucción (E.0=0).

3.14 TRUNC Truncar a entero doble

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Número real a convertir
OUT	DINT	E, A, M, L, D	Parte entera del valor de IN

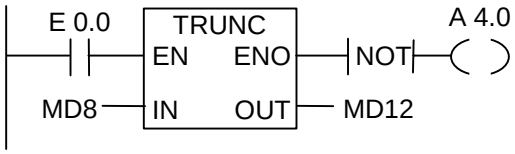
Descripción de la operación

TRUNC (Truncar a entero doble) lee el contenido del parámetro IN como número real y convierte este valor en un entero (de 32 bits). El resultado es la parte entera del número real, proporcionado por el parámetro OUT. Si se produce un desbordamiento, ENO = 0.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	-	-	x	x	0	x	x	1

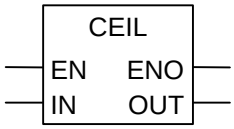
Ejemplo



Si la entrada E 0.0 es 1, el contenido de MD8 se lee como número real y se convierte en entero doble. El resultado es el componente entero del número real que se almacena en MD12. La salida A 4.0 será "1" si se produce un desbordamiento o en el caso de que no se procese la instrucción (E.0 = 0).

3.15 CEIL Redondear número real a entero doble superior

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Número real a convertir
OUT	DINT	E, A, M, L, D	Primer entero doble que es mayor que el número real

Descripción de la operación

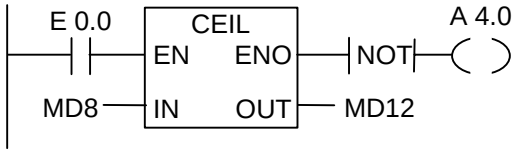
CEIL (Convertir número real en el entero doble más próximo) lee el contenido del parámetro IN como número real y convierte este número en entero doble (de 32 bits). El resultado es el primer entero que es mayor que el número real ("redondeo"). Si se produce un desbordamiento, ENO = 0.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe*	X	-	-	X	X	0	X	X	1
se escribe**	0	-	-	-	-	0	0	0	1

* La operación se ejecuta (=> EN = 1)
** La operación no se ejecuta (=> EN = 0)

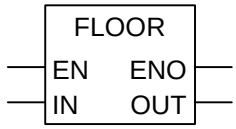
Ejemplo



Si la entrada E 0.0 es 1, el contenido de MD8 se lee como número real y éste se convierte en un entero doble aplicando además el redondeo a la siguiente cifra entera mayor que la real. El resultado se deposita en MD12. La salida A 4.0 será "1" si se produce un desbordamiento o si no se procesa la instrucción (E0.0 = 0).

3.16 FLOOR Redondear número real a entero doble inferior

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Número real a convertir
OUT	DINT	E, A, M, L, D	Primer entero doble que es menor que el número real

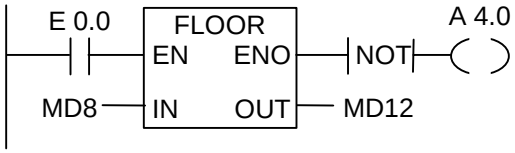
Descripción de la operación

FLOOR (Redondear número real a entero doble inferior) lee el contenido del parámetro IN como número real y convierte este número en entero (32 bits). El resultado es el primer entero doble que es menor que el número real ("redondeo"). Si se produce un desbordamiento, ENO es 0.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	-	-	x	x	0	x	x	1

Ejemplo



Si la entrada E 0.0 es 1, el contenido de MD8 se lee como número real y éste se convierte en un entero doble, aplicándose al mismo tiempo el redondeo al siguiente número entero inferior. El resultado se deposita en MD12. La salida A 4.0 será "1" si se produce un desbordamiento o si no se procesa la instrucción (E 0.0 = 0).

4 Operaciones de contaje

4.1 Lista de operaciones de contaje

Area de memoria

Los contadores tienen reservada un área de memoria en la CPU. Esta área de memoria reserva una palabra de 16 bits para cada contador. KOP asiste 256 contadores. Las operaciones de contaje son las únicas funciones que tienen acceso al área de memoria reservada para contadores.

Valor de contaje

Los bits 0 a 9 de la palabra de contaje contienen el valor de contaje en código binario. El valor fijado por el usuario se transfiere del acumulador al contador al activarse éste. El valor de contaje puede estar comprendido entre 0 y 999.

Dentro de este margen se puede variar dicho valor utilizando las operaciones siguientes:

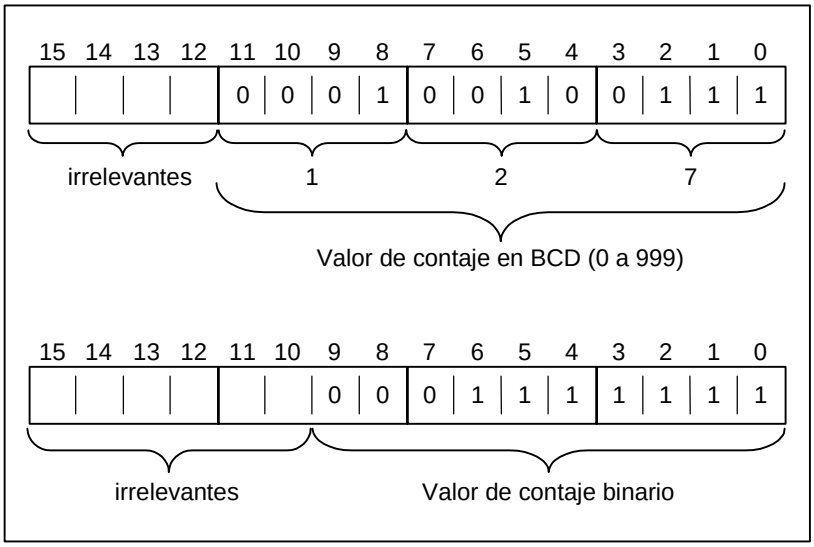
- ZAEHLER Parametrizar e incrementar/decrementar contador
- Z_VORW Parametrizar e incrementar contador
- Z_RUECK Parametrizar y decrementar contador
- ---(SZ) Poner contador al valor inicial
- ---(ZV) Incrementar contador
- ---(ZR) Decrementar contador

Configuración binaria en el contador

Para poner el contador a un valor determinado hay que introducir un número de 0 a 999, por ejemplo 127, en el siguiente formato: C# 127. C# sirve para indicar el formato decimal codificado en binario.

Los bits 0 a 11 del contador contienen el valor de contaje en formato BCD (formato BCD: cada conjunto de cuatro bits contiene el código binario de un valor decimal).

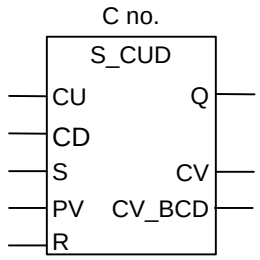
La figura muestra el contenido del contador después de haber cargado el valor de contaje 127 y el contenido de la palabra de contaje después de haber activado el contador.



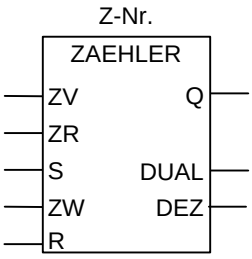
4.2 ZAEHLER Parametrizar e incrementar/decrementar contador

Símbolo

Inglés



Alemán



Parámetro Inglés	Parámetro Alemán	Tipo de datos	Area de memoria	Descripción
N.° de C	N.º de Z	COUNTER	Z	Número de identificación del contador; el área varía según CPU utilizada
CU	ZV	BOOL	E, A, M, L, D	Entrada de contaje adelante
CD	ZR	BOOL	E, A, M, L, D	Entrada de contaje atrás
S	S	BOOL	E, A, M, L, D	Entrada para predeterminar el contador
PV	ZW	WORD	E, A, M, L, D o constante	Valor numérico introducido en forma de C#<valor> en el margen comprendido entre 0 y 999
PV	ZW	WORD	E, A, M, L, D	Valor para inicializar el contador
R	R	BOOL	E, A, M, L, D	Entrada de puesta a 0
CV	DUAL	WORD	E, A, M, L, D	Valor actual del contador, número hexadecimal
CV_BCD	DEZ	WORD	E, A, M, L, D	Valor actual del contador, número BCD
Q	Q	BOOL	E, A, M, L, D	Estado del contador

Descripción de la operación

ZAEHLER (Parametrizar e incrementar/decrementar contador) queda inicializado con el valor de la entrada ZW cuando se produce un flanco ascendente en la entrada S. Si hay un 1 en la entrada R, el contador se pone a cero y el valor de contaje es 0.

El contador incrementa en "1" si el estado de señal de la entrada ZV cambia de "0" a "1" y el valor del contador era menor que "999".

El contador se decrementa en "1" si en la entrada ZR se produce un flanco ascendente y el valor del contador es mayor que "0".

Al producirse un flanco ascendente en ambas entradas de contaje se ejecutan ambas operaciones, y el valor de contaje no varía.

Si se inicializa el contador y el RLO de las entradas ZV/ZR = 1, el contador contará así en el siguiente ejemplo aunque no haya habido ningún cambio de flanco.

El estado de señal de la salida Q será "1" si el valor de contaje es mayor que cero, y será "0" si el valor de contaje es igual a cero.

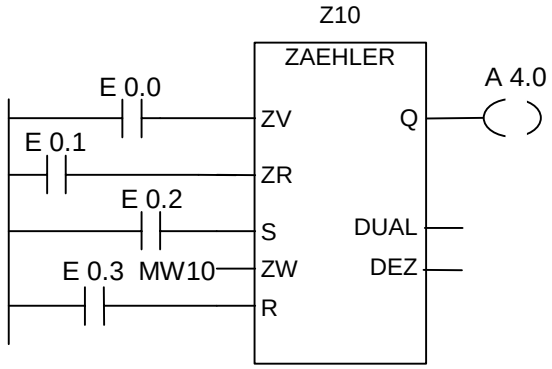
Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Nota

No utilice un mismo contador en varios puntos del programa (riesgo de errores de contaje).

Ejemplo

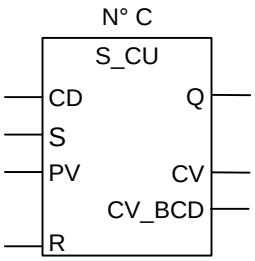


Al cambiar la entrada E 0.2 de "0" a "1", el contador toma el valor de preselección de MW10. Si el estado de señal en E 0.0 cambia de "0" a "1", el valor del contador Z10 incrementa en "1", a menos que el valor de Z10 fuera "999". Si E 0.1 cambia de "0" a "1", Z10 decrementa en "1", a no ser que el valor de Z10 fuera cero. La salida A 4.0 será "1" si el valor de Z10 no es cero.

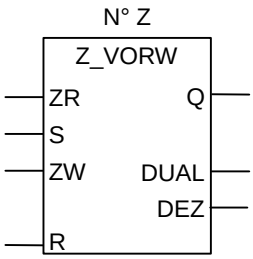
4.3 Z_VORW Parametrizar e incrementar contador

Símbolo

Inglés



Alemán



Parámetro Inglés	Parámetro Alemán	Tipo de datos	Area de memoria	Descripción
N.° de C	N.º de Z	COUNTER	Z	Número de identificación del contador, el área varía según la CPU utilizada
CU	ZV	BOOL	E, A, M, L, D	Entrada de contaje adelante
S	S	BOOL	E, A, M, L, D	Entrada para predeterminar el contador
PV	ZW	WORD	E, A, M, L, D o constante	Introducir valor numérico en forma de C#<valor> en el margen comprendido entre 0 y 999
PV	ZW	WORD	E, A, M, L, D	Valor para predeterminar el contador
R	R	BOOL	E, A, M, L, D	Entrada de puesta a 0
CV	DUAL	WORD	E, A, M, L, D	Valor actual del contador, número hexadecimal
CV_BCD	DEZ	WORD	E, A, M, L, D	Valor actual del contador, número BCD
Q	Q	BOOL	E, A, M, L, D	Estado del contador

Descripción de la operación

Z_VORW (Parametrizar e incrementar contador) toma el valor predeterminado de la entrada ZW si en la entrada S hay un flanco ascendente.

Si el estado de señal de la entrada R es "1" el contador se pone a 0, y entonces el valor de contaje es cero.

El contador incrementa en "1" si el estado de señal en la entrada ZV cambia de "0" a "1", siempre y cuando el valor de contaje sea menor que "999".

Si se inicializa el contador y el RLO de las entradas ZV/ZR = 1, el contador contará así en el siguiente ejemplo aunque no haya habido ningún cambio de flanco.

El estado de señal en la salida Q será "1" siempre que el valor de contaje sea mayor que cero, y será "0" si el valor de contaje es cero.

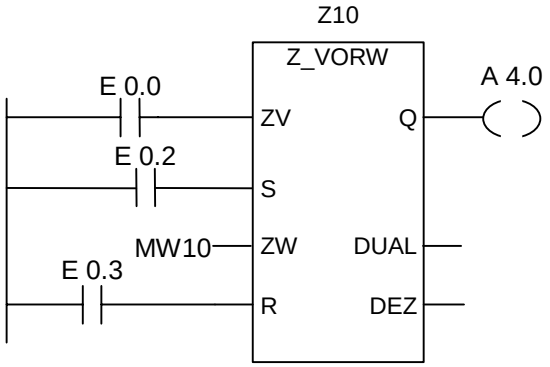
Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Nota

No utilice un mismo contador en varios puntos del programa (riesgo de errores de contaje).

Ejemplo

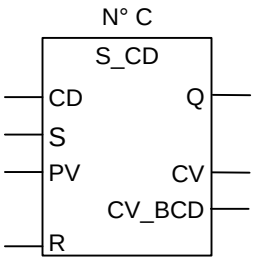


Al cambiar la entrada E 0.2 de "0" a "1", el contador toma el valor predeterminado para MW10. Si el estado de señal en E 0.0 cambia de "0" a "1", el valor del contador Z10 se incrementa en "1", a menos que el valor de Z10 fuera "999". La salida A 4.0 será "1" siempre que el valor de Z10 no sea cero.

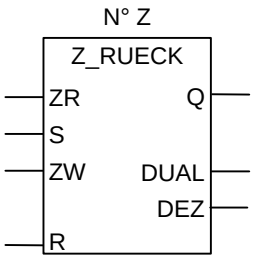
4.4 Z_RUECK Parametrizar y decrementar contador

Símbolo

Inglés



Alemán



Parámetro Inglés	Parámetro Alemán	Tipo de datos	Area de memoria	Descripción
N.° de C	N.º de Z	COUNTER	Z	Número de identificación del contador; el área varía según CPU utilizada
CU	ZV	BOOL	E, A, M, L, D	Entrada de contaje adelante
CD	ZR	BOOL	E, A, M, L, D	Entrada de contaje atrás
S	S	BOOL	E, A, M, L, D	Entrada para predeterminar el contador
PV	ZW	WORD	E, A, M, L, D o constante	Valor numérico introducido en forma de C#<valor> en el margen comprendido entre 0 y 999
PV	ZW	WORD	E, A, M, L, D	Valor para inicializar el contador
R	R	BOOL	E, A, M, L, D	Entrada de puesta a 0
CV	DUAL	WORD	E, A, M, L, D	Valor actual del conta-dor, número hexadecimal
CV_BCD	DEZ	WORD	E, A, M, L, D	Valor actual del contador, número BCD
Q	Q	BOOL	E, A, M, L, D	Estado del contador

Descripción de la operación

Z_RUECK (Parametrizar y decrementar contador) toma el valor predeterminado de la entrada ZW si en la entrada S hay un flanco ascendente.

Si el estado de señal de la entrada R es "1" el contador se pone a 0, y entonces el valor de contaje es cero.

El contador decrementa en "1" si el estado de señal en la entrada ZR cambia de "0" a "1" y el valor de contaje era mayor que cero.

Si se inicializa el contador y el RLO de las entradas ZV/ZR = 1, el contador contará así en el siguiente ejemplo aunque no haya habido ningún cambio de flanco.

El estado de señal en la salida Q será "1" si el valor de contaje es mayor que cero, y será "0" si el valor de contaje es cero.

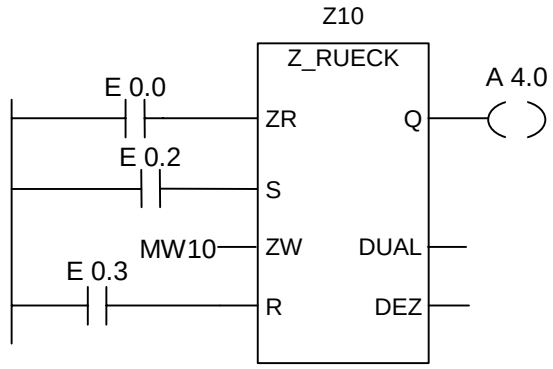
Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Nota

No utilice un mismo contador en varios puntos del programa (riesgo de errores de contaje).

Ejemplo



Al cambiar la entrada E 0.2 de "0" a "1", el contador toma el valor de preselección de MW10. Si el estado de señal en E 0.0 cambia de "0" a "1", el valor del contador Z10 decrementa en "1", a menos que el valor de Z10 fuera "0". La salida A 4.0 será "1" siempre que el valor de Z10 no sea cero.

4.5 ---(SZ) Poner contador al valor inicial

Símbolo

Inglés	**Alemán**
<Nº de C>	<Nº de Z>
---(SC)	**---(SZ)**
<Valor predeterminado>	<Valor predeterminado>

Parámetro Inglés	**Parámetro Alemán**	**Tipo de datos**	**Area de memoria**	**Descripción**
N.° de C	<Nº de Z>	COUNTER	Z	Número del contador a predeterminar
<Valor pre-determina-do>	<Valor prede-terminado>	WORD	E, A, M, L, D	Valor para la preselección BCD (0-999)

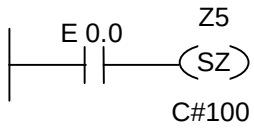
Descripción de la operación

---(SZ) (Inicializar el contador) se ejecuta solamente en caso de que haya un flanco ascendente en el RLO. En este caso se transmite el valor predeterminado al contador indicado.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	x	-	0

Ejemplo



El valor "100" quedará preseleccionado para el contador Z5 si en la entrada E 0.0 se produce un flanco ascendente (cambio de "0" a "1"). El valor del contador Z5 no se altera en caso de que no se produzca ningún flanco ascendente.

4.6 ---(ZV) Incrementar contador

Símbolo

Inglés	**Alemán**
<Nº de C>	<Nº de Z>
---(CU)	**---(ZV)**

Parámetro Inglés	**Parámetro Alemán**	**Tipo de datos**	**Area de memoria**	**Descripción**
N.° de C	<Nº de Z>	COUNTER	Z	Número específico del contador; el área varía según la CPU utilizada

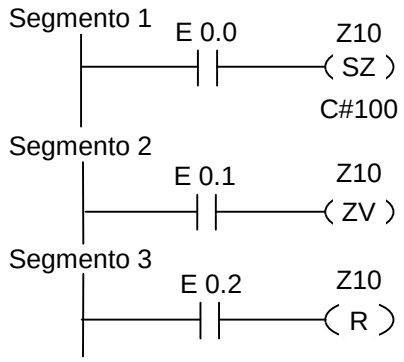
Descripción de la operación

---(ZV) (Contar adelante) incrementa en "1" el valor del contador indicado si hay un flanco ascendente en el RLO y el valor del contador es menor que "999". El valor del contador no se altera si no hay ningún flanco ascendente, ni tampoco en caso de que el el contador tenga ya el valor "999".

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	-	-	0

Ejemplo



Si el estado de señal de E 0.0 cambia de "0" a "1" (flanco ascendente en el RLO), se carga el valor predeterminado de "100" en el contador Z10.

Si el estado de señal de E 0.1 cambia de "0" a "1" (flanco ascendente en el RLO), se aumenta en "1" el valor de contaje del contador Z10, a menos que el valor de contaje sea igual a "999". El valor del contador Z10 no se altera si no hay ningún flanco ascendente en el RLO.

Si el estado de señal de E 0.2 es 1, el contador se pone a "0".

4.7 ---(ZR) Decrementar contador

Símbolo

Inglés	**Alemán**
<Nº de C>	<Nº de Z>
---(CD)	**---(ZR)**

Parámetro Inglés	**Parámetro Alemán**	**Tipo de datos**	**Area de memoria**	**Descripción**
N.° de C	<Nº de Z>	COUNTER	Z	Número específico del contador; el área varía según la CPU utilizada

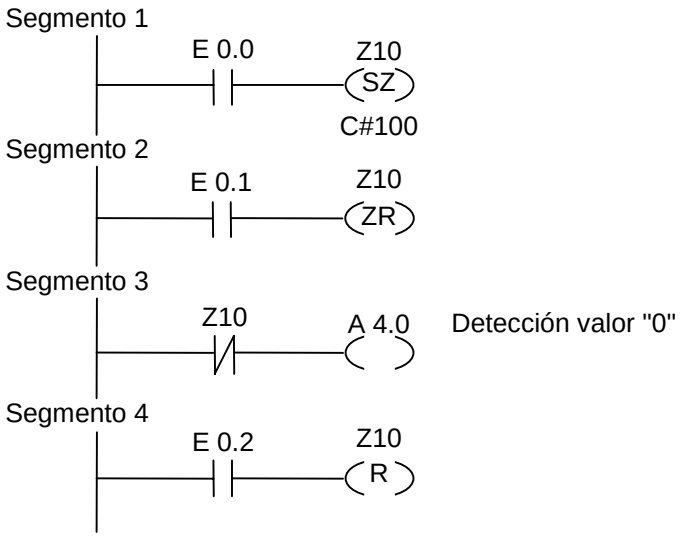
Descripción de la operación

---(ZR) (Contar atrás) decrementa en "1" el valor del contador indicado si hay un flanco ascendente en el RLO y el valor del contador es mayor que "0". El valor del contador no se altera si no hay ningún flanco ascendente, ni tampoco en caso de que el contador tenga ya el valor "0.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	-	-	0

Ejemplo



Si el estado de señal de E 0.0 cambia de "0" a "1" (flanco ascendente en el RLO), se carga el valor predeterminado de "100" en el contador Z10.

Si el estado de señal de E 0.1 cambia de "0" a "1" (flanco ascendente en el RLO), se decrementa en "1" el valor de contaje del contador Z10, a menos que el valor de contaje sea igual a "0". El valor del contador Z10 no se altera si no hay ningún flanco ascendente en el RLO.

A 4.0 se conecta si el valor de contaje equivale a cero.

Si el estado de señal de E 0.2 es "1", el contador se pone a "0".

5 Operaciones con bloques de datos

5.1 ---(OPN) Abrir bloque de datos

Símbolo

<N.º de DB> ó <N.º de DI>

---(OPN)

Parámetro	Tipo de datos	Area de memoria	Descripción
<N.º de DB> <N.º de DI>	BLOCK_DB	DB, DI	Número del DB/DI; el área varía según la CPU utilizada

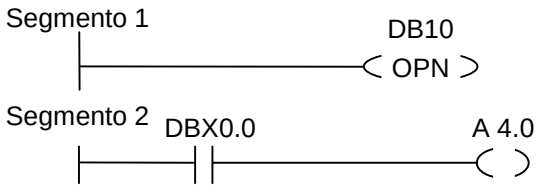
Descripción de la operación

---(OPN) (Abrir bloque de datos) abre un bloque de datos (DB global o DB de instancia). La operación ---(OPN) es una llamada absoluta a un bloque de datos. El número del bloque de datos se transmite al registro DB o DI. Los comandos de DB y DI siguientes acceden a los bloques correspondientes en función de cuáles sean los contenidos del registro.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	-	-	-	-

Ejemplo



Se abre el bloque de datos 10 (DB10). La dirección del contacto (DBX0.0) se refiere al bit cero del byte de datos cero del registro actual que hay en DB10. El estado de señal de este bit se asigna a la salida A 4.0.

6 Operaciones de salto

6.1 Lista de operaciones de salto

Descripción

Estas operaciones se pueden utilizar en todos los bloques lógicos: bloques de organización (OBs), bloques de función (FBs) y funciones (FCs).

Se dispone de las operaciones de salto siguientes:

- ---(JMP)--- Salto absoluto
- ---(JMP)--- Salto condicional
- ---(JMPN) Saltar si la señal es 0

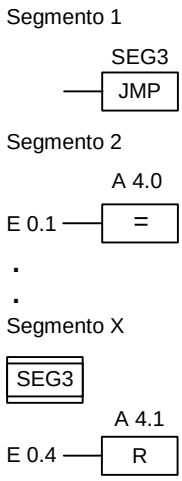
Meta como operando

El operando de una operación de salto es una meta. La meta indica el destino a donde se desea saltar en el programa. La meta se introduce encima de la bobina de salto,

Una meta se compone de cuatro caracteres como máximo. El primer carácter ha de ser una letra del alfabeto; los restantes caracteres pueden ser letras o números (p.ej. SEG3).

Meta como destino

La meta de destino ha de encontrarse siempre al principio de un segmento. Para introducirla hay que seleccionar LABEL en el cuadro KOP. En seguida aparece un cuadro vacío. Introducir en el cuadro el nombre de la meta.



6.2 ---(JMP)--- Salto absoluto

Símbolo

<Meta>

---(JMP)

Descripción de la operación

---(JMP) (Saltar si la señal es 1) funciona como un salto absoluto cuando no hay otro elemento KOP entre el conductor izquierdo y la operación (v. ejemplo).

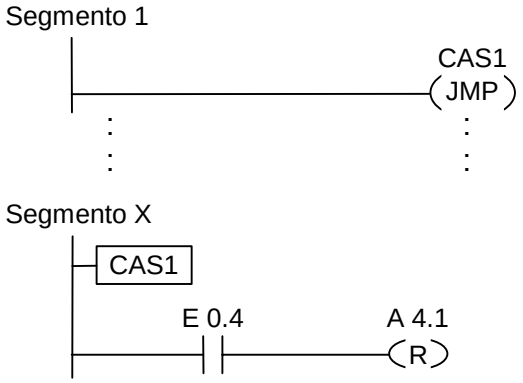
Cada salto ---(JMP) tiene que tener una meta (LABEL).

¡No se ejecutarán las operaciones que se encuentren entre la operación de salto y la meta!

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	-	-	-	-

Ejemplo



El salto se ejecuta en todos los casos, omitiéndose (”pasando por alto”) las operaciones que se encuentren entre la operación de salto y la meta.

6.3 ---(JMP)--- Salto condicional

Símbolo

<Meta>

---(JMP)

Descripción de la operación

---(JMP) (Saltar en el bloque si es 1) funciona como un salto condicional cuando el RLO de la combinación lógica anterior es "1".

Cada salto ---(JMP) tiene que tener una meta (LABEL).

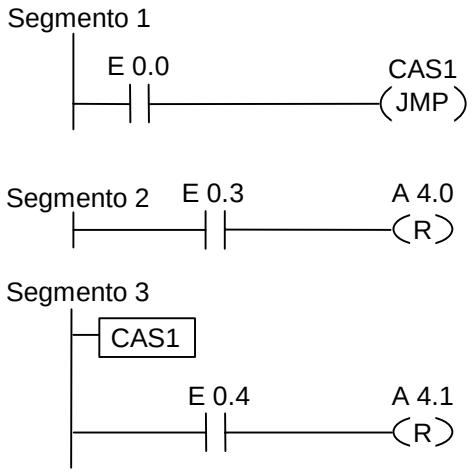
¡No se ejecutarán las operaciones que se encuentren entre la operación de salto y la meta!

Si un salto condicional no se ejecuta, el RLO cambia a "1" después de la operación de salto.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	1	1	0

Ejemplo



Si la entrada E 0.0 es 0 se ejecuta el salto a la meta CAS1. Al llevarse a cabo el salto, en la salida A 4.0 no se ejecuta la operación ”Poner salida a 0”, aunque E 0.3 sea 1.

6.4 ---(JMPN) Saltar si la señal es 0

Símbolo

<Meta>

---(JMPN)

Descripción de la operación

---(JMPN) (Saltar si la señal es 0) funciona como un salto condicional cuando el RLO de la combinación lógica anterior es "0".

Cada salto ---(JMPN) tiene que tener una meta (LABEL).

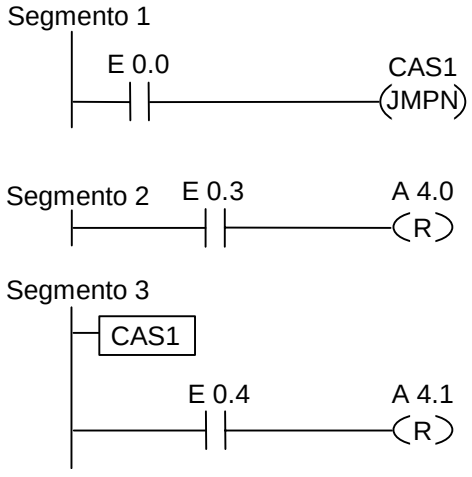
¡No se ejecutarán las operaciones que se encuentran entre la operación de salto y la meta!

Si un salto condicional no se lleva a cabo, el RLO cambia a "1" después de la operación de salto.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	0	1	1	0

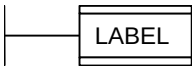
Ejemplo



Si la entrada E 0.0 es 0, se ejecuta el salto a la meta CAS1. Al ejecutarse el salto, en la salida A 4.0 no se lleva a cabo la operación ”Poner salida a 0”, aunque la entrada E 0.3 sea 1.

6.5 LABEL Meta del salto

Símbolo

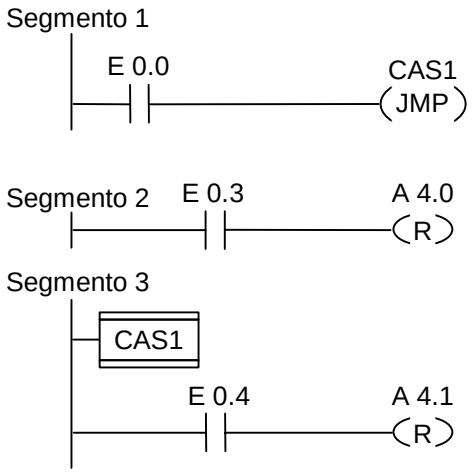


Descripción de la operación

LABEL marca la meta de una operación de salto. Esta meta puede tener hasta un màximo 4 caracteres. Primer carácter: letra; demás caracteres: letras o alfanuméricos, p.ej. CAS1.

Cada salto ---(JMP) o ---(JMPN) tiene que tener una meta del salto (LABEL).

Ejemplo



Si E 0.0 = 1 se ejecuta el salto a la meta CAS1. Al llevarse a cabo el salto, en la salida A 4.0 no se ejecuta la operación ”Poner salida a 0”, aunque E 0.3 sea 1.

7 Operaciones aritméticas con enteros

7.1 Lista de operaciones aritméticas con enteros

Descripción

Las operaciones aritméticas con enteros sirven para ejecutar las siguientes operaciones aritméticas con **dos** enteros (16 y 32 bits):

- ADD_I Sumar enteros
- SUB_I Restar enteros
- MUL_I Multiplicar enteros
- DIV_I Dividir enteros
- ADD_DI Sumar enteros dobles
- SUB_DI Restar enteros dobles
- MUL_DI Multiplicar enteros dobles
- DIV_DI Dividir enteros dobles
- MOD_DI Obtener el resto de una división de enteros dobles

7.2 Evaluar bits de la palabra de estado en operaciones en coma fija

Descripción

Las operaciones aritméticas básicas influyen sobre los siguientes bits de la palabra de datos:

- A1 y A0
- OV
- OS

Las tablas siguientes muestran el estado de señal de los bits de la palabra de estado para los resultados de las operaciones con números en coma fija (16 bit, 32 bit).

