

# SIEMENS

## SIMATIC

### Lista de instrucciones (AWL) para S7-300 y S7-400

Manual de referencia

Este manual forma parte del paquete de documentación con la referencia:

**6ES7810-4CA10-8DW1**

**05/2010**  
A5E02790285-01

Operaciones lógicas con bits	1
Operaciones de comparación	2
Operaciones de conversión	3
Operaciones de contaje	4
Operaciones con los bloques de datos	5
Operaciones de salto	6
Aritmética de enteros	7
Aritmética en coma flotante	8
Operaciones de carga y transferencia	9
Control de programa	10
Operaciones de desplazamiento y rotación	11
Operaciones de temporización	12
Operaciones lógicas con palabras	13
Operaciones con acumuladores	14
Sinopsis de las operaciones AWL	A
Ejemplos de programación	B
Transferencia de parámetros	C

## Notas jurídicas

### Filosofía en la señalización de advertencias y peligros

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.

 <b>PELIGRO</b>
Significa que, si no se adoptan las medidas preventivas adecuadas <b>se producirá</b> la muerte, o bien lesiones corporales graves.
 <b>ADVERTENCIA</b>
Significa que, si no se adoptan las medidas preventivas adecuadas <b>puede producirse</b> la muerte o bien lesiones corporales graves.
 <b>PRECAUCIÓN</b>
con triángulo de advertencia significa que si no se adoptan las medidas preventivas adecuadas, pueden producirse lesiones corporales.
<b>PRECAUCIÓN</b>
sin triángulo de advertencia significa que si no se adoptan las medidas preventivas adecuadas, pueden producirse daños materiales.
<b>ATENCIÓN</b>
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 producto/sistema tratado en esta documentación sólo deberá ser manejado o manipulado por **personal cualificado** para la tarea encomendada y observando lo indicado en la documentación correspondiente a la misma, particularmente las consignas de seguridad y advertencias en ella incluidas. Debido a su formación y experiencia, el personal cualificado está en condiciones de reconocer riesgos resultantes del manejo o manipulación de dichos productos/sistemas y de evitar posibles peligros.

### Uso previsto o de los productos de Siemens

Considere lo siguiente:

 <b>ADVERTENCIA</b>
Los productos de Siemens sólo deberán usarse para los casos de aplicación previstos en el catálogo y la documentación técnica asociada. De usarse productos y componentes de terceros, éstos deberán haber sido recomendados u homologados por Siemens. El funcionamiento correcto y seguro de los productos exige que su transporte, almacenamiento, instalación, montaje, manejo y mantenimiento hayan sido realizados de forma correcta. Es preciso respetar las condiciones ambientales permitidas. También deberán seguirse las indicaciones y advertencias que figuran en la documentación asociada.

### 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 las correcciones se incluyen en la siguiente edición.

# Prológo

## Objetivo del manual

Este manual le servirá de ayuda al crear programas de usuario con el lenguaje de programación AWL. Describe los elementos del lenguaje de programación AWL, 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 XP, MS Windows Server 2003 o MS Windows 7.

## Objeto del manual

El software en el que se basan las indicaciones del manual es STEP 7 V5.5.

## Cumplimiento de la normativa IEC 1131-3

AWL sigue los principios del lenguaje "Lista de Instrucciones" (en inglés 'Instruction List') 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.RTF de STEP 7 encontrará información más detallada sobre el cumplimiento de las normas.

## Requisitos

Para entender correctamente el presente manual de AWL 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: <ul style="list-style-type: none"> <li>• STEP 7 : Introducción y ejercicios prácticos</li> <li>• Programar con STEP 7</li> <li>• Configurar el hardware y la comunicación con STEP 7</li> <li>• De S5 a S7, Guía para facilitar la transición</li> </ul>	Nociones básicas para el personal técnico. Describe cómo realizar soluciones de control con el software STEP 7 para los sistemas S7-300/400.	6ES7810-4CA10-8DW0
Información de referencia para STEP 7, compuesta por <ul style="list-style-type: none"> <li>• Manuales KOP/FUP/AWL para S7-300/400</li> <li>• Funciones estándar y funciones de sistema para S7-300/400 Tomo 1 y Tomo 2</li> </ul>	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 7.	6ES7810-4CA10-8DW1

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 AWL. 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 90026 Nuernberg.

Internet: <http://www.sitrain.com>

## **Technical Support**

Podrá acceder al Technical Support de todos los productos de Industry Automation and Drive Technology

- a través del formulario de Internet para el Support Request  
<http://www.siemens.com/automation/support-request>

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 "Support al producto" 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 Industry Automation and Drive Technology de su región,
- Informaciones sobre reparaciones, piezas de repuesto y consulting.

# Índice

<b>1</b>	<b>Operaciones lógicas con bits</b>	<b>13</b>
1.1	Lista de operaciones lógicas con bits .....	13
1.2	U Y .....	15
1.3	UN Y-No .....	16
1.4	O O .....	17
1.5	ON O-No .....	18
1.6	X O-exclusiva .....	19
1.7	XN O-exclusiva-NO .....	20
1.8	O Y antes de O .....	21
1.9	U( Y con abrir paréntesis .....	22
1.10	UN( Y-No con abrir paréntesis .....	23
1.11	O( O con abrir paréntesis .....	23
1.12	ON( O-No con abrir paréntesis .....	24
1.13	X( O-exclusiva con abrir paréntesis .....	24
1.14	XN( O-exclusiva-NO con abrir paréntesis .....	25
1.15	) Cerrar paréntesis .....	25
1.16	= Asignar .....	27
1.17	R Desactivar .....	28
1.18	S Activar .....	29
1.19	NOT Negar el RLO .....	30
1.20	SET Activar el RLO (=1) .....	30
1.21	CLR Desactivar RLO (=0) .....	32
1.22	SAVE Memorizar el RLO en el registro RB .....	33
1.23	FN Flanco negativo .....	34
1.24	FP Flanco positivo .....	36
<b>2</b>	<b>Operaciones de comparación</b>	<b>39</b>
2.1	Lista de operaciones de comparación .....	39
2.2	? I Comparar enteros .....	40
2.3	? D Comparar enteros dobles .....	41
2.4	? R Comparar números en coma flotante (32 bits) .....	42
<b>3</b>	<b>Operaciones de conversión</b>	<b>43</b>
3.1	Lista de operaciones de conversión .....	43
3.2	BTI Convertir BCD a entero .....	44
3.3	ITB Convertir entero en BCD .....	45
3.4	BTD Convertir número BCD a entero doble .....	46
3.5	ITD Convertir entero en entero doble .....	47
3.6	DTB Convertir entero doble en BCD .....	48
3.7	DTR Convertir entero doble en número en coma flotante (32 bits, IEEE 754) .....	49
3.8	INVI Complemento a uno de un entero .....	50
3.9	INVD Complemento a uno de un entero doble .....	51
3.10	NEGI Complemento a dos de un entero .....	52
3.11	NEGD Complemento a dos de un entero doble .....	53
3.12	NEGR Invertir un número en coma flotante (32 bits, IEEE 754) .....	54
3.13	TAW Cambiar el orden de los bytes en el ACU 1-L (16 bits) .....	55
3.14	TAD Invertir el orden de los bytes en el ACU 1 (32 bits) .....	56

3.15	RND Redondear un número en coma flotante a entero .....	57
3.16	TRUNC Truncar .....	58
3.17	RND+ Redondear un número real al próximo entero superior .....	59
3.18	RND- Redondear un número real al próximo entero inferior .....	60
<b>4</b>	<b>Operaciones de contaje</b> .....	<b>61</b>
4.1	Lista de operaciones de contaje .....	61
4.2	FR Habilitar contador .....	62
4.3	L Cargar valor actual del contador en ACU 1 en forma de entero .....	63
4.4	LC Cargar valor actual del contador en ACU 1 como número BCD.....	64
4.5	R Desactivar contador.....	66
4.6	S Poner contador al valor inicial .....	67
4.7	ZV Incrementar contador .....	68
4.8	ZR Decrementar contador.....	69
<b>5</b>	<b>Operaciones con los bloques de datos</b> .....	<b>71</b>
5.1	Lista de operaciones con bloques .....	71
5.2	AUF Abrir bloque de datos.....	72
5.3	TDB Intercambiar bloque de datos global y bloque de datos de instancia.....	73
5.4	L DBLG Cargar la longitud del DB global en el ACU 1.....	73
5.5	L DBNO Cargar número del bloque de datos global en ACU 1.....	74
5.6	L DILG Cargar longitud del bloque de datos de instancia en ACU 1 .....	74
5.7	L DINO Cargar número del bloque de datos de instancia en ACU 1 .....	75
<b>6</b>	<b>Operaciones de salto</b> .....	<b>77</b>
6.1	Lista de operaciones de salto .....	77
6.2	SPA Salto incondicionado.....	79
6.3	SPL Saltar utilizando una lista de metas .....	80
6.4	SPB Saltar si RLO = 1 .....	82
6.5	SPBN Saltar si RLO = 0.....	83
6.6	SPBB Saltar si RLO = 1 y salvaguardar RLO en RB.....	84
6.7	SPBNB Saltar si RLO = 0 y salvar RLO en RB .....	85
6.8	SPBI Saltar si RB = 1.....	86
6.9	SPBIN Saltar si RB = 0 .....	87
6.10	SPO Saltar si OV = 1 .....	88
6.11	SPS Saltar si OS = 1.....	89
6.12	SPZ Saltar si el resultado = 0 .....	91
6.13	SPN Saltar si resultado <> 0.....	92
6.14	SPP Saltar si el resultado > 0 .....	93
6.15	SPM Saltar si resultado < 0 .....	94
6.16	SPPZ Saltar si el resultado >= 0.....	95
6.17	SPMZ Saltar si el resultado <= 0 .....	96
6.18	SPU Saltar si el resultado no es válido.....	97
6.19	LOOP Bucle .....	99

<b>7</b>	<b>Aritmética de enteros</b>	<b>101</b>
7.1	Lista de operaciones aritméticas con enteros .....	101
7.2	Evaluar bits de la palabra de estado en operaciones en coma fija .....	102
7.3	+I Sumar ACU 1 y 2 como entero .....	103
7.4	-I Restar ACU 1 de ACU 2 como entero .....	104
7.5	*I Multiplicar ACU 1 por ACU 2 como entero .....	105
7.6	/I Dividir ACU 2 por ACU 1 como entero .....	106
7.7	+ Sumar constante entera o entera doble .....	108
7.8	+D Sumar ACU 1 y 2 como entero doble .....	110
7.9	-D Restar ACU 1 de ACU 2 como entero doble .....	111
7.10	*D Multiplicar ACU 1 por ACU 2 como entero doble .....	112
7.11	/D Dividir ACU 2 por ACU 1 como entero doble .....	113
7.12	MOD Resto de la división de enteros dobles.....	114
<b>8</b>	<b>Aritmética en coma flotante</b>	<b>115</b>
8.1	Lista de operaciones aritméticas con números en coma flotante.....	115
8.2	Evaluar los bits de la palabra de estado en operaciones en coma flotante .....	116
8.3	Operaciones básicas .....	117
8.3.1	+R Sumar ACU 1 y 2 como número en coma flotante (32 bits) .....	117
8.3.2	-R Restar ACU 1 de ACU 2 como número en coma flotante (32 bits) .....	119
8.3.3	*R Multiplicar ACU 1 por ACU 2 como número en coma flotante (32 bits) .....	120
8.3.4	/R Dividir ACU 2 por ACU 1 como número en coma flotante (32 bits).....	121
8.3.5	ABS Valor absoluto de un número en coma flotante (32 bits, IEEE 754) .....	122
8.4	Operaciones ampliadas .....	123
8.4.1	SQR Calcular el cuadrado de un número en coma flotante (32 bits).....	123
8.4.2	SQRT Calcular la raíz cuadrada de un número en coma flotante (32 bits).....	124
8.4.3	EXP Calcular el exponente de un número en coma flotante (32 bits).....	125
8.4.4	LN Calcular el logaritmo natural de un número en coma flotante (32 bits) .....	126
8.4.5	SIN Calcular el seno de ángulos como números en coma flotante (32 bits).....	127
8.4.6	COS Calcular el coseno de ángulos como números en coma flotante (32 bits) .....	128
8.4.7	TAN Calcular la tangente de ángulos como números en coma flotante (32 bits) .....	128
8.4.8	ASIN Calcular el arcoseno de un número en coma flotante (32 bits).....	129
8.4.9	ACOS Calcular el arcocoseno de un número en coma flotante (32 bits).....	130
8.4.10	ATAN Calcular la arcotangente de un número en coma flotante (32 bits) .....	131
<b>9</b>	<b>Operaciones de carga y transferencia</b>	<b>133</b>
9.1	Lista de operaciones de cargar y transferencia.....	133
9.2	L Cargar .....	134
9.3	L STW Cargar palabra de estado en ACU 1 .....	136
9.4	LAR1 Cargar registro de direcciones 1 con contenido del ACU 1.....	137
9.5	LAR1 <D> Cargar registro de direcciones 1 con puntero (formato de 32 bits) .....	138
9.6	LAR1 AR2 Cargar registro de direcciones 1 con contenido del registro de direcciones 2.....	139
9.7	LAR2 Cargar registro de direcciones 2 con contenido del ACU 1.....	139
9.8	LAR2 <D> Cargar registro de direcciones 2 con puntero (formato de 32 bits) .....	140
9.9	T Transferir.....	141
9.10	T STW Transferir ACU 1 a la palabra de estado.....	142
9.11	TAR Intercambiar registro de direcciones 1 y registro de direcciones 2 .....	143
9.12	TAR1 Transferir registro de direcciones 1 a ACU 1 .....	143
9.13	TAR1 <D> Transferir registro de direcciones 1 a dirección de destino (puntero de 32 bits) ...	144
9.14	TAR1 AR2 Transferir registro de direcciones 1 a registro de direcciones 2 .....	145
9.15	TAR2 Transferir registro de direcciones 2 a ACU 1 .....	145
9.16	TAR2 <D> Transferir registro de direcciones 2 a dirección de destino (puntero de 32 bits) ...	146