Margen válido	A1	A0	OV	OS
0 (cero)	0	0	0	*
enteros: –32 768 < =resultado < 0 (número negativo) enteros dobles: –2 147 483 648 <=resultado < 0 (número negativo)	0	1	0	*
enteros: 32 767 > =resultado > 0 (número positivo) enteros dobles: 2 147 483 647 > = resultado > 0 (número positivo)	1	0	0	*

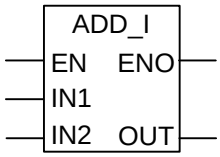
* El bit OS no se ve influido por el resultado de la operación.

Margen no válido	A1	A0	OV	OS
Desbordamiento negativo en la suma enteros: resultado = –65536 enteros dobles: resultado = –4 294 967 296	0	0	1	1
Desbordamiento negativo en la multiplicación enteros: resultado < –32 768 (número negativo) enteros dobles: resultado < –2 147 483 648 (número negativo)	0	1	1	1
Desbordamiento positivo en la suma, resta enteros: resultado > 32 767 (número positivo) enteros dobles: resultado > 2 147 483 647 (número positivo)	0	1	1	1
Desbordamiento positivo en la multiplicación, división enteros: resultado > 32 767 (número positivo) enteros dobles: resultado > 2 147 483 647 (número positivo)	1	0	1	1
Desbordamiento negativo en la suma, resta enteros: resultado < –32 768 (número negativo) enteros dobles: resultado < –2 147 483 648 (número negativo)	1	0	1	1
División por cero	1	1	1	1

Operación	A1	A0	OV	OS
+D: resultado = -4 294 967 296	0	0	1	1
/D o MOD: división por cero	1	1	1	1

7.3 ADD_I Sumar enteros

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	INT	E, A, M, L, D o constante	Primer sumando
IN2	INT	E, A, M, L, D o constante	Segundo sumando
OUT	INT	E, A, M, L, D	Resultado de la suma

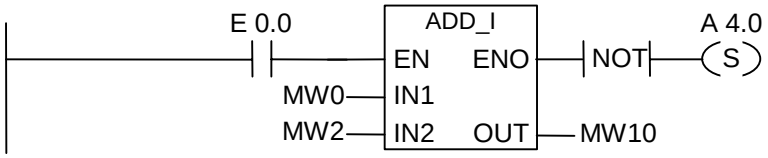
Descripción de la operación

ADD_I (Sumar enteros) suma las entradas IN1 y IN2 si el estado de señal en la entrada de habilitación (EN) es "1". La salida OUT proporciona el resultado. Si el resultado es un valor fuera del margen válido para enteros (de 16 bits), los bits OV y OS son 1 y ENO es 0, de forma que otras operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

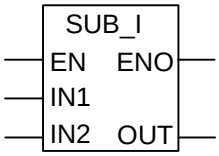
Ejemplo



El cuadro ADD_I se activa si E 0.0 es 1. El resultado de la suma MW0 + MW2 se deposita en MW10. Si el resultado es un valor fuera del margen válido para enteros o si el estado de señal dE E 0.0 es 0, la salida A 4.0 se pone a 1.

7.4 SUB_I Restar enteros

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	INT	E, A, M, L, D o constante	Sustraendo
IN2	INT	E, A, M, L, D o constante	Minuendo
OUT	INT	E, A, M, L, D	Resultado de la sustracción

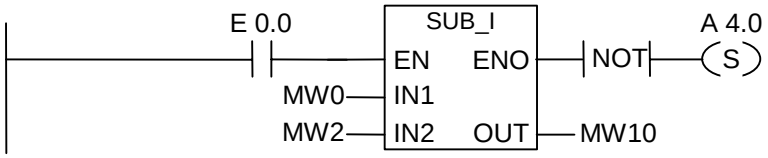
Descripción de la operación

SUB_I (Restar enteros) resta el valor de IN2 del valor de IN1 si el estado de señal en la entrada de habilitación (EN) es "1". La salida OUT proporciona el resultado. Si el resultado es un valor fuera del margen válido para enteros (de 16 bits), los bits OV y OS son "0" y ENO es "0", de forma que otras operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

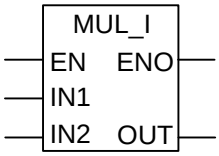
Ejemplo



El cuadro SUB_I se activa si E 0.0 es 1. El resultado de la sustracción MW0 - MW2 se deposita en MW10. Si el resultado es un valor fuera del margen válido para enteros (de 16 bits) o si el estado de señal de E 0.0 es 0, la salida A 4.0 se pone a 1.

7.5 MUL_I Multiplicar enteros

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	INT	E, A, M, L, D o constante	Multiplicando
IN2	INT	E, A, M, L, D o constante	Multiplicador
OUT	INT	E, A, M, L, D	Resultado de la multiplicación

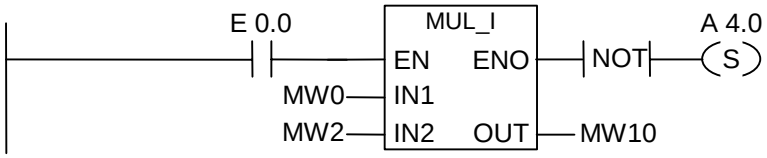
Descripción de la operación

MUL_I (Multiplicar enteros) multiplica los valores de las entradas IN1 y IN2 si el estado de señal en la entrada de habilitación (EN) es "1". La salida OUT proporciona el resultado. Si el resultado es un valor fuera del margen válido para enteros (de 16 bits), los bits OV y OS son "1" y ENO es "0", de forma que otras operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

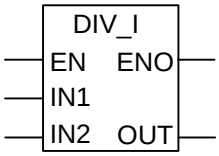
Ejemplo



El cuadro MUL_I se activa si E 0.0 es 1. El resultado de la multiplicación MW0 x MW2 se deposita en MW10. Si el resultado es un valor fuera del margen válido para enteros, o si el estado de señal de E 0.0 es 0, la salida A 4.0 se pone a 1.

7.6 DIV_I Dividir enteros

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	INT	E, A, M, L, D o constante	Dividendo
IN2	INT	E, A, M, L, D o constante	Divisor
OUT	INT	E, A, M, L, D	Cociente la división

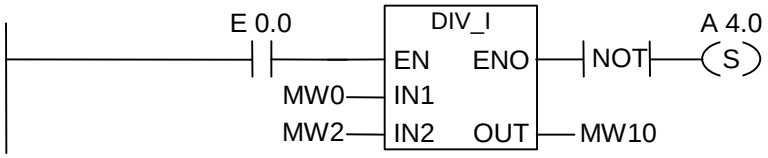
Descripción de la operación

DIV_I (Dividir enteros) divide el valor de IN1 entre el valor de IN2 si el estado de señal en la entrada de habilitación (EN) es "1". La salida OUT proporciona el resultado. Si el resultado es un valor fuera del margen válido para enteros, los bits OV y OS son "1" y ENO es "0", de forma que otras operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

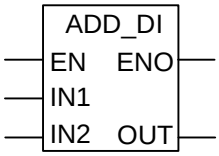
Ejemplo



El cuadro DIV_I se activa si E 0.0 es 1. El resultado de la división de MW0 entre MW2 se deposita en MW10. Si el resultado es un valor fuera del margen válido para enteros, o si el estado de señal de E 0.0 es 0, la salida A 4.0 se pone a 1.

7.7 ADD_DI Sumar enteros dobles

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	DINT	E, A, M, L, D o constante	Primer sumando
IN2	DINT	E, A, M, L, D o constante	Segundo sumando
OUT	DINT	E, A, M, L, D	Resultado de la suma

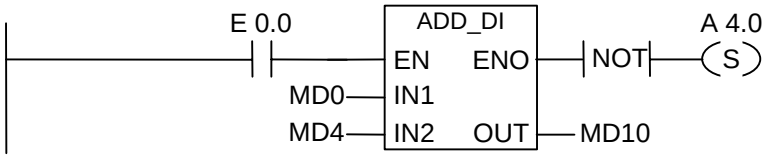
Descripción de la operación

ADD_DI (Sumar enteros dobles) suma las entradas IN1 y IN2 si el estado de señal en la salida de habilitación es "1". La salida OUT proporciona el resultado. Si el resultado es un valor fuera del margen válido para enteros dobles, los bits OV y OS son 1 y ENO es 0, de forma que otras operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

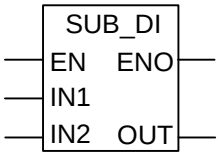
Ejemplo



El cuadro ADD_DI se activa si E 0.0 es 1. El resultado de la suma MD0 + MD4 se deposita en MD10. Si el resultado es un valor fuera del margen válido para enteros dobles, o si el estado de señal de E 0.0 es 0, la salida A 4.0 se pone a 1.

7.8 SUB_DI Restar enteros dobles

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	DINT	E, A, M, L, D o constante	Sustraendo
IN2	DINT	E, A, M, L, D o constante	Minuendo
OUT	DINT	E, A, M, L, D	Resultado de la sustracción

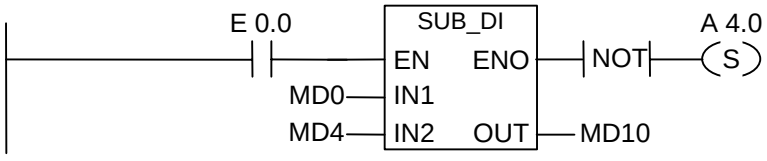
Descripción de la operación

SUB_DI (Restar enteros dobles) resta el valor IN2 del valor de IN1 si el estado de señal en la entrada de habilitación (EN) es "1". La salida OUT proporciona el resultado. Si el resultado es un valor fuera del margen válido para enteros dobles, los bits OV y OS son "1" y ENO es "0", de forma que otras operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

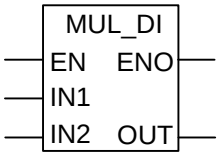
Ejemplo



El cuadro SUB_DI se activa si E 0.0 es 1. El resultado de la sustracción MD0 - MD4 se deposita en MD10. Si el resultado es un valor fuera del margen válido para enteros dobles, o si el estado de señal de E 0.0 es 0, la salida A 4.0 se pone a 1.

7.9 MUL_DI Multiplicar enteros dobles

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	DINT	E, A, M, L, D o constante	Multiplicando
IN2	DINT	E, A, M, L, D o constante	Multiplicador
OUT	DINT	E, A, M, L, D	Producto de la multiplicación

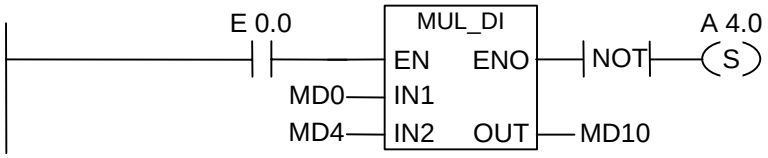
Descripción de la operación

MUL_DI (Multiplicar enteros dobles) multiplica los valores de las entradas IN1 y IN2 si el estado de señal en la entrada de habilitación es "1". La salida OUT proporciona el resultado. Si el resultado es un valor fuera del margen válido para enteros dobles, los bits OV y OS son "1" y ENO es "0", de forma que otras operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

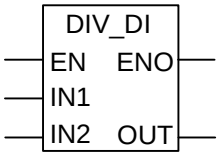
Ejemplo



El cuadro MUL_DI se activa si E 0.0 es 1. El resultado de la multiplicación MD0 x MD4 se deposita en MD10. Si el resultado es un valor fuera del margen válido para enteros dobles, o si el estado de señal de E 0.0 es 0, la salida A 4.0 se pone a 1.

7.10 DIV_DI Dividir enteros dobles

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	DINT	E, A, M, L, D o constante	Dividendo
IN2	DINT	E, A, M, L, D o constante	Divisor
OUT	DINT	E, A, M, L, D	Cociente de la división

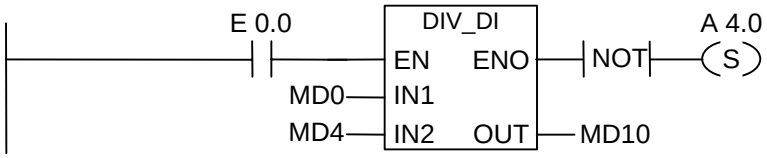
Descripción de la operación

DIV_DI (Dividir enteros dobles) divide el valor de IN1 entre el valor de IN2 si el estado de señal en la entrada de habilitación (EN) es "1". La salida OUT proporciona el resultado (parte entera). El elemento Dividir enteros dobles no genera ningún resto de división. Si el resultado es un valor fuera del margen válido para enteros dobles, los bits OV y OS son "1" y ENO es "0", de forma que otras operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

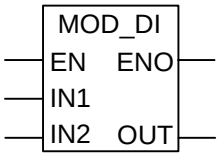
Ejemplo



El cuadro DIV_DI se activa si E 0.0 es 1. El resultado de la división de MD0 por MD4 se deposita en MD10. Si el resultado es un valor fuera del margen válido para enteros dobles, o si el estado de señal de E 0.0 es 0, la salida A 4.0 se pone a 1.

7.11 MOD_DI Obtener el resto de una división de enteros dobles

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	DINT	E, A, M, L, D o constante	Dividendo
IN2	DINT	E, A, M, L, D o constante	Divisor
OUT	DINT	E, A, M, L, D	Resto de la división

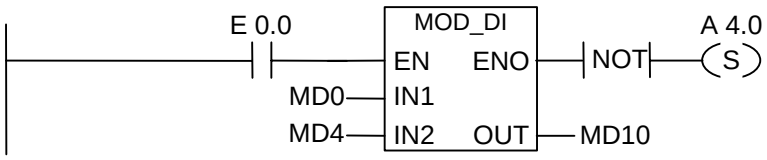
Descripción de la operación

MOD_DI (Obtener el resto de una división de enteros dobles) divide el valor de IN1 entre el valor de IN2 si el estado de señal en la entrada de habilitación (EN) es "1". La salida OUT proporciona el resultado, esto es, el resto de la división. Si el resultado es un valor fuera del margen válido para enteros dobles, los bits OV y OS son "1" y ENO es "0", de forma que otras operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

Ejemplo



El cuadro DIV_DI se activa si E 0.0 es 1. El resto de la división de MD0 entre MD4 se deposita en MD10. Si el resto de la división es un valor fuera del margen válido para enteros dobles, o si el estado de señal de E 0.0 es 0, la salida A 4.0 se pone a 1.

8 Operaciones aritméticas en coma flotante

8.1 Lista de operaciones aritméticas con números en coma flotante

Descripción

Los números de 32 bits IEEE en coma flotante pertenecen al tipo de datos denominado "REAL". Las operaciones aritméticas con números en coma flotante sirven para ejecutar las siguientes operaciones aritméticas con **dos** números en coma flotante IEEE de 32 bits:

- ADD_R Sumar números en coma flotante
- SUB_R Restar números en coma flotante
- MUL_R Multiplicar números en coma flotante
- DIV_R Dividir números en coma flotante

Con las operaciones aritméticas de números en coma flotante se pueden ejecutar las siguientes funciones con **un** número en coma flotante (32 bit, IEEE-FP):

- Calcular el valor absoluto (ABS)
- Calcular el cuadrado (SQR) o la raíz cuadrada (SQRT)
- Calcular el logaritmo natural (LN)
- Calcular el valor exponencial (EXP) en base e (= 2,71828...)
- Calcular las funciones trigonométricas siguientes (en un ángulo como número en coma flotante (32 bit, IEEE–FP))
  - seno (SIN) y arcoseno (ASIN)
  - coseno (COS) y arcocoseno (ACOS)
  - tangente (TAN) y arcotangentE (ATAN)

8.2 Evaluar los bits de la palabra de estado en operaciones en coma flotante

Descripción

Las operaciones aritméticas básicas afectan a los siguientes bits de la palabra de estado:

- A1 y A0
- OV
- OS

Las tablas siguientes muestran el estado de señal de los bits de la palabra de estado para los resultados de operaciones con números en coma flotante (32 bits).

Margen válido	A1	A0	OV	OS
+0, -0 (Cero)	0	0	0	*
-3.402823E+38 < Resultado < -1.175494E-38 (número negativo)	0	1	0	*
+1.175494E-38 < Resultado < +3.402823E+38 (número positivo)	1	0	0	*

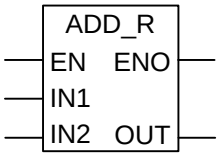
* El bit OS no es afectado por el resultado de la operación.

Margen inválido	A1	A0	OV	OS
Desbordamiento negativo -1.175494E-38 < Resultado < -1.401298E-45 (número negativo)	0	0	1	1
Desbordamiento negativo +1.401298E-45 < Resultado < +1.175494E-38 (número positivo)	0	0	1	1
Desbordamiento Resultado < -3.402823E+38 (número negativo)	0	1	1	1
Desbordamiento Resultado > 3.402823E+38 (número positivo)	1	0	1	1
Número en coma flotante no válido u operación no permitida (valor de entrada fuera del margen válido de valores)	1	1	1	1

8.3 Operaciones básicas

8.3.1 ADD_R Sumar números en coma flotante

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	REAL	E, A, M, L, D	Primer sumando
IN2	REAL	E, A, M, L, D	Segundo sumando
OUT	REAL	E, A, M, L, D	Resultado de la suma

Descripción de la operación

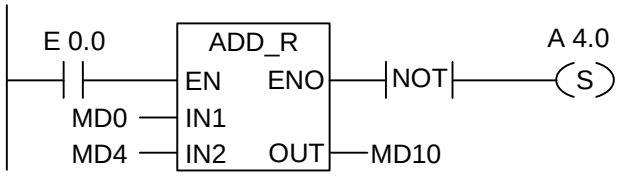
ADD_R (Sumar números en coma flotante) suma los valores de las entradas IN1 y IN2 cuando la entrada de habilitación (EN) tiene el estado de señal "1". El resultado se deposita en la salida OUT. Si el resultado se encuentra fuera del margen válido para números en coma flotante (desbordamiento positivo o negativo), el bit OV y el bit OS son 1 y ENO es 0, de forma que las demás operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

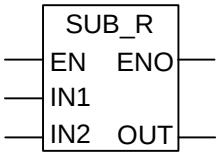
Ejemplo



El cuadro ADD_R se activa si E 0.0 es 1. El resultado de la suma MD0 + MD4 se deposita en MD10. Si el resultado se encuentra fuera del margen válido para números en coma flotante, o si no se ejecuta esta instrucción (E 0.0 = 0), se activa la salida A 4.0.

8.3.2 SUB_R Restar números en coma flotante

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	REAL	E, A, M, L, D	Minuendo
IN2	REAL	E, A, M, L, D	Sustraendo
OUT	REAL	E, A, M, L, D	Resultado de la sustracción

Descripción de la operación

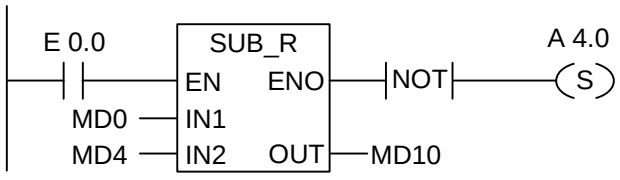
SUB_R (Restar números en coma flotante) resta los valores IN2 de IN1 cuando la entrada de habilitación (EN) tiene el estado de señal "1". El resultado de la sustracción se deposita en la salida OUT. Si el resultado se encuentra fuera del margen válido para números en coma flotante (desbordamiento positivo o negativo), los bits OV y OS son 1 y ENO es 0, de forma que las demás operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

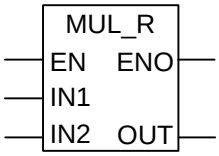
Ejemplo



El cuadro SUB_R se activa cuando E 0.0 es 1. El resultado de la sustracción MD0 - MD4 se deposita en MD10. Si el resultado se encuentra fuera del margen válido para números en coma flotante, o si no se ejecuta esta instrucción (E 0.0 = 0), se activará la salida A 4.0.

8.3.3 MUL_R Multiplicar números en coma flotante

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	REAL	E, A, M, L, D	Multiplicando
IN2	REAL	E, A, M, L, D	Multiplicador
OUT	REAL	E, A, M, L, D	Producto de la multiplicación

Descripción de la operación

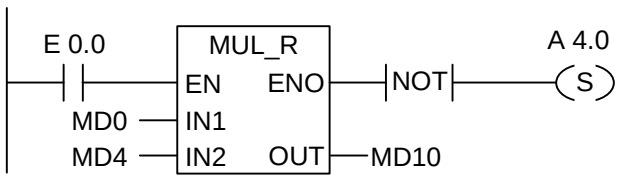
MUL_R (Multiplicar números en coma flotante) multiplica el valor de IN1 por el valor de IN2 cuando la entrada de habilitación (EN) tiene el estado de señal "1". El resultado se deposita en la salida OUT. Si el resultado se encuentra fuera del margen válido para números en coma flotante (desbordamiento positivo o negativo), el bit OV y el bit OS son 1 y ENO es 0, de forma que las demás operaciones operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

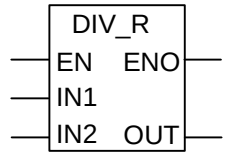
Ejemplo



El cuadro MUL_R se activa cuando E 0.0 es 1. El resultado de la multiplicación MD0 x MD4 se deposita en MD0. Si el resultado se encuentra fuera del margen válido para números en coma flotante, o si no se ejecuta esta instrucción (E 0.0 = 0), se activará la salida A 4.0.

8.3.4 DIV_R Dividir números en coma flotante

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	REAL	E, A, M, L, D	Dividendo
IN2	REAL	E, A, M, L, D	Divisor
OUT	REAL	E, A, M, L, D	Cociente de la división

Descripción de la operación

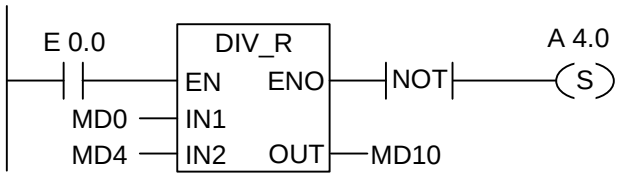
DIV_R (Dividir números en coma flotante) divide el valor de IN1 entre el valor de IN2 cuando la entrada de habilitación (EN) tiene el estado de señal "1". El resultado se deposita en la salida OUT. Si el resultado se encuentra fuera del margen válido para números en coma flotante (desbordamiento positivo o negativo), el bit OV y el bit OS son 1 y ENO es 0, de forma que las demás operaciones que siguen a esta operación aritmética, combinadas a través de ENO (ejecución en cascada), no se ejecutan.

Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

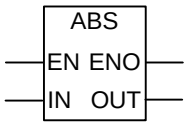
Ejemplo



El cuadro DIV_R se activa cuando E 0.0 es 1. El resultado de la división de MD0 entre MD4 se deposita en MD10. Si el resultado se encuentra fuera del margen válido para números en coma flotante, o si no se ejecuta esta instrucción (E 0.0 = 0), se activará la salida A 4.0.

8.3.5 ABS Calcular el valor absoluto de un número en coma flotante

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Valor absoluto del número en coma flotante

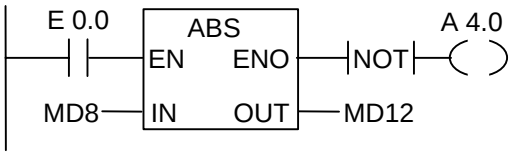
Descripción de la operación

ABS (Calcular el valor absoluto de un número en coma flotante) calcula el valor absoluto de un número en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	-	-	-	-	0	1	1	1

Ejemplo



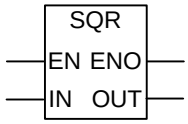
Si E 0.0 = 1 se calcula el valor absoluto de MD8 y el resultado se escribe en MD12.

De MD8 = + 6,234 resulta MD12 = 6,234. La salida A 4.0 será "1" si no se lleva a cabo la conversión (ENO = EN = 0).

8.4 Operaciones ampliadas

8.4.1 SQR Calcular el cuadrado

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Cuadrado del número en coma flotante

Descripción de la operación

SQR (Calcular el cuadrado de un número en coma flotante) calcula el cuadrado de un número flotante.

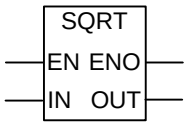
Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

8.4.2 SQRT Calcular la raíz cuadrada

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Raíz cuadrada del número en coma flotante

Descripción de la operación

SQRT (Calcular la raíz cuadrada de un número en coma flotante) calcula la raíz cuadrada de un número en coma flotante. Esta operación arroja un resultado positivo si el operando es mayor que "0". Única excepción: la raíz cuadrada de -0 es -0.

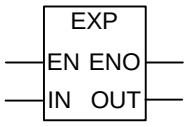
Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

8.4.3 EXP Calcular el exponente

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Exponente del número en coma flotante

Descripción de la operación

EXP (Calcular el exponente de un número en coma flotante) calcula el exponente de un número en coma flotante con la base e (=2,71828...).

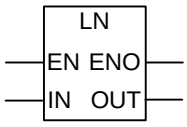
Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

8.4.4 LN Calcular el logaritmo natural

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Logaritmo natural del número en coma flotante

Descripción de la operación

LN (Calcular el logaritmo natural de un número en coma flotante) calcula el logaritmo natural de un número en coma flotante.

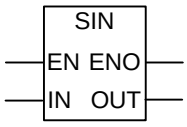
Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

8.4.5 SIN Calcular el seno

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Seno del número en coma flotante

Descripción de la operación

SIN (Calcular el seno de un número en coma flotante) calcula el seno de un número en coma flotante. El número en coma flotante representa aquí un ángulo en radianes.

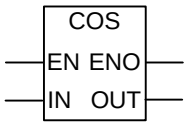
Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

8.4.6 COS Calcular el coseno

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Coseno del número en coma flotante

Descripción de la operación

COS (Calcular el coseno de un número en coma flotante) calcula el coseno de un número en coma flotante, siendo éste el valor de un ángulo expresado en radianes.

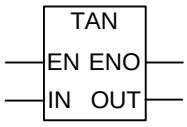
Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

8.4.7 TAN Calcular la tangente

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Tangente del número en coma flotante

Descripción de la operación

TAN (Calcular la tangente de un número en coma flotante) calcula la tangente de un número en coma flotante, siendo éste el valor de un ángulo expresado en radianes.

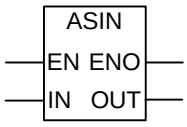
Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

8.4.8 ASIN Calcular el arcoseno

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Arcoseno del número en coma flotante

Descripción de la operación

ASIN (Calcular el arcoseno de un número en coma flotante) calcula el arcoseno de un número en coma flotante, cuyo margen de definición es -1 <= Valor de entradA <= 1. El resultado representa aquí un ángulo en radianes en el margen de valores

$$-\pi/2 \leq \text{Valor de salida} \leq +\pi/2$$

siendo π = 3,1415....

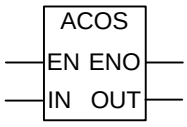
Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

8.4.9 ACOS Calcular el arcocoseno

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Arcocoseno del número en coma flotante

Descripción de la operación

ACOS (Calcular el arcocoseno de un número en coma flotante) calcula el arcocoseno de un número en coma flotante, cuyo margen de definición es -1 <= Valor de entradA <= 1. El resultado es el valor de un ángulo expresado en radianes, valor que queda dentro del margen de valores

$$0 \leq \text{valor de salida} \leq +\pi$$

siendo π = 3,1415....

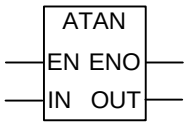
Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

8.4.10 ATAN Calcular la arcotangente

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	REAL	E, A, M, L, D	Valor de entrada: Número en coma flotante
OUT	REAL	E, A, M, L, D	Valor de salida: Arcotangente del número en coma flotante

Descripción de la operación

ATAN (Calcular la arcotangente de un número en coma flotante) calcula la arcotangente de un número en coma flotante. El resultado es un ángulo expresado en radianes que queda dentro del margen

$$-\pi/2 \leq \text{valor de salida} \leq \pi/2$$

siendo π = 3,1415....

Consulte también Evaluar los bits de la palabra de estado en operaciones en coma flotante.

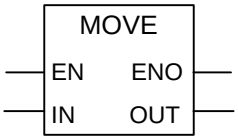
Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	x	0	x	x	1

9 Operaciones de transferencia

9.1 MOVE Asignar un valor

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	Todos los tipos de datos simples con una longitud de 8, 16 o 32 bits	E, A, M, L, D o constante	Valor de fuente
OUT	Todos los tipos de datos simples con una longitud de 8, 16 o 32 bits	E, A, M, L, D	Dirección de destino

Descripción de la operación

MOVE (Asignar un valor) es activada por la entrada de habilitación EN. El valor indicado por la entrada IN se copia en la dirección que la salida OUT. La salida de habilitación ENO tiene el mismo estado de señal que la entrada de habilitación EN. La operación MOVE sólo puede copiar los objetos de datos que tengan las longitudes de BYTE, WORD o de DWORD. Los tipos de datos de usuario tales como los arrays o las estructuras han de copiarse con SFC 20 “BLKMOV”.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	-	-	-	-	0	1	1	1

Dependencia del MCR (Master Control Relay)

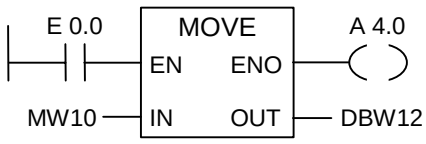
La dependencia del MCR solamente se activa si el cuadro MOVE se coloca dentro de un área de MCR activa. En área MCR los datos direccionados se copian tal como se ha descrito arriba, siempre que el MCR esté conectado y haya corriente en la entrada de habilitación, el bit direccionado se pone a "1" (se activa) ó a "0" (se desactiva), tal como se ha descrito más arriba. Si el MCR está desconectado y se ejecuta una operación MOVE, en la dirección indicada por OUT se escribirá siempre el valor "0", independientemente de cuál sea el estado actual de IN.

Nota

Al transferir un valor a un tipo de datos de longitud diferente los bytes más significativos se truncan o se rellenan con ceros si es preciso:

Palabra doble	**1111 1111**	**0000 1111**	**1111 0000**	**0101 0101**
Transferencia	**Resultado**			
a una palabra doble:	1111 1111	0000 1111	1111 0000	0101 0101
a un byte:				0101 0101
a una palabra:			1111 0000	0101 0101
Byte				**1111 0000**
Transferencia	**Resultado**			
a un byte:				1111 0000
a una palabra:			0000 0000	1111 0000
a una palabra doble:	0000 0000	0000 0000	0000 0000	1111 0000

Ejemplo



La operación se ejecuta si E 0.0 es 1. El contenido de MW10 se copia entonces en la palabra 12 del bloque de datos que está abierto.

La salida A 4.0 será "1" si se ejecuta la operación.

Al encontrarse los circuitos del ejemplo dentro de un área MCR activada:

Si el MCR está conectado, los datos se copian de MW10 a DBW12 , tal como se ha explicado arriba en la descripción de la operación.

Si el MCR está desconectado, en DBW12 se escribe el valor "0".

10 Operaciones de control del programa

10.1 Lista de operaciones de control del programa

Descripción

Se dispone de las operaciones de control del programa siguientes:

- ---(Call) Llamar a una FC/SFC sin parámetros
- CALL_FB Llamar a un FB desde un cuadro
- CALL_FC Llamar a una FC desde un cuadro
- CALL_SFB Llamar a un SFB desde un cuadro
- CALL_SFC Llamar a una SFC desde un cuadro
- Llamar a una multiinstancia
- Llamar a un bloque de una librería

- Notas importantes sobre el uso de la función MCR
- ---(MCR<) Conectar un Master Control Relay
- ---(MCR>) Desconectar un Master Control Relay
- ---(MCRA) Inicio de un Master Control Relay
- ---(MCRD) Final de un Master Control Relay

- RET Retorno

10.2 ---(Call) Llamar a una FC/SFC sin parámetros

Símbolo

< N.º de FC/SFC>

---(CALL)

Parámetro	Tipo de datos	Area de memoria	Descripción
< N.º de FC/SFC>	BLOCK_FC BLOCK_SFC	-	Número de FC/SFC; el área varía según la CPU que se utilice

Descripción de la operación

---(Call) (Llamar a una FC/SFC sin parámetros) llama a una función (FC) o a una función de sistema (SFC) que no tiene parámetros. La llamada se ejecuta únicamente si el RLO de la bobina CALL es "1". Al ejecutarse la operación ---(CALL) sucede lo siguiente:

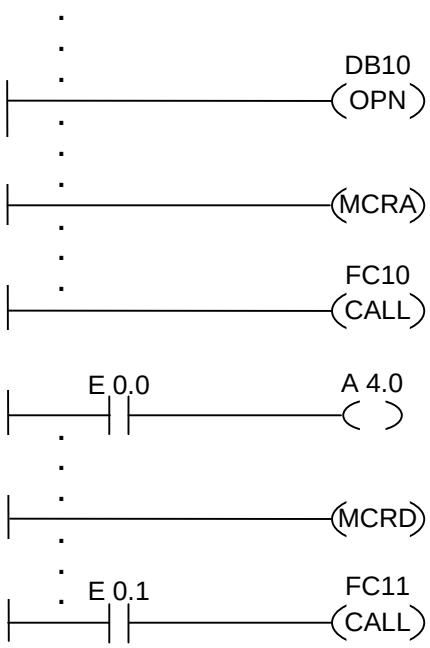
- se memoriza la dirección de retorno del bloque que efectúa la llamada,
- se sustituye el área de datos locales anterior por el área de datos locales actual,
- se crea un nuevo área de datos locales para la función que se ha llamado.
- se desplaza el bit MA (bit MCR activo) a la pila BSTACK y

Seguidamente, la ejecución del programa continúa en la función o función de sistema que se ha llamado.

Palabra de estado

		RB	A1	A0	OV	OS	OR	STA	RLO	/ER
Llamada condicional:	se escribe:	-	-	-	-	0	0	1	1	0
Llamada absoluta:	se escribe:	-	-	-	-	0	0	1	-	0

Ejemplo



Los circuitos del esquema de contactos representados en el ejemplo son elementos del programa de un bloque de función escrito por el usuario. En este bloque de función se abre DB10 y se activa el MCR. Si se ejecuta la llamada absoluta a la FC10 sucede lo siguiente:

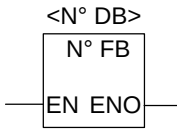
Se memorizan la dirección de retorno del bloque de función que efectúa la llamada, los datos seleccionados para el DB10 y los datos para el bloque de datos de instancia perteneciente al bloque de función que efectúa la llamada. El bit MA que fue puesto a "1" por la operación MCRA se desplaza a la pila BSTACK y seguidamente es puesto a "0" para el bloque (FC10) que se ha llamado. La ejecución del programa continúa en FC10. Si FC10 necesita el MCR, hay que volver a activar el MCR FC10. Una vez finalizada la ejecución de FC10, el programa vuelve al FB que efectúa la llamada. El bit MA se restablece. El DB10 y el bloque de datos de instancia perteneciente al bloque de función escrito por el usuario convierten de nuevo en los DB actuales. El programa continúa en el siguiente circuito, donde se asigna el estado de E 0.0 a la salida A 4.0. La llamada a FC11 es una llamada condicional. Esta llamada se ejecuta únicamente si E 0.1 es 1. Al ejecutarse la llamada, el control de programa es transferido a FC11, del mismo modo que se ha descrito para FC10, y retorna después de haberse ejecutado FC11.

Nota

Después de retornar al bloque que efectúa la llamada puede ocurrir que el DB que se había abierto anteriormente ahora ya no esté abierto. Sírvase tener en cuenta la indicación al respecto en el archivo README.

10.3 CALL_FB Llamar a un FB desde un cuadro

Símbolo



El símbolo varía según el bloque de función (dependiendo de si hay parámetros o de cuántos parámetros existen). EN, ENO y el nombre o número del FB tienen que estar siempre presentes.

Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
N.° de FB N.° de DB	BLOCK_FB BLOCK_DB	- -	Número del FB/DB; el área varía según la CPU que se utiliza

Descripción de la operación

CALL_FB (Llamar a un FB desde un cuadro) se ejecuta si EN es 1. Al ejecutarse la operación CALL_FB sucede lo siguiente:

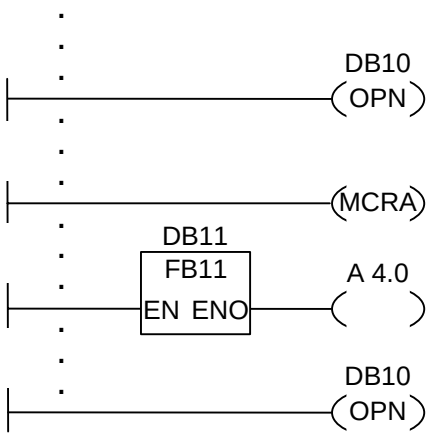
- se memoriza la dirección de retorno del bloque que efectúa la llamada,
- se memorizan los datos seleccionados para los dos bloques de datos actuales (DB y DB de instancia),
- se sustituye el área de datos locales anterior por el área de datos locales actual,
- se crea un nuevo área de datos locales para el bloque de función llamado.
- se desplaza el bit MA (bit MCR activo) a la pila BSTACK y

Seguidamente, la ejecución del programa continúa en el bloque de función llamado. Para determinar ENO se consulta el bit RB, el usuario tiene que asignarle a éste, en el bloque llamado, con ---(SAVE) el estado deseado (evaluación de errores).

Palabra de estado

		RB	A1	A0	OV	OS	OR	STA	RLO	/ER
condicional:	se escribe:	x	-	-	-	0	0	x	x	x
absoluto:	se escribe:	-	-	-	-	0	0	x	x	x

Ejemplo



Los circuitos del esquema de contactos arriba representados son elementos del programa de un bloque de función escrito por el usuario. En este bloque de función se abre DB10 y se activa el MCR. Si se ejecuta la llamada absoluta al FB11 sucede lo siguiente:

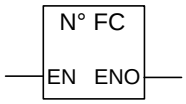
Se memorizan la dirección de retorno del bloque de función que efectúa la llamada, los datos seleccionados para el DB10 y los datos para el bloque de datos de instancia perteneciente al bloque de función que efectúa la llamada. El bit MA que fue puesto a "1" por la función MCRA se desplaza a la pila BSTACK y seguidamente es puesto a "0" para el bloque FB11 llamado. La ejecución del programa continúa en FB11. Si el FB11 necesita el MCR, hay que volver a activar el MCR en el bloque de función. El estado del RLO tiene que almacenarse a través de la operación ---(SAVE) en el bit RB para poder evaluar los posibles errores en el FB que efectúa la llamada. Una vez finalizada la ejecución del FB11, el programa vuelve al bloque de función que efectúa la llamada. El bit MA se restablece y el bloque de datos de instancia perteneciente al bloque de función escrito por el usuario se vuelve a convertir en el DB actual. Si el FB11 es ejecutado correctamente, ENO es 1 y, por tanto, A 4.0 es 1.

Nota

El número del bloque de datos abierto anteriormente se pierde al llamar FB/SFB. Habrá que volver a abrir el DB que se necesite.

10.4 CALL_FC Llamar a una FC desde un cuadro

Símbolo



El símbolo varía según la función (dependiendo de si hay parámetros o de cuántos parámetros existen). EN, ENO y el nombre o número de la FC tienen que estar siempre presentes.

Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
N.° de FC	BLOCK_FC	-	Número de la FC; el área varía según la CPU que se utiliza

Descripción de la operación

CALL_FC (Llamar a una FC desde un cuadro) llama a una función (FC). La llamada se ejecuta si EN es 1. Al ejecutarse la operación CALL_FC sucede lo siguiente:

- se memoriza la dirección de retorno del bloque que efectúa la llamada,
- se sustituye el área de datos locales anterior por el área de datos locales actual,
- se crea un nuevo área de datos locales para la función que se ha llamado.
- se desplaza el bit MA (bit MCR activo) a la pila BSTACK y

Seguidamente, la ejecución del programa continúa en la función que se ha llamado.

Para determinar ENO se consulta el bit RB, el usuario tiene que asignarle a éste, en el bloque llamado, con ---(SAVE) el estado deseado (evaluación de errores).

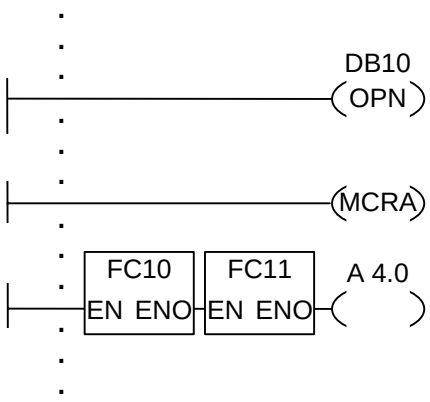
Si llama una FC y la tabla de declaración de variables del bloque llamado dispone de declaraciones del tipo IN, OUT y IN_OUT, dichas variables se visualizarán en la lista de parámetros formales en el programa del bloque que realiza la llamada.

En la llamada de las FCs **es imprescindible** asignar parámetros actuales a los parámetros formales en el punto donde se encuentre la llamada. Si hubiere valores iniciales en la declaración de la FC carecen de importancia.

Palabra de estado

		RB	A1	A0	OV	OS	OR	STA	RLO	/ER
condicional:	se escribe:	x	-	-	-	0	0	x	x	x
absoluto:	se escribe:	-	-	-	-	0	0	x	x	x

Ejemplo



Los circuitos del esquema de contactos representados en el ejemplo son elementos del programa de un bloque de función escrito por el usuario. En este bloque de función se abre DB10 y se activa el MCR. Si se ejecuta la llamada absoluta a la FC10 sucede lo siguiente:

Se memorizan la dirección de retorno del bloque de función que efectúa la llamada, los datos seleccionados para el DB10 y los datos para el bloque de datos de instancia perteneciente al bloque de función que efectúa la llamada. El bit MA que fue puesto a "1" por la operación MCRA se desplaza a la pila BSTACK y seguidamente es puesto a "0" para el bloque FC10 que se ha llamado. La ejecución del programa continúa en FC10. Si FC10 necesita el MCR, hay que volver a activar el MCR en FC10. El estado del RLO tiene que almacenarse a través de la operación ---(SAVE) en el bit RB para poder realizar una evaluación de errores en el FB que ejecuta la llamada. Una vez finalizada la ejecución de la FC10, el programa vuelve al bloque de función que efectúa la llamada. El bit MA se restablece. Al finalizar la ejecución de la FC10 el programa continúa, en función de la señal de ENO, en el FB que efectúa la llamada:

ENO = 1 se ejecuta la FC11

ENO = 0 la ejecución comienza en el segmento siguiente.

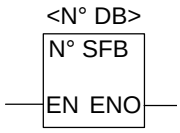
Si la ejecución de FC11 también es correcta, ENO es 1 y, por tanto, A 4.0 es 1.

Nota

Después de retornar al bloque que efectúa la llamada puede ocurrir que el DB que se había abierto anteriormente ahora ya no esté abierto. Sírvase tener en cuenta la indicación al respecto en el archivo README.

10.5 CALL_SFB Llamar a un SFB desde un cuadro

Símbolo



El símbolo varía según el bloque de función de sistema (dependiendo de si hay parámetros o de cuántos parámetros existen). EN, ENO y el nombre o número del SFB tienen que estar siempre presentes.

Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
N.° de SFB N.° de DB	BLOCK_SFB BLOCK_DB	- -	Número del SFB; el área varía según la CPU que se utiliza

Descripción de la operación

CALL_SFB (Llamar a un SFB desde un cuadro) se ejecuta si EN es 1. Al ejecutarse la operación CALL_SFB sucede lo siguiente:

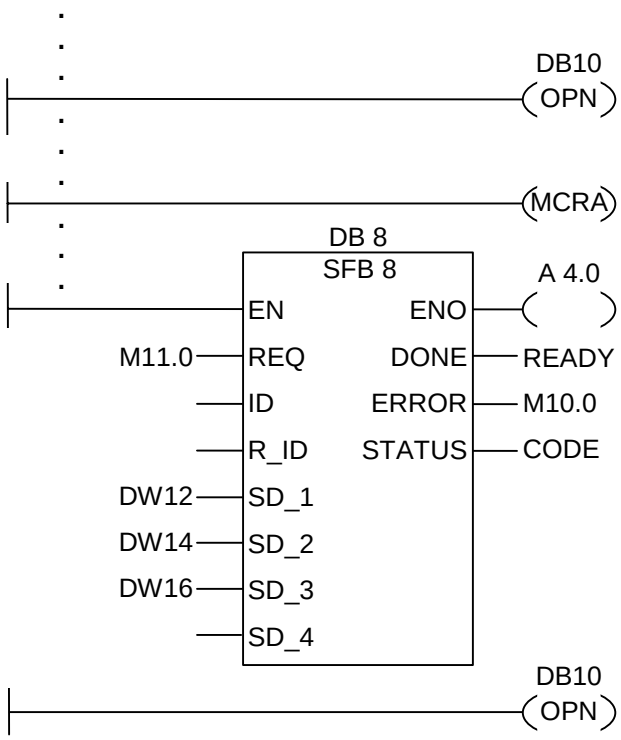
- se memoriza la dirección de retorno del bloque que efectúa la llamada,
- se memorizan los datos seleccionados para los dos bloques de datos actuales (DB y DB de instancia),
- se sustituye el área de datos locales anterior por el área de datos locales actual,
- se crea un nuevo área de datos locales para el bloque de función de sistema que se ha llamado.
- se desplaza el bit MA (bit MCR activo) a la pila BSTACK y

Seguidamente, la ejecución del programa continúa en el bloque de función de sistema llamado. ENO es "1" si la llamada al bloque de función de sistema (EN = 1) se ejecutó sin errores.

Palabra de estado

		RB	A1	A0	OV	OS	OR	STA	RLO	/ER
condicional:	se escribe:	x	-	-	-	0	0	x	x	x
absoluto:	se escribe:	-	-	-	-	0	0	x	x	x

Ejemplo



Los circuitos del esquema de contactos arriba representados son elementos del programa de un bloque de función escrito por el usuario. En este bloque de función se abre DB10 y se activa el MCR. Al ejecutarse la llamada absoluta al SFB8 sucede lo siguiente:

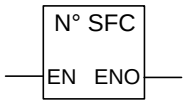
Se memorizan la dirección de retorno del bloque de función que efectúa la llamada, los datos seleccionados para el DB10 y los datos para el bloque de datos de instancia pertenciente al bloque de función que efectúa la llamada. El bit MA que fue puesto a "1" por la función MCRA se desplaza a la pila BSTACK y seguidamente puesto a "0" para el SFB8 llamado. La ejecución del programa continúa en SFB8. Una vez finalizada la ejecución de SFB8, el programa vuelve al bloque de función que efectúa la llamada. El bit MA se restablece y el bloque de datos de instancia pertenciente al bloque de función escrito por el usuario se vuelve a convertir en el DB de instancia actual. Si el SFB8 es ejecutado correctamente, ENO es 1 y, por tanto, A4.0 es 1.

Nota

El número del bloque de datos abierto anteriormente se pierde al llamar FB/SFB. Habrá que volver a abrir el DB que se necesite.

10.6 CALL_SFC Llamar a una SFC desde un cuadro

Símbolo



El símbolo varía según la función de sistema (dependiendo de si hay parámetros o de cuántos parámetros existen). EN, ENO y el nombre o número de SFC tienen que estar siempre presentes.

Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
N.° de SFC	BLOCK_SFC	-	Número de SFC; el área varía según la CPU que se utiliza

Descripción de la operación

CALL_SFC (Llamar a una SFC desde un cuadro) llama a una función de sistema. La llamada se ejecuta si EN es 1. Al ejecutarse la operación CALL_SFC sucede lo siguiente:

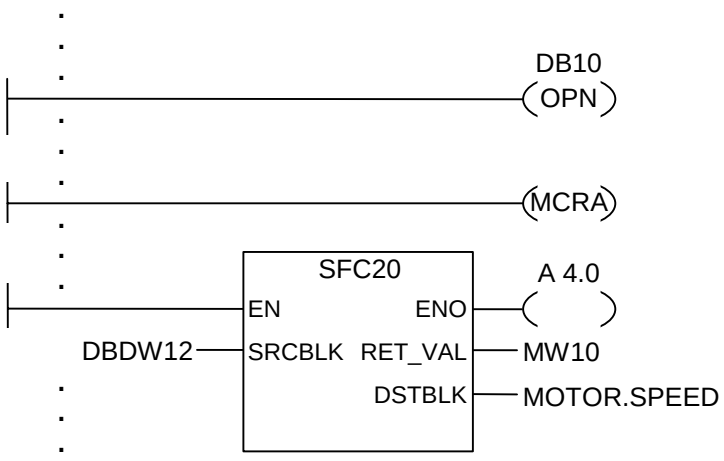
- se memoriza la dirección de retorno del bloque que efectúa la llamada,
- se sustituye el área de datos locales anterior por el área de datos locales actual,
- se crea un nuevo área de datos locales para la función que se ha llamado.
- se desplaza el bit MA (bit MCR activo) a la pila BSTACK y

Seguidamente, la ejecución del programa continúa en la función de sistema que se ha llamado. ENO es "1" si la llamada a la función (EN = 1) se produjo sin errores.

Palabra de estado

		RB	A1	A0	OV	OS	OR	STA	RLO	/ER
condicional:	se escribe:	x	-	-	-	0	0	x	x	x
absoluto:	se escribe:	-	-	-	-	0	0	x	x	x

Ejemplo



Los circuitos del esquema de contactos arriba representados son elementos del programa de un bloque de función escrito por el usuario. En este bloque se abre DB10 y se activa el MCR. Si se ejecuta la llamada absoluta a la SFC20 sucede lo siguiente:

Se memorizan la dirección de retorno del bloque de función que efectúa la llamada, los datos seleccionados para el DB10 y los datos para el bloque de datos de instancia perteneciente al bloque de función que efectúa la llamada. El bit MA que fue puesto a "1" por la operación MCRA se desplaza a la pila BSTACK y seguidamente es puesto a "0" para el bloque SFC20 que se ha llamado. La ejecución del programa continúa en SFC20. Una vez finalizada la ejecución de la SFC20, el programa vuelve al bloque de función que efectúa la llamada. El bit MA se restablece.

Una vez finalizada la ejecución de la SFC20 el programa continúa, en función de cuál sea la señal en ENO, en el FB que efectúa la llamada:

ENO = 1 A 4.0 = 1

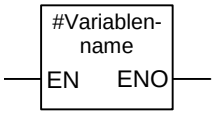
ENO = 0 A 4.0 = 0

Nota

Después de retornar al bloque que efectúa la llamada puede ocurrir que el DB que se había abierto anteriormente ahora ya no esté abierto. Sírvase tener en cuenta la indicación al respecto en el archivo README.

10.7 Llamar a una multiinstancia

Símbolo



El símbolo varía según cuál sea la multiinstancia (dependiendo de si hay parámetros o de cuántos parámetros existen). EN, ENO y el nombre o número del FB/SFB siempre tienen que estar presentes.

Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
# Variablenname	FB, SFB	-	Nombre de la multiinstancia

Descripción

Para generar una multiinstancia se debe declarar una variable estática del tipo de datos de un blqoue de función. Sólo las multiinstancias ya declaradas se listarán en el catálogo de elementos del programa.
El símbolo de una multiinstancia se modifica dependiendo de si hay parámetros y, en caso afirmativo, de qué tipo de parámetros se trata. EN, EN0 y el nombre de la variable existen siempre.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	0	0	x	x	x

10.8 Llamar a un bloque de una librería

Se ofrecen las librerías que el Administrador SIMATIC haya encontrado. De ese conjunto de librerías, se pueden escoger:

- los bloques que están integrados en el sistema operativo de la CPU utilizada (librería "Standard Library" para proyectos de STEP 7 de la versión 3 y "stdlibs (V2)" para proyectos de STEP 7 de la versión 2),
- los bloques que el usuario mismo haya guardado en librerías con el fin de poder utilizarlas varias veces.

10.9 Notas importantes sobre el uso de la función MCR

Prestar atención al usar la función en bloques en los que se activó el Master Control Relay con MCRA

- Si está desconectado el MCR, en la parte del programa que se encuentra entre ---(MCR<) y ---(MCR>) todas las asignaciones (T, =) escribirán el valor 0. Esto también es aplicable a **todos** los cuadros que contienen una asignación, incluida la transferencia de parámetros a bloques.
- El MCR se desconecta siempre que un RLO = 0 preceda a una instrucción **MCR(**.

Peligro: STOP del AS o comportamiento no definido en runtime

Para el cálculo de direcciones, el compilador también tiene acceso de escritura a los datos locales después de las variables temporales definidas en VAR_TEMP. Para ello, las siguientes secuencias de comandos ponen el PLC en STOP o provocan comportamientos indefinidos en runtime:

Acceso a parámetros formales

- Accesos a componentes de parámetros FC compuestos del tipo STRUCT, UDT, ARRAY, STRING.
- Accesos a componentes de parámetros FB compuestos del tipo STRUCT, UDT, ARRAY, STRING del área IN_OUT en un bloque apto para multiinstancia (de la versión 2).
- Accesos a parámetros de un FB multiinstancia (de la versión 2) si su dirección es mayor que 8180.0.
- El acceso en el FB multiinstancia (de la versión 2) a un parámetro del tipo BLOCK_DB abre el DB 0. Los siguientes accesos a datos ponen la CPU en STOP. Con TIMER, COUNTER, BLOCK_FC, BLOCK_FB se utiliza siempre T 0, Z 0, FC 0 o FB 0.

Transferencia de parámetros

- Calls en las que se transfieren parámetros.

KOP/FUP

- Las ramas T y los conectores en KOP o FUP arrancan con RLO = 0.

Remedio

Active las órdenes mencionadas en función del MCR:

1. **Desactive** el Master Control Relay con Final de un Master Control Relay **antes** de la instrucción correspondiente o antes del segmento involucrado.
2. **Active** nuevamente el Master Control Relay con Inicio de un Master Control Relay **después de** la instrucción correspondiente o después del segmento involucrado.

10.10 ---(MCR<) Conectar un Master Control Relay

Notas importantes sobre el uso de la función MCR

Símbolo

---(MCR<)

Descripción de la operación

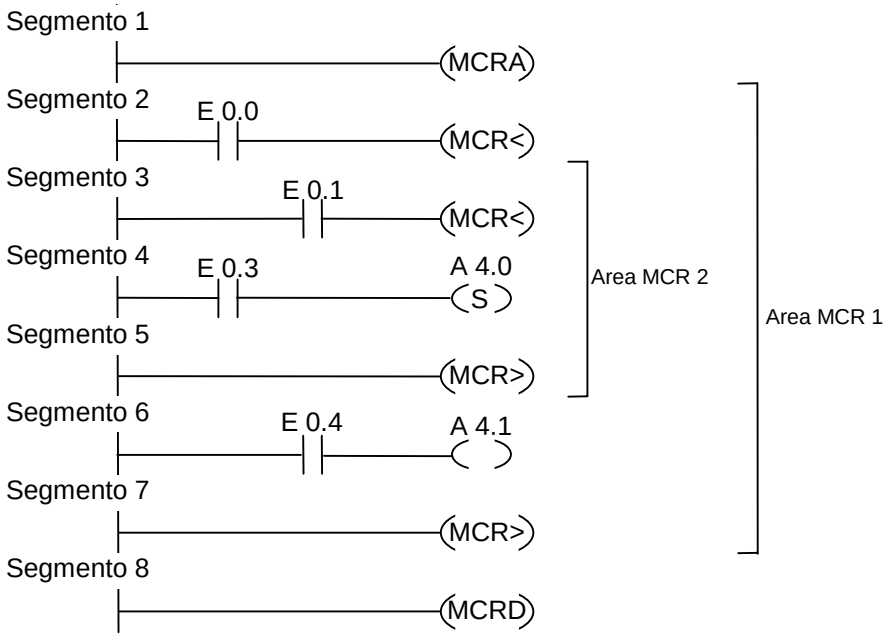
---(MCR<) (Conectar un Master Control Relay) almacena el RLO en la pila MCR y abre un área MCR. La pila de paréntesis MCR es una pila LIFO (last in, first out) que da cabida, como máximo, a 8 registros (8 niveles). Si la pila ya está llena, la operación ---(MCR<) provoca un error de la pila MCR (MCRF). Los siguientes elementos dependen del MCR y varían según cuál sea el estado de señal del RLO que se almacena en la pila MCR mientras está abierta un área MCR:

- --(#) Conector
- --( ) Bobina de relé, salida
- --(S) Activar salida
- --(R) Desactivar salida
- RS Desactivar flip-flop de activación
- SR Activar flip-flop de desactivación
- MOVE Asignar un valor

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	1	-	0

Ejemplo



El circuito MCRA activa el MCR. Entonces puede crearse hasta un máximo de ocho áreas MCR anidadas. En este ejemplo hay dos áreas MCR. Las operaciones se ejecutan de la siguiente manera:

E 0.0 = 1 (el MCR está ON en el área 1): A la salida A 4.1 se le asigna el estado de señal de la entrada E 0.4.

E 0.0 = 0 (el MCR está OFF en el área 1): la salida A 4.1 es "0", independientemente de cuál sea el estado de la entrada E 0.4.

E 0.1 Y E 0.1 = 1 (el MCR está ON en el área 2): la salida A 4.0 se pone a "1" si E 0.3 es 1

E 0.0 Y E 0.1 = 0 (el MCR está OFF en el área 2): la salida A 4.0 no varía, independientemente del estado de E 0.3

10.11 ---(MCR>) Desconectar un Master Control Relay

Símbolo

---(MCR>)

Descripción de la operación

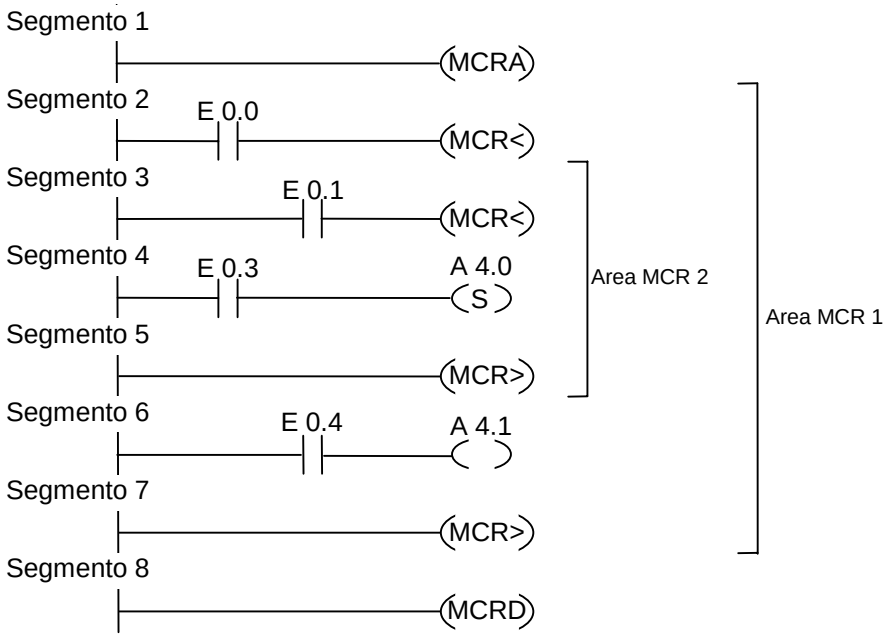
---(MCR>) (Desconectar un Master Control Relay) borra un registro RLO de la pila MCR. La pila de paréntesis MCR es una pila LIFO (last in, first out) que da cabida a un máximo de 8 registros (8 niveles). Si la pila ya está vacía, la operación ---(MCR >) provoca un error de la pila MCR (MCRF). Los siguientes elementos dependen del MCR y se ven afectados por el estado de señal del RLO que se almacena en la pila MCR mientras está abierta un área MCR:

- --(#) Conector
- --( ) Bobina de relé, salida
- --(S) Activar salida
- --(R) Desactivar salida
- RS Desactivar flip-flop de activación
- SR Activar flip-flop de desactivación
- MOVE Asignar un valor

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	1	-	0

Ejemplo



La operación ---(MCRA) activa el MCR. En este caso puede crearse hasta un máximo de ocho áreas MCR. En este ejemplo hay dos áreas MCR. El primero circuito ---(MCR>) (MCR OFF) forma parte del segundo circuito ---(MCR<) (MCR ON). Todos los circuitos que hay entre estos dos pertenecen al área MCR 2. Las funciones se ejecutan de la siguiente manera:

E 0.0 = 1: el estado de señal de la entrada E 0.4 se asigna a la salida A 4.1

E 0.0 = 0: la salida A 4.1 es "0", independientemente del estado de E 0.4

E 0.0 Y E 0.1 = 1: la salida A 4.0 se pone a "1" si E 0.3 es 1

E 0.0 Y E 0.1 = 0: la salida A 4.0 no varía, independientemente del estado de E 0.3

10.12 ---(MCRA) Inicio de un Master Control Relay

Símbolo

---(MCRA)

Descripción de la operación

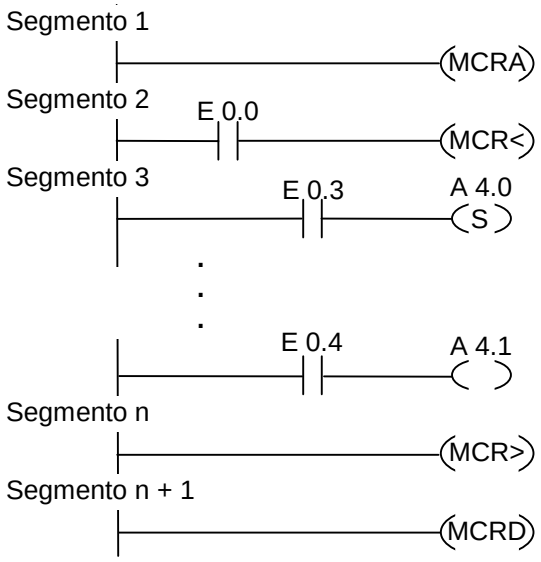
---(MCRA) (Inicio de un Master Control Relay) activa un Master Control Relay. Una vez efectuada esta operación se pueden programar las áreas MCR utilizando las siguientes operaciones:

- ---(MCR<)
- ---(MCR>)

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	-	-	-	-

Ejemplo



El circuito MCRA activa el MCR. Los circuitos entre las operaciones MCR< y MCR> (salidas A 4.0, A 4.1) se ejecutan de la siguiente manera:

E 0.0 = 1 (MCR está ON): la salida A 4.0 se pone a "1" si la entrada E 0.3 está en el estado "1", y no cambia si la entrada E 0.3 está en el estado "0". A la salida A 4.1 se le asigna el estado de la entrada E 0.4.

E 0.0 = 0 (MCR está OFF): la salida A 4.0 no varía, independientemente del estado de E 0.3; la salida A 4.1 es "0", independientemente de cuál sea el estado de E0.4.

En el circuito siguiente, la operación ---(MCRD) desactiva el MCR. Esto significa que ya no se pueden programar áreas MCR con las dos operaciones ---(MCR<) y ---(MCR>).

10.13 ---(MCRD) Final de un Master Control Relay

Símbolo

---(MCRD)

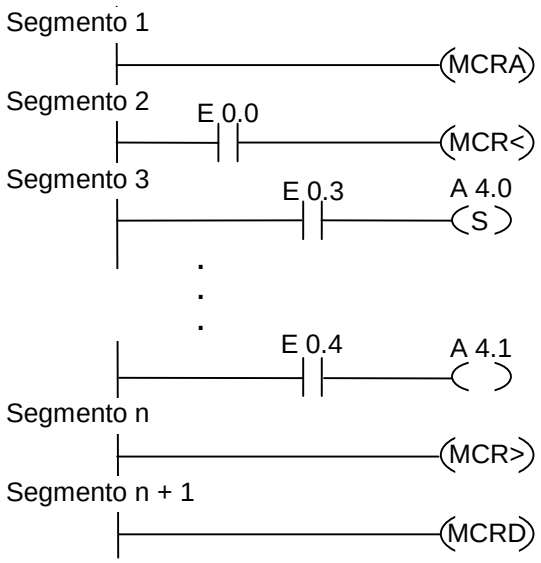
Descripción de la operación

---(MCRD) (Final de un Master Control Relay) desactiva un MCR. Después de esta operación no se pueden programar áreas MCR.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	-	-	-	-

Ejemplo



El circuito ---(MCRA) activa el MCR. Los circuitos entre las operaciones MCR< y MCR> (salidas A 4.0, A 4.1) se ejecutan de la siguiente manera:

E 0.0 = 1 (MCR está ON): la salida A 4.0 se pone a "1" si la entrada E 0.3 está en el estado "1"; A 4.0 no se modifica si la entrada E 0.3 está en el estado "0". A la salida A 4.1 se le asigna el estado de la entrada E 0.4.