<b>10</b>	<b>Control de programa</b>	<b>147</b>
10.1	Lista de operaciones de control del programa.....	147
10.2	BE Fin de bloque.....	148
10.3	BEB Fin de bloque condicionado.....	149
10.4	BEA Fin de bloque incondicionado.....	150
10.5	CALL Llamada.....	151
10.6	Llamar a un FB.....	154
10.7	Llamar a una FC.....	156
10.8	Llamar a un SFB.....	158
10.9	Llamar a una SFC.....	160
10.10	Llamar a una multiinstancia.....	161
10.11	Llamar a un bloque de una librería.....	161
10.12	CC Llamada condicionada.....	162
10.13	UC Llamada incondicionada.....	163
10.14	EI MCR (Master Control Relay).....	164
10.15	Notas importantes sobre el uso de la función MCR.....	166
10.16	MCR( Almacenar el RLO en pila MCR, inicio área MCR.....	167
10.17	)MCR Fin área MCR.....	169
10.18	MCRA Activar área MCR.....	170
10.19	MCRD Desactivar área MCR.....	171
<b>11</b>	<b>Operaciones de desplazamiento y rotación</b>	<b>173</b>
11.1	Operaciones de desplazamiento.....	173
11.1.1	Lista de operaciones de desplazamiento.....	173
11.1.2	SSI Desplazar signo de número entero a la derecha (16 bits).....	174
11.1.3	SSD Desplazar signo de número entero a la derecha (32 bits).....	176
11.1.4	SLW Desplazar palabra a la izquierda (16 bits).....	178
11.1.5	SRW Desplazar palabra a la derecha (16 bits).....	180
11.1.6	SLD Desplazar doble palabra a la izquierda (32 bits).....	182
11.1.7	SRD Desplazar doble palabra a la derecha (32 bits).....	184
11.2	Operaciones de rotación.....	186
11.2.1	Lista de operaciones de rotación.....	186
11.2.2	RLD Rotar doble palabra a la izquierda (32 bits).....	187
11.2.3	RRD Rotar doble palabra a la derecha (32 bits).....	189
11.2.4	RLDA Rotar ACU 1 a la izquierda vía A1 (32 bits).....	191
11.2.5	RRDA Rotar ACU 1 a la derecha vía A1 (32 bits).....	192
<b>12</b>	<b>Operaciones de temporización</b>	<b>193</b>
12.1	Lista de operaciones de temporización.....	193
12.2	Area de memoria y componentes de un temporizador.....	194
12.3	FR Habilitar temporizador.....	197
12.4	L Cargar valor actual del temporizador en ACU 1 como entero.....	199
12.5	LC Cargar el valor actual de temporización en ACU 1 como número BCD.....	201
12.6	R Desactivar temporizador.....	203
12.7	SI Temporizador como impulso.....	204
12.8	SV Temporizador como impulso prolongado.....	206
12.9	SE Temporizador como retardo a la conexión.....	208
12.10	SS Temporizador como retardo a la conexión con memoria.....	210
12.11	SA Temporizador como retardo a la desconexión.....	212

<b>13</b>	<b>Operaciones lógicas con palabras</b>	<b>215</b>
13.1	Lista de operaciones lógicas con palabras.....	215
13.2	UW Y con palabra (16 bits).....	216
13.3	OW O con palabra (16 bits).....	218
13.4	XOW O-exclusiva con palabra (16 bits).....	220
13.5	UD Y con doble palabra (32 bits).....	222
13.6	OD O con doble palabra (32 bits).....	224
13.7	XOD O-exclusiva con doble palabra (32 bits).....	226
<b>14</b>	<b>Operaciones con acumuladores</b>	<b>229</b>
14.1	Lista de operaciones con acumuladores.....	229
14.2	TAK Intercambiar ACU 1 y ACU 2.....	230
14.3	PUSH CPU con dos acumuladores.....	231
14.4	PUSH CPU con cuatro acumuladores.....	232
14.5	POP CPU con dos acumuladores.....	233
14.6	POP CPU con cuatro acumuladores.....	234
14.7	ENT Introducir pila de ACU.....	235
14.8	LEAVE Salir de la pila de ACU.....	235
14.9	INC Incrementar ACU 1-L-L.....	236
14.10	DEC Decrementar ACU 1-L-L.....	237
14.11	+AR1 Sumar el ACU 1 al registro de direcciones 1.....	238
14.12	+AR2 Sumar el ACU1 al registro de direcciones 2.....	239
14.13	BLD Estructuración de imagen (operación nula).....	240
14.14	NOP 0 Operación nula 0.....	241
14.15	NOP 1 Operación nula 1.....	241
<b>A</b>	<b>Sinopsis de las operaciones AWL</b>	<b>243</b>
A.1	Operaciones AWL ordenadas según la nemotécnica alemana (SIMATIC).....	243
A.2	Operaciones AWL ordenadas según la nemotécnica inglesa (internacional).....	248
<b>B</b>	<b>Ejemplos de programación</b>	<b>253</b>
B.1	Lista de ejemplos de programación.....	253
B.2	Ejemplos: Operaciones lógicas con bits.....	254
B.3	Ejemplo: Operaciones de temporización.....	257
B.4	Ejemplo: Operaciones de contaje y comparación.....	260
B.5	Ejemplo: Operaciones de aritmética con enteros.....	262
B.6	Ejemplo: Operaciones lógicas con palabras.....	263
<b>C</b>	<b>Transferencia de parámetros</b>	<b>265</b>

Index

Fehler! Textmarke nicht definiert.



# 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 constituyen 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.

Las operaciones básicas para las operaciones lógicas con bits son:

- U Y
- UN Y-No
- O O
- ON O-No
- X O-exclusiva
- XN O-exclusiva-No

Las siguientes operaciones permiten ejecutar una cadena lógica encerrada entre paréntesis:

- U( Y con abrir paréntesis
- UN( Y-No con abrir paréntesis
- O( O con abrir paréntesis
- ON( O-No con abrir paréntesis
- X( O-exclusiva con abrir paréntesis
- XN( O-exclusiva-NO con abrir paréntesis
- ) Cerrar paréntesis

Para terminar una cadena lógica se puede utilizar una de las tres operaciones:

- = Asignar
- R Desactivar
- S Activar

*1.1 Lista de operaciones lógicas con bits*

Las operaciones siguientes permiten modificar el resultado lógico (RLO):

- NOT Negar el RLO
- SET Activar el RLO (=1)
- CLR Desactivar RLO (=0)
- SAVE Memorizar el RLO en el registro RB

Otras operaciones detectan cambios en el resultado lógico y reaccionan correspondientemente:

- FN Flanco negativo
- FP Flanco positivo

## 1.2 U Y

### Formato

U <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D, T, Z

### Descripción de la operación

U consulta el bit direccionado para saber si tiene el estado de señal "1", y combina el resultado de la consulta con el RLO realizando una Y lógica.

#### Consultar el estado de los bits de la palabra de estado:

Utilizando la operación Y también se puede consultar directamente la palabra de estado. A tal fin, empléense los siguientes operandos: ==0, <>0, >0, <0, >=0, <=0, UO, RB, OS, OV.

### Palabra de estado

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

### Ejemplo

Programa AWL	Esquema de conexiones de relé		
	Barra de alimentación		
U	E 1.0	E 1.0 Estado de señal 1	▼ Contacto normal. abierto
U	E 1.1	E 1.1 Estado de señal 1	▼ Contacto normal. abierto
=	A 4.0	A 4.0 Estado de señal 1	□ Bobina
▼	Indica un contacto cerrado.		

### 1.3 UN Y-No

#### Formato

UN <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D, T, Z

#### Descripción de la operación

UN consulta el bit direccionado para saber si tiene el estado de señal "0" y combina el resultado de la consulta con el RLO realizando una Y lógica.

#### Consultar el estado de los bits de la palabra de estado:

Con la operación Y-No también se puede consultar directamente la palabra de estado. A tal fin, empléense los siguientes operandos: ==0, <>0, >0, <0, >=0, <=0, UO, RB, OS, OV.

#### Palabra de estado

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

#### Ejemplo

Programa AWL	Esquema de conexiones de relé		
	Barra de alimentación		
U E 1.0	E 1.0 Estado de señal 0		Contacto normal. abierto
UN E 1.1	E 1.1 Estado de señal 1		Contacto normal. cerrado
= A 4.0	A 4.0 Estado de señal 0		Bobina

## 1.4 O O

### Formato

O <bit>

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

### Descripción de la operación

O consulta el bit direccionado para saber si tiene el estado de señal "1", y combina el resultado de la consulta con el RLO realizando una O lógica.

#### Consultar el estado de los bits de la palabra de estado:

Con la operación O también se puede consultar directamente la palabra de estado. A tal fin, empléense los siguientes operandos: ==0, <>0, >0, <0, >=0, <=0, UO, RB, OS, OV.

### Palabra de estado

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

### Ejemplo

Programa AWL	Esquema conexiones de relé
O E 1.0	E 1.0 Estado de señal 1 Contacto
O E 1.1	E 1.1 Estado de señal 0 Contacto
= A 4.0	A 4.0 Estado de señal 1 Bobina
	<p>Indica un contacto cerrado.</p>

## 1.5 ON O-No

### Formato

ON <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D, T, Z

### Descripción de la operación

ON consulta el bit direccionado para saber si tiene el estado de señal "0", y combina el resultado de la consulta con el RLO realizando una O lógica.

#### Consultar el estado de los bits de la palabra de estado:

Con la operación O-No también se puede consultar directamente la palabra de estado. A tal fin, empléense los siguientes operandos: ==0, <>0, >0, <0, >=0, <=0, UO, RB, OS, OV.

### Palabra de estado

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

### Ejemplo

Programa AWL		Esquema de conexiones de relé	
		Barra de alimentación	
O	E 1.0	E 1.0 Estado de señal 0	Contacto normalm. abierto
ON	E 1.1	E 1.1 Estado de señal 1	Contacto normalm. cerrado
=	A 4.0	A 4.0 Estado de señal 1	Bobina

## 1.6 X O-exclusiva

### Formato

X <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D, T, Z

### Descripción de la operación

X consulta el bit direccionado para saber si su estado de señal es "1", y combina el resultado de la consulta con el RLO realizando una operación lógica O-exclusiva.

La función O-exclusiva se puede utilizar varias veces consecutivas. Entonces el resultado lógico común será "1" si un número impar de los operandos consultados da un "1" como resultado.

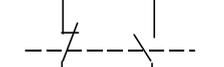
#### Consultar el estado de los bits de la palabra de estado:

Aplicando la operación O-exclusiva también se puede consultar directamente la palabra de estado. A tal fin, empléense los siguientes operandos: ==0, <>0, >0, <0, >=0, <=0, UO, RB, OS, OV.

### Palabra de estado

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

### Ejemplo

Programa AWL	Esquema de conexiones de relé
	Barra de alimentación 
X E 1.0	Contacto E 1.0 
X E 1.1	Contacto E 1.1 
= A 4.0	A 4.0 Bobina 
	

## 1.7 XN O-exclusiva-NO

### Formato

XN <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D, T, Z

### Descripción de la operación

**XN** consulta el bit direccionado para saber si tiene el estado de señal "0", y combina el resultado de la consulta con el RLO realizando una operación lógica O-exclusiva.

#### Consultar el estado de los bits de la palabra de estado:

Aplicando la operación O-exclusiva-NO también se puede consultar directamente la palabra de estado. A tal fin, empléense los siguientes operandos: ==0, <>0, >0, <0, >=0, <=0, UO, RB, OS, OV.

### Palabra de estado

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

### Ejemplo

Programa AWL	Esquema de conexiones de relé
X E 1.0	Contacto E 1.0
XN E 1.1	Contacto E 1.1
= A 4.0	A 4.0 Bobina

## 1.8 O Y antes de O

### Formato

O

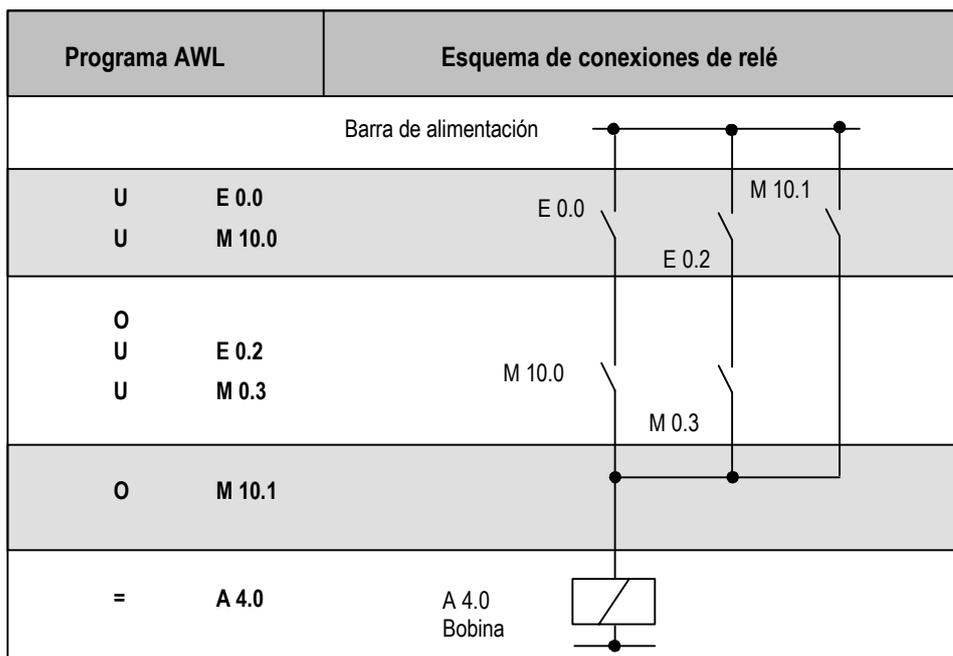
### Descripción de la operación

La operación **O** realiza una O lógica de combinaciones Y siguiendo la regla Y antes de O.

### Palabra de estado

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

### Ejemplo



## 1.9 U( Y con abrir paréntesis

### Formato

U(

### Descripción de la operación

U( (Y con abrir paréntesis) almacena en la pila de paréntesis los bits RLO y OR y un identificador de la operación. La pila de paréntesis puede contener un máximo de 7 entradas.