E 0.0 = 0 (MCR está OFF): la salida A 4.0 no varía, independientemente del estado de E 0.3, y la salida A 4.1 es "0", independientemente del estado de E 0.4

En el circuito siguiente, la operación **---**(MCRD) desactiva el MCR. Esto significa que ya no se pueden programar áreas MCR con la pareja de operaciones ---(MCR<) y ---(MCR>).

10.14 ---(RET) Retorno

Símbolo

---(RET)

Descripción de la operación

RET (Retorno) sirve para salir de los bloques condicionalmente. Para emplear esta salida se necesita una combinación lógica previa.

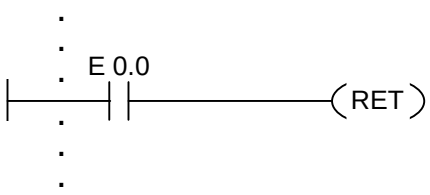
Palabra de estado

Retorno condicional (retorno, si RLO = 1):

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	*	-	-	-	0	0	1	1	0

* La operación **RET** se representa internamente en la secuencia "SAVE; BEB;", por lo que también se influye sobre el bit RB.

Ejemplo



Se sale del bloque si E 0.0 es 1.

11 Operaciones de desplazamiento y rotación

11.1 Operaciones de desplazamiento

11.1.1 Lista de operaciones de desplazamiento

Descripción

Las operaciones de desplazamiento sirven para desplazar bit a bit el contenido de la entrada IN, hacia la izquierda o hacia la derecha (v. Registros de la CPU). El desplazamiento hacia la izquierda multiplica el contenido de la entrada IN por potencias de 2; el desplazamiento hacia la derecha divide el contenido de la entrada IN por potencias de 2. Por ejemplo, desplazando el equivalente binario del valor decimal 3 tres bits hacia la izquierda se obtiene en el acumulador el equivalente binario del valor decimal 24. Desplazando el equivalente binario del valor decimal 16 dos bits hacia la derecha se obtiene en el acumulador el equivalente binario del valor decimal 4.

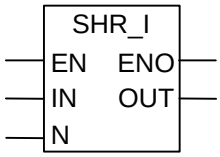
El número que se introduce en el parámetro de entrada N indica el número de bits a desplazar. Las posiciones que quedan libres después de ejecutar la operación de desplazamiento se rellenan con ceros o con el estado de señal del bit de signo (0 significa positivo y 1 significa negativo). El estado de señal del último bit desplazado se carga en el bit A1 de la palabra de estado. Los bits A0 y OV de la palabra de estado se ponen a 0. Para interpretar el bit A1 pueden utilizarse las operaciones de salto.

Se dispone de las operaciones de desplazamiento siguientes:

- SHR_I Desplazar entero a la derecha
- SHR_DI Desplazar entero doble a la derecha
- SHL_W Desplazar 16 bits a la izquierda
- SHR_W Desplazar 16 bits a la derecha
- SHL_DW Desplazar 32 bits a la izquierda
- SHR_DW Desplazar 32 bits a la derecha

11.1.2 SHR_I Desplazar entero a la derecha

Símbolo

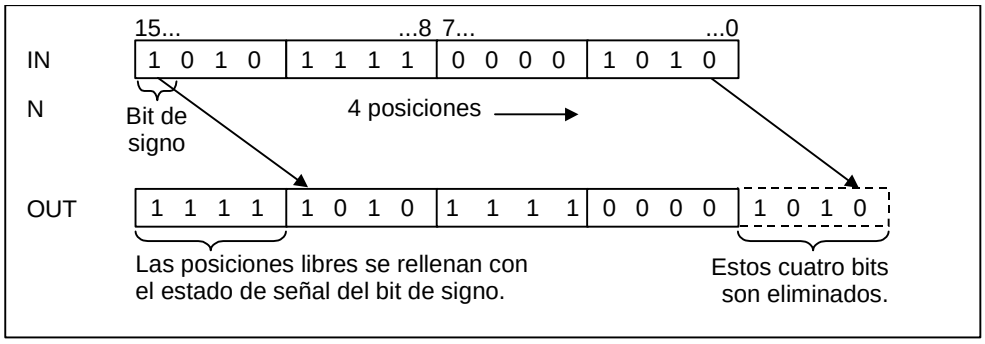


Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	INT	E, A, M, L, D	Valor a desplazar
N	WORD	E, A, M, L, D	Número de posiciones (bits) a desplazar
OUT	INT	E, A, M, L, D	Resultado de la operación de desplazamiento

Descripción de la operación

SHR_I (Desplazar entero a la derecha) se activa si la entrada de habilitación (EN) tiene el estado de señal "1". Con la operación SHR_I se desplazan los bits 0 a 15 de la entrada IN bit a bit a la derecha. A los bits 16 a 31 no les afecta esta operación de desplazamiento. La entrada N indica el número de posiciones de bit en que se va a efectuar un desplazamiento. Si N es mayor que 16, la instrucción trabaja como si N fuera igual a 16. Las posiciones de bit que se arrastran de la izquierda para ocupar las posiciones libres reciben el estado de señal del bit 15 (este es el bit de signo del entero). Esto significa que estas posiciones de bit se ocupan con el valor "0" si se trata de un entero positivo, y que se ocupan con el valor "1" si se trata de un entero negativo. El resultado de la operación de desplazamiento queda depositado en la salida OUT. La operación SHR_I pone los bits A0 y OV a "0" si N es diferente de 0.

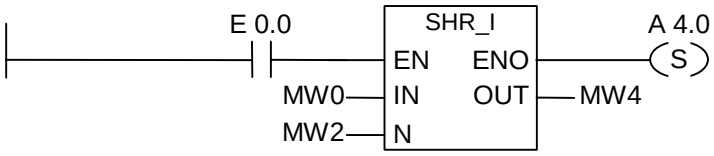
El estado de señal de ENO es igual al de EN.



Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	x	x	x	x	-	x	x	x	1

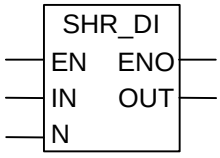
Ejemplo



El cuadro SHR_I se activa si E 0.0 es "1". MW0 se carga y se desplaza a la derecha tantos bits como indica MW2. El resultado se escribe en MW4. La salida A 4.0 se pone a 1.

11.1.3 SHR_DI Desplazar entero doble a la derecha

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	DINT	E, A, M, L, D	Valor a desplazar
N	WORD	E, A, M, L, D	Número de posiciones (bits) a desplazar
OUT	DINT	E, A, M, L, D	Resultado de la operación de desplazamiento

Descripción de la operación

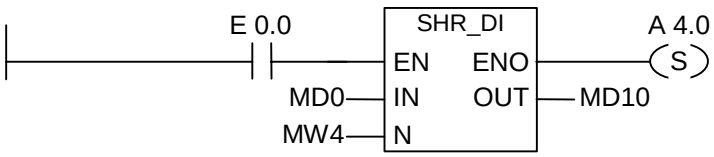
SHR_DI (Desplazar entero de 32 bits a la derecha) se activa si la entrada de habilitación (EN) tiene el estado de señal "1". Con la operación SHR_DI se desplazan los bits 0 a 31 de la entrada IN bit a bit a la derecha. La entrada N indica el número de posiciones de bit en que se va a efectuar un desplazamiento. Si N es mayor que 32, la instrucción trabaja como si N fuera igual a 32. Las posiciones de bit que se arrastran de la izquierda para ocupar las posiciones libres reciben el estado de señal del bit 31 (este es el bit de signo del entero). Esto significa que estas posiciones de bit se ocupan con el valor "0" si se trata de un entero positivo, y que se ocupan con el valor "1" si se trata de un entero negativo. El resultado de la operación de desplazamiento queda depositado en la salida OUT. La operación SHR_DI pone los bits A0 y OV a "0" si N es diferente de 0.

El estado de señal de ENO es igual al de EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	-	x	x	x	1

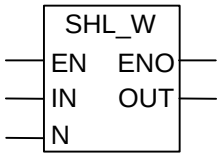
Ejemplo



El cuadro SHR_DI se activa si E 0.0 es 1. MD0 se carga y se desplaza a la derecha tantos bits como indica MW4. El resultado se escribe en MD10. La salida A 4.0 se pone a 1.

11.1.4 SHL_W Desplazar 16 bits a la izquierda

Símbolo

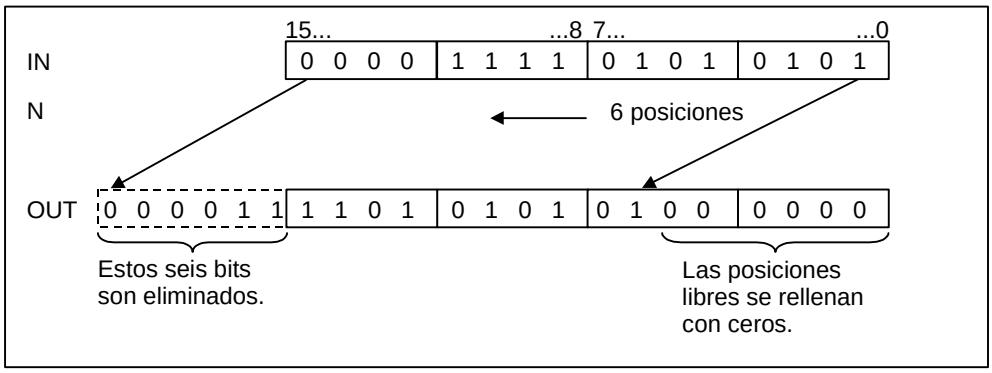


Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	WORD	E, A, M, L, D	Valor a desplazar
N	WORD	E, A, M, L, D	Número de posiciones (bits) a desplazar
OUT	WORD	E, A, M, L, D	Resultado de la operación de desplazamiento

Descripción de la operación

SHL_W (Desplazar 16 bits a la izquierda) se activa si la entrada de habilitación (EN) tiene el estado de señal "1". Con la operación SHL_W se desplazan los bits 0 a 15 de la entrada IN bit a bit a la izquierda. A los bits 16 a 31 no les afecta la operación de desplazamiento. La entrada N indica el número de posiciones de bit en que se va a efectuar un desplazamiento. Si N es mayor que 16, la instrucción en la salida OUT escribe un "0" y pone los bits A0 y OV de la palabra de estado a "0". Desde la derecha se desplaza el mismo número (N) de ceros para ocupar las posiciones que quedaron libres. El resultado de la operación de desplazamiento queda depositado en la salida OUT. La operación SHL_W pone a "0" al bit A0 y al bit OV si N es diferente de 0.

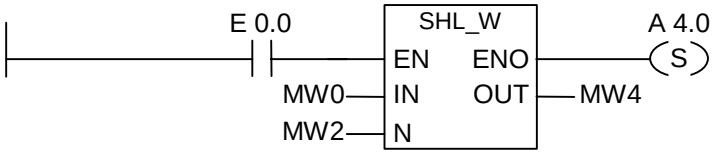
El estado de señal de ENO es igual al de de EN.



Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	x	x	x	x	-	x	x	x	1

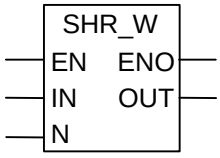
Ejemplo



El cuadro SHL_W se activa si E 0.0 es 1. MW0 se carga en el ACU 1 y se desplaza a la izquierda tantos bits como indica MW2. La palabra del resultado se escribe en MW4. La salida A 4.0 se pone a 1.

11.1.5 SHR_W Desplazar 16 bits a la derecha

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	WORD	E, A, M, L, D	Valor a desplazar
N	WORD	E, A, M, L, D	Número de posiciones (bits) a desplazar
OUT	WORD	E, A, M, L, D	Palabra del resultado de la operación de desplazamiento

Descripción de la operación

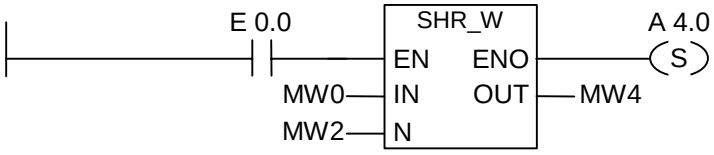
SHR_W (Desplazar 16 bits a la derecha) se activa si la entrada de habilitación (EN) tiene el estado de señal "1". Con la operación SHR_W se desplazan los bits 0 a 15 de la entrada IN bit a bit a la derecha. A los bits 16 a 31 no les afecta esta operación de desplazamiento. La entrada N indica el número de posiciones de bit en las que se va a efectuar un desplazamiento. Si N es mayor que 16, la instrucción escribe un "0" en la salida OUT y pone a "0" los bits A0 y OV de la palabra de estado. Desde la izquierda se desplaza el mismo número (N) de ceros para ocupar las posiciones libres. El resultado de la operación de desplazamiento queda depositado en la salida OUT. La operación SHR_W pone los bits A0 y OV a "0" si N es diferente de 0.

El estado de señal de ENO es igual al de EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	-	x	x	x	1

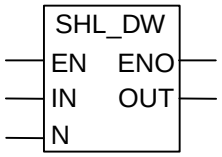
Ejemplo



El cuadro SHR_W se activa si E 0.0 es 1. MW0 se carga y se desplaza a la derecha tantos bits como indica MW2. La palabra del resultado se escribe en MW4. La salida A 4.0 se pone a 1.

11.1.6 SHL_DW Desplazar 32 bits a la izquierda

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	DWORD	E, A, M, L, D	Valor a desplazar
N	WORD	E, A, M, L, D	Número de posiciones (bits) a desplazar
OUT	DWORD	E, A, M, L, D	Palabra doble del resultado de la operación de desplazamiento

Descripción de la operación

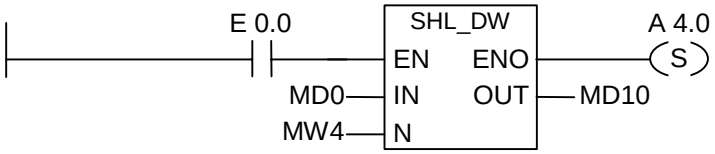
SHL_DW (Desplazar 32 bits a la izquierda) se activa si la entrada de habilitación (EN) tiene el estado de señal "1". Con la operación SHL_DW se desplazan los bits 0 a 31 de la entrada IN bit a bit a la izquierda. La entrada N indica el número de posiciones de bit en que se va a efectuar un desplazamiento. Si N es mayor que 32, la instrucción escribe un "0" en la salida OUT y pone los bits A0 y OV a "0". Desde la derecha se desplaza el mismo número (N) de ceros para ocupar las posiciones libres. La palabra doble del resultado de la operación de desplazamiento queda depositada en la salida OUT. La operación SHL_DW pone los bits A0 y OV a "0" si N es diferente de 0.

El estado de señal de ENO es igual al de EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	-	x	x	x	1

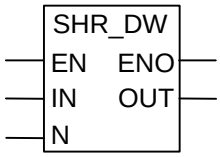
Ejemplo



El cuadro SHL_DW se activa si E 0.0 tiene el estado de señal "1". MD0 se carga y se desplaza a la izquierda tantos bits como indica MW4. La palabra doble del resultado se escribe en MD10. La salida A 4.0 se pone a 1.

11.1.7 SHR_DW Desplazar 32 bits a la derecha

Símbolo

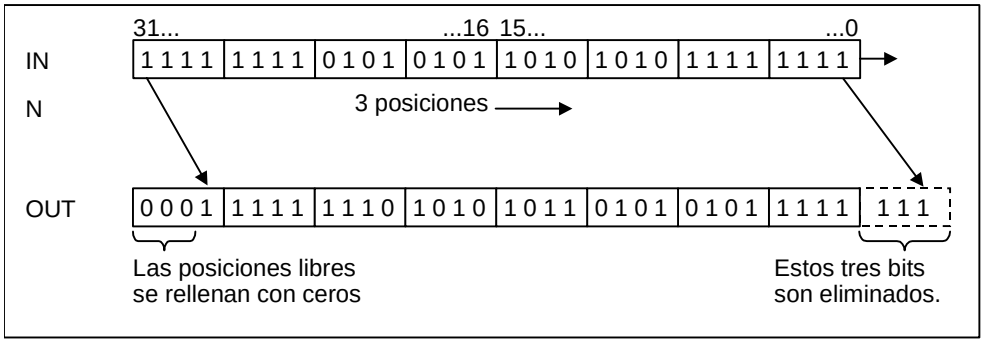


Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	DWORD	E, A, M, L, D	Valor a desplazar
N	WORD	E, A, M, L, D	Número de posiciones (bits) a desplazar
OUT	DWORD	E, A, M, L, D	Palabra doble del resultado de la operación de desplazamiento

Descripción de la operación

SHR_DW (Desplazar 32 bits a la derecha) se activa si la entrada de habilitación (EN) tiene el estado de señal "1". Con la operación SHR_DW se desplazan los bits 0 a 31 de la entrada EN bit a bit a la derecha. La entrada N indica el número de posiciones de bit en que se va a efectuar un desplazamiento. Si N es mayor que 32, la instrucción escribe un "0" en la salida OUT y pone los bits A0 y OV a "0". Desde la izquierda se desplaza el mismo número (N) de ceros para ocupar las posiciones libres. La palabra doble del resultado de la operación de desplazamiento queda depositada en la salida OUT. La operación SHR_DW pone los bits A0 y OV a "0" si N es diferente dE 0.

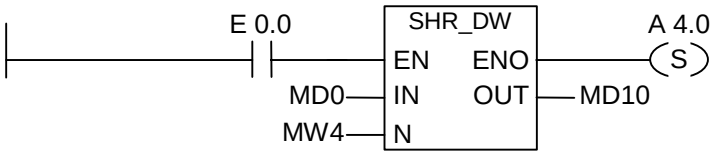
El estado de señal de ENO es igual al de EN.



Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	-	x	x	x	1

Ejemplo



El cuadro SHR_DW se activa si E 0.0 tiene el estado de señal "1". MD0 se carga y se desplaza a la derecha tantos bits como indicA MW4. La palabra doble del resultado se escribe en MD10. La salida A 4.0 se pone a 1.

11.2 Operaciones de rotacíon

11.2.1 Lista de operaciones de rotación

Descripción

Las operaciones de rotación sirven para rotar bit a bit todo el contenido de la entrada IN, hacia la izquierda o hacia la derecha (v. Registros de la CPU). Las posiciones libres de los bits se rellenan con los estados de señal de los bits que se desplazan fuera de la entrada IN. El número que se introduce en el parámetro de entrada N indica el número de bits que se va a rotar.

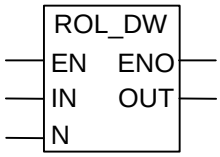
Dependiendo de la operación, la rotación tiene lugar vía el bit A1 de la palabra de estado. El bit A0 de la palabra de estado se pone a 0.

Se dispone de las siguientes operaciones de rotación:

- ROL_DW Rotar 32 bits a la izquierda
- ROR_DW Rotar 32 bits a la derecha

11.2.2 ROL_DW Rotar 32 bits a la izquierda

Símbolo

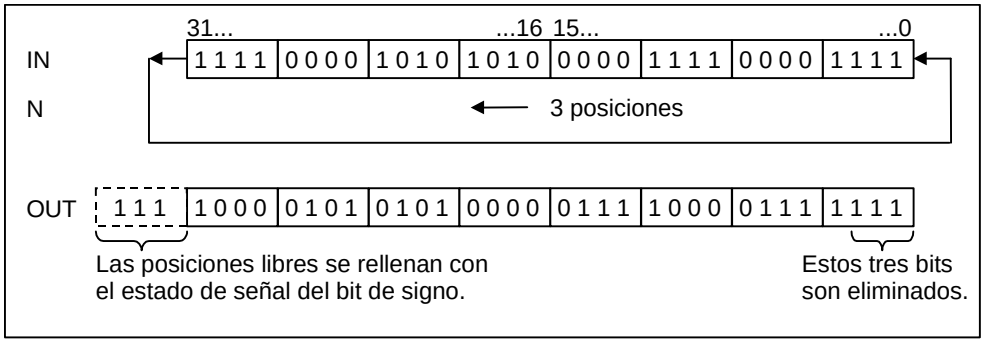


Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	DWORD	E, A, M, L, D	Valor a rotar
N	WORD	E, A, M, L, D	Número de posiciones (bits) a rotar
OUT	DWORD	E, A, M, L, D	Palabra doble del resultado de la operación de rotación

Descripción de la operación

ROL_DW (Rotar 32 bits a la izquierda) se activa si la entrada de habilitación (EN) tiene el estado de señal "1". La operación ROL_DW hace rotar el contenido completo de la entrada IN bit a bit a la izquierda. La entrada N indica el número de posiciones de bit en que se va a efectuar la rotación. Si N es mayore que 32 la palabra doble IN es rotada en ((N-1) modulo 32)+1 posiciones. Las posiciones de bit que se arrastran de la derecha se ocupan con el estado de señal de los bits que fueron rotados a la izquierda (rotación a la izquierda). La palabra doble del resultado de la operación de rotación queda depositada en la salida OUT. La operación ROL_DW pone los bits A0 y OV a "0" si N es diferente de 0.

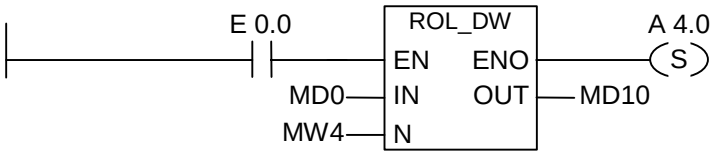
El estado de señal de ENO es igual al de EN.



Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	x	x	x	x	-	x	x	x	1

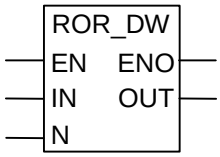
Ejemplo



El cuadro ROL_DW se activa si E 0.0 es 1. MD0 se carga y se rota a la izquierda tantos bits como indica MW4. La palabra doble del resultado se escribe en MD10. La salida A 4.0 se pone a 1.

11.2.3 ROR_DW Rotar 32 bits a la derecha

Símbolo

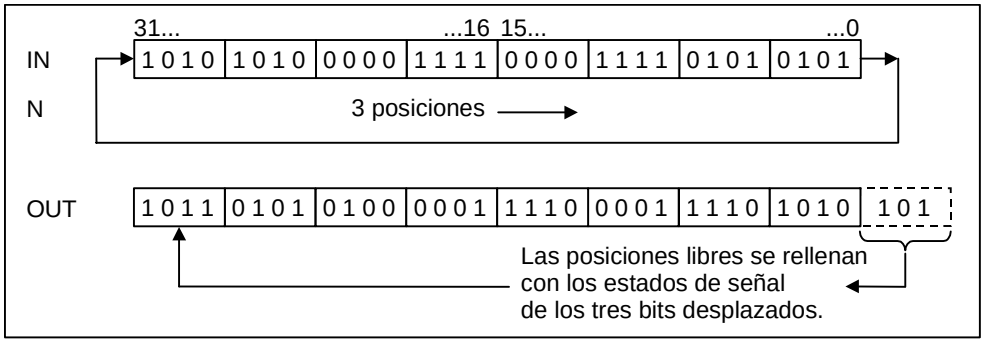


Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN	DWORD	E, A, M, L, D	Valor a rotar
N	WORD	E, A, M, L, D	Número de posiciones (bits) a rotar
OUT	DWORD	E, A, M, L, D	Palabra doble del resultado de la operación de rotación

Descripción de la operación

ROR_DW (Rotar 32 bits a la derecha) se activa si la entrada de habilitación (EN) tiene el estado de señal "1". La operación ROR_DW hace rotar el contenido completo de la entrada IN bit a bit a la derecha. La entrada N indica el número de posiciones de bit en que se va a efectuar la rotación. Si N es mayor que 32 la palabra doble IN es rotada en ((N-1) modulo 32)+1 posiciones. Las posiciones de bit que se arrastran de la izquierda se ocupan con el estado de señal de los bits que fueron rotados a la derecha (rotación a la derecha). La palabra doble del resultado de la operación de rotación queda depositada en la salida OUT. La operación ROR_DW pone los bits A0 y OV a "0" si N es diferente dE 0.

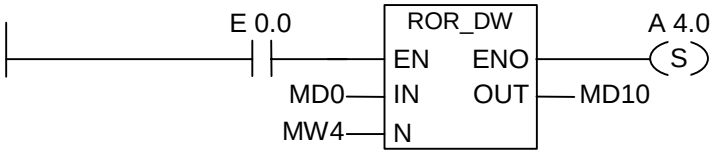
El estado de señal de ENO es igual al de EN.



Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	x	x	x	x	-	x	x	x	1

Ejemplo



El cuadro ROR_DW se activa si E 0.0 es 1. MD0 se carga y se rota a la derecha tantos bits como indica MW4. La palabra doble del resultado se escribe en MD10. La salida A 4.0 se pone a 1.

12 Operaciones con bits de la palabra de estado

12.1 Lista de operaciones con bits de la palabra de estado

Descripción

Las operaciones con bits de la palabra de estado son operaciones lógicas, que trabajan con los bits de la palabra de estado. Estas operaciones reaccionan ante una de las condiciones expuestas a continuación, representadas por uno o más bits de la palabra de estado:

- El bit de resultado binario (RB ---I I---) está activado (es decir, su estado de señal es 1).
- Una función aritmética ha causado un desbordamiento (OV ---I I---) o un desbordamiento memorizado (OS ---I I---).
- El resultado de una función aritmética no es válida (UO ---I I---).
- El resultado de una función aritmética referido a 0 puede ser:
  == 0, <> 0, > 0, < 0, >= 0, <= 0

Si la operación con bits de la palabra de estado está conectada en serie, ésta combina el resultado de la consulta de su estado de señal con el resultado lógico precedente según la tabla de verdad Y. Si la operación con bits de la palabra de estado está conectada en paralelo, ésta combina su resultado con el RLO precedente según la tabla de verdad O.

Palabra de estado

La palabra de estado es un registro de la memoria de la CPU que contiene bits que pueden direccionarse en el operando de las operaciones lógicas con bits y con palabras. La estructura de la palabra de estado:

2^{15}... ...2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER

Los bits de la palabra de estado se pueden evaluar

- en operaciones en coma fija
- en operaciones en coma flotante

12.2 OV ---| |--- Bit de anomalía "desbordamiento"

Símbolo

Descripción de la operación

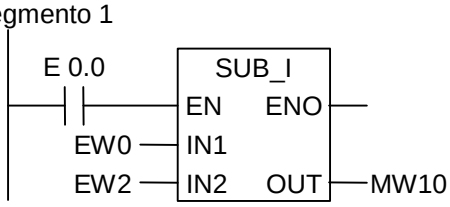
OV ---| |--- (Bit de anomalía "desbordamiento") y **OV ---| / |---** (Negación del bit de anomalía "desbordamiento") detectan los desbordamientos que se producen en la última operación aritmética procesada. Esta detección indica que el resultado se encuentra fuera de los márgenes admisibles, ya sea del positivo o del negativo. El resultado de la consulta está combinado con el RLO mediante una Y lógica cuando las conexiones son en serie. En las conexiones en paralelo el resultado está combinado con el RLO mediante una O lógica.

Palabra de estado

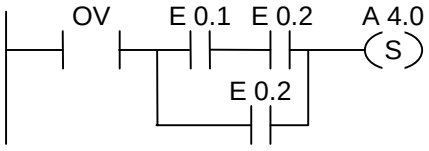
	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo

Segmento 1



Segmento 2



El estado "1" en E 0.0 activa el cuadro. Si el resultado de la operación aritmética EW0 - EW2 queda fuera del margen admisible para un número entero, el bit OV se pone a 1.

La consulta del estado de señal en OV da como resultado "1". A 4.0 se pone a 1 si la consulta de desbordamiento es 1 y si el RLO del segmento 2 es 1.

Nota

La consulta de desbordamiento sólo es necesaria porque hay dos segmentos separados. De no ser así, cuando el resultado queda fuera del margen admisible se puede utilizar la salida ENO de la operación aritmética que tenga el estado "0".

12.3 OS ---| |--- Bit de anomalía "desbordamiento memorizado"

Símbolo

Descripción de la operación

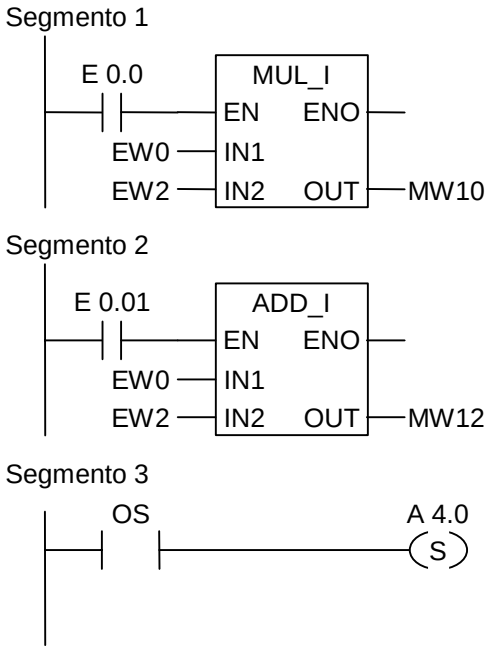
OS ---| |--- (Bit de anomalía "desbordamiento memorizado") y **OS ---| / |---** (Negación del bit de anomalía "desbordamiento memorizado") detectan un desbordamiento cuando se produce en una operación aritmética y lo memorizan. Si el resultado de la operación queda fuera de los márgenes positivo o negativo admisibles, el bit OS de la palabra de estado se activa. A diferencia del bit OV, que se vuelve a escribir en las siguientes operaciones aritméticas, el bit OS memoriza el desbordamiento que se haya producido. El bit OS permanece activado hasta abandonar el bloque.

El resultado de la consulta está combinado con el RLO mediante una Y lógica cuando las conexiones son en serie. En las conexiones en paralelo el resultado está combinado con el RLO mediante una O lógica.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo



El estado "1" en E 0.0 activa el cuadro MUL_I. El estado "1" en E 0.1 activa el cuadro ADD_I. Si el resultado de una de las operaciones aritméticas queda fuera del margen admisible para un número entero, el bit OS en la palabra de estado se pone a "1". La salida A 4.0 se pone a 1 si la consulta de desbordamiento memorizado es 1.

Nota

La consulta de desbordamiento memorizado sólo es necesaria porque hay varios segmentos. De no ser así, se podría conectar la salida ENO de la primera operación aritmética a la entrada EN de la segunda operación aritmética (ejecución en cascada).

12.4 UO ---| |--- Bit de anomalía "operación no válida"

Símbolo

Descripción de la operación

UO ---| |--- (Bit de anomalía "operación no válida") y **UO ---| / |---** (Negación del bit de anomalía "operación no válida") averiguan si el resultado de una operación aritmética con números reales no es válida (o sea, si uno de los valores en la operación aritmética no es un número real válido).

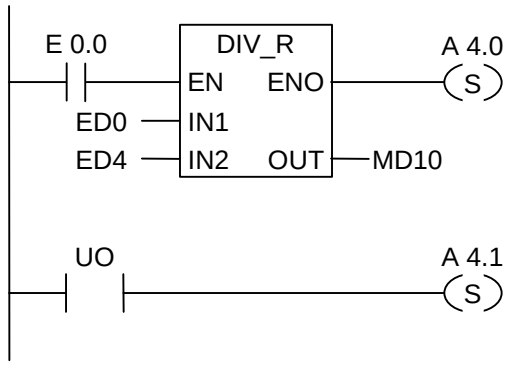
Si si el resultado de una operacion aritmética con números reales (UO) es inválido, la consulta del estado de señal da 1. Si la combinación en A1 y A0 indica ”válido”, el resultado de la consulta del estado de señal es "0".

El resultado de la consulta está combinado con el RLO mediante una Y lógica cuando las conexiones son en serie. En las conexiones en paralelo el resultado está combinado con el RLO mediante una O lógica.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo



El estado "1" en E 0.0 activa el cuadro. Si el valor de ED0 o ED4 no es un número real válido, la operación aritmética no es válida. Si el estado de señal de EN es 1 (activado) y si se produce un error durante el procesamiento de la función DIV_R, el estado de señal de ENO será 0.

A 4.0 se pone a 1 si se ejecuta la operación DIV_R pero uno de los valores no es un **número real** válido.

12.5 RB ---| |--- Bit de anomalía "registro RB"

Símbolo

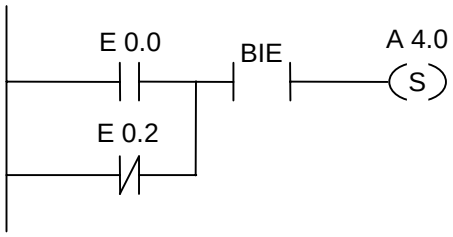
Descripción de la operación

RB ---| |--- (Bit de anomalía "registro RB") y **RB ---| / |---** (Negación del bit de anomalía "registro RB") comprueban el estado del bit RB en la palabra de estado. El resultado de la consulta está combinado con el RLO mediante una Y lógica cuando las conexiones son en serie. En las conexiones en paralelo el resultado está combinado con el RLO mediante una O lógica. El bit RB se emplea al cambiar del procesamiento de palabras al procesamiento de bits.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo



A 4.0 se pone a 1 si E 0.0 es 1 ó E 0.2 es 0 **Y** además de este RLO el bit RB es 1.

12.6 Bit de resultado igual a 0

Símbolo

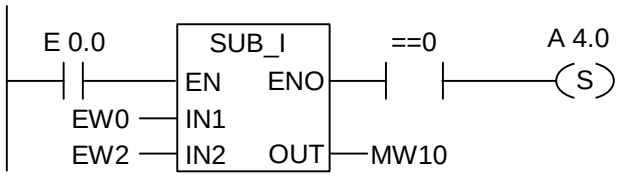
Descripción de la operación

==0 ---| |--- (Bit de resultado igual a 0) y **==0 ---| / |---** (Negación del bit de resultado igual a 0) averiguan si el resultado de una operacion aritmética es igual a "0", o no. Las operaciones consultan los códigos de condición A1 y A0 en la palabra de estado para determinar la relación del resultado con respecto a "0". Cuando las conexiones son en serie, el resultado de consulta está combinado con el RLO mediante una Y lógica; en las conexiones en paralelo, el resultado está combinado con el RLO mediante una O lógica.

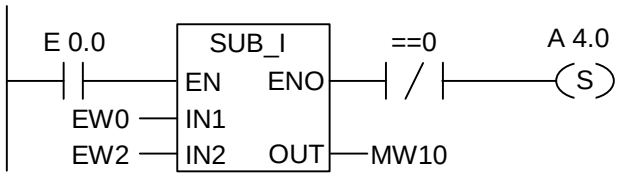
Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplos



El estado "1" en E 0.0 activa la operación. Si el valor de EW0 es igual al valor de EW2, el resultado de la operación aritmética EW0 - EW2 será igual a "0". La salida A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado es igual a "0".



A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado no es igual a "0".

12.7 Bit de resultado diferente de 0

Símbolo

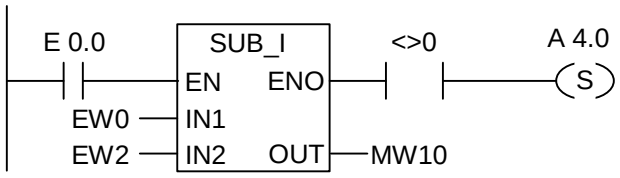
Descripción de la operación

<>0 ---| |--- (Bit de resultado diferente de 0) y **<>0 ---| / |---** (Negación del bit de resultado diferente de 0) averiguan si el resultado de una operacion aritmética es diferente de "0", o no. Las operaciones consultan los códigos de condición A1 y A0 en la palabra de estado para determinar la relación del resultado con respecto a "0". Cuando las conexiones son en serie, el resultado de consulta está combinado con el RLO mediante una Y lógica; en las conexiones en paralelo, el resultado está combinado con el RLO mediante una O lógica.

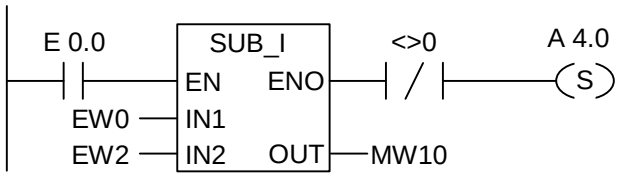
Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplos



El estado "1" en E 0.0 activa la operación. Si el valor de EW0 es diferente del valor de EW2, el resultado de la operación aritmética EW0 - EW2 será diferente de "0". La salida A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado es diferente de "0".



A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado es igual a "0".

12.8 Bit de resultado mayor o igual a 0

Símbolo

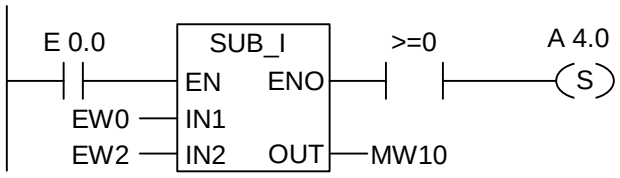
Descripción de la operación

>=0 ---| |--- (Bit de resultado mayor o igual a 0) y **>=0 ---| / |---** (Negación del bit de resultado mayor o igual a 0) averiguan si el resultado de una operacion aritmética es mayor o igual a "0", o no. Las operaciones consultan los códigos de condición A1 y A0 en la palabra de estado para determinar la relación con respecto a "0". En las conexiones en serie el resultado de la consulta está combinado con el RLO mediante una Y lógica; cuando las conexiones son en paralelo, el resultado está combinado con el RLO mediante una O lógica.

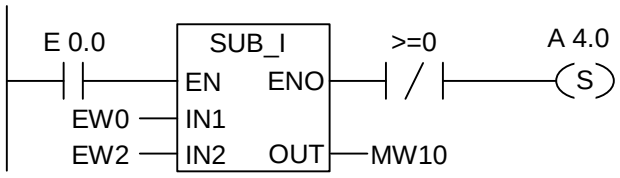
Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplos



El estado "1" en E 0.0 activa la operación. Si el valor de EW0 es mayor o igual al valor de EW2, el resultado de la operación aritmética EW0 - EW2 será mayor o igual a "0". A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado es mayor o igual a "0".



A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado no es mayor o igual a "0".

12.9 Bit de resultado menor o igual a 0

Símbolo

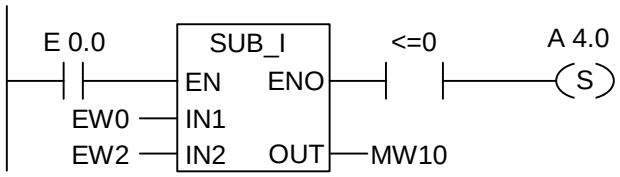
Descripción de la operación

<=0 ---| |--- (Bit de resultado menor o igual a 0) y **<=0 ---| / |---** (Negación del bit de resultado menor o igual a 0) averiguan si el resultado de una operacion aritmética es menor o igual a "0", o no. Las operaciones consultan los códigos de condición A1 y A0 en la palabra de estado para determinar la relación del resultado con respecto a respecto a 0. Cuando las conexiones son en serie, el resultado de consulta está combinado con el RLO mediante una Y lógica; en las conexiones en paralelo, el resultado está combinado con el RLO mediante una O lógica.

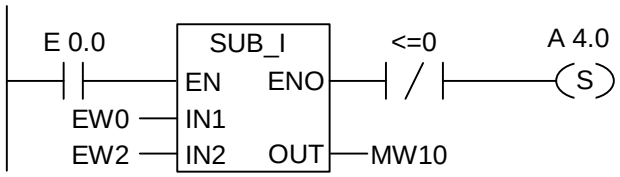
Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplos



El estado "1" en E 0.0 activa la operación. Si el valor de EW0 es menor o igual al valor de EW2, el resultado de la operación aritmética EW0 - EW2 será menor o igual a "0". A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado es menor o igual a "0".



A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado no es menor o igual a "0".

12.10 Bit de resultado mayor que 0

Símbolo

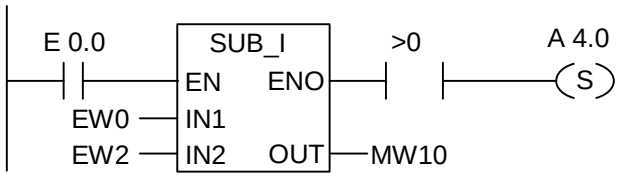
Descripción de la operación

>0 ---| |--- (Bit de resultado mayor que 0) y **>0 ---| / |---** (Negación del bit de resultado mayor que 0) averiguan si el resultado de una operacion aritmética es mayor que 0, o no. Las operaciones consultan los códigos de condición A1 y A0 para determinar la relación existente con respecto a "0". En las conexiones en serie el resultado de la consulta está combinado con el RLO mediante una Y lógica; en las conexiones en paralelo el resultado está combinado con el RLO mediante una O lógica.

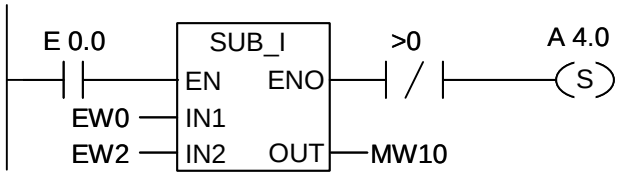
Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo



El estado "1" en E 0.0 activa el cuadro. Si el valor de EW0 es mayor que el valor de EW2, el resultado de la operación aritmética EW0 - EW2 será mayor que "0". A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado es mayor que "0".



A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado no es mayor que "0".

12.11 <0 ---| |--- Bit de resultado menor que 0

Símbolo

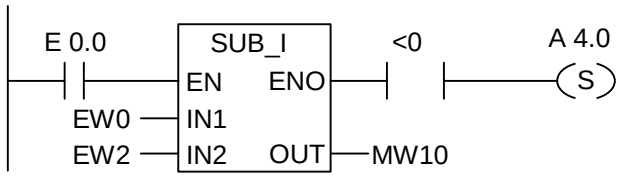
Descripción de la operación

<0 ---| |--- (Bit de resultado menor que 0) y **<0 ---| / |---** (Negación del bit de resultado menor que 0) averiguan si el resultado de una operacion aritmética es menor que "0", o no. Las operaciones consultan los códigos de condición A1 y A0 en la palabra de estado para determinar la relación del resultado con respecto a "0". En las conexiones en serie el resultado de la consulta está combinado con el RLO mediante una Y lógica; en las conexiones en paralelo, el resultado está combinado con el RLO mediante una O lógica.

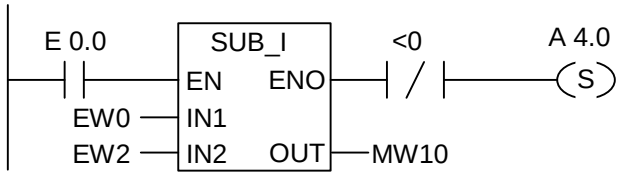
Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplos



El estado "1" en E 0.0 activa la operación. Si el valor de EW0 es menor que el valor de EW2, el resultado de la operación aritmética EW0 - EW2 será menor que "0". A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado es menor que "0".



A 4.0 se pone a 1 si la operación se ejecuta sin errores y el resultado no es menor que "0".

13 Operaciones de temporización

13.1 Lista de operaciones de temporización

Descripción

Bajo Area de memoria y componentes de un temporizador encontrará información sobre cómo ajustar y seleccionar los temporizadores.

Se dispone de las operaciones de temporización siguientes:

- S_IMPULS Parametrizar y arrancar temporizador como impulso
- S_VIMP Parametrizar y arrancar temporizador como impulso prolongado
- S_EVERZ Parametrizar y arrancar temporizador como retardo a la conexión
- S_SEVERZ Parametrizar y arrancar temporizador como retardo a la conexión con memoria
- S_ABRES Parametrizar y arrancar temporizador como retardo a la desconexión
- ---(SI) Arrancar temporizador como impulso
- ---(SV) Arrancar temporizador como impulso prolongado
- ---(SE) Arrancar temporizador como retardo a la conexión
- ---(SS) Arrancar temporizador como retardo a la conexión con memoria
- ---(SA) Arrancar temporizador como retardo a la desconexión

13.2 Area de memoria y componentes de un temporizador

Area de memoria

Los temporizadores tienen un área reservada en la memoria de la CPU. Esta área de memoria reserva una palabra de 16 bits para cada operando de temporizador. La programación con KOP asiste 256 temporizadores. Consulte los datos técnicos de la CPU para saber de cuántas palabras de temporización dispone ésta.

Las siguientes funciones tienen acceso al área de memoria de temporizadores:

- Operaciones de temporización
- Actualización por reloj de palabras de temporización. Esta función de la CPU en el estado RUN decrementa en una unidad un valor de temporización dado en el intervalo indicado por la base de tiempo hasta alcanzar el valor 0.

Valor de temporización

Los bits 0 a 9 de la palabra de temporización contienen el valor de temporización en código binario. Este valor indica un número de unidades. La actualización decrementa el valor de temporización en una unidad y en el intervalo indicado por la base de tiempo hasta alcanzar el valor 0. El valor de temporización se puede cargar en los formatos binario, hexadecimal o decimal codificado en binario (BCD). El área de temporización va de 0 a 9 990 segundos. Para cargar un valor de temporización redefinido, se observarán las siguientes reglas sintácticas.

El valor de temporización se puede cargar en cualesquiera de los siguientes formatos:

- w#16#wxyz
  - siendo: w= la base de tiempo (es decir, intervalo de tiempo o resolución)
  - xyz = el valor de temporización en formato BCD
- S5T#a**H**_b**M**_c**S**_d**MS**
  - siendo: H (horas), M (minutos), S (segundos), MS (milisegundos);
    a, b, c, d los define el usuario
  - La base de tiempo se selecciona automáticamente y el valor de temporización se redondea al próximo número inferior con esa base de tiempo.

El valor de temporización máximo que puede introducirse es de 9 900 segundos ó 2H_46M_30S. Ejemplos:

S5TIME#4S --> 4 segundos
s5t#2h_15m --> 2 horas y 15 minutos
S5T#1H_12M_18S --> 1 hora 12 minutos y 18 segundos

Base de tiempo

Los bits 12 y 13 de la palabra de temporización contienen la base de tiempo en código binario. La base de tiempo define el intervalo en que se decrementa en una unidad el valor de temporización. La base de tiempo más pequeña es 10 ms, la más grande 10 s.

Base di tiempo	Base di tiempo en código binario
10 ms	00
100 ms	01
1 s	10
10 s	11

Los valores no deben exceder 2H_46M_30S. Los valores con un margen o una resolución demasiado grandes (p. ej. 2H_10MS) se redondean de tal forma que correspondan a la tabla para el margen y la resolución.

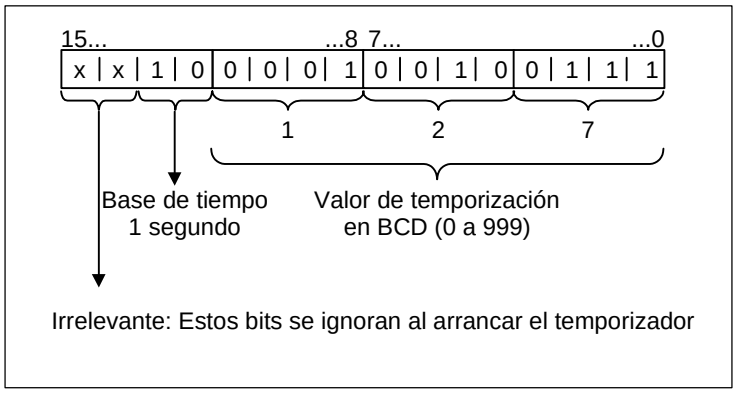
El formato general para el tipo de datos S5TIME tiene los siguientes valores límite para el margen y la resolución:

Resolución	Margen
10MS A 9S_990MS	10MS A 9S_990MS
100MS A 1M_39S_900MS	100MS A 1M_39S_900MS
1S A 16M_39S	1S A 16M_39S
10S A 2H_46M_30S	10S A 2H_46M_30S

Configuración binaria en la palabra de temporización

Cuando se dispara un temporizador, el contenido de la palabra de temporización 1 se utiliza como valor de temporización. Los bits 0 a 11 de la palabra de temporización almacenan el valor de temporización en formato decimal codificado en binario (formato BCD: cada grupo de cuatro bits contiene el código binario de un valor decimal). Los bits 12 a 13 almacenan la base de tiempo en código binario.

La figura muestra el contenido de la palabra de temporización cargado con el valor 127 y una base de tiempo de 1 segundo.

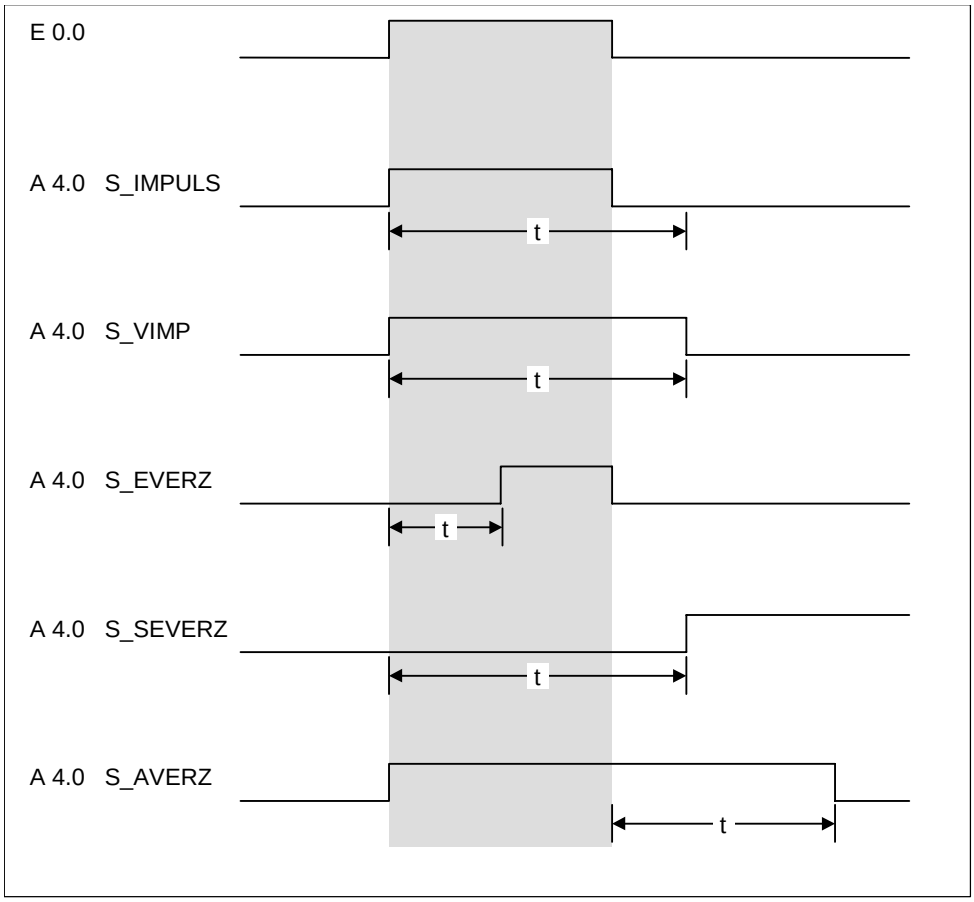


Leer el temporizador y la base de tiempo

Todos los cuadros de temporizadores tienen dos salidas, DUAL y DEZ, para las que se puede indicar una dirección de palabra. La salida DUAL indica el valor de temporización en formato binario. La salida DEZ indica la base de tiempo y el valor de temporización en formato decimal codificado en binario (BCD).

Elegir el temporizador apropiado

El resumen breve de los cinco tipos de temporizadores sirve de ayuda para la elección del temporizador que se adapte mejor a sus necesidades.

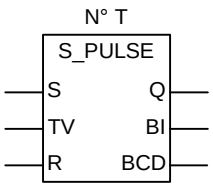


Temporizadores	Descripción
S_IMPULS Temporizador de impulso	El tiempo máximo que la señal de salida permanece a 1 corresponde al valor de temporización t programado. La señal de salida permanece a 1 durante un tiempo inferior si la señal de entrada cambia a 0.
S_VIMP Temporizador de impulso prolongado	La señal de salida permanece a 1 durante el tiempo programado, independientemente del tiempo en que la señal de entrada esté a 1.
S_EVERZ Temporizador de retardo a la conexión	La señal de salida es 1 solamente si ha finalizado el tiempo programado y la señal de entrada sigue siendo 1.
S_SEVERZ Temporizador de retardo a la conexión con memoria	La señal de salida cambia de 0 a 1 solamente si ha finalizado el tiempo programado, independientemente del tiempo en que la señal de salida esté a 1.
S_AVERZ Temporizador de retardo a la desconexión	La señal de salida es 1 cuando la señal de entrada es 1 o cuando el temporizador está en marcha. El temporizador arranca cuando la señal de entrada cambia de 1 a 0.

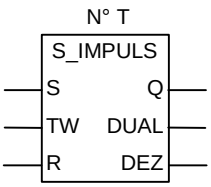
13.3 S_IMPULS Parametrizar y arrancar temporizador como impulso

Símbolo

Inglés



Aléman



Parámetro Inglés	Parámetro Aléman	Tipo de datos	Area de memoria	Descripción
N.° de T	N.º de T	TIMER	T	Número de identificación del temporizador, el área varía según la CPU que se utilice
S	S	BOOL	E, A, M, L, D	Entrada de arranque
TV	TW	S5TIME	E, A, M, L, D	Valor de temporización predeterminado
R	R	BOOL	E, A, M, L, D	Valor de temporización predeterminado
BI	DUAL	WORD	E, A, M, L, D	Valor de temporización actual, codificado en binario
BCD	DEZ	WORD	E, A, M, L, D	Tiempo restante, formato BCD
Q	Q	BOOL	E, A, M, L, D	Estado del temporizador

Descripción de la operación

S_IMPULS (Parametrizar y arrancar temporizador como impulso) arranca el temporizador indicado cuando hay un flanco creciente en la entrada de arranque S. Para arrancar un temporizador tiene que producirse necesariamente un cambio de señal. El temporizador funciona mientras que el estado de señal en la entrada S sea "1", pero como máximo durante el tiempo indicado por el valor de temporización en la entrada TV/TW. El estado de señal en la salida Q es "1" mientras que funcione el temporizador. Si el estado de señal en la entrada S cambia de "1" a "0" antes de transcurrir el intervalo de tiempo, el temporizador se para. En este caso el estado de señal en la salida Q es "0".

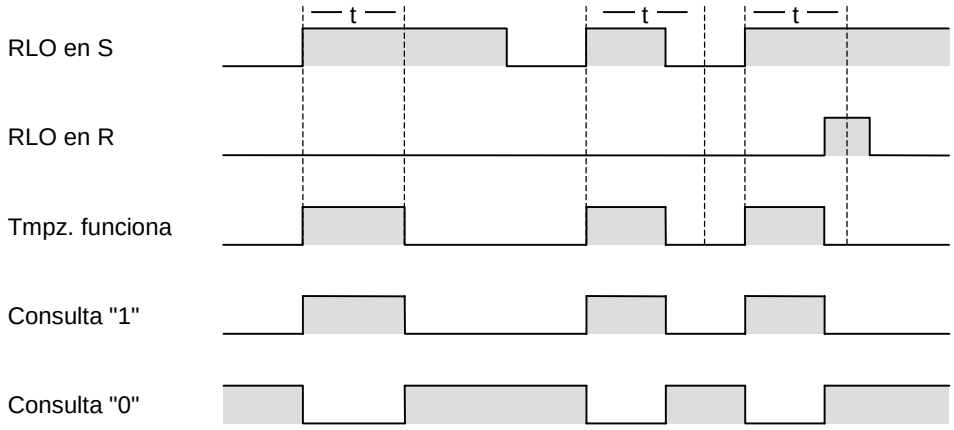
El temporizador se pone a 0 si la entrada de desactivación R del temporizador se pone a "1" mientras funciona el temporizador. El valor de temporización actual y la base de tiempo también se ponen a 0. Un "1" en la entrada R del temporizador no tiene efecto alguno si el temporizador no está en marcha.

El valor de temporización actual queda depositado en las salidas BI/DUAL y BCD/DEZ. El valor de temporización en la salida BI/DUAL está en código binario, el valor en la salida BCD/DEZ está en formato decimal codificado en binario. El valor de temporización actual equivale al valor inicial de TV/TW menos el valor de temporización que ha transcurrido desde el arranque del temporizador.

Consulte también Area de memoria y componentes de un temporizador.

Diagrama de temporización

Características del temporizador como impulso:

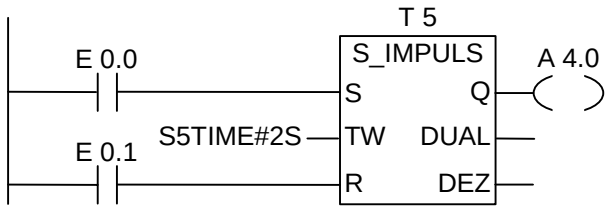


t = tiempo programado

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo



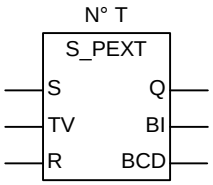
Si el estado de señal de la entrada E 0.0 cambia de "0" a "1" (flanco creciente en el RLO), se activa el temporizador T5. El temporizador continúa en marcha con el valor de temporización indicado de 2 segundos (2 s) mientras la entrada E 0.0 sea 1. Si el estado de señal de la entrada E 0.0 cambia de "1" a "0" antes de transcurrir el tiempo, el temporizador se para.

La salida A 4.0 es "1" mientras esté en marcha el temporizador, y "0" si el tiempo ha transcurrido o si el temporizador fue puesto a 0.

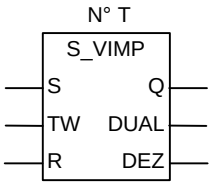
13.4 S_VIMP Parametrizar y arrancar temporizador como impulso prolongado

Símbolo

Inglés



Aléman



Parámetro Inglés	Parámetro Aléman	Tipo de datos	Area de memoria	Descripción
N.° de T	N.º de T	TIMER	T	Número de identificación del temporizador, el área varía según la CPU que se utilice
S	S	BOOL	E, A, M, L, D	Entrada de arranque
TV	TW	S5TIME	E, A, M, L, D	Valor de temporización predeterminado
R	R	BOOL	E, A, M, L, D	Entrada de desactivación
BI	DUAL	WORD	E, A, M, L, D	Valor de temporización actual, codificado en binario
BCD	DEZ	WORD	E, A, M, L, D	Tiempo restante, formato BCD
Q	Q	BOOL	E, A, M, L, D	Estado del temporizador

Descripción de la operación

S_VIMP (Parametrizar y arrancar temporizador como impulso prolongado) arranca el temporizador indicado cuando hay un flanco creciente en la entrada de arranque S. Para arrancar un temporizador tiene que producirse necesariamente un cambio de señal. El temporizador continúa en marcha durante el tiempo predeterminado -indicado en la entrada TV/TW-, aunque el estado de señal en la entrada S se ponga a "0" antes de haber transcurrido el intervalo de tiempo. El estado de señal en la salida Q es "1" mientras el temporizador esté en marcha. El temporizador vuelve a arrancar con el valor de temporización predeterminado si el estado de señal en la entrada S cambia de "0" a "1" mientras está en marcha el temporizador.

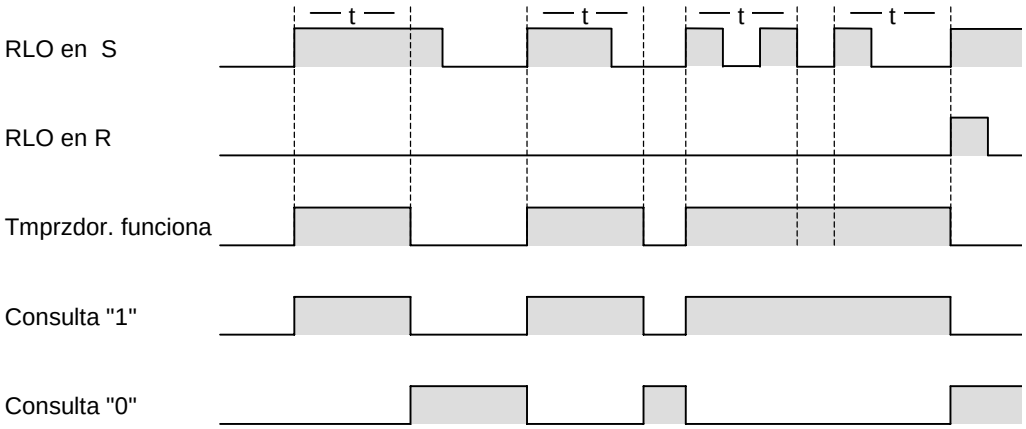
El temporizador se pone a 0 si la entrada de desactivación R del temporizador se pone a "1" mientras el temporizador está funcionando. El valor de temporización actual y la base de tiempo se ponen a 0.

El valor de temporización actual queda depositado en las salidas BI/DUAL y BCD/DEZ. El valor de temporización en la salida BI/DUAL está en código binario, el valor en la salida BCD/DEZ está en formato decimal codificado en binario. El valor de temporización actual equivale al valor inicial de TV/TW menos el valor de temporización que ha transcurrido desde el arranque del temporizador.

Consulte también Area de memoria y componentes de un temporizador.

Diagrama de temporización

Características del temporizador como impulso prolongado:

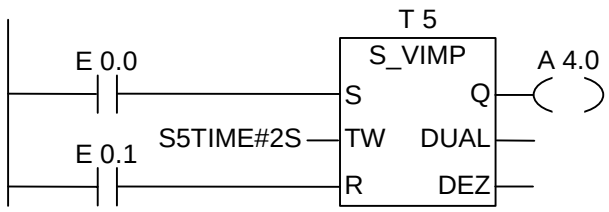


t = tiempo programado

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo

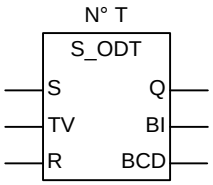


Si el estado de señal de la entrada E 0.0 cambia de "0" a "1" (flanco creciente en el RLO), se activa el temporizador T5. El temporizador continúa en marcha con el valor de temporización indicado de dos segundos sin ser afectado por un flanco decreciente en la entrada S. Si el estado de señal de la entrada E 0.0 cambia de "0" a "1" antes de transcurrir el tiempo, el temporizador vuelve a arrancar. Si el estado de señal de la entrada E 0.1 cambia de "0" a "1" mientras el temporizador está en marcha, éste se pone a 0. La salida A 4.0 es "1" mientras esté en marcha el temporizador.