### Palabra de estado

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

### Ejemplo

Programa AWL	Esquema de conexiones de relé
	Barra de alimentación
U( O    E 0.0 O    M 10.0 )	
U    M 10.1	
=    A 4.0	A 4.0 Bobina

## 1.10 UN( Y-No con abrir paréntesis

### Formato

UN(

### Descripción de la operación

UN( (Y-No con abrir paréntesis) almacena en la pila de paréntesis los bits RLO y OR y el identificador de la operación. La pila de paréntesis puede contener un máximo de 7 entradas.

### Palabra de estado

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

## 1.11 O( O con abrir paréntesis

### Formato

O(

### Descripción de la operación

O( (O con abrir paréntesis) almacena en la pila de paréntesis los bits RLO y OR y el identificador de la operación. La pila de paréntesis puede contener un máximo de 7 entradas.

### Palabra de estado

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

## 1.12 ON( O-No con abrir paréntesis

### Formato

ON(

### Descripción de la operación

ON( (O-No con abrir paréntesis) almacena en la pila de paréntesis los bits RLO y OR y un identificador de la operación. La pila de paréntesis puede contener un máximo de 7 entradas.

### Palabra de estado

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

## 1.13 X( O-exclusiva con abrir paréntesis

### Formato

X(

### Descripción de la operación

X( (O-exclusiva con abrir paréntesis) almacena en la pila de paréntesis los bits RLO y OR y un identificador de la operación. La pila de paréntesis puede contener un máximo de 7 entradas.

### Palabra de estado

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

## 1.14 XN( O-exclusiva-NO con abrir paréntesis

### Formato

XN(

### Descripción de la operación

XN( (O-exclusiva-NO con abrir paréntesis) almacena en la pila de paréntesis los bits RLO y OR y un identificador de la operación. La pila de paréntesis puede contener un máximo de 7 entradas.

### Palabra de estado

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

## 1.15 ) Cerrar paréntesis

### Formato

)

### Descripción de la operación

) (Cerrar paréntesis) borra una entrada de la pila de paréntesis, restablece el bit OR, combina el RLO que hay en la entrada de pila con el RLO actual conforme al identificador de la operación y asigna el resultado al RLO. Si el identificador de la operación es Y o Y-No también se tiene en cuenta el bit OR.

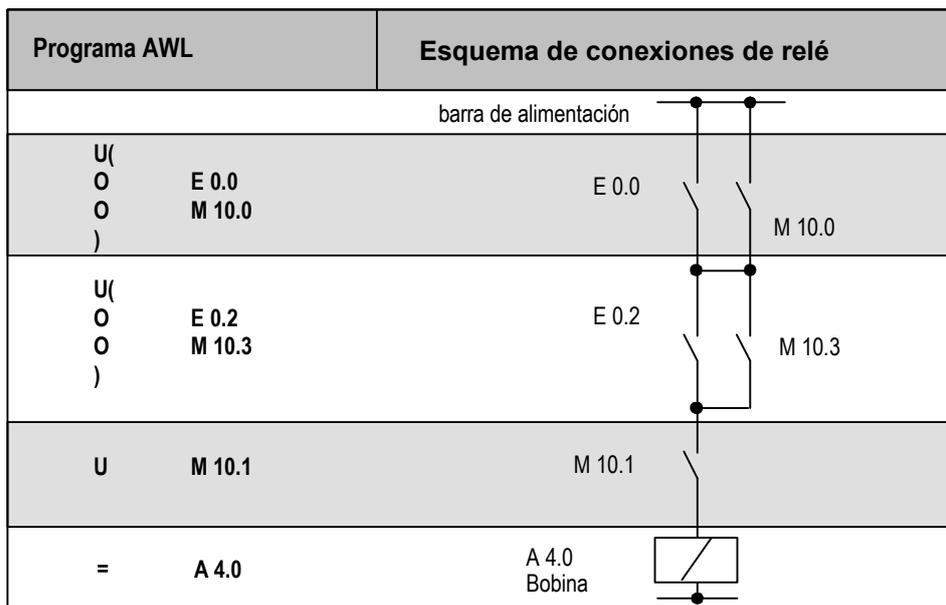
Operaciones para abrir paréntesis:

- U( Y con abrir paréntesis
- UN( Y-No con abrir paréntesis
- O( O con abrir paréntesis
- ON( O-No con abrir paréntesis
- X( O-exclusiva con abrir paréntesis
- XN( O-exclusiva-NO con abrir paréntesis

**Palabra de estado**

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

**Ejemplo**



## 1.16 = Asignar

### Formato

= <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D, T, Z

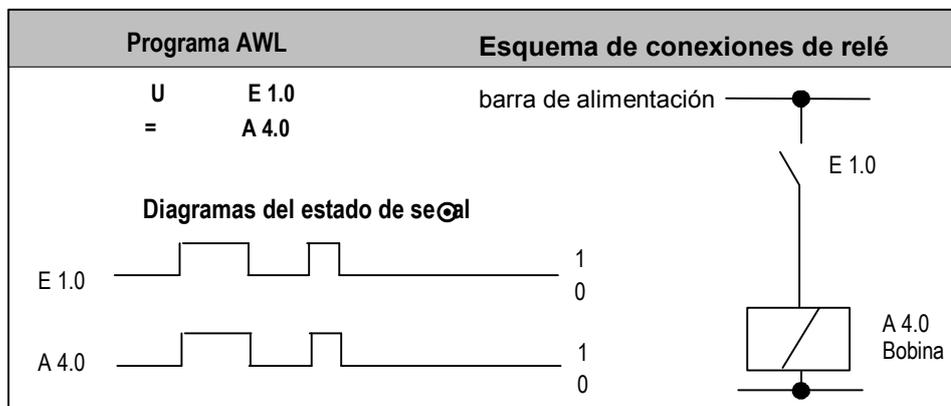
### Descripción de la operación

= <bit> escribe el RLO en el bit direccionado si el Master Control Relay está conectado (MCR = 1). Si el MCR es 0, en el bit direccionado se escribe el valor "0" en vez del RLO.

### Palabra de estado

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

### Ejemplo



## 1.17 R Desactivar

### Formato

R <bit>

Operando	Tipo de datos	Area de memoria
<bit>	BOOL	E, A, M, L, D

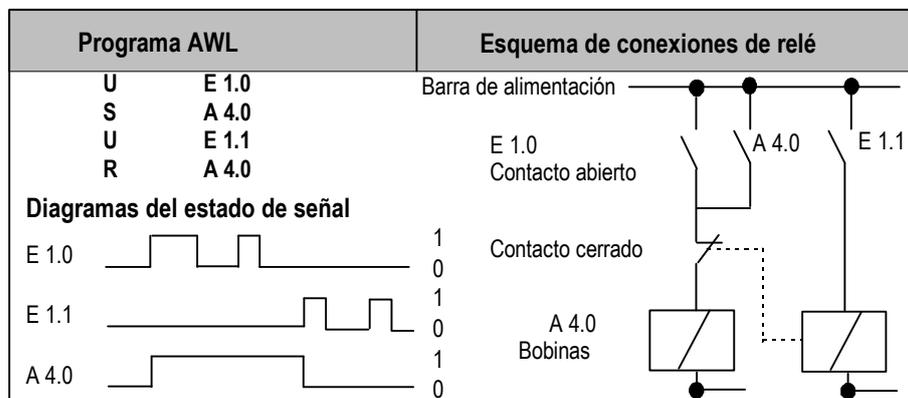
### Descripción de la operación

R (Desactivar bit) escribe el valor "0" en el bit direccionado si el RLO es 1 y si el Master Control Relay (MCR = 1) está conectado. Si el MCR es 0, el bit direccionado no varía.

### Palabra de estado

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

### Ejemplo





## 1.19 NOT Negar el RLO

### Formato

NOT

### Descripción de la operación

NOT niega el RLO.

### Palabra de estado

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

## 1.20 SET Activar el RLO (=1)

### Formato

SET

### Descripción de la operación

SET pone el RLO al estado de señal "1".

### Palabra de estado

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

### Ejemplo

Programa AWL	Estado de señal	Resultado lógico (RLO)
SET		1
= M 10.0	1	←
= M 15.1	1	
= M 16.0	1	
CLR		0
= M 10.1	0	←
= M 10.2	0	

## 1.21 CLR Desactivar RLO (=0)

### Formato

CLR

### Descripción de la operación

CLR pone el RLO al estado de señal "0".

### Palabra de estado

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

### Ejemplo

Programa AWL	Estado de señal	Resultado lógico (RLO)
<b>SET</b>		1
= M 10.0	1	←
= M 15.1	1	
= M 16.0	1	
<b>CLR</b>		0
= M 10.1	0	←
= M 10.2	0	

## 1.22 SAVE Memorizar el RLO en el registro RB

### Formato

SAVE

### Descripción de la operación

**SAVE** almacena el RLO (resultado lógico) en el bit RB (bit de resultado binario). El bit de primera consulta /ER no se pone a 0.

Por esta razón, en el caso de una operación lógica Y (AND) se combinará también el estado del bit RB en el siguiente segmento.

No se recomienda utilizar SAVE y consultar directamente después el bit RB en el mismo bloque o en bloques subordinados, ya que el bit RB puede ser modificado entretanto por muchas operaciones. Resulta conveniente usar SAVE antes de salir de un bloque, ya que así la salida ENO (bit RB) se pone al valor del bit RLO, lo que permite tratar a continuación los errores del bloque.

### Palabra de estado

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

## 1.23 FN Flanco negativo

### Formato

FN <bit>

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

### Descripción de la operación

FN <bit> (Flanco negativo) detecta un flanco negativo si el RLO cambia de "1" a "0", y lo indica con RLO = 1.

El estado de señal del bit RLO se compara durante cada ciclo del programa con el estado de señal del bit RLO del ciclo anterior para determinar los cambios de estado. Para poder ejecutar la comparación hay que almacenar el estado del bit RLO anterior en la dirección de la marca de flancos (<bit>). Si el estado de señal actual del bit RLO es distinto que el estado anterior ("1") (detección de un flanco negativo), tras ejecutarse esta operación el bit RLO será "1".

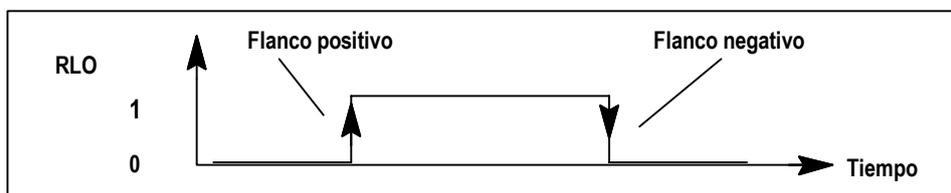
### Nota

Esta operación no conviene ejecutarla si el bit que se está vigilando se encuentra en la imagen del proceso, ya que los datos locales de un bloque sólo son válidos mientras se ejecuta dicho bloque.

### Palabra de estado

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

### Definición



**Ejemplo**

Si el autómata programable detecta un flanco negativo en el contacto E 1.0, activa la salida A 4.0 para un ciclo del OB1.

Programa AWL	Diagramas del estado de señal																		
<b>U</b> <b>E 1.0</b>	E 1.0										1 0								
<b>FN</b> <b>M 1.0</b>	M 1.0										1 0								
<b>=</b> <b>A 4.0</b>	A 4.0										1 0								
<b>N</b> de ciclo del OB1:	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">1</td> <td style="width: 20px; text-align: center;">2</td> <td style="width: 20px; text-align: center;">3</td> <td style="width: 20px; text-align: center;">4</td> <td style="width: 20px; text-align: center;">5</td> <td style="width: 20px; text-align: center;">6</td> <td style="width: 20px; text-align: center;">7</td> <td style="width: 20px; text-align: center;">8</td> <td style="width: 20px; text-align: center;">9</td> </tr> </table>										1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9											

## 1.24 FP Flanco positivo

### Formato

FP <bit>

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

### Descripción de la operación

**FP <bit>** (Flanco positivo) detecta un flanco positivo si el RLO cambia de "0" a "1", y lo indica con RLO = 1.

El estado de señal del bit RLO se compara durante cada ciclo del programa con el estado de señal del bit RLO del ciclo anterior para determinar los cambios de estado. Para poder ejecutar la comparación hay que almacenar el estado del bit RLO anterior en la dirección de la marca de flancos (<bit>). Si el estado de señal actual del bit RLO es distinto que el estado anterior ("0") (detección de un flanco positivo), tras ejecutarse esta operación el bit RLO será "1".

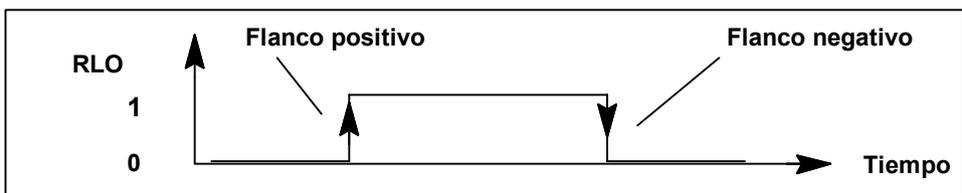
### Nota

Esta operación no conviene ejecutarla si el bit que se está vigilando se encuentra en la imagen del proceso, ya que los datos locales de un bloque sólo son válidos durante la ejecución del bloque.

### Palabra de estado

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

### Definition



### Ejemplo

Si el autómata programable detecta un flanco ascendente en el contacto E 1.0, activa la salida A 4.0 para un ciclo del OB1.

Programa AWL		Diagramas del estado de señal																		
<b>U</b>	<b>E 1.0</b>	E 1.0																		
<b>FP</b>	<b>M 1.0</b>	M 1.0																		
<b>=</b>	<b>A 4.0</b>	A 4.0																		
<b>N# de ciclo del OB1:</b>		<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">1</td> <td style="width: 20px; text-align: center;">2</td> <td style="width: 20px; text-align: center;">3</td> <td style="width: 20px; text-align: center;">4</td> <td style="width: 20px; text-align: center;">5</td> <td style="width: 20px; text-align: center;">6</td> <td style="width: 20px; text-align: center;">7</td> <td style="width: 20px; text-align: center;">8</td> <td style="width: 20px; text-align: center;">9</td> </tr> </table>										1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9												



## 2 Operaciones de comparación

### 2.1 Lista de operaciones de comparación

#### Descripción

Las operaciones Comparar enteros (de 16 bits) comparan el contenido del ACU2-L con el contenido del ACU1-L según los tipos de comparación siguientes:

- == ACU 2 es igual al ACU 1
- <> ACU 2 es diferente al ACU 1
- > ACU 2 es mayor que ACU 1
- < ACU 2 es menor que ACU 1
- >= ACU 2 es mayor que o igual al ACU 1
- <= ACU 2 es menor que o igual al ACU 1

RLO = 1 indica que el resultado de la comparación es verdadero. RLO = 0 indica que el resultado de la comparación es falso. Los bits A1 y A0 de la palabra de estado indican la relación "menor que", "igual que" o "mayor que".

Se dispone de las operaciones de comparación siguientes:

- ? I Comparar enteros ==, <>, >, <, >=, <=
- ? D Comparar enteros dobles ==, <>, >, <, >=, <=
- ? R Comparar números en coma flotante (32 bits) ==, <>, >, <, >=, <=

## 2.2 ? I Comparar enteros

### Formato

==I, <>I, >I, <I, >=I, <=I

### Descripción de la operación

Las operaciones **Comparar enteros (de 16 bits)** comparan el contenido del ACU2-L con el contenido del ACU1-L. Los contenidos de ACU2-L y ACU1-L se evalúan como enteros (de 16 bits). El RLO y los bits relevantes de la palabra de estado indican el resultado de la comparación. RLO = 1 indica que el resultado de la comparación es verdadero. RLO = 0 indica que el resultado de la comparación es falso. Los bits A1 y A0 de la palabra de estado indican la relación "menor que", "igual que" o "mayor que".

### Palabra de estado

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

### Valores del RLO

Operación de comparación ejecutada	RLO si ACU 2 > ACU 1	RLO si ACU 2 = ACU 1	RLO si ACC 2 < ACC 1
==I	0	1	0
<>I	1	0	1
>I	1	0	0
<I	0	0	1
>=I	1	1	0
<=I	0	1	1

### Ejemplo

AWL	Explicación
L MW10	//Cargar el contenido de MW10 (entero de 16 bits).
L EW24	//Cargar el contenido de EW24 (entero de 16 bits).
>I	//Comparar si el ACU2-L (MW10) es mayor (>) que el ACU1-L (EW24).
= M 2.0	//RLO = 1, si MW10 > EW24.

## 2.3 ? D Comparar enteros dobles

### Formato

==D, <>D, >D, <D, >=D, <=D

### Descripción de la operación

Las operaciones **Comparar enteros dobles** comparan el contenido del ACU 2 con el contenido del ACU 1. Los contenidos del ACU 2 y del ACU 1 se evalúan como enteros (de 32 bits). El RLO y los bits relevantes de la palabra de estado indican el resultado de la comparación. RLO = 1 indica que el resultado de la comparación es verdadero. RLO = 0 indica que el resultado de la comparación es falso. Los bits A1 y A0 de la palabra de estado indican la relación "menor que", "igual" o "mayor que".

### Palabra de estado

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

### Valores del RLO

Operación de comparación ejecutada	RLO si ACU 2 > ACU 1	RLO si ACU 2 = ACU 1	RLO si ACC 2 < ACC 1
==D	0	1	0
<>D	1	0	1
>D	1	0	0
<D	0	0	1
>=D	1	1	0
<=D	0	1	1

### Ejemplo

AWL	Explicación
L MD10	//Cargar el contenido de MD10 (entero de 32 bits).
L ED24	//Cargar el contenido de ED24 (entero de 32 bits).
>D	//Comparar si el ACU 2 (MD10) es mayor (>) que el ACU 1 (ED24).
= M 2.0	//RLO = 1, si MD10 > ED24.

## 2.4 ? R Comparar números en coma flotante (32 bits)

### Formato

==R, <>R, >R, <R, >=R, <=R

### Descripción de la operación

Las operaciones **Comparar números en coma flotante (32 bits, IEEE 754)** comparan el contenido del ACU 2 con el contenido del ACU 1. Los contenidos del ACU 1 y del ACU 2 se evalúan como números en coma flotante (32 bits, IEEE 754). El RLO y los bits relevantes de la palabra de estado indican el resultado de la comparación. RLO = 1 indica que el resultado de la comparación es verdadero. RLO = 0 indica que el resultado de la comparación es falso. Los bits A1 y A0 de la palabra de estado indican la relación "menor que", "igual que" o "mayor que".

### Palabra de estado

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

### Valores del RLO

Operación de comparación ejecutada	RLO si ACU 2 > ACU 1	RLO si ACU 2 = ACU 1	RLO si ACC 2 < ACC 1
==R	0	1	0
<>R	1	0	1
>R	1	0	0
<R	0	0	1
>=R	1	1	0
<=R	0	1	1

### Ejemplo

AWL	Explicación
L MD10	//Cargar el contenido de MD10 (número real).
L 1.359E+02	//Cargar la constante 1.359E+02.
>R	//Comparar si el ACU 2 (MD10) es mayor (>) que el ACU 1 (1.359E+02).
= M 2.0	//RLO = 1, si MD10 > 1.359E+02.

## 3 Operaciones de conversión

### 3.1 Lista de operaciones de conversión

#### Descripción

Las siguientes operaciones se utilizan para convertir números decimales codificados en binario y enteros a otros tipos de números:

- BTI Convertir BCD a entero
- ITB Convertir entero en BCD
- BTD Convertir número BCD a entero doble
- ITD Convertir entero en entero doble
- DTB Convertir entero doble en BCD
- DTR Convertir entero doble en número en coma flotante (32 bits, IEEE 754)

Para formar complementos de números enteros o para cambiar el signo de un número en coma flotante se utilizan las siguientes operaciones:

- INVI Complemento a uno de un entero
- INVD Complemento a uno de un entero doble
- NEGI Complemento a dos de un entero
- NEGD Complemento a dos de un entero doble
- NEGR Invertir un número en coma flotante (32 bits, IEEE 754)

Las siguientes operaciones permiten cambiar la secuencia de bytes de la palabra baja del ACU 1 o de todo el acumulador:

- TAW Cambiar el orden de los bytes en el ACU 1-L (16 bits)
- TAD Invertir el orden de los bytes en el ACU 1 (32 bits)

Para convertir un número en coma flotante de 32 bits IEEE 754 en un entero de 32 bits (entero doble) se utilizan las operaciones descritas a continuación. Las operaciones difieren en el método de redondeo.

- RND Redondear un número en coma flotante a entero
- TRUNC Truncar
- RND+ Redondear un número real al próximo entero superior
- RND- Redondear un número real al próximo entero inferior

### 3.2 BTI Convertir BCD a entero

#### Formato

BTI

#### Descripción de la operación

**BTI** (Conversión de un número BCD de tres dígitos en un número entero) evalúa el contenido del ACU1-L en formato de número decimal codificado en binario (BCD) de tres dígitos y convierte ese número en un entero (de 16 bits). El resultado se almacena en el ACU1-L. El ACU1-H y el ACU 2 no se alteran.

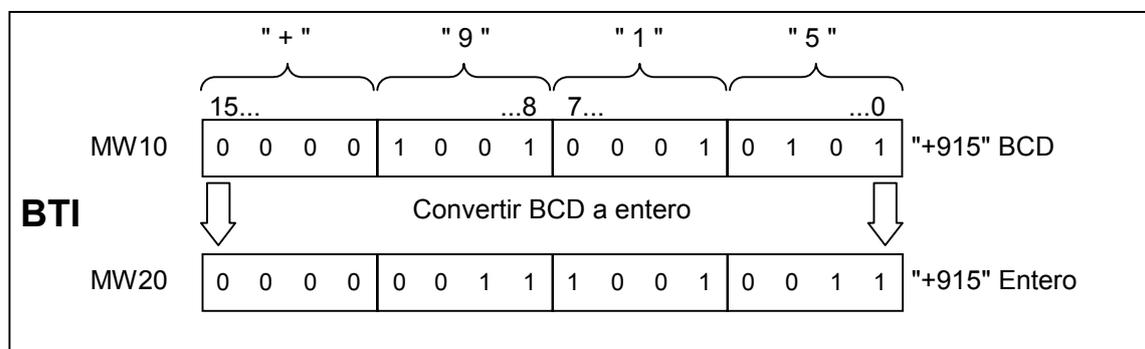
**Número BCD en el ACU1-L:** valores admisibles de "-999" a "+999". Los bits 0 a 11 indican el valor del número BCD, y el bit 15 el signo del mismo (0 = positivo, 1 = negativo). Los bits 12 a 14 no se utilizan al efectuar la conversión. Si un dígito (una tetrada de 4 bits en el formato BCD) del número BCD está dentro del margen inválido de 10 a 15 se producirá un error BCDF al intentar efectuar la conversión. En este caso, el autómata programable cambia generalmente al estado operativo STOP. No obstante, utilizando el OB121 se puede programar una reacción diferente a este error síncrono.

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L MW10	//Cargar el número BCD en el ACU1-L.
BTI	//Convertir el número BCD en número entero, almacenar el resultado en el ACU1-L.
T MW20	//Transferir el resultado (entero de 16 bits) a MW20.



### 3.3 ITB Convertir entero en BCD

#### Formato

ITB

#### Descripción de la operación

**ITB** (Conversión de un entero de 16 bits en número de formato BCD) evalúa el contenido del ACU1- L como número entero (16 bits) y convierte ese número en el correspondiente número decimal codificado en binario de tres dígitos (BCD). El resultado se almacena en el ACU1-L. Los bits 0 a 11 indican el valor del número BCD; los bits 12 a 15 indican el signo del mismo (0000 = positivo, 1111 = negativo). ACU1-H y ACU 2 no se alteran.

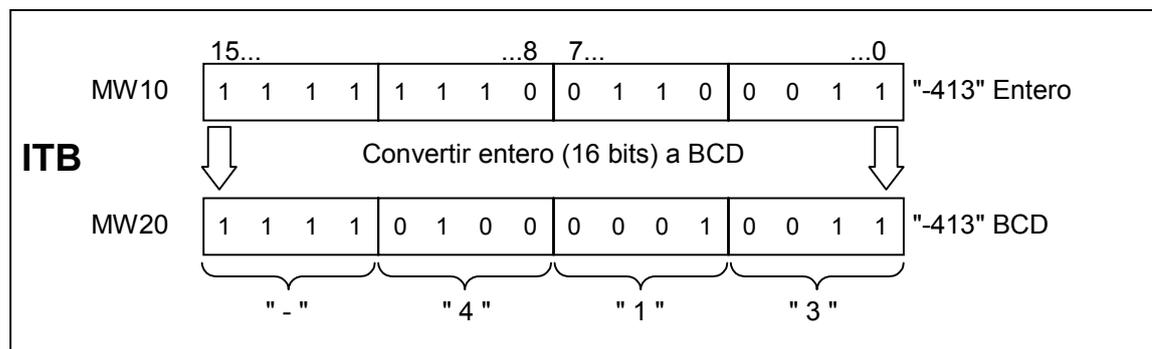
El número BCD puede tener un valor dentro del margen entre "-999" y "+999". Si el número se encuentra fuera del margen admisible, los bits OV y OS de la palabra de estado se ponen a "1".

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L MW10	//Cargar el entero en el ACU1-L.
ITB	//Convertir el entero (16 bits) en número BCD, almacenar el resultado en //el ACU1-L.
T MW20	//Transferir el resultado (número BCD) a MW20.



### 3.4 BTD Convertir número BCD a entero doble

#### Formato

BTB

#### Descripción de la operación

**BTB** (Conversión de un número BCD de siete dígitos en un número entero doble) evalúa el contenido del ACU 1 en formato de número decimal codificado en binario (BCD) de siete dígitos y convierte ese número en un entero doble (32 bits). El resultado se almacena en el ACU 1. El ACU 2 no se altera.

**Número BCD en el ACU 1:** valores admisibles de "-9,999,999" a "+9,999,999". Los bits 0 a 27 indican el valor del número BCD, y el bit 31 indica el signo del mismo (0 = positivo, 1 = negativo). Los bits 28 a 30 no se utilizan en la conversión.

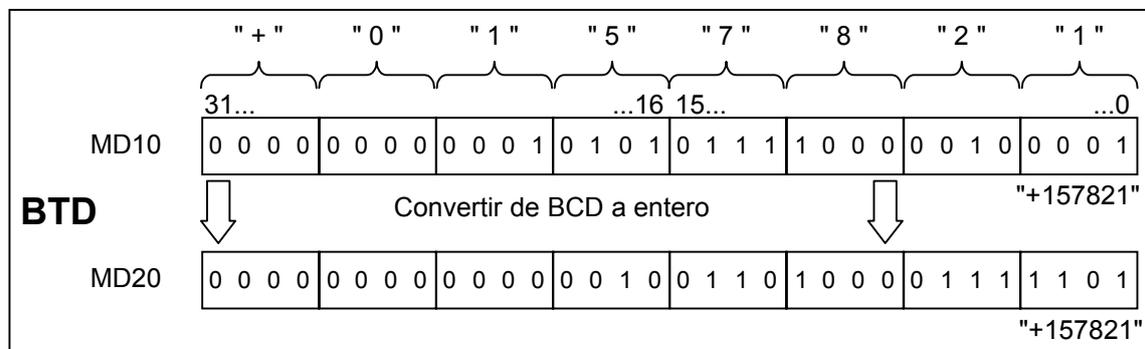
Si una cifra decimal (una tetrada de 4 bits en la representación BCD) está dentro del margen inválido de 10 a 15 se producirá un error BCDF al intentar efectuar la conversión. En este caso, el autómata programable cambia generalmente al estado operativo STOP. No obstante, utilizando el OB121 se puede programar una reacción diferente a este error síncrono.