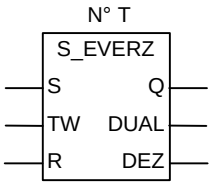
13.5 S_EVERZ Parametrizar y arrancar temporizador como retardo a la conexión

Símbolo

Inglés



Aléman



Parámetro Inglés	Parámetro Aléman	Tipo de datos	Area de memoria	Descripción
N.° de T	N.° de T	TIMER	T	Número de identificación del temporizador, el área varía según la CPU que se utilice
S	S	BOOL	E, A, M, L, D	Entrada de arranque
TV	TW	S5TIME	E, A, M, L, D	Valor de temporización predeterminado
R	R	BOOL	E, A, M, L, D	Entrada de desactivación
BI	DUAL	WORD	E, A, M, L, D	Valor de temporización actual, codificado en binario
BCD	DEZ	WORD	E, A, M, L, D	Valor de temporización actual, formato BCD
Q	Q	BOOL	E, A, M, L, D	Estado del temporizador

Descripción de la operación

S_EVERZ (Parametrizar y arrancar temporizador como retardo a la conexión) arranca el temporizador indicado cuando hay un flanco creciente en la entrada de arranque S. Para arrancar un temporizador tiene que producirse necesariamente un cambio de señal. El temporizador continúa en marcha con el valor de temporización indicado en la entrada TV/TW mientras el estado de señal en la entrada S sea positivo. El estado de señal en la salida Q será "1" si el tiempo ha transcurrido sin que se produjeran errores y si el estado de señal en la entrada S es "1". Si el estado de señal en la entrada S cambia de "1" a "0" mientras está en marcha el temporizador, éste se para. En este caso, el estado de señal en la salida Q será "0".

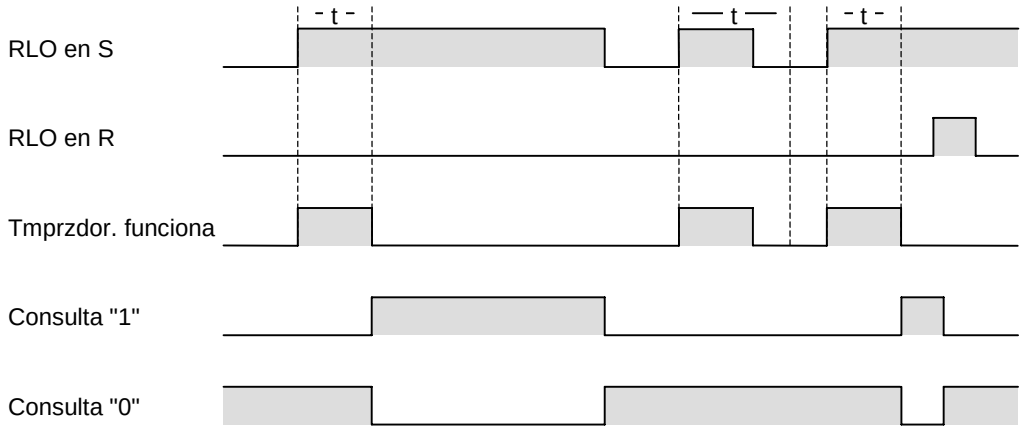
El temporizador se pone a 0 si la entrada de desactivación R del temporizador se pone a "1" mientras funciona el temporizador. El valor de temporización y la base de tiempo se ponen a 0. Entonces el estado de señal en la salida Q es "0". El temporizador también se pone a 0 si en la entrada de desactivación R el valor es "1", mientras el temporizador no está en marcha y el RLO en la entrada S es "1".

El valor de temporización actual queda depositado en las salidas BI/DUAL y BCD/DEZ. El valor de temporización en la salida BI/DUAL está en código binario, el valor en la salida BCD/DEZ está en formato decimal codificado en binario. El valor de temporización actual equivale al valor inicial de TV/TW menos el valor de temporización que ha transcurrido desde el arranque del temporizador.

Consulte también Area de memoria y componentes de un temporizador.

Diagrama de temporización

Características del temporizador de retardo a la conexión:

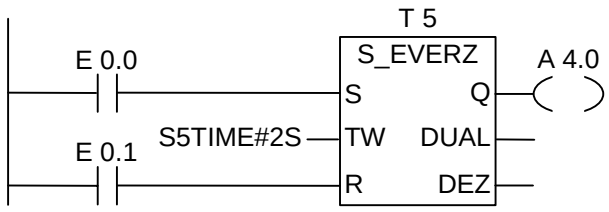


t = tiempo programado

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo

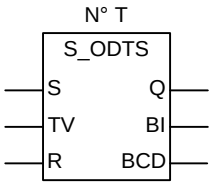


Si el estado de señal de la entrada E 0.0 cambia de "0" a "1" (flanco creciente en el RLO), se activa el temporizador T5. Si transcurre el tiempo de dos segundos y el estado de señal en la entrada E 0.0 sigue siendo "1", la salida A 4.0 será "1". Si el estado de señal de la entrada E 0.0 cambia de "1" A "0", el temporizador se para y la salida A 4.0 será "0". (Si el estado de señal de la entrada E 0.1 cambia de "0" a "1", el temporizador se pone a 0, tanto si estaba funcionando como si no).

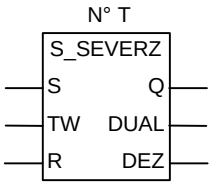
13.6 S_SEVERZ Parametrizar y arrancar temporizador como retardo a la conexión con memoria

Símbolo

Inglés



Aléman



Parámetro Inglés	Parámetro Aléman	Tipo de datos	Area de memoria	Descripción
N.° de T	N.º de T	TIMER	T	Número de identificación del temporizador, el área varía según la CPU que se utilice
S	S	BOOL	E, A, M, L, D	Entrada de arranque
TV	TW	S5TIME	E, A, M, L, D	Valor de temporización predeterminado
R	R	BOOL	E, A, M, L, D	Entrada de desactivación
BI	DUAL	WORD	E, A, M, L, D	Valor de temporización actual, codificado en binario
BCD	DEZ	WORD	E, A, M, L, D	Valor de temporización actual, formato BCD
Q	Q	BOOL	E, A, M, L, D	Estado del temporizador

Descripción de la operación

S_SEVERZ (Parametrizar y arrancar temporizador como retardo a la conexión con memoria) arranca el temporizador indicado cuando hay un flanco creciente en la entrada de arranque S. Para arrancar un temporizador tiene que producirse necesariamente un cambio de señal. El temporizador continúa en marcha con el valor de temporización indicado en la entrada TV/TW aunque el estado de señal en la entrada S se ponga a "0" antes de que haya transcurrido el tiempo. El estado de señal en la salida Q será "1" si ha transcurrido el tiempo, independientemente del estado de señal que tenga la entrada S. El temporizador vuelve a arrancar con el valor de temporización indicado si el estado de señal en la entrada S cambia de "0" a "1" mientras el temporizador está en marcha.

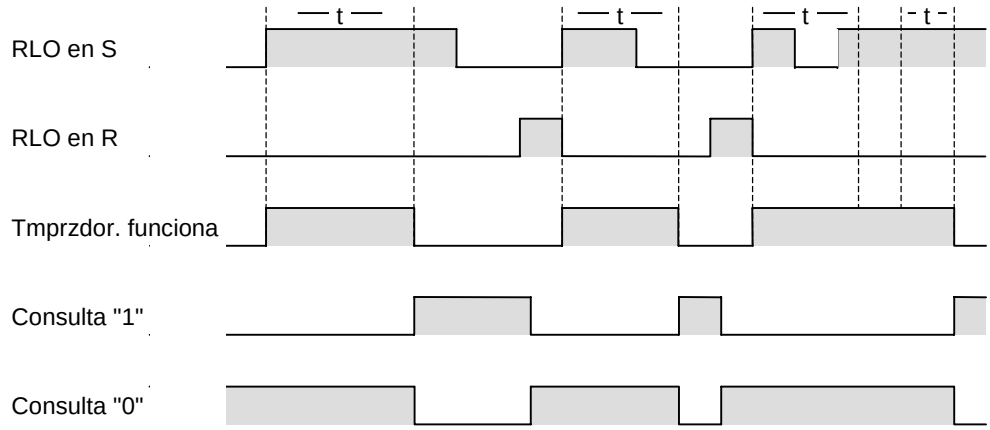
El temporizador se pone a 0 si la entrada de desactivación R del temporizador se pone a "1", independientemente del RLO en la entrada S. Entonces el estado de señal en la salida Q es "0".

El valor de temporización actual queda depositado en las salidas BI/DUAL y BCD/DEZ. El valor de temporización en la salida BI/DUAL está en código binario, el valor en la salida BCD/DEZ está en formato decimal codificado en binario. El valor de temporización actual equivale al valor inicial de TV/TW menos el valor de temporización que ha transcurrido desde el arranque del temporizador.

Consulte también Area de memoria y componentes de un temporizador.

Diagrama de temporización

Características del temporizador de retardo a la conexión con memoria:

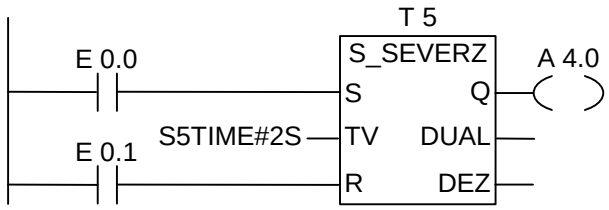


t = tiempo programada

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo

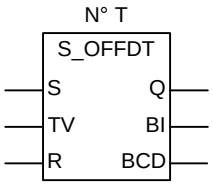


Si el estado de señal de la entrada E 0.0 cambia de "0" a "1" (flanco creciente en el RLO), se activa el temporizador T5. El temporizador continúa en marcha sin que un cambio de señal de "1" a "0" en la entrada E 0.0 repercuta en él. Si el estado de señal de la entrada E 0.0 cambia de "1" a "0" antes de que haya transcurrido el tiempo, el temporizador vuelve a arrancar. La salida A 4.0 será "1" si ha transcurrido el tiempo (Si el estado de señal de la entrada E 0.1 cambia de "0" a "1", el temporizador se pone a "0", independientemente de cuál sea el RLO en S).

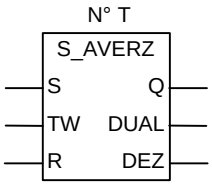
13.7 S_AVERZ Parametrizar y arrancar temporizador como retardo a la desconexión

Símbolo

Inglés



Aléman



Parámetro Inglés	Parámetro Aléman	Tipo de datos	Area de memoria	Descripción
N.° de T	N.º de T	TIMER	T	Número de identificación del temporizador, el área varía según la CPU que se utilice
S	S	BOOL	E, A, M, L, D	Entrada de arranque
TV	TW	S5TIME	E, A, M, L, D	Valor de temporización predeterminado
R	R	BOOL	E, A, M, L, D	Entrada de desactivación
BI	DUAL	WORD	E, A, M, L, D	Valor de temporización actual, codificado en binario
BCD	DEZ	WORD	E, A, M, L, D	Valor de temporización actual, formato BCD
Q	Q	BOOL	E, A, M, L, D	Estado del temporizador

Descripción de la operación

S_AVERZ (Parametrizar y arrancar temporizador como retardo a la desconexión) arranca el temporizador indicado cuando hay un flanco decreciente en la entrada de arranque S. Para arrancar un temporizador tiene que producirse necesariamente un cambio de señal. El estado de señal en la salida Q será "1" si el estado de señal en la entrada S es "1", y también mientras el temporizador esté en marcha. El temporizador se para si el estado de señal en la entrada S cambia de "0" a "1" mientras el temporizador está en marcha. El temporizador sólo vuelve a arrancar si el estado de señal en la entrada S vuelve a cambiar de "1" a "0".

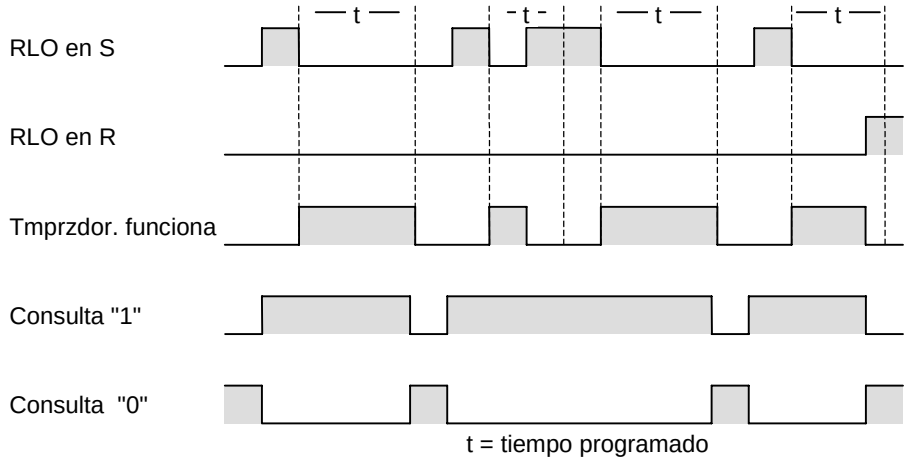
El temporizador se pone a 0 si la entrada de desactivación R se pone a "1" mientras el temporizador está en marcha.

El valor de temporización actual queda depositado en las salidas BI/DUAL y BCD/DEZ. El valor de temporización en la salida BI/DUAL está en código binario, el valor en la salida BCD/DEZ está en formato decimal codificado en binario. El valor de temporización actual equivale al valor inicial de TV/TW menos el valor de temporización que ha transcurrido desde el arranque del temporizador.

Consulte también Area de memoria y componentes de un temporizador.

Diagrama de temporización

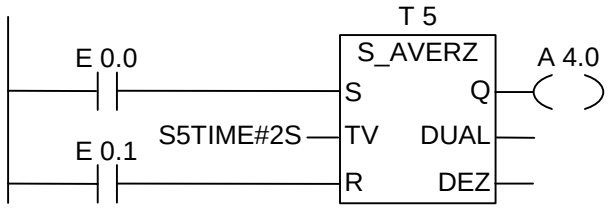
Características del temporizador como retardo a la desconexión:



Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	x	x	x	1

Ejemplo



El temporizador arranca si el estado de señal en la entrada E 0.0 cambia de "1" a "0".

A 4.0 es "1" si E 0.0 es "1" o el temporizador está en marcha (Si el estado de señal en E 0.1 cambia de "0" a "1", mientras está en marcha el temporizador, éste se pone a 0).

13.8 ---(SI) Arrancar temporizador como impulso

Símbolo

Inglés	**Aléman**
<Nº de T>	<Nº de T>
---(SP)	**---(SI)**
<Valor de temporización>	<Valor de temporización>

Parámetro	**Tipo de datos**	**Area de memoria**	**Descripción**
<Nº de T>	TIMER	T	Número específico del temporizador; el área varía según la CPU utilizada
<Valor de temporización>	S5TIME	E, A, M, L, D	Valor de temporización predeterminado

Descripción de la operación

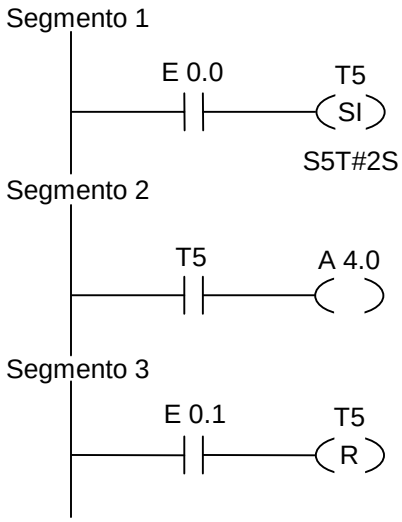
---(SI) (Arrancar temporizador como impulso) arranca el temporizador indicado con el **<valor de temporización>** si en el RLO se produce un flanco creciente. El temporizador continúa funcionando con el intervalo de tiempo indicado mientras el RLO sea positivo ("1"). El estado del contador es "1" mientras está en marcha el temporizador. Si el RLO cambia de "1" a "0" antes de transcurrir el valor de temporización, el temporizador se para. En este caso el estado del contador es "0".

Consulte también Area de memoria y componentes de un temporizador y S_IMPULS Parametrizar y arrancar temporizador como impulso.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	-	-	0

Ejemplo



Si el estado de señal en la entrada E 0.0 cambia de "0" a "1" (flanco creciente en el RLO), el temporizador T5 arranca. El temporizador continúa en marcha con el valor de temporización indicado de 2 s mientras E 0.0 sea "1". Si el estado de señal en E 0.0 cambia de "1" a "0" antes de transcurrir el tiempo, el temporizador se para. La salida A 4.0 es "1" mientras el temporizador está en marcha. Si el estado de señal en la entrada E 0.1 cambia de "0" a "1", el temporizador T5 se pone a 0, es decir, se para y el valor de temporización restante se pone a "0".

13.9 ---(SV) Arrancar temporizador como impulso prolongado

Símbolo

Inglés	**Aléman**
<Nº de T>	<Nº de T>
---(SE)	**---(SV)**
<Valor de temporización>	<Valor de temporización>

Parámetro	**Tipo de datos**	**Area de memoria**	**Descripción**
<Nº de T>	TIMER	T	Número específico del temporizador; el área varía según la CPU utilizada
<Valor de temporización>	S5TIME	E, A, M, L, D	Valor de temporización predeterminado

Descripción de la operación

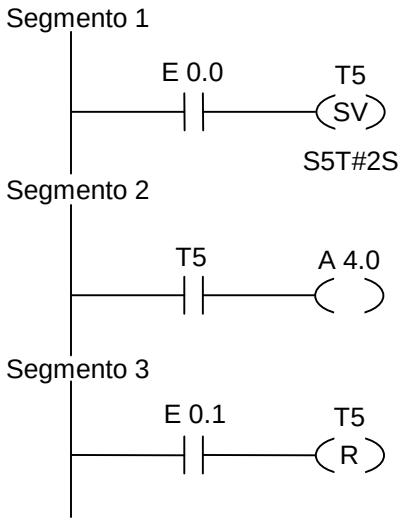
---(SV) (Arrancar temporizador como impulso prolongado (bobina)) arranca el temporizador indicado con el **<valor de temporización>** si en el RLO se produce un flanco creciente. El temporizador continúa funcionando con el intervalo de tiempo indicado aunque el RLO se ponga a "0" antes de transcurrir el tiempo. El estado del contador es "1" mientras está en marcha el temporizador. El temporizador se vuelve a arrancar con el valor de temporización indicado si el RLO cambia de "0" a "1" mientras el temporizador está en marcha.

Consulte también Area de memoria y componentes de un temporizador y S_VIMP Parametrizar y arrancar temporizador como impulso prolongado.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	-	-	0

Ejemplo



Si el estado de señal en la entrada E 0.0 cambia de "0" a "1" (flanco creciente en el RLO), el temporizador T5 arranca. El temporizador continúa en marcha con el valor de temporización indicado sin ser afectado por un flanco negativo en el RLO. Si el estado de señal en E 0.0 cambia de "0" a "1" antes de transcurrir el tiempo, el temporizador se vuelve a arrancar. La salida A 4.0 es "1" mientras el temporizador está en marcha. Si el estado de señal en la entrada E 0.1 cambia de "0" a "1", el temporizador T5 se pone a 0, es decir que se para y que el valor de temporización restante se pone a "0".

13.10 ---(SE) Arrancar temporizador como retardo a la conexión

Símbolo

Inglés	**Aléman**
<Nº de T>	<Nº de T>
---(SD)	**---(SE)**
<Valor de temporización>	<Valor de temporización>

Parámetro	**Tipo de datos**	**Area de memoria**	**Descripción**
<Nº de T>	TIMER	T	Número específico del temporizador; el área varía según la CPU utilizada
<Valor de temporización>	S5TIME	E, A, M, L, D	Valor de temporización predeterminado

Descripción de la operación

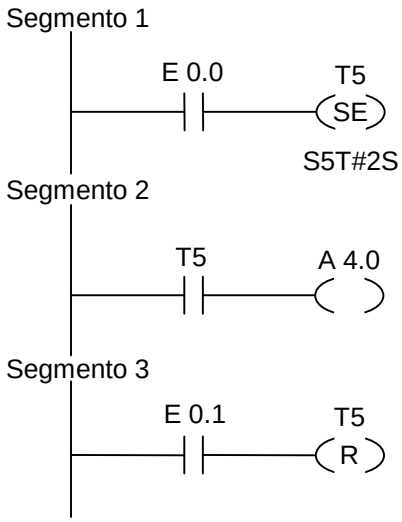
---(SE) (Arrancar temporizador como retardo a la conexión) arranca el temporizador indicado con el **<valor de temporización>** si en el RLO se produce un flanco creciente. El estado de señal del temporizador es "1" si el **<valor de temporización>** ha transcurrido sin errores y el RLO sigue siendo "1". Si el RLO cambia de "1" a "0" mientras el temporizador está en marcha, éste cambia a la marcha en vacío. En este caso, una consulta de "1" da como resultado el valor "0".

Consulte también Area de memoria y componentes de un temporizador y S_EVERZ Parametrizar y arrancar temporizador como retardo a la conexión.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	-	-	0

Ejemplo



Si el estado de señal en la entrada E 0.0 cambia de "0" a "1" (flanco creciente en el RLO), el temporizador T5 arranca. Si transcurre el tiempo y el estado de señal en E 0.0 sigue siendo "1", la salida A 4.0 es "1". Si el estado de señal en la entrada E 0.0 cambia de "1" a "0", el temporizador cambia a la marcha en vacío y A 4.0 es "0". Si el estado de señal en la entrada E 0.1 cambia de "0" a "1", el temporizador T5 se pone a 0, es decir, se para y el valor de temporización restante se pone A "0".

13.11 ---(SS) Arrancar temporizador como retardo a la conexión con memoria

Símbolo

Inglés	**Aléman**
<Nº de T>	<Nº de T>
---(SS)	**---(SS)**
<Valor de temporización>	<Valor de temporización>

Parámetro	**Tipo de datos**	**Area de memoria**	**Descripción**
<Nº de T>	TIMER	T	Número específico del temporizador; el área varía según la CPU utilizada
<Valor de temporización>	S5TIME	E, A, M, L, D	Valor de temporización predeterminado

Descripción de la operación

---(SS) (Arrancar temporizador como retardo a la conexión con memoria) arranca el temporizador indicado si en el RLO se produce un flanco creciente. El estado de señal del temporizador es "1" si el tiempo ha transcurrido. Un rearranque del temporizador sólo es posible si éste se ha puesto expresamente a 0. El estado del temporizador sólo se puede poner a "0" mediante una puesta a cero.

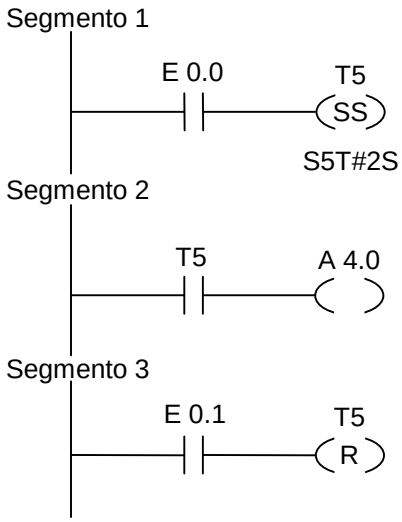
El temporizador se vuelve a arrancar con el valor de temporización indicado si el RLO cambia de "0" a "1" mientras transcurre el tiempo.

Consulte también Area de memoria y componentes de un temporizador y S_SEVERZ Parametrizar y arrancar temporizador como retardo a la conexión con memoria.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	-	-	0

Ejemplo



Si el estado de señal en la entrada E 0.0 cambia de "0" a "1" (flanco creciente en el RLO), el temporizador T5 arranca. Si el estado de señal en la entrada E 0.0 cambia de "0" a "1" antes de transcurrir el tiempo, el temporizador se vuelve a arrancar. La salida A 4.0 es "1" si ha transcurrido el tiempo. Si el estado de señal en la entrada E 0.1 es "1", el temporizador T5 se pone a 0, es decir, se para y el valor de temporización restante se pone a "0".

13.12 ---(SA) Arrancar temporizador como retardo a la desconexión

Símbolo

Inglés	**Aléman**
<Nº de T>	<Nº de T>
---(SF)	**---(SA)**
<Valor de temporización>	<Valor de temporización>

Parámetro	**Tipo de datos**	**Area de memoria**	**Descripción**
<Nº de T>	TIMER	T	Número específico del temporizador; el área varía según la CPU utilizada
<Valor de temporización>	S5TIME	E, A, M, L, D	Valor de temporización predeterminado

Descripción de la operación

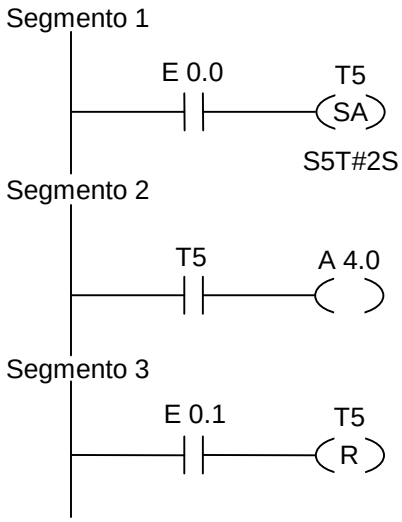
---(SA) (Arrancar temporizador como retardo a la desconexión) arranca el temporizador indicado si en el RLO se produce un flanco decreciente. El estado de señal del temporizador es "1" si el RLO es "1" o mientras funcione el temporizador con el **<valor de temporización>**. El temporizador se pone a cero si el RLO cambia de "0" a "1" mientras está en marcha el temporizador. El temporizador siempre se vuelve a arrancar si el RLO cambia de "1" a "0".

Consulte también Area de memoria y componentes de un temporizador y S_AVERZ Parametrizar y arrancar temporizador como retardo a la desconexión.

Palabra de estado

	RB	**A1**	**A0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/ER**
se escribe:	-	-	-	-	-	0	-	-	0

Ejemplo



Si el estado de señal en la entrada E 0.0 cambia de "1" a "0", el temporizador arranca.

A 4.0 es "1" si E 0.0 es "1" o si el temporizador está en marcha. Si el estado de señal en la entrada E 0.1 cambia de "0" a "1", el temporizador T5 se pone a 0, es decir que se para y que el valor de temporización restante se pone a "0".

14 Operaciones lógicas con palabras

14.1 Lista de operaciones lógicas con palabras

Descripción

Las operaciones lógicas con palabras comparan bit a bit pares de palabras (16 bits) y palabras dobles (32 bits) según la lógica de Boole.

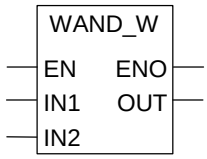
Si el resultado en la salida OUT es diferente de 0, el bit A1 de la palabra de estado se pone a "1".
Si el resultado en la salida OUT es igual a 0, el bit A1 de la palabra de estado se pone a "0".

Se dispone de las operaciones lógicas con palabras siguientes:

- WAND_W Y lógica con palabras
- WOR_W O lógica con palabras
- WXOR_W O exclusiva con palabras
- WAND_DW Y lógica con dobles palabras
- WOR_DW O lógica con dobles palabras
- WXOR_DW O exclusiva con dobles palabras

14.2 WAND_W Y lógica con palabras

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	WORD	E, A, M, L, D	Primer valor de la combinación lógica
IN2	WORD	E, A, M, L, D	Segundo valor de la combinación lógica
OUT	WORD	E, A, M, L, D	Palabra del resultado de la combinación lógica

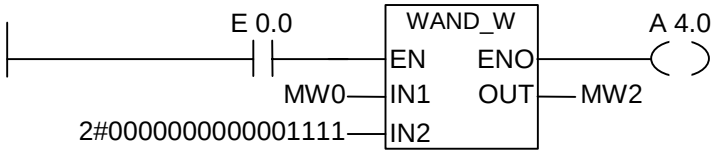
Descripción de la operación

WAND_W (Y lógica con palabras) se activa cuando la entrada de habilitación (EN) tiene el estado de señal "1". Esta operación combina entonces los dos valores de palabra de IN1 y IN2 bit a bit realizando una Y lógica. Los valores se interpretan como puras configuraciones binarias. El resultado queda depositado en la salida OUT. La salida de habilitación ENO tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	x	0	0	-	x	1	1	1

Ejemplo



La operación se ejecuta si E 0.0 es 1. Sólo son relevantes los bits de 0 a 3 de MW0; los demás bits son enmascarados por la configuración binaria de la palabra en IN2:

MW0 = 01010101 01010101

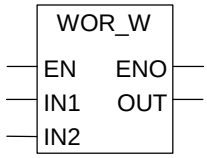
IN2 = 00000000 00001111

MW0 Y IN2 = MW2 = 00000000 00000101

A 4.0 será "1" si se ejecuta la operación.

14.3 WOR_W O lógica con palabras

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	WORD	E, A, M, L, D	Primer valor de la combinación lógica
IN2	WORD	E, A, M, L, D	Segundo valor de la combinación lógica
OUT	WORD	E, A, M, L, D	Palabra del resultado de la combinación lógica

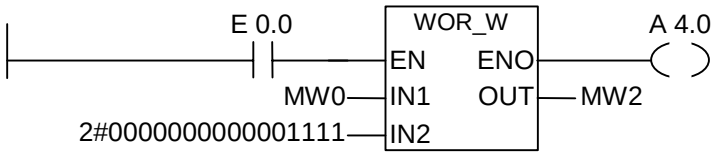
Descripción de la operación

WOR_W (O lógica con palabras) se activa cuando la entrada de habilitación (EN) tiene el estado de señal “1”. Esta operación combina los dos valores de las palabras IN1 y IN2 bit a bit realizando una O lógica. Los valores se interpretan como puras configuraciones binarias. El resultado queda depositado en la salida OUT. La salida de habiliatación ENO tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	x	0	0	-	x	1	1	1

Ejemplo



La operación se ejecuta si E 0.0 es 1. Los bits 0 a 3 se ponen a "1", los demás bits de MW0 no varían.

MW0 = 01010101 01010101

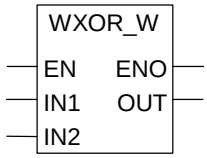
IN2 = 00000000 00001111

MW0 Ó IN2 = MW2 = 01010101 01011111

A 4.0 será “1” si se ejecuta la operación.

14.4 WXOR_W O-exclusiva con palabras

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	WORD	E, A, M, L, D	Primer valor de la combinación lógica
IN2	WORD	E, A, M, L, D	Segundo valor de la combinación lógica
OUT	WORD	E, A, M, L, D	Palabra del resultado de la combinación lógica

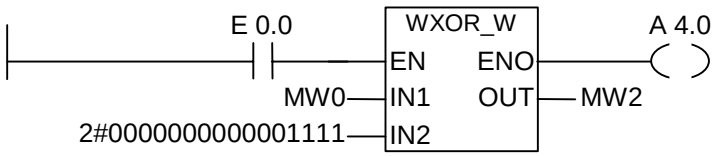
Descripción de la operación

WXOR_W (O exclusiva con palabras) se activa cuando la entrada de habilitación (EN) tiene el estado de señal “1”. Esta operación lógica combina los dos valores de las palabra IN1 y IN2 bit a bit realizando una O exclusiva. Los valores se interpretan como puras configuraciones binarias. El resultado queda depositado en la salida OUT. La salida de habiliatación ENO tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	x	0	0	-	x	1	1	1

Ejemplo



La operación se ejecuta si E 0.0 es 1:

MW0 = 01010101 01010101

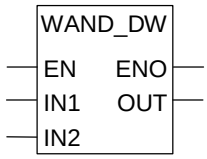
IN2 = 00000000 00001111

MW0 XOR IN2 = MW2 = 01010101 01011010

A 4.0 será “1” si se ejecuta la operación.