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L MD10	//Cargar el número BCD en el ACU 1.
BTD	//Convertir el número BCD en número entero, almacenar el resultado en el ACU 1.
T MD20	//Transferir el resultado (en formato de entero doble) a MD20.



## 3.5 ITD Convertir entero en entero doble

### Formato

ITD

### Descripción de la operación

ITD (Convertir un entero en un entero doble) evalúa el contenido del ACU1-L como entero de 16 bits y convierte este número en entero de 32 bits. El resultado se almacena en el ACU 1. El ACU 2 no se altera.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L MW12	//Cargar el entero (16 bits) en el ACU 1-L.
ITD	//Convertir el entero de 16 bits en un entero de 32 bits, almacenar el //resultado en el ACU 1.
T MD20	//Transferir el resultado (entero doble) a MD20.

### Ejemplo MW12 = "-10" (entero de 16 bits)

Contenido	ACU1-H				ACU1-L			
Bit	31...	..	..	... 16	15...	..	..	... 0
antes de ejecutar ITD	XXXX	XXXX	XXXX	XXXX	1111	1111	1111	0110
después de ejecutar ITD	1111	1111	1111	1111	1111	1111	1111	0110
	(X = 0 o 1, los bits no son necesarios para la conversión)							

### 3.6 DTB Convertir entero doble en BCD

#### Formato

DTB

#### Descripción de la operación

**DTB** (Conversión de un entero doble en el correspondiente número con formato BCD) evalúa el contenido del ACU 1 como entero (de 32 bits) y convierte ese número en el correspondiente decimal codificado en binario de siete dígitos. El resultado se almacena en el ACU 1. Los bits 0 a 27 indican el valor del número BCD. Los bits 28 a 31 indican el signo del número BCD (0000 = positivo, 1111 = negativo). El ACU 2 no se altera.

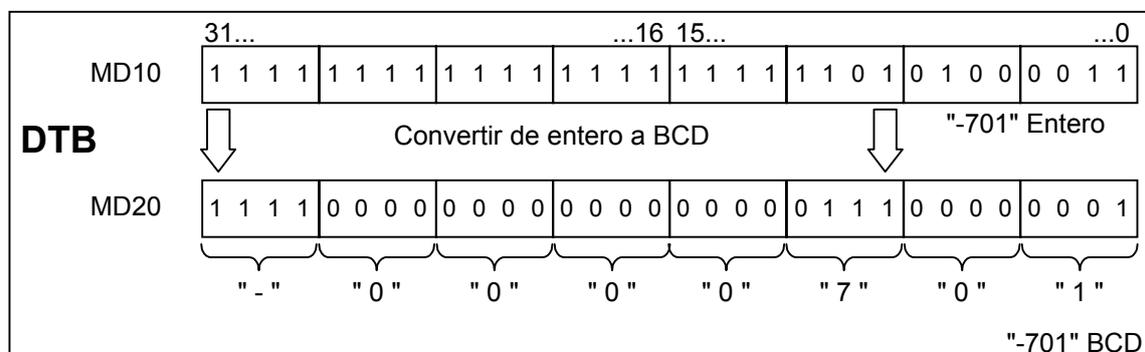
El número BCD puede estar dentro del margen de valores entre "-9,999,999" y "+9,999,999". Si el número se encuentra fuera del margen admisible los bits OV y OS de la palabra de estado se ponen a "1".

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L MD10	//Cargar el entero (32 bits) en el ACU 1.
DTB	//Convertir el entero (32 bits) en un número BCD, almacenar el resultado //en el ACU 1.
T MD20	//Transferir el resultado (número BCD) a MD20.



### 3.7 DTR Convertir entero doble en número en coma flotante (32 bits, IEEE 754)

#### Formato

DTR

#### Descripción de la operación

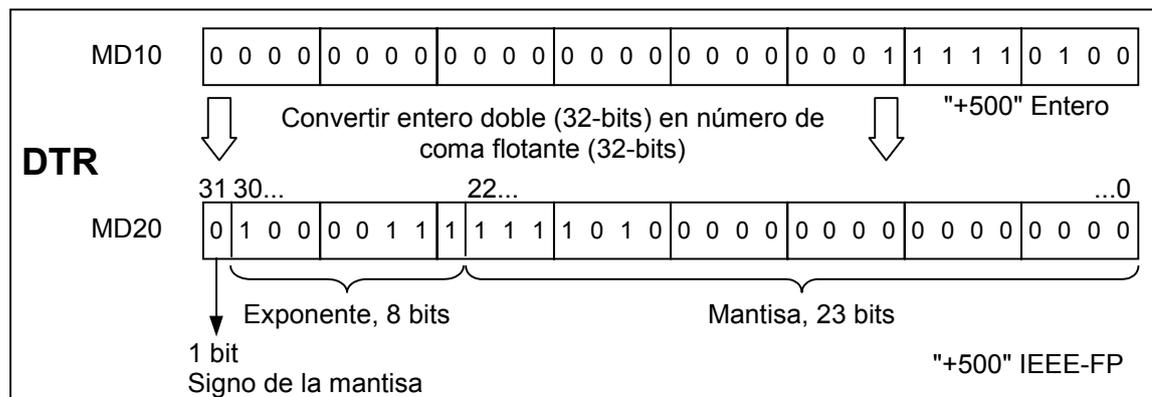
**DTR** (Convertir entero de 32 bits en un número en coma flotante de 32 bits, IEEE 754) evalúa el contenido del ACU 1 como entero (de 32 bits) y convierte ese número en el correspondiente número en coma flotante (32 bits, IEEE 754). Si es necesario, la operación redondea el resultado (el grado de exactitud de un entero de 32 bits es mayor que el de un número en coma flotante de 32 bits, IEEE 754). El resultado se almacena en el ACU 1.

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L MD10	//Cargar el entero (32 bits) en el ACU 1.
DTR	//Convertir el entero (32 bits) en un número en coma flotante (32 bits, IEEE- FP), almacenar el resultado en el ACU 1.
T MD20	//Transferir el resultado (número BCD) a MD20.



## 3.8 INVI Complemento a uno de un entero

### Formato

INVI

### Descripción de la operación

**INVI** (Complemento a uno de entero de 16 bits) calcula el complemento a uno de un valor de 16 bits en el ACU 1-L; al realizar esta operación se invierten todos los bits, es decir, los ceros se sustituyen por unos, y viceversa. El resultado se almacena en el ACU1-L.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L EW8	//Cargar el valor en el ACU1-L.
INVI	//Formar el complemento a uno (16 bits).
T MW10	//Transferir el resultado a MW10.

Contenido	ACU1-L			
Bit	15 ...	..	..	... 0
antes de ejecutar <b>INVI</b>	0110	0011	1010	1110
después de ejecutar <b>INVI</b>	1001	1100	0101	0001

### 3.9 INVD Complemento a uno de un entero doble

#### Formato

**INVD**

#### Descripción de la operación

**INVD** (Complemento a 1 de un entero doble) calcula el complemento a uno de un valor de 32 bits en el ACU 1; al realizar esta operación se invierten todos los bits, es decir, los ceros se sustituyen por unos, y viceversa. El resultado se almacena en el ACU 1.

#### Palabra de estado

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

#### Ejemplo

AWL		Explicación
L	ED8	//Cargar el valor en el ACU 1.
INVD		//Formar el complemento a uno (32 bits).
T	MD10	//Transferir el resultado a MD10.

Contenido	ACU1-H				ACU1-L			
	31 ...	..	..	... 16	15 ...	..	..	... 0
antes de ejecutar <b>INVD</b>	0110	1111	1000	1100	0110	0011	1010	1110
después de ejecutar <b>INVD</b>	1001	0000	0111	0011	1001	1100	0101	0001

### 3.10 NEGI Complemento a dos de un entero

#### Formato

NEGI

#### Descripción de la operación

NEGI (Complemento a dos de un entero) calcula el complemento a dos de un valor de 16 bits en el ACU1-L; al realizar esta operación se invierten todos los bits, es decir, los ceros se sustituyen por unos, y viceversa. Seguidamente se suma un "1". El resultado se almacena en el ACU1-L. La operación Complemento a dos de un entero equivale a una multiplicación por "-1". Una vez ejecutada la operación se activan los bits A1, A0, OS y OV de la palabra de estado.

#### Palabra de estado

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

Calcular los bits de la palabra de estado	A1	A0	OV	OS
Resultado = 0	0	0	0	-
-32768 <= Resultado <= -1	0	1	0	-
32767 >= Resultado >= 1	1	0	0	-
Resultado = 2768	0	1	1	1

#### Ejemplo

AWL	Explicación
L EW8	//Cargar el valor en el ACU1-L.
NEGI	//Formar el complemento a dos (16 bits).
T MW10	//Transferir el resultado a MW10.

Contenido	ACU1-L			
Bit	15 ...	..	..	... 0
antes de ejecutar NEGI	0101	1101	0011	1000
después de ejecutar NEGI	1010	0010	1100	1000

### 3.11 NEGD Complemento a dos de un entero doble

#### Formato

**NEGD**

#### Descripción de la operación

**NEGD** (Complemento a dos de un entero doble) calcula el complemento a dos de un valor de 32 bits en el ACU 1; al realizar esta operación se invierten todos los bits, es decir, los ceros se sustituyen por unos, y viceversa. Seguidamente se suma un "1". El resultado se almacena en el ACU 1. La operación Complemento a dos de un entero doble equivale a una multiplicación por "-1". Una vez ejecutada la operación se activan los bits A1, A0, OS y OV de la palabra de estado.

#### Palabra de estado

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

Calcular los bits de la palabra de estado	A1	A0	OV	OS
Resultado = 0	0	0	0	-
-2.147.483.647 <= Resultado <= -1	0	1	0	-
2.147.483.647 >= Resultado >= 1	1	0	0	-
Resultado = -2 147 483 648	0	1	1	1

#### Ejemplo

AWL	Explicación
L ED8	//Cargar el valor en el ACU 1.
NEGD	//Formar el complemento a dos (32 bits).
T MD10	//Transferir el resultado a MD10.

Contenido	ACU1-H				ACU1-L			
Bit	31 ...	..	..	... 16	15 ...	..	..	... 0
antes de ejecutar <b>NEGD</b>	0101	1111	0110	0100	0101	1101	0011	1000
después de ejecutar <b>NEGD</b>	1010	0000	1001	1011	1010	0010	1100	1000

### 3.12 NEGR Invertir un número en coma flotante (32 bits, IEEE 754)

#### Formato

NEGR

#### Descripción de la operación

**NEGR** (Invertir un número en coma flotante de 32 bits, IEEE 754) invierte el número en coma flotante (32 bits, IEEE 754) en el ACU 1. La operación invierte el estado del bit 31 en el ACU 1 (signo de la mantisa). El resultado se almacena en el ACU 1.

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L ED8	//Cargar el valor en el ACU 1 (ejemplo: ED8 = 1.5E+02).
NEGR	//Invertir número en coma flotante (32 bits, IEEE 754), almacenar el //resultado //en el ACU 1.
T MD10	//Transferir el resultado a MD10 (ejemplo: resultado = - 1.5E+02).

### 3.13 TAW Cambiar el orden de los bytes en el ACU 1-L (16 bits)

#### Formato

**TAW**

#### Descripción de la operación

**TAW** invierte el orden de los bytes en el ACU1-L. El resultado se almacena en el ACU1-L. El ACU1-H y el ACU 2 no se alteran.

El contenido de ACU1-L-H se desplaza a: ACU1-L-L.

El contenido de ACU1-L-L se desplaza a: ACU1-L-H.

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L MW10	//Cargar el valor de MW10 en el ACU 1.
TAW	//Invertir el orden de los bytes en el ACU1-L.
T MW20	//Transferir el resultado a MW20.

Contenido	ACU 1-H-H	ACU 1-H-L	ACU 1-L-H	ACU 1-L-L
antes de ejecutar la operación <b>TAW</b>	valor A	valor B	valor C	valor D
después de ejecutar la operación <b>TAW</b>	valor A	valor B	valor D	valor C

### 3.14 TAD Invertir el orden de los bytes en el ACU 1 (32 bits)

#### Formato

TAD

#### Descripción de la operación

**TAD** invierte el orden de los bytes en el ACU 1. El resultado se almacena en el ACU 1. El ACU 2 no se altera.

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L MD10	//Cargar el valor de MD10 en el ACU 1.
TAD	//Invertir el orden de los bytes en el ACU 1.
T MD20	//Transferir el resultado a MD20.

Contenido	ACU 1-H-H	ACU 1-H-L	ACU 1-L-H	ACU 1-L-L
antes de ejecutar la operación <b>TAD</b>	valor A	valor B	valor C	valor D
después de ejecutar la operación <b>TAD</b>	valor D	valor C	valor B	valor A

### 3.15 RND Redondear un número en coma flotante a entero

#### Formato

RND

#### Descripción de la operación

**RND** (Conversión de un número en coma flotante (32 bits, IEEE 754) en un entero de 32 bits) evalúa el contenido del ACU 1 como número en coma flotante (32 bits, IEEE 754); la operación convierte a continuación el número en coma flotante (32 bits, IEEE 754) en el correspondiente número entero (32 bits), y redondea el resultado al número entero más próximo. Si la fracción del número convertido se encuentra exactamente en medio de un resultado par y de un resultado impar, la operación redondea al resultado par. Si el número está fuera del margen admisible, los bits OV y OS de la palabra de estado se ponen a "1".

Si se produce un error (una NaN o un número en coma flotante que no se pueda representar como entero de 32 bits), no se ejecuta la conversión y se señala un desbordamiento.

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L MD10	//Cargar el número en coma flotante (32 bits, IEEE 754) en el ACU1-L.
RND	//Convertir el número en coma flotante (32 bits, IEEE 754) en entero de //32 bits y redondear el resultado. Almacenar el resultado en el ACU1.
T MD20	//Transferir el resultado (entero de 32 bits) a MD20.

Valor antes de la conversión		Valor después de la conversión
MD10 = "100.5"	=> RND =>	MD20 = "+100"
MD10 = "-100.5"	=> RND =>	MD20 = "-100"

## 3.16 TRUNC Truncar

### Formato

TRUNC

### Descripción de la operación

**TRUNC** (Conversión de un número en coma flotante (32 bits, IEEE 754) en un entero de 32 bits) evalúa el contenido del ACU 1 como número en coma flotante (32 bits, IEEE 754); la operación convierte a continuación el número en coma flotante (32 bits, IEEE 754) en entero (32 bits). El resultado es la parte entera del número en coma flotante convertido (modo de redondeo IEEE 'Round to Zero'). Si el número está fuera del margen admisible, los bits OV y OS de la palabra de estado se ponen a "1". El resultado se almacena en el ACU 1.

Si se produce un error (una NaN o un número en coma flotante que no se pueda representar como entero de 32 bits), no se ejecuta la conversión y se señala un desbordamiento.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L MD10	//Cargar el número en coma flotante (32 bits, IEEE 754) en el ACU1-L.
TRUNC	//Convertir el número en coma flotante (32 bits, IEEE 754) en entero de //32 bits y redondear el resultado. Almacenar el resultado en el ACU1.
T MD20	//Transferir el resultado (entero de 32 bits) a MD20.

Valor antes de la conversión		Valor después de la conversión
MD10 = "100.5"	=> TRUNC =>	MD20 = "+100"
MD10 = "-100.5"	=> TRUNC =>	MD20 = "-101"

### 3.17 RND+ Redondear un número real al próximo entero superior

#### Formato

RND+

#### Descripción de la operación

**RND+** (Conversión de un número en coma flotante (32 bits, IEEE 754) en un entero de 32 bits) evalúa el contenido del ACU 1 como número en coma flotante (32 bits, IEEE 754); la operación convierte a continuación el número en coma flotante (32 bits, IEEE 754) en entero (32 bits) y redondea el resultado al próximo número entero que sea mayor o igual al número en coma flotante convertido (modo de redondeo IEEE "Round to +infinity"). Si el número está fuera del margen admisible, los bits OV y OS de la palabra de estado se ponen a "1". El resultado se almacena en el ACU 1.

Si se produce un error (una NaN o un número en coma flotante que no se pueda representar como entero de 32 bits), no se ejecuta la conversión y se señala un desbordamiento.

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L MD10	//Cargar el número en coma flotante (32 bits, IEEE 754) en el ACU1-L.
RND+	//Convertir el número en coma flotante (32 bits, IEEE 754) en entero de //32 bits y redondear el resultado. Almacenar el resultado en el ACU1.
T MD20	//Transferir el resultado (entero de 32 bits) a MD20.

Valor antes de la conversión		Valor después de la conversión
MD10 = "100.5"	=> RND+ =>	MD20 = "+101"
MD10 = "-100.5"	=> RND + =>	MD20 = "-100"

### 3.18 RND- Redondear un número real al próximo entero inferior

#### Formato

RND-

#### Descripción de la operación

**RND-** (Conversión de un número en coma flotante (32 bits, IEEE 754) en un entero de 32 bits) evalúa el contenido del ACU 1 como número en coma flotante (32 bits, IEEE 754); la operación convierte a continuación el número en coma flotante (32 bits, IEEE 754) en entero (32 bits) y redondea el resultado al próximo número entero que sea menor o igual al número en coma flotante convertido (modo de redondeo IEEE "Round to +infinity"). Si el número está fuera del margen admisible, los bits OV y OS de la palabra de estado se ponen a "1". El resultado se almacena en el ACU 1.

Si se produce un error (una NaN o un número en coma flotante que no se pueda representar como entero de 32 bits), no se ejecuta la conversión y se señala un desbordamiento.

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
L MD10	//Cargar el número en coma flotante (32 bits, IEEE 754) en el ACU1-L.
RND-	//Convertir el número en coma flotante (32 bits, IEEE 754) en entero (32 bits) y redondear el resultado. Almacenar el resultado en el ACU1.
T MD20	//Transferir el resultado (entero de 32 bits) a MD20.

Valor antes de la conversión		Valor después de la conversión
MD10 = "100.5"	=> RND- =>	MD20 = "+100"
MD10 = "-100.5"	=> RND- =>	MD20 = "-101"

## 4 Operaciones de contaje

### 4.1 Lista de operaciones de contaje

#### Descripción

Un contador es un elemento funcional del software de programación STEP 7. Los contadores tienen reservada un área de memoria propia en la CPU. Dicha área de memoria reserva una palabra de 16 bits para cada contador. La programación con AWL asiste un máximo de 256 contadores. En los datos técnicos de la CPU encontrará la cantidad de contadores de que puede disponer. Las operaciones de contaje son las únicas funciones que tienen acceso al área de memoria reservada para contadores.

Se dispone de las operaciones de contaje siguientes:

- FR Habilitar contador
- L Cargar valor actual del contador en ACU 1 en forma de entero
- LC Cargar valor actual del contador en ACU 1 como número BCD
- R Desactivar contador
- S Poner contador al valor inicial
- ZV Incrementar contador
- ZR Decrementar contador

## 4.2 FR Habilitar contador

### Formato

FR <contador>

Operando	Tipo de datos	Area de memoria	Descripción
<contador>	COUNTER	Z	Contador; el área varía según la CPU utilizada.

### Descripción de la operación

Si el RLO cambia de "0" a "1", la operación **FR <contador>** borra la marca de flancos que el contador direccionado pone en incrementar / decrementar contador. No es necesario habilitar un contador para activarlo ni para ejecutar la función normal de contaje. Es decir, aunque el RLO sea constantemente 1 en las instrucciones **Poner contador al valor inicial**, **Incrementar contador** o **Decrementar contador**, después de la habilitación se vuelven a ejecutar estas operaciones.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
U E 2.0	//Consultar el estado de señal en la entrada E 2.0.
FR Z3	//Habilitar el contador Z3 si el RLO cambia de "0" a "1".

## 4.3 L Cargar valor actual del contador en ACU 1 en forma de entero

### Formato

L <contador>

Operando	Tipo de datos	Area de memoria	Descripción
<contador>	COUNTER	Z	Contador; el área varía según la CPU utilizada.

### Descripción de la operación

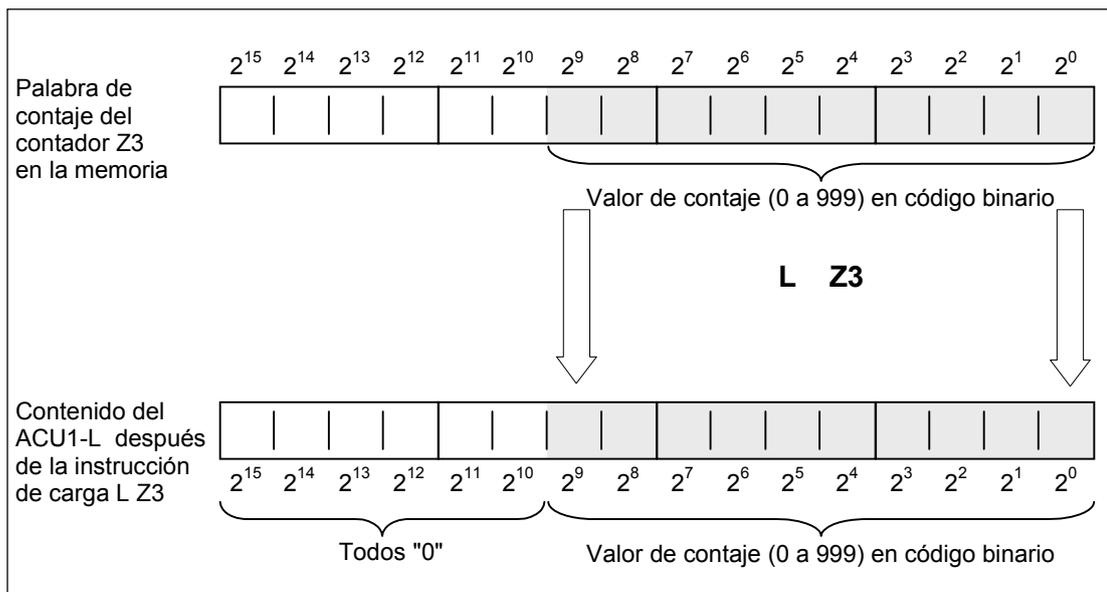
L <contador> carga el valor de contaje del contador direccionado en forma de número entero en ACU1-L, después de que se haya almacenado el contenido del ACU 1 en el ACU 2.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L Z3	//Cargar el valor de contaje del contador Z3 en formato binario en el //ACU1-L.



## 4.4 LC Cargar valor actual del contador en ACU 1 como número BCD

### Formato

LC <contador>

Operando	Tipo de datos	Area de memoria	Descripción
<contador>	COUNTER	Z	Contador; el área varía según la CPU utilizada.

### Descripción de la operación

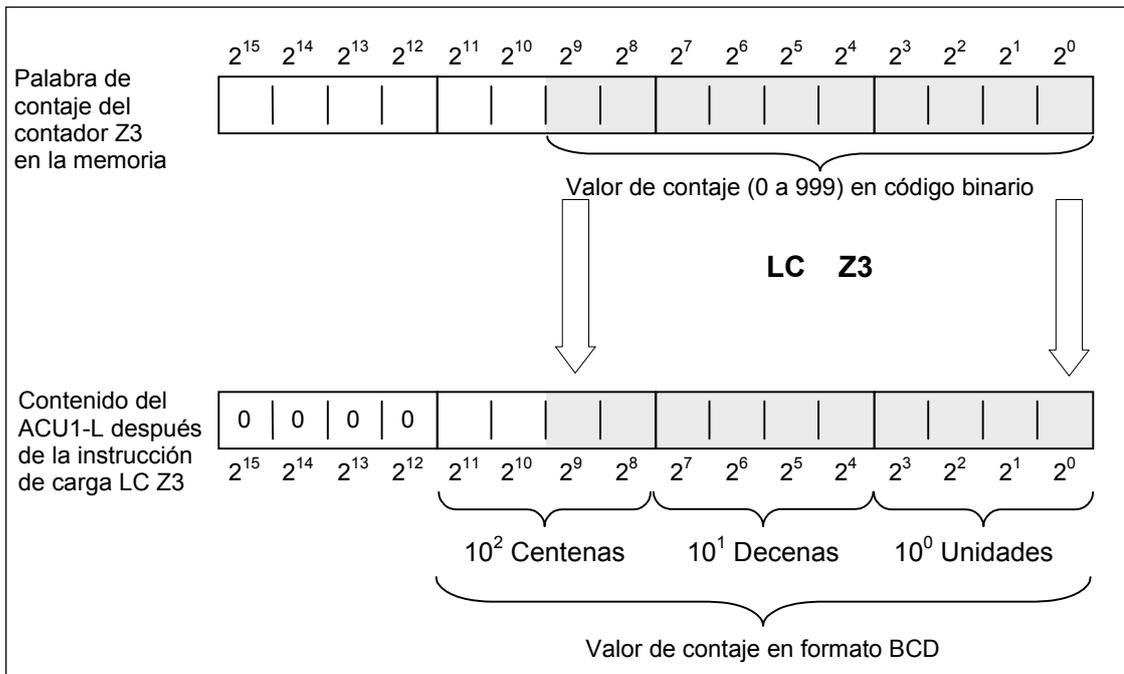
LC <contador> carga el valor de contaje del contador direccionado en formato BCD en el ACU 1, después de que se haya almacenado el contenido del ACU 1 en el ACU 2.

### Palabra de estado

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

## Ejemplo

AWL	Explicación
LC Z3	//Cargar el valor de contaje del contador Z3 en formato BCD en ACU1-L.



## 4.5 R Desactivar contador

### Formato

R <contador>

Operando	Tipo de datos	Area de memoria	Descripción
<contador>	COUNTER	Z	Contador a desactivar; el área varía según la CPU utilizada.

### Descripción de la operación

R <contador> carga el valor de contaje "0" en el contador direccionado, si el RLO es 1.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
U E 2.3	//Consultar el estado de señal en la entrada E 2.3.
R Z3	//Poner el contador Z3 al valor "0" si el RLO cambia de "0" a "1".

## 4.6 S Poner contador al valor inicial

### Formato

S <contador>

Operando	Tipo de datos	Area de memoria	Descripción
<contador>	COUNTER	Z	Contador a predeterminar; el área varía según la CPU utilizada.

### Descripción de la operación

S <contador> carga el valor de contaje del ACU1-L en el contador direccionado, si el RLO cambia de "0" a "1". El valor de contaje en el ACU 1 tiene que estar en formato BCD y tener un valor entre "0" y 999.

### Palabra de estado

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

### Ejemplo

AWL		Explicación
U	E 2.3	//Consultar el estado de señal de la entrada E 2.3.
L	C#3	//Cargar el valor de contaje 3 en el ACU1-L.
S	Z1	//Poner el contador Z1 al valor de contaje si el RLO cambia de "0" a "1".

## 4.7 ZV Incrementar contador

### Formato

ZV <contador>

Operando	Tipo de datos	Area de memoria	Descripción
<contador>	COUNTER	Z	Contador; el área varía según la CPU utilizada.

### Descripción de la operación

**ZV <contador>** incrementa en "1" el valor de contaje del contador direccionado si el RLO cambia de "0" a "1" y el valor de contaje es menor que "999". Cuando el valor de contaje llegue al valor límite superior "999" dejará de incrementarse; los cambios posteriores del RLO no tendrán efecto alguno. El bit de desbordamiento (OV) no se activa.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
U E 2.1	//Consultar el estado de señal en la entrada E 2.1.
ZV Z3	//Incrementar en 1 el valor del contador Z3 si el RLO cambia de "0" a "1".

## 4.8 ZR Decrementar contador

### Formato

ZR <contador>

Operando	Tipo de datos	Area de memoria	Descripción
<contador>	COUNTER	Z	Contador; el área varía según la CPU utilizada.

### Descripción de la operación

**ZR <contador>** decrementa en "1" el valor de contaje del contador direccionado si el RLO cambia de "0" a "1" y el valor de contaje es mayor que "0". Cuando el valor de contaje llega al valor límite inferior "0" deja de decrementarse; los cambios posteriores del RLO no tendrán efecto alguno, ya que el contador no opera con valores negativos.