14.5 WAND_DW Y lógica con dobles palabras

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	DWORD	E, A, M, L, D	Primer valor de la combinación
IN2	DWORD	E, A, M, L, D	Segundo valor de la combinación
OUT	DWORD	E, A, M, L, D	Doble palabra del resultado de la combinación lógica

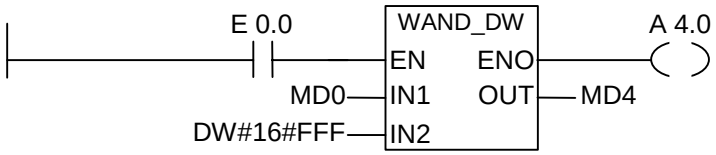
Descripción de la operación

WAND_DW (Y lógica con dobles palabras) se activa cuando la entrada de habilitación (EN) tiene el estado de señal "1". Esta operación combina los dos valores de las dobles palabras IN1 y IN2 bit a bit realizando una Y lógica. Los valores se interpretan como puras configuraciones binarias. El resultado queda depositado en la salida OUT. La salida de habilitación ENO tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	x	0	0	-	x	1	1	1

Ejemplo



La operación se ejecuta si E 0.0 es 1. Sólo son relevantes los bits 0 y 11 de MD0, los demás bits son enmascarados por la configuración binaria de IN2:

MD0 = 01010101 01010101 01010101 01010101

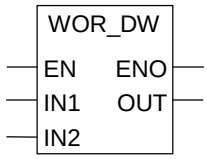
IN2 = 00000000 00000000 00001111 11111111

MD0 Y IN2 = MD4 = 00000000 00000000 00000101 01010101

A 4.0 será “1” si se ejecuta la operación.

14.6 WOR_DW O lógica con dobles palabras

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	DWORD	E, A, M, L, D	Primer valor de la combinación lógica
IN2	DWORD	E, A, M, L, D	Segundo valor de la combinación lógica
OUT	DWORD	E, A, M, L, D	Doble palabra del resultado de la combinación lógica

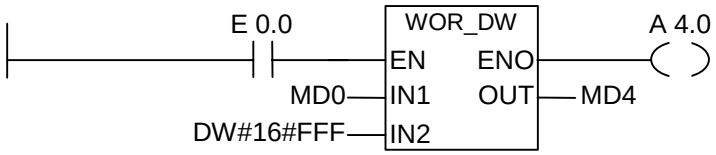
Descripción de la operación

WOR_DW (O lógica con dobles palabras) se activa cuando la entrada de habilitación (EN) tiene el estado de señal “1”. Esta operación combina los dos valores de las dobles palabras IN1 y IN2 bit a bit realizando una O lógica. Los valores se interpretan como puras configuraciones binarias. El resultado queda depositado en la salida OUT. La salida de habiliatación ENO tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	x	0	0	-	x	1	1	1

Ejemplo



La operación se ejecuta si E 0.0 es 1. Los bits de 0 a 11 se ponen a "1". Los demás bits de MWZ no cambian:

MD0 = 01010101 01010101 01010101 01010101

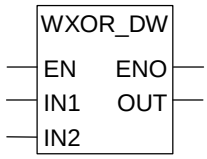
IN2 = 00000000 00000000 00001111 11111111

MD0 Ó IN2 = MD4 = 01010101 01010101 01011111 11111111

A 4.0 será “1” si se ejecuta la operación.

14.7 WXOR_DW O-exclusiva con dobles palabras

Símbolo



Parámetro	Tipo de datos	Area de memoria	Descripción
EN	BOOL	E, A, M, L, D	Entrada de habilitación
ENO	BOOL	E, A, M, L, D	Salida de habilitación
IN1	DWORD	E, A, M, L, D	Primer valor de la combinación lógica
IN2	DWORD	E, A, M, L, D	Segundo valor de la combinación lógica
OUT	DWORD	E, A, M, L, D	Doble palabra del resultado de la combinación lógica

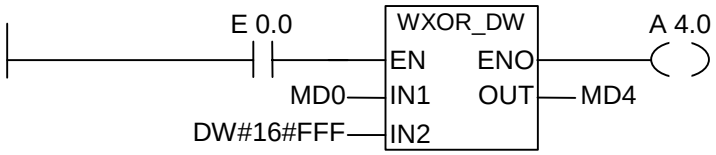
Descripción de la operación

WXOR_DW (O exclusiva con dobles palabras) se activa cuando la entrada de habilitación (EN) tiene el estado de señal “1”. Esta operación lógica combina los dos valores de las dobles palabras IN1 y IN2 bit a bit realizando una O exclusiva. Los valores se interpretan como puras configuraciones binarias. El resultado queda depositado en la salida OUT. La salida de habiliatación ENO tiene el mismo estado de señal que EN.

Palabra de estado

	RB	A1	A0	OV	OS	OR	STA	RLO	/ER
se escribe:	1	x	0	0	-	x	1	1	1

Ejemplo



La operación se ejecuta si E 0.0 es 1:

MD0 = 01010101 01010101 01010101 01010101

IN2 = 00000000 00000000 00001111 11111111

MD4 = MD0 XOR IN2 = 01010101 01010101 01011010 10101010

A 4.0 será “1” si se ejecuta la operación.

A Sinopsis de las operaciones KOP

A.1 Operaciones KOP ordenadas según las abreviaturas nemotécnicas alemanas (SIMATIC)

Nemotécnica alemana	Nemotécnica inglesa	Catálogo de elementos del programa	Descripción
---| |---	---| |---	Operaciones lógicas con bits	Contacto normalmente abierto
---|/|---	---|/|---	Operaciones lógicas con bits	Contacto normalmente cerrado
---( )	---( )	Operaciones lógicas con bits	Bobina de relé, salida
---(#)---	---(#)---	Operaciones lógicas con bits	Conector
==0 ---| |---	==0 ---| |---	Bits de estado	Bit de resultado igual a 0
>0 ---| |---	>0 ---| |---	Bits de estado	Bit de resultado mayor que 0
>=0 ---| |---	>=0 ---| |---	Bits de estado	Bit de resultado mayor o igual a 0
<=0 ---| |---	<=0 ---| |---	Bits de estado	Bit de resultado menor o igual a 0
<0 ---| |---	<0 ---| |---	Bits de estado	Bit de resultado menor que 0
<>0 ---| |---	<>0 ---| |---	Bits de estado	Bit de resultado diferente de 0
ABS	ABS	Función en coma flotante	Calcular el valor absoluto de un número de coma flotante
ACOS	ACOS	Función en coma flotante	Calcular el arcocoseno
ADD_DI	ADD_DI	Función en coma fija	Sumar enteros dobles
ADD_I	ADD_I	Función en coma fija	Sumar enteros
ADD_R	ADD_R	Función en coma flotante	Sumar números de coma flotante
ASIN	ASIN	Función en coma flotante	Calcular el arcoseno
ATAN	ATAN	Función en coma flotante	Calcular la arcotangente
BCD_DI	BCD_DI	Convertidor	Convertir BCD en entero doble
BCD_I	BCD_I	Convertidor	Convertir BCD en entero
BIE ---| |---	BR ---| |---	Bits de estado	Bit de anomalía "registro RB"
----(CALL)	----(CALL)	Control del programa	Llamar a una FC/SFC sin parámetros
CALL_FB	CALL_FB	Control del programa	Llamar a un FB desde un cuadro
CALL_FC	CALL_FC	Control del programa	Llamar a una FC desde un cuadro

Nemotécnica alemana	**Nemotécnica inglesa**	**Catálogo de elementos del programa**	**Descripción**
CALL_SFB	CALL_SFB	Control del programa	Llamar a un SFB desde un cuadro
CALL_SFC	CALL_SFC	Control del programa	Llamar a una SFC desde un cuadro
CEIL	CEIL	Convertidor	Redondear número real a entero doble superior
CMP ? D	CMP ? D	Comparador	Comparar enteros dobles
CMP ? I	CMP ? I	Comparador	Comparar enteros
CMP ? R	CMP ? R	Comparador	Comparar números de coma flotante
COS	COS	Función en coma flotante	Calcular el coseno
DI_BCD	DI_BCD	Convertidor	Convertir entero doble en BCD
DI_R	DI_R	Convertidor	Convertir entero doble en real
DIV_DI	DIV_DI	Función en coma flotante	Dividir enteros dobles
DIV_I	DIV_I	Función en coma fija	Dividir enteros
DIV_R	DIV_R	Función en coma flotante	Dividir números de coma flotante
EXP	EXP	Función en coma flotante	Calcular el exponente
FLOOR	FLOOR	Convertidor	Redondear número real a entero doble inferior
I_BCD	I_BCD	Convertidor	Convertir entero en BCD
I_DI	I_DI	Convertidor	Convertir entero en entero doble
INV_I	INV_I	Convertidor	Complemento a 1 de un entero
INV_DI	INV_DI	Convertidor	Complemento a 1 de un entero doble
---(JMP)	---(JMP)	Saltos	Saltar si la señal es 1
---(JMP)	---(JMP)	Saltos	Salto absoluto
---(JMP)	---(JMP)	Saltos	Salto condicional
---(JMPN)	---(JMPN)	Saltos	Saltar si la señal es 0
LABEL	LABEL	Saltos	Meta del salto
LN	LN	Función en coma flotante	Calcular el logaritmo natural
---(MCR>)	---(MCR>)	Control del programa	Desconectar un Master Control Relay
---(MCR<)	---(MCR<)	Control del programa	Conectar un Master Control Relay
---(MCRA)	---(MCRA)	Control del programa	Inicio de un Master Control Relay
---(MCRD)	---(MCRD)	Control del programa	Final de un Master Control Relay
MOD_DI	MOD_DI	Función en coma fija	Obtener el resto de una división de enteros dobles
MOVE	MOVE	Desplazamiento	Asignar un valor
MUL_DI	MUL_DI	Función en coma fija	Multiplicar enteros dobles
MUL_I	MUL_I	Función en coma fija	Multiplicar enteros
MUL_R	MUL_R	Función en coma flotante	Multiplicar números de coma flotante
---(N)---	---(N)---	Operaciones lógicas con bits	Detectar flanco decreciente (1 --> 0)
NEG	NEG	Operaciones lógicas con bits	Detectar flanco de señal negativo (1 --> 0)

Nemotécnica alemana	**Nemotécnica inglesa**	**Catálogo de elementos del programa**	**Descripción**
NEG_DI	NEG_DI	Convertidor	Complemento a 2 de un entero doble
NEG_I	NEG_I	Convertidor	Complemento a 2 de un entero
NEG_R	NEG_R	Convertidor	Invertir signo de un número real
---| NOT |---	---| NOT |---	Operaciones lógicas con bits	Invertir resultado lógico (RLO)
---(OPN)	---(OPN)	Llamada DB	Abrir bloque de datos
OS ---| |---	OS ---| |---	Bits de estado	Bit de anomalía "desbordamiento memorizado"
OV ---| |---	OV ---| |---	Bits de estado	Bit de anomalía "desbordamiento"
---(P)---	---(P)---	Operaciones lógicas con bits	Detectar flanco creciente RLO (0 --> 1)
POS	POS	Operaciones lógicas con bits	Detectar flanco de señal positivo (0 --> 1)
---(R)	---(R)	Operaciones lógicas con bits	Desactivar salida
---(RET)	---(RET)	Control del programa	Retorno
ROL_DW	ROL_DW	Desplazar/rotar	Rotar 32 bits a la izquierda
ROR_DW	ROR_DW	Desplazar/rotar	Rotar 32 bits a la derecha
ROUND	ROUND	Convertidor	Redondear a entero doble
RS	RS	Operaciones lógicas con bits	Activar flip-flop de desactivación
---(S)	---(S)	Operaciones lógicas con bits	Activar salida
---(SA)	---(SF)	Temporizadores	Arrancar temporizador como retardo a la desconexión
---(SAVE)	---(SAVE)	Operaciones lógicas con bits	Cargar resultado lógico (RLO) en el registro RB
S_AVERZ	S_OFFDT	Temporizadores	Parametrizar y arrancar temporizador como retardo a la desconexión
---(SE)	---(SD)	Temporizadores	Arrancar temporizador como retardo a la conexión
S_EVERZ	S_ODT	Temporizadores	Parametrizar y arrancar temporizador como retardo a la conexión
SHL_DW	SHL_DW	Desplazar/rotar	Desplazar 32 bits a la izquierda
SHL_W	SHL_W	Desplazar/rotar	Desplazar 16 bits a la izquierda
SHR_DI	SHR_DI	Desplazar/rotar	Desplazar entero doble a la derecha
SHR_DW	SHR_DW	Desplazar/rotar	Desplazar 32 bits a la derecha
SHR_I	SHR_I	Desplazar/rotar	Desplazar entero a la derecha
SHR_W	SHR_W	Desplazar/rotar	Desplazar 16 bits a la derecha
---(SI)	---(SP)	Temporizadores	Arrancar temporizador como impulso
S_IMPULS	S_PULSE	Temporizadores	Parametrizar y arrancar temporizador como impulso
SIN	SIN	Función en coma flotante	Calcular el seno
SQR	SQR	Función en coma flotante	Calcular el cuadrado

Nemotécnica alemana	**Nemotécnica inglesa**	**Catálogo de elementos del programa**	**Descripción**
SQRT	SQRT	Función en coma flotante	Calcular la raíz cuadrada
SR	SR	Operaciones lógicas con bits	Desactivar flip-flop de activación
---(SS)	---(SS)	Temporizadores	Arrancar temporizador como retardo a la conexión con memoria
S_SEVERZ	S_ODTS	Temporizadores	Parametrizar y arrancar temporizador como retardo a la conexión con memoria
SUB_DI	SUB_DI	Función en coma fija	Restar enteros dobles
SUB_I	SUB_I	Función en coma fija	Restar enteros
SUB_R	SUB_R	Función en coma flotante	Restar números de coma flotante
---(SV)	---(SE)	Temporizadores	Arrancar temporizador como impulso prolongado
S_VIMP	S_PEXT	Temporizadores	Parametrizar y arrancar temporizador como impulso prolongado
---(SZ)	---(SC)	Contadores	Poner contador al valor inicial
TAN	TAN	Función en coma flotante	Calcular la tangente
TRUNC	TRUNC	Convertidor	Truncar a entero doble
UO ---| |---	UO ---| |---	Bits de estado	Bit de anomalía "operación no válida"
WAND_DW	WAND_DW	Operaciones lógicas con palabras	Y lógica con dobles palabras
WAND_W	WAND_W	Operaciones lógicas con palabras	Y lógica con palabras
WOR_DW	WOR_DW	Operaciones lógicas con palabras	O lógica con dobles palabras
WOR_W	WOR_W	Operaciones lógicas con palabras	O lógica con palabras
WXOR_DW	WXOR_DW	Operaciones lógicas con palabras	O-exclusiva con dobles palabras
WXOR_W	WXOR_W	Operaciones lógicas con palabras	O-exclusiva con palabras
ZAEHLER	S_CUD	Contadores	Parametrizar e incrementar/decrementar contador
----(ZR)	----(CD)	Contadores	Decrementar contador
Z_RUECK	----(S_CD)	Contadores	Parametrizar y decrementar contador
---(ZV)	----(CU)	Contadores	Incrementar contador
Z_VORW	S_CU	Contadores	Parametrizar e incrementar contador

A.2 Operaciones KOP ordenadas según las abreviaturas nemotécnicas inglesas (internacional)

Nemotécnica inglesa	Nemotécnica alemana	Catálogo de elementos del programa	Descripción
---|/|---	---|/|---	Operaciones lógicas con bits	Contacto normalmente cerrado
---| |---	---| |---	Operaciones lógicas con bits	Contacto normalmente abierto
---( )	---( )	Operaciones lógicas con bits	Bobina de relé, salida
---(#)---	---(#)---	Operaciones lógicas con bits	Conector
==0 ---| |---	==0 ---| |---	Bits de estado	Bit de resultado igual a 0
>0 ---| |---	>0 ---| |---	Bits de estado	Bit de resultado mayor que 0
>=0 ---| |---	>=0 ---| |---	Bits de estado	Bit de resultado mayor o igual a 0
<=0 ---| |---	<=0 ---| |---	Bits de estado	Bit de resultado menor o igual a 0
<0 ---| |---	<0 ---| |---	Bits de estado	Bit de resultado menor que 0
<>0 ---| |---	<>0 ---| |---	Bits de estado	Bit de resultado diferente de 0
ABS	ABS	Función en coma flotante	Calcular el valor absoluto de un número de coma flotante
ACOS	ACOS	Función en coma flotante	Calcular el arcocoseno
ADD_DI	ADD_DI	Función en coma fija	Sumar enteros dobles
ADD_I	ADD_I	Función en coma fija	Sumar enteros
ADD_R	ADD_R	Función en coma flotante	Sumar números de coma flotante
ASIN	ASIN	Función en coma flotante	Calcular el arcoseno
ATAN	ATAN	Función en coma flotante	Calcular la arcotangente
BCD_DI	BCD_DI	Convertidor	Convertir BCD en entero doble
BCD_I	BCD_I	Convertidor	Convertir BCD en entero
BR ---| |---	BIE ---| |---	Bits de estado	Bit de anomalía "registro RB"
----(CALL)	----(CALL)	Control del programa	Llamar a una FC/SFC sin parámetros
CALL_FB	CALL_FB	Control del programa	Llamar a un FB desde un cuadro
CALL_FC	CALL_FC	Control del programa	Llamar a una FC desde un cuadro
CALL_SFB	CALL_SFB	Control del programa	Llamar a un SFB desde un cuadro
CALL_SFC	CALL_SFC	Control del programa	Llamar a una SFC desde un cuadro
----(CD)	----(ZR)	Contadores	Decrementar contador
CEIL	CEIL	Convertidor	Redondear número real a entero doble superior
CMP ? D	CMP ? D	Comparador	Comparar enteros dobles
CMP ? I	CMP ? I	Comparador	Comparar enteros
CMP ? R	CMP ? R	Comparador	Comparar números de coma flotante

Nemotécnica inglesa	**Nemotécnica alemana**	**Catálogo de elementos del programa**	**Descripción**
COS	COS	Función en coma flotante	Calcular el coseno
----(CU)	---(ZV)	Contadores	Incrementar contador
DI_BCD	DI_BCD	Convertidor	Convertir entero doble en BCD
DI_R	DI_R	Convertidor	Convertir entero doble en real
DIV_DI	DIV_DI	Función en coma fija	Dividir enteros dobles
DIV_I	DIV_I	Función en coma fija	Dividir enteros
DIV_R	DIV_R	Función en coma fija	Dividir números de coma flotante
EXP	EXP	Función en coma fija	Calcular el exponente
FLOOR	FLOOR	Convertidor	Redondear número real a entero doble inferior
I_BCD	I_BCD	Convertidor	Convertir entero en BCD
I_DI	I_DI	Convertidor	Convertir entero en entero doble
INV_I	INV_I	Convertidor	Complemento a 1 de un entero
INV_DI	INV_DI	Convertidor	Complemento a 1 de un entero doble
---(JMP)	---(JMP)	Saltos	Saltar si la señal es 1
---(JMP)	---(JMP)	Saltos	Salto absoluto
---(JMP)	---(JMP)	Saltos	Salto condicional
---(JMPN)	---(JMPN)	Saltos	Saltar si la señal es 0
LABEL	LABEL	Saltos	Meta del salto
LN	LN	Función en coma flotante	Calcular el logaritmo natural
---(MCR>)	---(MCR>)	Control del programa	Desconectar un Master Control Relay
---(MCR<)	---(MCR<)	Control del programa	Conectar un Master Control Relay
---(MCRA)	---(MCRA)	Control del programa	Inicio de un Master Control Relay
---(MCRD)	---(MCRD)	Control del programa	Final de un Master Control Relay
MOD_DI	MOD_DI	Función en coma fija	Obtener el resto de una división de enteros dobles
MOVE	MOVE	Desplazar	Asignar un valor
MUL_DI	MUL_DI	Función en coma fija	Multiplicar enteros dobles
MUL_I	MUL_I	Función en coma fija	Multiplicar enteros
MUL_R	MUL_R	Función en coma flotante	Multiplicar números de coma flotante
---(N)---	---(N)---	Operaciones lógicas con bits	Detectar flanco decreciente (1 --> 0)
NEG	NEG	Operaciones lógicas con bits	Detectar flanco de señal negativo (1 --> 0)
NEG_DI	NEG_DI	Convertidor	Complemento a 2 de un entero doble
NEG_I	NEG_I	Convertidor	Complemento a 2 de un entero
NEG_R	NEG_R	Convertidor	Invertir signo de un número real
---| NOT |---	---| NOT |---	Operaciones lógicas con bits	Invertir resultado lógico (RLO)
---(OPN)	---(OPN)	Llamada DB	Abrir bloque de datos
OS ---| |---	OS ---| |---	Bits de estado	Bit de anomalía "desbordamiento memorizado"

Nemotécnica inglesa	**Nemotécnica alemana**	**Catálogo de elementos del programa**	**Descripción**
OV ---| |---	OV ---| |---	Bits de estado	Bit de anomalía "desbordamiento"
---(P)---	---(P)---	Operaciones lógicas con bits	Detectar flanco creciente RLO (0 --> 1)
POS	POS	Operaciones lógicas con bits	Detectar flanco de señal positivo (0 --> 1)
---(R)	---(R)	Operaciones lógicas con bits	Desactivar salida
---(RET)	---(RET)	Control del programa	Retorno
ROL_DW	ROL_DW	Desplazar/rotar	Rotar 32 bits a la izquierda
ROR_DW	ROR_DW	Desplazar/rotar	Rotar 32 bits a la derecha
ROUND	ROUND	Convertidor	Redondear a entero doble
RS	RS	Operaciones lógicas con bits	Activar flip-flop de desactivación
---(S)	---(S)	Operaciones lógicas con bits	Activar salida
---(SAVE)	---(SAVE)	Operaciones lógicas con bits	Cargar resultado lógico (RLO) en el registro RB
---(SC)	---(SZ)	Contadores	Poner contador al valor inicial
----(S_CD)	Z_RUECK	Contadores	Parametrizar y decrementar contador
S_CU	Z_VORW	Contadores	Parametrizar e incrementar contador
S_CUD	ZAEHLER	Contadores	Parametrizar e incrementar/decrementar contador
---(SD)	---(SE)	Temporizadores	Arrancar temporizador como retardo a la conexión
---(SE)	---(SV)	Temporizadores	Arrancar temporizador como impulso prolongado v
---(SF)	---(SA)	Temporizadores	Arrancar temporizador como retardo a la desconexión
SHL_DW	SHL_DW	Desplazar/rotar	Desplazar 32 bits a la izquierda
SHL_W	SHL_W	Desplazar/rotar	Desplazar 16 bits a la izquierda
SHR_DI	SHR_DI	Desplazar/rotar	Desplazar entero doble a la derecha
SHR_DW	SHR_DW	Desplazar/rotar	Desplazar 32 bits a la derecha
SHR_I	SHR_I	Desplazar/rotar	Desplazar entero a la derecha
SHR_W	SHR_W	Desplazar/rotar	Desplazar 16 bits a la derecha
SIN	SIN	Función en coma flotante	Calcular el seno
S_ODT	S_EVERZ	Temporizadores	Parametrizar y arrancar temporizador como retardo a la conexión
S_ODTS	S_SEVERZ	Temporizadores	Parametrizar y arrancar temporizador como retardo a la conexión con memoria
S_OFFDT	S_AVERZ	Temporizadores	Parametrizar y arrancar temporizador como retardo a la desconexión
---(SP)	---(SI)	Temporizadores	Arrancar temporizador como impulso
S_PEXT	S_VIMP	Temporizadores	Parametrizar y arrancar temporizador como impulso prolongado
S_PULSE	S_IMPULS	Temporizadores	Parametrizar y arrancar temporizador como impulso
SQR	SQR	Función en coma flotante	Calcular el cuadrado

Nemotécnica inglesa	**Nemotécnica alemana**	**Catálogo de elementos del programa**	**Descripción**
SQRT	SQRT	Función en coma flotante	Calcular la raíz cuadrada
SR	SR	Operaciones lógicas con bits	Desactivar flip-flop de activación
---(SS)	---(SS)	Temporizadores	Arrancar temporizador como retardo a la conexión con memoria
SUB_DI	SUB_DI	Función en coma fija	Restar enteros dobles
SUB_I	SUB_I	Función en coma fija	Restar enteros
SUB_R	SUB_R	Función en coma flotante	Restar números de coma flotante
TAN	TAN	Función en coma flotante	Calcular la tangente
TRUNC	TRUNC	Convertidor	Truncar a entero doble
UO ---| |---	UO ---| |---	Bits de estado	Bit de anomalía "operación no válida"
WAND_DW	WAND_DW	Operaciones lógicas con palabras	Y lógica con dobles palabras
WAND_W	WAND_W	Operaciones lógicas con palabras	Y lógica con palabras
WOR_DW	WOR_DW	Operaciones lógicas con palabras	O lógica con dobles palabras
WOR_W	WOR_W	Operaciones lógicas con palabras	O lógica con palabras
WXOR_DW	WXOR_DW	Operaciones lógicas con palabras	O-exclusiva con dobles palabras
WXOR_W	WXOR_W	Operaciones lógicas con palabras	O-exclusiva con palabras

B Ejemplos de programación

B.1 Lista de ejemplos de programación

Aplicaciones prácticas

Todas las instrucciones KOP activan una operación determinada. Combinando estas operaciones en un programa se puede llevar a cabo una gran variedad de tareas de automatización. Este capítulo contiene los siguientes ejemplos:

- Controlar una cinta transportadora usando operaciones lógicas con bits
- Detectar el sentido de marcha de una cinta transportadora usando operaciones lógicas con bits
- Generar un impulso de reloj usando operaciones de temporización
- Supervisión del depósito usando operaciones de contaje y de comparación
- Resolver un problema usando operaciones aritméticas con enteros
- Ajustar el tiempo de calentamiento de una caldera

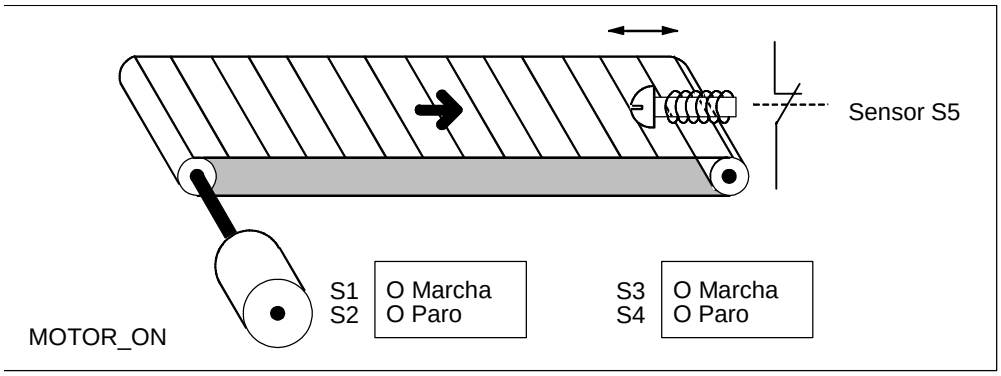
Operaciones utilizadas

Nemotécnica alemana	Operación	Descripción
WAND_W	Lógica de palabras	Y con palabras
WOR_W	Lógica de palabras	O con palabras
Z_RUECK	Contadores	Decrementar contador (bobina)
Z_VORW	Contadores	Incrementar contador (bobina)
---(R)	Operaciones lógicas con bits	Desactivar bobina
---(S)	Operaciones lógicas con bits	Activar bobina
---(P)	Operaciones lógicas con bits	Detectar flanco creciente RLO 0 $\rightarrow$ 1
ADD_I	Función en coma fija	Sumar enteros
DIV_I	Función en coma fija	Dividir enteros
MUL_I	Función en coma fija	Multiplicar enteros
CMP >=I	Comparadores	Comparar enteros
CMP <=I	Comparadores	Comparar enteros
—| |—	Operaciones lógicas con bits	Contacto normalmente abierto (operando)
—| / |—	Operaciones lógicas con bits	Contacto normalmente cerrado (operando)
—( )	Operaciones lógicas con bits	Bobina de relé (salida)
---(JMPN)	Saltos	Saltar si es 0 (condicional)
---(RET)	Control del programa	Retorno
MOVE	Desplazamiento	Asignar un valor
---(SV)	Temporizadores	Temporizador de impulso prolongado

B.2 Ejemplos: Operaciones lógicas con bits

Ejemplo 1: Controlar una cinta transportadora

La figura muestra una cinta transportadora que se pone en marcha eléctricamente. Al principio de la cinta (es decir, en el extremo izquierdo) se encuentran dos pulsadores: S1 para MARCHA (start) y S2 para PARO (stop). Al final de la cinta, es decir, en el extremo derecho se encuentran otros dos pulsadores: S3 para MARCHA y S4 para PARO. La cinta puede ponerse en marcha o pararse desde cualesquiera de ambos extremos. Asimismo, el sensor S5 detiene la cinta cuando un paquete alcanza el final de la cinta.



Programación absoluta y simbólica

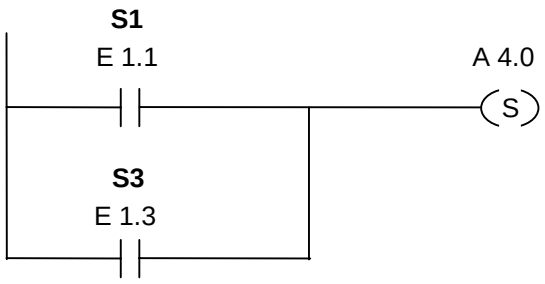
Se puede escribir un programa que controle la cinta transportadora usando **valores absolutos** o **símbolos** para representar los distintos componentes del sistema de transporte.

Los símbolos los define el usuario en la tabla de símbolos (v. la Ayuda en pantalla de STEP 7).

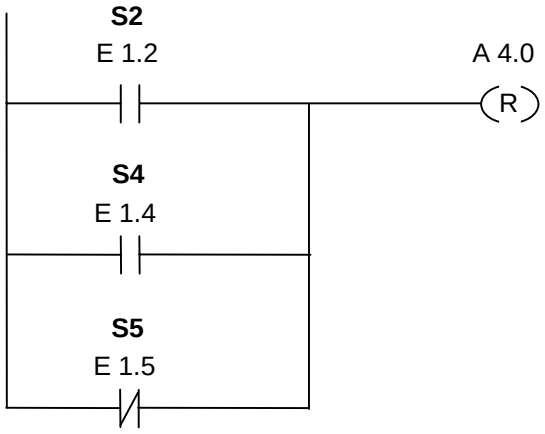
Componente del sistema	Dirección absoluta	Símbolo	Tabla de símbolos
Pulsador de marcha	E 1.1	S1	E 1.1 S1
Pulsador de paro	E 1.2	S2	E 1.2 S2
Pulsador de marcha	E 1.3	S3	E 1.3 S3
Pulsador de paro	E 1.4	S4	E 1.4 S4
Sensor	E 1.5	S5	E 1.5 S5
Motor	A 4.0	MOTOR_ON	A 4.0 MOTOR_ON

Esquema de contactos para controlar una cinta transportadora

Segmento 1: Pulsando cualquiera de los pulsadores de marcha se pone el motor en marcha.

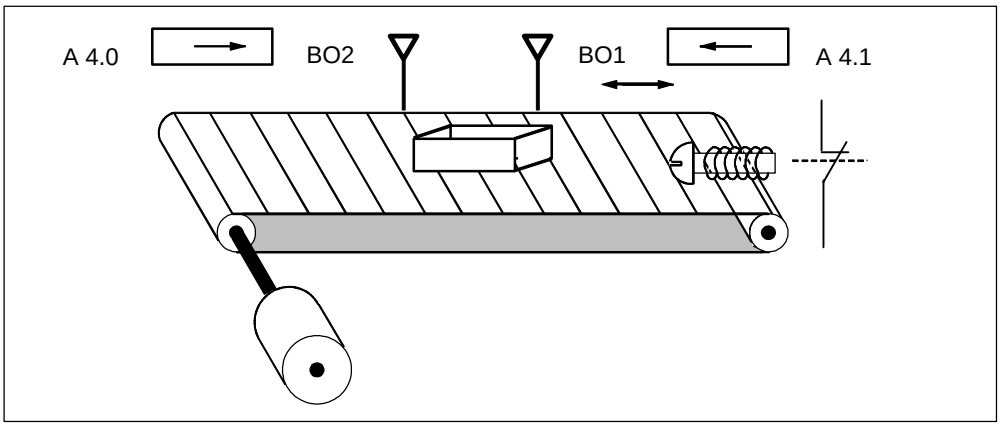


Segmento 2: Pulsando cualquiera de los pulsadores de paro o abriendo el contacto normalmente cerrado al final de la cinta se desconecta el motor.