### Palabra de estado

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

### Ejemplo

AWL	Explicación	
L	C#14	//Valor preajustado del contador.
U	E 0.1	//Contador preajustado tras detectar un flanco positivo en la entrada //E 0.1.
S	Z1	//Cargar el contador Z1 con el preajuste, si éste está habilitado.
U	E 0.0	//Decrementar en "1" cada vez que se dé un flanco positivo en E 0.0.
ZR	Z1	//Decrementar en "1" el contador Z1 si el RLO cambia de "0" a "1" //(en función de la entrada E 0.0).
UN	Z1	//Detección de cero mediante el bit Z1.
=	A 0.0	//A 0.0 es 1 si el valor del contador Z1 es "0".



## 5 Operaciones con los bloques de datos

### 5.1 Lista de operaciones con bloques

#### Descripción

La operación AUF (Abrir bloque de datos) sirve para abrir un bloque de datos global o un bloque de datos de instancia. En el programa pueden estar abiertos simultáneamente un bloque de datos global y un bloque de instancia.

Se dispone de las operaciones con bloques siguientes:

- AUF Abrir bloque de datos
- TDB Intercambiar bloque de datos global y bloque de datos de instancia
- L DBLG Cargar la longitud del DB global en el ACU 1
- L DBNO Cargar número del bloque de datos global en ACU 1
- L DILG Cargar longitud del bloque de datos de instancia en ACU 1
- L DINO Cargar número del bloque de datos de instancia en ACU 1

## 5.2 AUF Abrir bloque de datos

### Formato

AUF <bloque de datos>

Operando	Tipo del bloque de datos	Dirección fuente
<bloque de datos>	DB, DI	De 1 a 65.535

### Descripción de la operación

**AUF <bloque de datos>** abre un bloque de datos como bloque de datos global o como bloque de datos de instancia. Pueden estar abiertos simultáneamente un bloque de datos global y un bloque de datos de instancia, respectivamente.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
AUF DB10	//Abrir el bloque de datos DB10 como bloque de datos global.
L DBW35	//Cargar en ACU1-L la palabra de datos DBW35 del bloque de datos abierto.
T MW22	//Transferir el contenido del ACU1-L a la palabra de marcas MW22.
AUF DI20	//Abrir el bloque de datos DB20 como bloque de datos de instancia.
L DIB12	//Cargar en el ACU1-L-L el byte de datos DIB12 del bloque de datos de //instancia abierto.
T DBB37	//Transferir el contenido del ACU1-L-L al byte de datos DBB37 del bloque //de datos global abierto.

## 5.3 TDB Intercambiar bloque de datos global y bloque de datos de instancia

### Formato

TDB

### Descripción de la operación

**TDB** intercambia los registros de los bloques de datos. Al efectuarse la permutación, un bloque de datos global en un bloque de datos de instancia, y viceversa.

### Palabra de estado

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

## 5.4 L DBLG Cargar la longitud del DB global en el ACU 1

### Formato

L DBLG

### Descripción de la operación

**L DBLG** (Cargar la longitud del bloque de datos global) carga la longitud del bloque de datos global en el ACU 1, después de que se haya almacenado el contenido del ACU 1 en el ACU 2.

### Palabra de estado

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

### Ejemplo

AWL	Explicación	
AUF DB10	//Abrir el bloque de datos DB10 como bloque de datos global.	
L DBLG	//Cargar la longitud del bloque de datos global (longitud de DB10).	
L MD10	//Valor para comparar si el bloque de datos es suficientemente largo.	
<D		
SPB ERRO	//Saltar a meta ERRO si la longitud es menor que el valor en MD10.	

## 5.5 L DBNO Cargar número del bloque de datos global en ACU 1

### Formato

L DBNO

### Descripción de la operación

L DBNO (Cargar el número del bloque de datos global) carga en el ACU1 el número del bloque de datos global que está abierto, después de que se haya almacenado el contenido del ACU 1 en el ACU 2.

### Palabra de estado

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

## 5.6 L DILG Cargar longitud del bloque de datos de instancia en ACU 1

### Formato

L DILG

### Descripción de la operación

L DILG (Cargar la longitud del bloque de datos de instancia) carga la longitud del bloque de datos de instancia en el ACU 1, después de que se haya almacenado el contenido del ACU 1 en el ACU 2.

### Palabra de estado

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

### Ejemplo

AWL	Explicación	
AUF DI20	//Abrir el bloque de datos DB20 como bloque de datos de instancia.	
L DILG	//Cargar la longitud del bloque de datos de instancia (longitud de DB20).	
L MW10	//Valor para comparar si el bloque de datos es suficientemente largo.	
<I		
SPB ERRO	//Saltar a meta ERRO si la longitud es menor que el valor en MW10.	

## 5.7 L DINO Cargar número del bloque de datos de instancia en ACU 1

### Formato

L DINO

### Descripción de la operación

**L DINO** (Cargar número del bloque de datos de instancia) carga en el ACU 1 el número del bloque de datos de instancia que está abierto, después de que se haya almacenado el contenido del ACU 1 en el ACU 2.

### Palabra de estado

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



## 6 Operaciones de salto

### 6.1 Lista de operaciones de salto

#### Descripción

Las operaciones de salto sirven para controlar el desarrollo del programa. Estas operaciones permiten al programa interrumpir su desarrollo lineal para continuar el procesamiento en un punto diferente del programa. La operación LOOP (bucle) sirve para llamar varias veces un segmento del programa.

El operando de una operación de salto o LOOP es una meta. Esta meta puede tener hasta un máximo de 4 caracteres, el primero de los cuales tiene que ser una letra. La meta termina con un signo de dos puntos ":", y da inicio a la instrucción de la línea (p.ej. SEG3).

---

#### Nota

Al escribir programas para las CPUs S7- 300, atender a que en operaciones de salto el destino del salto esté siempre al **comienzo** de una cadena de combinaciones lógicas (no necesario con 318-2). El destino del salto no deberá encontrarse dentro de una cadena de combinaciones.

---

Las siguientes operaciones de salto se utilizan para interrumpir el desarrollo normal del programa sin condiciones:

- SPA Salto incondicionado
- SPL Saltar utilizando una lista de metas

Las siguientes operaciones de salto interrumpen el desarrollo del programa dependiendo del resultado lógico (RLO) dado en la instrucción anterior:

- SPB Saltar si RLO = 1
- SPBN Saltar si RLO = 0
- SPBB Saltar si RLO = 1 y salvaguardar RLO en RB
- SPBNB Saltar si RLO = 0 y salvar RLO en RB

Las siguientes operaciones de salto interrumpen el desarrollo del programa en función del estado de señal de un determinado bit de la palabra estado:

- SPA Salto incondicionado
- SPBI Saltar si RB = 1
- SPBIN Saltar si RB = 0
- SPO Saltar si OV = 1
- SPS Saltar si OS = 1

Las siguientes operaciones de salto interrumpen el desarrollo del programa en función del resultado de una operación anterior:

- SPZ Saltar si el resultado = 0
- SPN Saltar si resultado  $\neq$  0
- SPP Saltar si el resultado > 0
- SPM Saltar si resultado < 0
- SPPZ Saltar si el resultado  $\geq$  0
- SPMZ Saltar si el resultado  $\leq$  0
- SPU Saltar si el resultado no es válido

## 6.2 SPA Salto incondicionado

### Formato

SPA <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

**SPA <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado, independientemente de cuál sea el contenido de la palabra de estado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

### Ejemplo

AWL	Explicación	
U	E 1.0	
U	E 1.2	
SPB	DELE	//Si el RLO es 1 saltar a la meta DELE.
L	MB10	
INC	1	
T	MB10	
SPA	FORW	//Salto incondicionado a la meta FORW.
DELE:	L	0
	T	MB10
FORW:	U	E 2.1 //La ejecución del programa continúa aquí después de haber saltado a la //meta FORW.

## 6.3 SPL Saltar utilizando una lista de metas

### Formato

SPL <meta>

Operando	Descripción
<meta >	Nombre simbólico de la meta del salto

### Descripción de la operación

**SPL <meta>** (Saltar utilizando una lista de metas) permite programar distintos saltos. La lista de metas de salto, que como máximo contiene 255 registros, empieza directamente después de la operación SPL y termina antes de la marca del salto que indica el operando SPL. Cada meta contiene una operación SPA. La cantidad de metas (entre 0 y 255) se toma del ACU1-L-L.

Mientras el contenido del ACU sea inferior al número de metas entre la instrucción SPL y la meta del salto, la operación SPL saltará a una de las operaciones SPA. Si el ACU1-L-L 1 es 0 se salta a la primera operación SPA; si el ACU1-L-L es 1, se salta a la segunda operación SPA, y así sucesivamente. Si el número de destinos es demasiado grande, la operación SPL salta a la primera instrucción después de la última operación SPA de la lista de metas.

La lista de metas tiene que estar formada por operaciones SPA que se encuentren antes de la marca de salto que indica la instrucción SPL. Dentro de la lista de metas no se pueden realizar operaciones de otro tipo.

## Palabra de estado

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

## Ejemplo

AWL	Explicación	
L	MB0	//Cargar el número de la meta del salto en el ACU1-L-L.
SPL	LSTX	//Meta del salto si el ACU1-L-L es > 3.
SPA	SEG0	//Meta del salto si ACU1-L-L = 0.
SPA	SEG1	//Meta del salto si ACU1-L-L = 1.
SPA	COMM	//Meta del salto si ACU1-L-L = 2.
SPA	SEG3	//Meta del salto si ACU1-L-L = 3.
LSTX:	SPA	COMM
SEG0:	*	//Instrucción permitida.
	*	
	SPA	COMM
SEG1:	*	//Instrucción permitida.
	*	
	SPA	COMM
SEG3:	*	//Instrucción permitida.
	*	
	SPA	COMM
COMM:	*	
	*	

## 6.4 SPB Saltar si RLO = 1

### Formato

SPB <meta>

Operando	Descripción
<meta >	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el RLO es 1, la operación **SPB <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

Si el RLO es 0 no se ejecuta el salto. El RLO se pone a "1" y la ejecución del programa continúa con la instrucción siguiente.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
U E 1.0	
U E 1.2	
SPB JOVR	//Saltar a la meta JOVR si RLO = 1
L EW8	//La ejecución del programa continúa aquí en caso de que no se ejecute //el salto.
T MW22	
JOVR: U E 2.1	//La ejecución del programa continúa aquí después de haber saltado a la //meta JOVR.

## 6.5 SPBN Saltar si RLO = 0

### Formato

SPBN <meta>

Operando	Descripción
<meta >	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el RLO es 0, la operación **SPBN <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

Si el RLO es 1 no se ejecuta el salto. La ejecución del programa continúa con la instrucción siguiente.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
U E 1.0	
U E 1.2	
SPBN JOVR	//Saltar, si el RLO = 0, a la meta JOVR.
L EW8	//La ejecución del programa continúa aquí en caso de no ejecutarse el //salto.
T MW22	
JOVR: U E 2.1	//La ejecución del programa continúa aquí después de haber saltado a la //meta JOVR.

## 6.6 SPBB Saltar si RLO = 1 y salvaguardar RLO en RB

### Formato

SPBB <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el RLO es 1, la operación **SPBB <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

Si el RLO es 0 no se ejecuta el salto. El RLO se pone a "1" y la ejecución del programa continúa con la instrucción siguiente.

Independientemente de cuál sea el RLO, al ejecutarse la operación **SPBB <meta>** se copia el RLO en el RB.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
U E 1.0	
U E 1.2	
SPBB JOVR	//Si el RLO = 1, saltar a la meta JOVR. Copiar el contenido del bit RLO //en el bit RB.
L EW8	//La ejecución del programa continúa aquí en caso de que no se ejecute //el salto.
T MW22	
JOVR: U E 2.1	//La ejecución del programa continúa aquí después de haber saltado a la //meta JOVR.

## 6.7 SPBNB Saltar si RLO = 0 y salvar RLO en RB

### Formato

SPBNB <meta>

Operando	Descripción
<meta >	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el RLO es 0, la operación **SPBNB <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

Si el RLO es 1 no se ejecuta la operación; el RLO se pone a "1" y la ejecución del programa continúa con la instrucción siguiente.

Independientemente de cuál sea el RLO, al realizarse la operación **SPBNB <meta >** el RLO se copia automáticamente en el RB.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
U	E 1.0
U	E 1.2
SPBNB	JOVR //Si RLO = 0, saltar a la meta JOVR; copiar el contenido del bit RLO en //el bit RB.
L	EW //La ejecución del programa continúa aquí en caso de no ejecutarse el //salto.
T	MW22
JOVR: U	E 2.1 //La ejecución del programa continúa aquí después de haber saltado a la //meta JOVR.

## 6.8 SPBI Saltar si RB = 1

### Formato

SPBI <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el bit RB de la palabra de estado es 1, la operación **SPBI <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

## 6.9 SPBIN Saltar si RB = 0

### Formato

**SPBIN <meta>**

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el bit RB de la palabra de estado es 0, la operación **SPBIN <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

## 6.10 SPO Saltar si OV = 1

### Formato

SPO <meta>

Operando	Descripción
<meta >	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el bit OV de la palabra de estado es 1, la operación **SPO <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras). Si se trata de una operación aritmética compuesta hay que asegurarse de que después de cada una de las operaciones aritméticas no se produzca ningún desbordamiento, a fin de garantizar que cada resultado parcial quede dentro del margen admisible; de lo contrario habrá que utilizar la operación SPS.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L MW10	
L 3	
*I	//Multiplicar el contenido de MW10 por "3".
SPO OVER	//Saltar si el resultado excede
	//el margen máximo (OV = 1).
T MW10	//La ejecución del programa continúa aquí en caso de no ejecutarse el
	//salto.
U M 4.0	
R M 4.0	
SPA NEXT	
OVER: UN M 4.0	//La ejecución del programa continúa aquí después de haber saltado a la
	//meta OVER.
S M 4.0	
NEXT: NOP 0	//La ejecución del programa continúa aquí después de haber saltado a la
	//meta NEXT.