Ejemplo 2: Detectar el sentido de marcha de una cinta transportadora

La figura muestra una cinta transportadora equipada con dos barreras ópticas (BO1 y BO2) concebidas para detectar el sentido de marcha de la cinta transportadora. Cada barrera óptica funciona igual que un contacto normalmente abierto.



Programación absoluta y simbólica

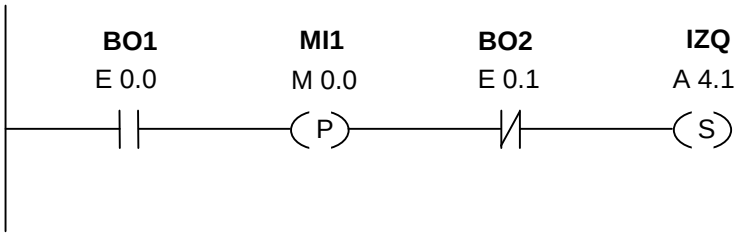
Se puede escribir un programa que controle la cinta transportadora usando **valores absolutos** o **símbolos** para representar los distintos componentes del sistema de transporte.

Los símbolos los define el usuario en la tabla de símbolos (v. la Ayuda en pantalla de STEP 7).

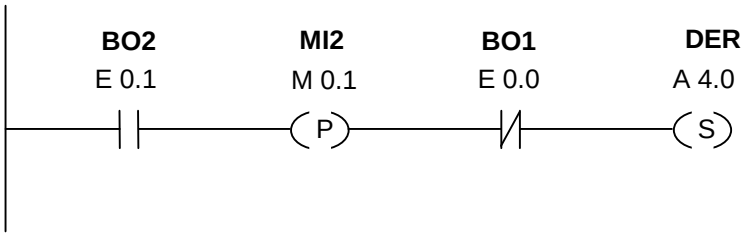
Componente del sistema	Dirección absoluta	Símbolo	Tabla de símbolos
Barrera óptica 1	E 0.0	BO1	E 0.0 BO1
Barrera óptica 2	E 0.1	BO2	E 0.1 BO2
Indicador de movimiento a la derecha	A 4.0	DER	A 4.0 DER
Indicador de movimiento a la izquierda	A 4.1	IZQ	A 4.1 IZQ
Marca de impulso 1	M 0.0	MI1	M 0.0 MI1
Marca de impulso 2	M 0.1	MI2	M 0.1 MI2

Esquema de contactos para detectar el sentido de marcha de una cinta transportadora

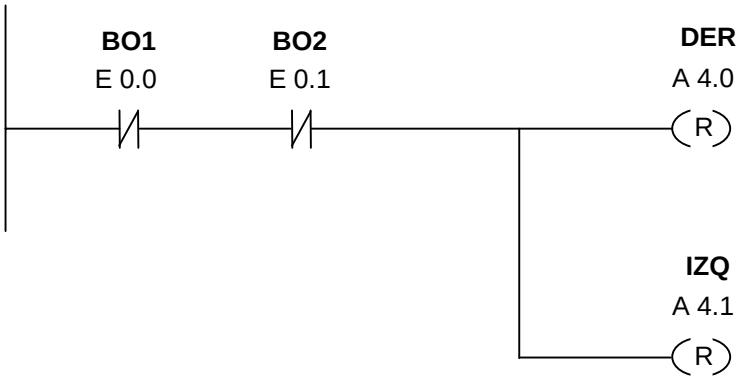
Segmento 1: Si el estado de señal de la entrada E 0.0 cambia de 0 a 1 (flanco positivo) y al mismo tiempo el estado de señal de la entrada E 0.1 es 0, entonces el paquete se está moviendo a la izquierda.



Segmento 2: Si el estado de señal de la entrada E 0.1 cambia de 0 a 1 (flanco positivo) y al mismo tiempo el estado de señal de la entrada E 0.0 es 0, entonces el paquete se esta moviendo a la derecha. Si se interrumpe una de las barreras ópticas, ésto significa que hay un paquete entre las barreras.



Segmento 3: Si una de las barreras ópticas es interrumpida, ésto significa que un paquete se encuentra entre las barreras. El indicador de sentido de marcha se desactiva.



B.3 Ejemplo: Operaciones de temporización

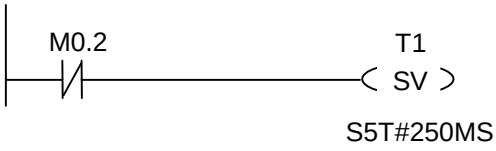
Reloj

Para generar una señal que se repita periódicamente se puede utilizar un reloj o un relé intermitente. Los relojes se suelen utilizar en sistemas de señalización que controlan la intermitencia de lámparas indicadoras.

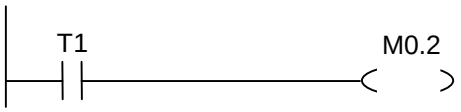
En el S7-300 se puede implementar la función Reloj usando un procesamiento tem-porizado en bloques de organización especiales. El ejemplo siguiente de un programa KOP muestra el uso de funciones temporizadas para generar un reloj.

Esquema de contactos para generar un impulso de reloj (relación impulso-pausa 1:1)

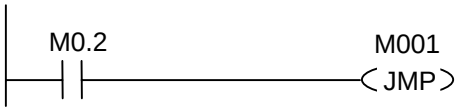
Segmento 1: Si el estado de señal del temporizador T1 es 0, se carga el valor de temporización 250 ms en T1 y T1 arranca como temporizador de impulso prolongado.



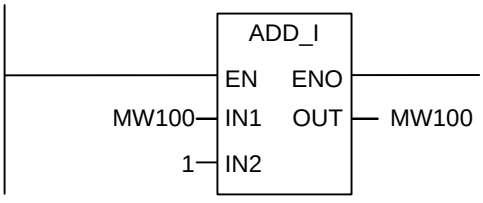
Segmento 2: El estado de señal del temporizador se almacena temporalmente en una marca auxiliar.



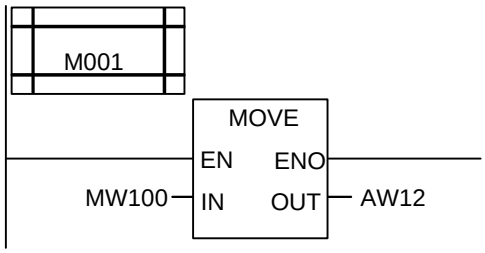
Segmento 3: Si el estado de señal del temporizador T1 es 1, salta a la meta M001.



Segmento 4: Cada vez que transcurre el tiempo programado en el temporizador T1 se incrementa en 1 la palabra de marcas 100.

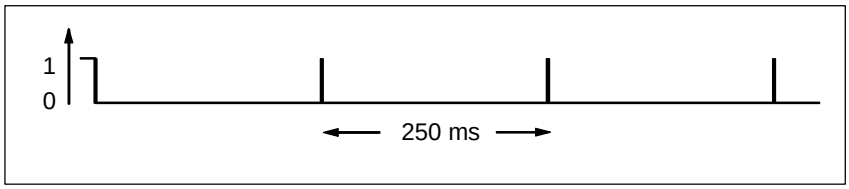


Segmento 5: La operación **MOVE** permite ver las distintas frecuencias de reloj en las salidas A 12.0 a A 13.7.



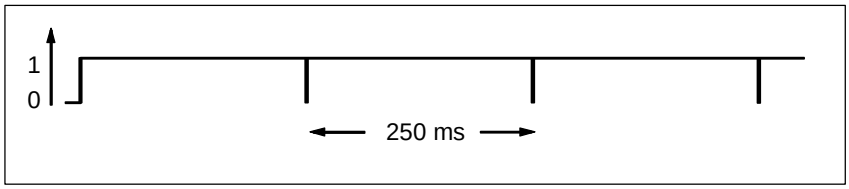
Consulta del estado de señal

La consulta de la señal del temporizador T1 arroja el siguiente resultado lógico para el contacto normalmente cerrado M0.2:



En cuanto finaliza el tiempo programado, el temporizador se vuelve a poner en marcha. Por este motivo, la consulta efectuada por —| / |— T produce sólo brevemente un estado de señal de 1.

La figura muestra el aspecto de un bit RLO negado (invertido):



El bit RLO es 0 cada 250 ms. El salto se ignora y el contenido de la palabra de marcas MW100 se incrementa en 1.

Programar una frecuencia determinada

Con los bits de los bytes de marcas MB101 y MB100 se consiguen las frecuencias siguientes:

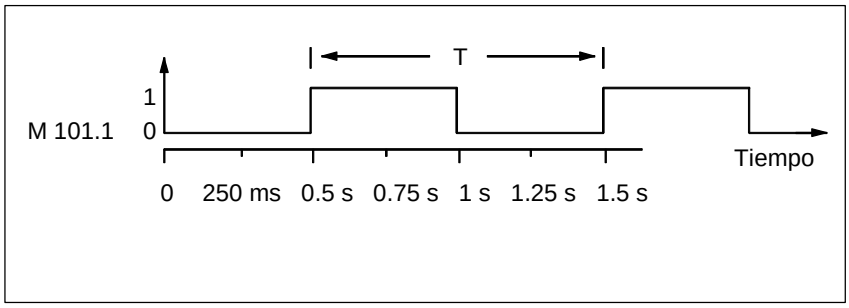
MB101, MB100	Frecuencia en hertzios	Duración	
M 101.0	2.0	0.5 s	(250 ms on / 250 ms off)
M 101.1	1.0	1 s	(0.5 s on / 0.5 s off)
M 101.2	0.5	2 s	(1 s on / 1 s off)
M 101.3	0.25	4 s	(2 s on / 2 s off)
M 101.4	0.125	8 s	(4 s on / 4 s off)
M 101.5	0.0625	16 s	(8 s on / 8 s off)
M 101.6	0.03125	32 s	(16 s on / 16 s off)
M 101.7	0.015625	64 s	(32 s on / 32 s off)
M 100.0	0.0078125	128 s	(64 s on / 64 s off)
M 100.1	0.0039062	256 s	(128 s on / 128 s off)
M 100.2	0.0019531	512 s	(256 s on / 256 s off)
M 100.3	0.0009765	1024 s	(512 s on / 512 s off)
M 100.4	0.0004882	2048 s	(1024 s on / 1024 s off)
M 100.5	0.0002441	4096 s	(2048 s on / 2048 s off)
M 100.6	0.000122	8192 s	(4096 s on / 4096 s off)
M 100.7	0.000061	16384 s	(8192 s on / 8192 s off)

Estados de señal de los bits del byte de marcas MB101

Ciclo	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor en ms
0	0	0	0	0	0	0	**0**	0	250
1	0	0	0	0	0	0	**0**	1	250
2	0	0	0	0	0	0	1	0	250
3	0	0	0	0	0	0	1	1	250
4	0	0	0	0	0	1	**0**	0	250
5	0	0	0	0	0	1	**0**	1	250
6	0	0	0	0	0	1	1	0	250
7	0	0	0	0	0	1	1	1	250
8	0	0	0	0	1	0	**0**	0	250
9	0	0	0	0	1	0	**0**	1	250
10	0	0	0	0	1	0	1	0	250
11	0	0	0	0	1	0	1	1	250
12	0	0	0	0	1	1	**0**	0	250

Estado de señal del bit 1 de MB101 (M 101.1)

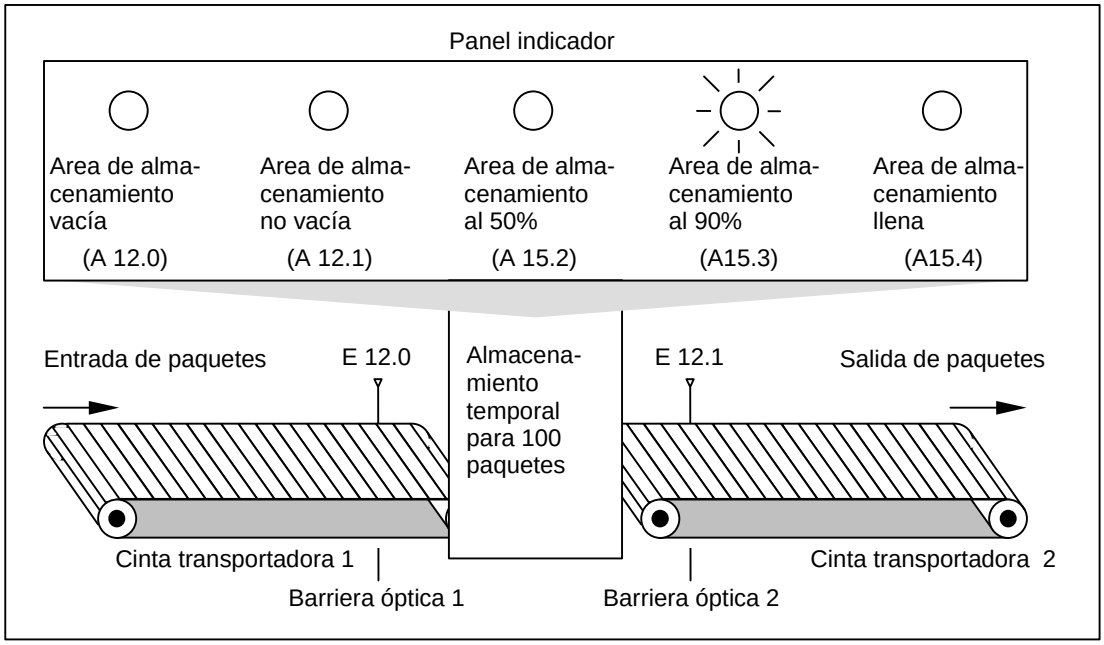
Frecuencia = 1/T = 1/1 s = 1 Hz



B.4 Ejemplo: Operaciones de contaje y comparación

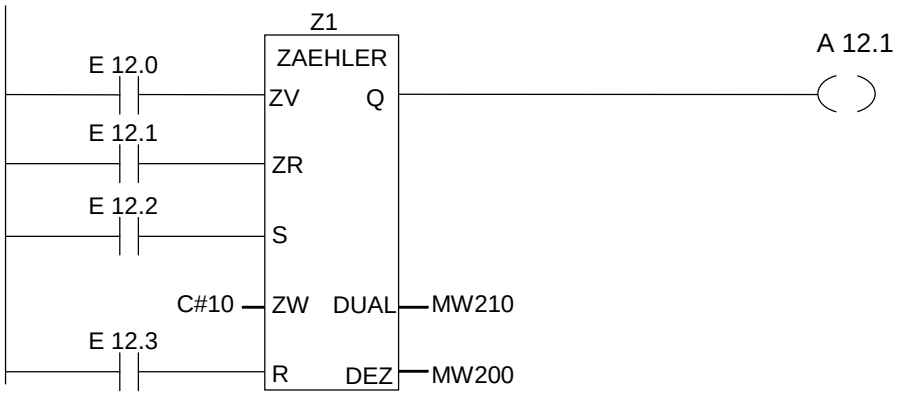
Area de almacenamiento con contador y comparador

La figura muestra un sistema con dos cintas transportadoras y un área de almacenamiento temporal colocada entre ambas. La cinta transportadora 1 transporta paquetes al área de almacenamiento. Una barrera óptica situada al final de la cinta 1 junto al área de almacenamiento determina cuántos paquetes se transportan a dicha área. La cinta transportadora 2 transporta paquetes desde el área de almacenamiento a una plataforma de carga donde llegan camiones y los recogen para suministrarlos a los clientes. Una barrera óptica situada al final de la cinta transportadora 2 junto al área de almacenamiento determina cuántos paquetes abandonan el área de almacenamiento para ser transportados a la plataforma de carga. Un panel indicador con cinco lámparas señala el nivel del área de almacenamiento temporal.



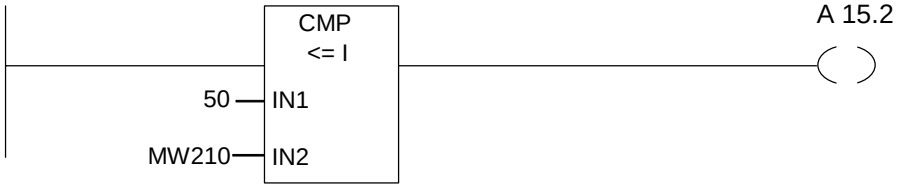
Esquema de contactos para activar las lámparas del panel indicador

Segmento 1: El contador Z1 incrementa con un flanco de señal de ”0” a ”1” en la entrada ZV y decrementa con un flanco de señal de ”0” a ”1”en la entrada ZR. Con un flanco de señal de ”0” a ”1” en la entrada S el valor del contador se pone en el valor de ZW. Con un flanco de señal de ”0” a ”1” en la entrada R el valor del contador se pone a ”0”. En el MW200 está depositado el valor actual del contador de Z1. A12.1 marca ”Area de almacenamiento no vacía”.

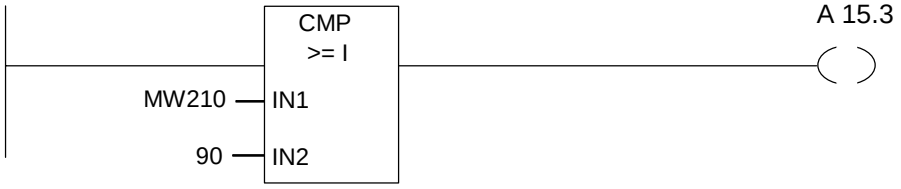


Segmento 2: A12.0 señaliza ”área de almacenamiento vacía”.

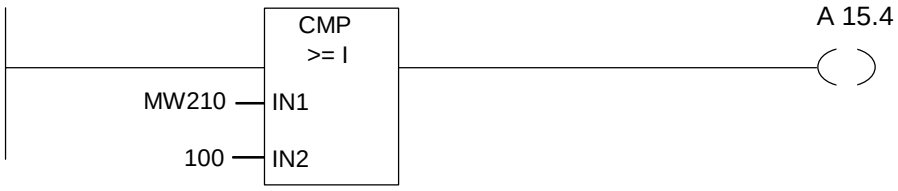
Segmento 3:Si 50 es menor o igual al valor del contador (o si el estado actual del contador es mayor igual que 50), se enciende la lámpara ”Area de almacenamiento al 50%.



Segmento 4: Si el valor del contador es mayor o igual a 90 se enciende la lámpara ”Area de almacenaciemto al 90%.



Segmento 5: Si el valor del contador es mayor o igual a 100 se enciende la lámpara ”Area de almacenamiento llena”.



B.5 Ejemplo: Operaciones de aritmétca con enteros

Resolver un Problema aritmético

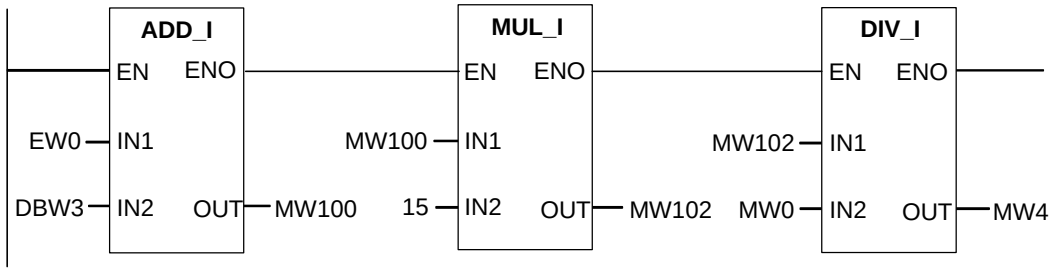
El programa de ejemplo siguiente muestra cómo obtener con tres operaciones aritméticas para enteros el mismo resultado que la ecuación:

MW4 = ((EW0 + DBW3) x 15) / MW0

Esquema de contactos

Segmento 1: Abrir bloque de datos DB1.

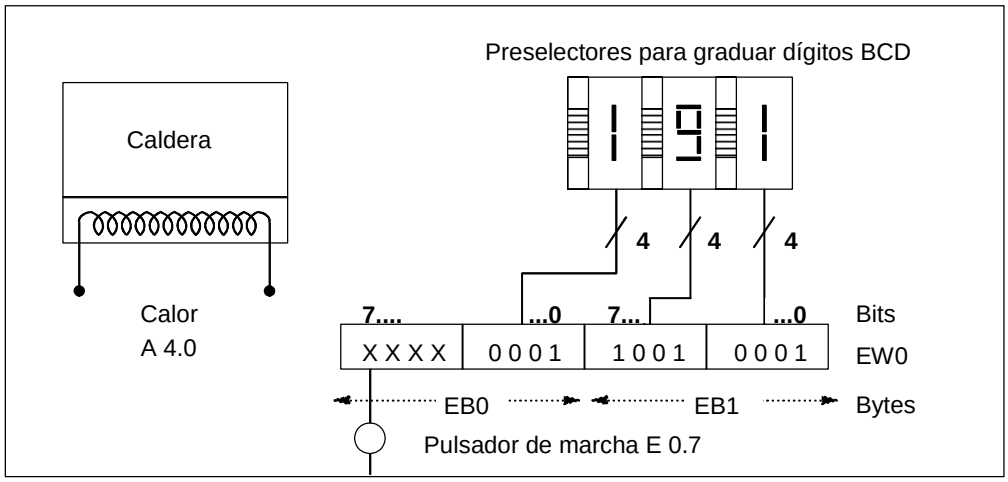
Segmento 2: La palabra de entrada EW0 se suma a la palabra de datos globales DBW3 (el bloque de datos tiene que estar definido y abierto) y la suma se carga en la palabra de marcas MW100. Después se multiplica MW100 por 15 y el resultado se deposita en la palabra doble de marcas MW102. Luego se divide MW102 entre MW0. Es resultado se guarda en MW4.



B.6 Ejemplo: Operaciones lógicas con palabras

Calentar una caldera

El operador de la caldera conecta la caldera accionando el pulsador de marcha. El operador puede graduar un tiempo de calentamiento utilizando los preselectores mecánicos. El valor fijado por el operador indica los segundos en formato decimal codificado en binario (BCD).



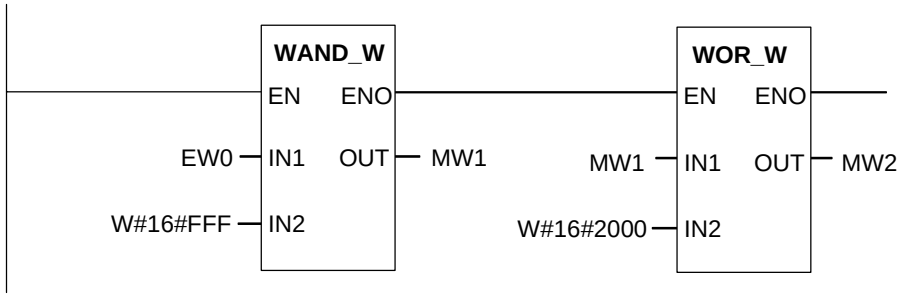
Componente del sistema	Dirección absoluta
Pulsador de marcha	E 0.7
Preselector digital para unidades	E 1.0 a E 1.3
Preselector digital para decenas	E 1.4 a E 1.7
Preselector digital para centenas	E 0.0 a E 0.3
Comienzo del proceso de calentamiento	A 4.0

Esquema de contactos

Segmento 1: Si el temporizador está en marcha, comienza el proceso de calentamiento.

Segmento 2: Si el temporizador está en marcha, la operación Retorno finaliza aquí.

Segmento 3: Enmascarar los bits de entrada E 0.4 a E 0.7 (es decir, ponerlos a 0). Estos bits de las entradas de los preselectores no se utilizan. Los 16 bits de las entradas de los preselectores se combinan con W#16#0FFF mediante la operación **Y con palabras**. El resultado se carga en la palabra de marcas MW1. Para regular la base de tiempo en segundos se combina el valor de preselección con W#16#2000 mediante la operación **O con palabras**, poniendo el bit 13 a 1 y el bit 12 a 0.



Segmento 4: Arrancar el temporizador T1 como temporizador de impulso prolongado, cuando se oprima el pulsador de marcha, cargando la palabra de marcas MW2 como valor de preselección (derivado de la lógica arriba descrita).

C Uso de KOP

C.1 Mecanismo EN/ENO

La habilitación (EN) y la salida de habilitación (ENO) de cuadros (boxes) FUP/KOP se realiza mediante el bit RB.

Si EN y ENO están conectados, rige:

ENO = EN AND NOT (error del cuadro)

Si no se produce ningún error (error del cuadro = 0), entonces ENO = EN.

El mecanismo EN/ENO se utiliza para:

- operaciones aritméticas
- operaciones de transferencia y de conversión
- operaciones de desplazamiento y de rotación
- llamadas de bloques

Este mecanismo **no se utiliza** para:

- comparaciones
- contadores
- temporizadores

Al rededor de las verdaderas instrucciones del cuadro o box se generan instrucciones adicionales para el mecanismo EN/ENO dependiendo de las operaciones lógicas precedentes o siguientes. Los cuatro casos posibles se indican a continuación con un sumando a modo de ejemplo:

- Sumando con conexión EN y con conexión ENO
- Sumando con conexión EN y sin conexión ENO
- Sumando sin conexión EN y con conexión ENO
- Sumando sin conexión EN y sin conexión ENO

Advertencia para la creación de bloques propios

Si desea escribir bloques y llamarlos después en FUP/KOP deberá vigilar que el bit RB esté activado al salir del bloque. El cuarto ejemplo demuestra que ésto no ocurre automáticamente. El RB no puede ser utilizado igual que una marca, ya que es sobreescrito continuamente por el mecanismo EN/ENO. Utilice en su lugar una variable temporal para guardar los errores que se hayan producido. Inicialice la variable con el valor 0. Active esta variable sirviéndose del mecanismo EN/ENO en cualquier punto del bloque donde crea que una operación fracasada pueda representar un error para todo el bloque. Para ello basta un NOT y una bobina de activación. Al final del bloque programa un segmento:

```
ende:  UN fehler

       SAVE
```

Tome las medidas necesarias para que este segmento se ejecute en cualquier caso, es decir, no utilice nunca una operación BEB dentro de este bloque e impida que se salte este segmento.

C.1.1 Sumando con conexión EN y ENO

Si el sumando tiene una conexión EN y una ENO se añaden las siguientes instrucciones AWL:

```
1    U   E   0.0        // Conexión EN
2    SPBNB _001         // Desplazar RLO al RB y saltar si RLO == 0
3    L     in1          // Parámetro del cuadro
4    L     in2          // Parámetro del cuadro
5    +I                 // Suma
6    T     out          // Parámetro del cuadro
7    UN    OV           // Detección de errores
8    SAVE               // Guardar error en RB
9    CLR                // Primera consulta
10   _001:    U    RB   // Desplaza RB al RLO
11   =     A   4.0
```

Después de la línea1 el RLO contiene el resultado de la combinación lógica precedente. La instrucción SPBNB copia el RLO en el RB y activa el bit de primera consulta.

- Si el RLO es 0, se salta a la línea 10 y se continúa con U RB. La suma no se lleva a cabo. En la línea 10 se copia el RB en el RLO y por lo tanto se le asigna un 0 a la salida.
- Si el RLO es 1, no se salta, es decir, se realiza la suma. En la línea 7 se determina si se ha producido un error al sumar, lo cual se almacena en el bit RB en la línea 8. La línea 9 activa el bit de primera consulta. En la línea 10 se vuelve a copiar el bit RB en el RLO, con lo cual en la salida se indica si la suma es correcta.
  El bit RB no se modifica en las líneas 10 y 11, por lo que también indica si la suma es correcta.

C.1.2 Sumando con conexión EN y sin conexión ENO

Si el sumando tiene una conexión EN, pero no tiene una conexión ENO, se añaden las siguientes instrucciones AWL:

```
1    U     E     0.0    // Conexión EN
2    SPBNB _001         // Desplazar RLO al RB y saltar si RLO == 0
3    L     in1          // Parámetro del cuadro
4    L     in2          // Parámetro del cuadro
5    +I                 // Suma
6    T     out          // Parámetro del cuadro
7    _001:       NOP   0
```

Después de la línea 1 el RLO contiene el resultado de la combinación lógica precedente. La instrucción SPBNB copia el RLO en el RB y activa el bit de primera consulta.

- Si el RLO es 0, se salta a la línea 7, no se realiza la suma y el RLO y RB son 0.
- Si el RLO era 1, no se salta, es decir, se realiza la suma. No se determina si se ha producido un error al sumar. El RLO y el RB son 1.

C.1.3 Sumando sin conexión EN y con conexión ENO

Si el sumando no tiene ninguna conexión EN, pero tiene una ENO, se añaden las siguientes instrucciones AWL:

```
1    L     in1     // Parámetro del cuadro
2    L     in2     // Parámetro del cuadro
3    +I            // Suma
4    T     out     // Parámetro del cuadro
5    UN    OV      // Detección de errores
6    SAVE          // Guardar error en RB
7    CLR           // Primera consulta
8    U     RB      // Desplazar RB a RLO
9    =     A     4.0
```

La suma se realiza en cualquier caso. En la línea 5 se determina si se ha producido un error al sumar, lo cual se guarda en el RB en la línea 6. La línea 7 activa el bit de primera consulta. En la línea 8 se vuelve a copiar el bit RB en el RLO, con lo cual se indica en la salida si la suma ha tenido éxito.

El bit RB no se modifica en las líneas 8 y 9, es decir que también indica si la suma ha tenido éxito.

C.1.4 Sumando sin conexión EN y sin conexión ENO

Si el sumando no tiene ninguna conexión EN y tampoco una ENO se añaden las siguientes instrucciones AWL:

```
1    L     in1   // Parámetro del cuadro
2    L     in2   // Parámetro del cuadro
3    +I          // Suma
4    T     out   // Parámetro del cuadro
5    NOP   0
```

La suma se ejecuta. El RLO y el bit RB no cambian.

C.2 Entrega de parámetros

Los parámetros de un bloque se entregan o transfieren en forma de valores. En el caso de los bloques de función (FB), el bloque llamado utiliza una copia del valor del parámetro actual (real) que se encuentra en el DB de instancia. En el caso de las funciones (FC), la pila de datos locales contiene una copia del valor actual (real). Antes de la llamada se copian los valores INPUT en el DB de instancia, o en la pila LSTACK, según el caso. Después de la llamada se copian los valores OUTPUT en las variables. Dentro del bloque llamado se opera solamente con una copia. Las instrucciones AWL necesarias se encuentran en el bloque que efectúa la llamada y quedan ocultos para el usuario.

Nota

Cuando en una función se utilizan marcas, entradas, salidas o entradas y salidas de la periferia como operandos actuales, éstas reciben otro tratamiento que los demás operandos. La actualización no se lleva a cabo a través de la pila LSTACK sino directamente.

Excepción:

Si el parámetro formal correspondiente es un parámetro de entrada del tipo de datos BOOL, la actualización de los parámetros actuales tiene lugar por LSTACK.

Atención

Vigile que al programar el bloque llamado también se escriban los parámetros declarados como OUTPUT, de lo contrario los valores emitidos serán arbitrarios. En el caso de los bloques de función se obtiene el valor del DB de instancia memorizado en la última llamada, mientras que en el caso de las funciones se obtiene aquel valor que se encuentre casualmente en la pila de datos locales (LSTACK).

Tenga en cuenta los puntos siguientes:

- Inicialice en lo posible todos los parámetros OUTPUT.
- Evite utilizar instrucciones de activación (set) y desactivación (reset). Estas instrucciones dependen del RLO. Si el RLO es 0 se mantiene el valor casual.
- Si programa un salto en el bloque, tenga cuidado de no saltarse ninguna parte en la que se escriban parámetros OUTPUT. No se olvide del BEB y del efecto que tienen las instrucciones MCR.

Índice alfabético

C

D

E

F

I

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z