## 6.11 SPS Saltar si OS = 1

### Formato

SPS <meta>

Operando	Descripción
<meta >	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el bit OS de la palabra de estado es 1, la operación **SPS <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

## Ejemplo

AWL	Explicación	
L	EW10	
L	MW12	
*I		
L	DBW25	
+I		
L	MW14	
-I		
SPS	OVER	//Saltar si se produce desbordamiento en una de las 3 operaciones, //OS = 1, (véase la nota).
T	MW16	//La ejecución del programa continúa aquí en caso de no ejecutarse el //salto.
U	M 4.0	
R	M 4.0	
SPA	NEXT	
OVER:	UN	M 4.0 //La ejecución del programa continúa aquí después de haber saltado a la //meta OVER.
	S	M 4.0
NEXT:	NOP 0	//La ejecución del programa continúa aquí después de haber saltado a la //meta NEXT.

## Nota

En este caso no debe emplearse la operación **SPO**. La operación **SPO** sólo consultaría la operación precedente **-I** en caso de un desbordamiento.

## 6.12 SPZ Saltar si el resultado = 0

### Formato

SPZ <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

Si los bits de la palabra de estado A1 = 0 y A0 = 0, la operación **SPZ <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L MW10	
SRW 1	
SPZ ZERO	//Saltar a la meta ZERO si el bit desplazado es 0.
L MW2	//La ejecución del programa continúa aquí en caso de no ejecutarse el //salto.
INC 1	
T MW2	
SPA NEXT	
ZERO: L MW4	//La ejecución del programa continúa aquí después de haber saltado a la //meta ZERO.
INC 1	
T MW4	
NEXT: NOP 0	//La ejecución del programa continúa aquí después de haber saltado a la //meta NEXT.

## 6.13 SPN Saltar si resultado <> 0

### Formato

SPN <meta>

Operando	Descripción
<meta >	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el resultado indicado por los bits A1 y A0 de la palabra de estado es mayor o menor que cero ( $A1 = 0/A0 = 1$  ó  $A1 = 1/A0 = 0$ ), la operación **SPN <meta>** (Saltar si el resultado <> 0) interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L EW8	
L MW12	
XOW	
SPN NOZE	//Saltar si el contenido del ACU1-L no es cero.
UN M 4.0	//La ejecución del programa continúa aquí en caso de no ejecutarse el //salto.
S M 4.0	
SPA NEXT	
NOZE: UN M 4.1	//La ejecución del programa continúa aquí después de haber saltado a la //meta NOZE.
S M 4.1	
NEXT: NOP 0	//La ejecución del programa continúa aquí después de haber saltado a la //meta NEXT.

## 6.14 SPP Saltar si el resultado > 0

### Formato

SPP <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

Si los bits de la palabra de estado A1 = 1 y A0 = 0, la operación **SPP <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L EW8	
L MW12	
-I	//Restar el contenido de MW12 del contenido de EW8.
SPP POS	//Saltar si el resultado > 0 (es decir, si el contenido de ACU 1 > 0).
UN M 4.0	//La ejecución del programa continúa aquí en caso de no ejecutarse el //salto.
S M 4.0	
SPA NEXT	
POS: UN M 4.1	//La ejecución del programa continúa aquí después de haber saltado a la //meta POS.
S M 4.1	
NEXT: NOP 0	//La ejecución del programa continúa aquí después de haber saltado a la //meta NEXT.

## 6.15 SPM Saltar si resultado < 0

### Formato

SPM <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

Si los bits de la palabra de estado A1 = 0 y A0 = 1, la operación **SPM <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L EW8	
L MW12	
-I	//Restar el contenido de MW12 del contenido de EW8.
SPM NEG	//Saltar si el resultado < 0 (es decir, si el contenido de ACU 1 < 0).
UN M 4.0	//La ejecución del programa continúa aquí en caso de no ejecutarse el //salto.
S M 4.0	
SPA NEXT	
NEG: UN M 4.1	//La ejecución del programa continúa aquí después haber saltado a la //meta NEG.
S M 4.1	
NEXT: NOP 0	//La ejecución del programa continúa aquí después de haber saltado a la //meta NEXT.

## 6.16 SPPZ Saltar si el resultado >= 0

### Formato

SPPZ <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el resultado indicado por los bits A1 y A0 de la palabra de estado es mayor o igual que cero ( $A1=0/A0 = 0$  ó  $A1 = 1/A0 = 0$ ), la operación **SPPZ <meta>** (Saltar si el resultado >= 0) interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L EW8	
L MW12	
-I	//Restar el contenido de MW12 del contenido de EW8.
SPPZ REG0	//Saltar si el resultado >= 0 (es decir, si el contenido de //ACU 1 >= 0).
UN M 4.0	//La ejecución del programa continúa aquí en caso de no ejecutarse el //salto.
S M 4.0	
SPA NEXT	
REG0: UN M 4.1	//La ejecución del programa continúa aquí después de haber saltado a la //meta REG0.
S M 4.1	
NEXT: NOP 0	//La ejecución del programa continúa aquí después de haber saltado a la //meta NEXT.

## 6.17 SPMZ Saltar si el resultado <= 0

### Formato

SPMZ <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

Si el resultado indicado por los bits A1 y A0 de la palabra de estado es menor o igual que cero ( $A1 = 0/A0 = 0$  ó  $A1 = 0/A0 = 1$ ), la operación **SPMZ <meta>** (Saltar si el resultado <= 0) interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L EW8	
L MW12	
-I	//Restar el contenido de MW12 del contenido de EW8.
SPMZ RGE0	//Saltar si el resultado <= 0 (es decir, si el contenido de //ACU 1 <= 0).
UN M 4.0	//La ejecución del programa continúa aquí en caso de no ejecutarse el //salto.
S M 4.0	
SPA NEXT	
RGE0: UN M 4.1	//La ejecución del programa continúa aquí después de haber saltado a la //meta RGE0.
S M 4.1	
NEXT: NOP 0	//La ejecución del programa continúa aquí después de haber saltado a la //meta NEXT.

## 6.18 SPU Saltar si el resultado no es válido

### Formato

SPU <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

Si los bits de la palabra de estado A1 = 1 y A0 = 1, la operación **SPU <meta>** interrumpe la ejecución lineal del programa y salta a la meta que se haya indicado. La ejecución lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

Los bits de la palabra de estado A1 y A0 se ponen a 1 en los siguientes casos:

- al dividir por cero, o
- al utilizar operaciones no permitidas, o
- cuando una comparación de números en coma flotante da un resultado "inválido", es decir, al utilizar un formato inválido.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L MD10	
L ED2	
/D	//Dividir el contenido de MD10 por el contenido de ED2.
SPU ERRO	//Saltar si hay una división por cero (es decir, ED2 = "0").
T MD14	//La ejecución del programa continúa aquí en caso de no ejecutarse el //salto.
U M 4.0	
R M 4.0	
SPA NEXT	
ERRO: UN M 4.0	//La ejecución del programa continúa aquí después de haber saltado a la //meta ERRO.
S M 4.0	
NEXT: NOP 0	//La ejecución del programa continúa aquí después de haber saltado a la //meta NEXT.

## 6.19 LOOP Bucle

### Formato

LOOP <meta>

Operando	Descripción
<meta>	Nombre simbólico de la meta del salto

### Descripción de la operación

**LOOP <meta>** (Decrementar el ACU1-L y saltar, si el ACU1-L  $\neq$  0) simplifica la programación de bucles. El contador de bucles es un entero (de 16 bits) sin signo y se encuentra en el ACU1-L. La instrucción salta a la meta que se haya indicado. El salto se ejecuta mientras el contenido del ACU1-L sea diferente de "0". El desarrollo lineal del programa continúa en la meta del salto, que está señalada por una marca. Se puede saltar tanto hacia adelante como hacia atrás. Los saltos sólo pueden ser ejecutados dentro de un bloque; esto implica que tanto la instrucción del salto como su meta tienen que encontrarse dentro del mismo bloque. La meta del salto sólo puede estar representada una sola vez dentro de este bloque. La distancia máxima del salto es de -32768 ó +32767 palabras del código de programa. El número máximo efectivo de las instrucciones que se pueden saltar depende de cuál sea la combinación de las instrucciones dentro del programa (instrucciones de una, dos o tres palabras).

### Palabra de estado

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

### Ejemplo de cálculo de la facultad de 5 (5!)

AWL	Explicación
L L#1	//Cargar la constante de entero (32 bits) en el ACU 1.
T MD20	//Transferir el contenido del ACU 1 a MD20 (Inicialización).
L 5	//Cargar el número de ciclos de bucles en el ACU1-L.
NEXT: T MW10	//Meta = inicio del bucle Transferir el ACU1-L al contador de bucles.
L MD20	
* D	//Multiplicar el contenido actual de MD20 por el contenido actual de //MB10.
T MD20	//Transferir el resultado de la multiplicación a MD20.
L MW10	//Cargar el contenido del contador de bucles en el ACU 1.
LOOP NEXT	//Decrementar el contenido del ACU 1 y saltar a la meta NEXT, //si el ACU1-L > 0.
L MW24	//La ejecución del programa continúa aquí después del fin del bucle.
L 200	
>I	



## 7 Aritmética de enteros

### 7.1 Lista de operaciones aritméticas con enteros

#### Descripción

Las operaciones aritméticas combinan el contenido de los ACUs 1 y 2. El resultado se deposita en el ACU 1. El contenido del ACU 2 permanece inalterado.

En las CPUs con 4 acumuladores se copia a continuación el contenido del ACU 3 en el ACU 2, y el contenido del ACU 4 en el ACU 3. El antiguo contenido del ACU 4 no varía.

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

- +I Sumar ACU 1 y 2 como entero
- -I Restar ACU 1 de ACU 2 como entero
- /I Dividir ACU 2 por ACU 1 como entero
- \*I Multiplicar ACU 1 por ACU 2 como entero
- + Sumar constante entera o entera doble
- +D Sumar ACU 1 y 2 como entero doble
- -D Restar ACU 1 de ACU 2 como entero doble
- \*D Multiplicar ACU 1 por ACU 2 como entero doble
- /D Dividir ACU 2 por ACU 1 como entero doble
- MOD Resto de la 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: $-32\ 768 \leq \text{resultado} < 0$ (número negativo) enteros dobles: $-2\ 147\ 483\ 648 \leq \text{resultado} < 0$ (número negativo)	0	1	0	*
enteros: $32\ 767 > \text{resultado} > 0$ (número positivo) enteros dobles: $2\ 147\ 483\ 647 > \text{resultado} > 0$ (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 +I Sumar ACU 1 y 2 como entero

### Formato

+I

### Descripción de la operación

+I (Sumar enteros) suma el contenido del ACU1-L al contenido del ACU2-L y almacena el resultado en el ACU1-L. Los contenidos del ACU1-L y ACU2-L se evalúan como enteros (de 16 bits). La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits de la palabra de estado A1, A0, OS y OV. En caso de un desbordamiento, ya sea por defecto o por exceso, el resultado de la operación no es un entero doble (de 32 bits), sino un entero (de 16 bits).

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU 3. El contenido del ACU 4 queda inalterado.

### Palabra de estado

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

Configuración de los bits de la palabra de estado	A1	A0	OV	OS
Suma = 0	0	0	0	-
-32768 <= Suma < 0	0	1	0	-
32767 >= Suma > 0	1	0	0	-
Suma = -65536	0	0	1	1
65534 >= Suma > 32767	0	1	1	1
-65535 <= Suma < -32768	1	0	1	1

### Ejemplo

AWL	Explicación
L EW10	//El valor de EW10 se carga en el ACU1-L.
L MW14	//Cargar el contenido del ACU1-L en el ACU2-L. Cargar el valor de MW14 //en el ACU1-L.
+I	//Sumar ACU2-L y ACU1-L, almacenar el resultado en el ACU1-L.
T DB1.DBW25	//El contenido del ACU1-L (resultado) se transfiere del DB1 a DBW25.

## 7.4 -I Restar ACU 1 de ACU 2 como entero

### Formato

-I

### Descripción de la operación

-I (Restar enteros) resta el contenido del ACU1-L del contenido del ACU2-L y almacena el resultado en el ACU1-L. Los contenidos del ACU1-L y ACU2-L se evalúan como enteros (de 16 bits). La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits de la palabra de estado A1, A0, OS y OV. En caso de producirse un desbordamiento, ya sea por exceso o por defecto, el resultado de la operación no es un entero doble (de 32 bits), sino un entero (de 16 bits).

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU 3. El contenido del ACU 4 queda inalterado.

### Palabra de estado

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

Configuración de los bits de la palabra de estado	A1	A0	OV	OS
Diferencia = 0	0	0	0	-
-32768 <= Diferencia < 0	0	1	0	-
32767 >= Diferencia > 0	1	0	0	-
65535 >= Diferencia > 32767	0	1	1	1
-65535 <= Diferencia < -32768	1	0	1	1

### Ejemplo

AWL	Explicación
L EW10	//El valor de EW10 se carga en el ACU1-L.
L MW14	//Cargar el contenido del ACU1-L en el ACU2-L. Cargar el valor de MW14 //en el ACU1-L.
-I	//Restar ACU1-L de ACU2-L, almacenar el resultado en el ACU1-L.
T DB1.DBW25	//El contenido del ACU1-L (resultado) se transfiere del DB1 a DBW25.

## 7.5 \*I Multiplicar ACU 1 por ACU 2 como entero

### Formato

\*I

### Descripción de la operación

\*I (Multiplicar enteros) multiplica el contenido del ACU2-L por el contenido del ACU1-L. Los contenidos del ACU1-L y ACU2-L se evalúan como enteros (16 bits). El resultado se almacena como entero doble (32 bits) en el ACU 1. Si los bits de la palabra de estado OV = 1 y OS = 1, el resultado queda fuera del margen válido para un entero (de 16 bits).

La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits de la palabra de estado A1, A0, OS y OV.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU 3. El contenido del ACU 4 queda inalterado.

### Palabra de estado

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

Configuración de los bits de la palabra de estado	A1	A0	OV	OS
Resultado = 0	0	0	0	-
-32768 <= Resultado < 0	0	1	0	-
32767 >= Resultado > 0	1	0	0	-
1.073.741.824 >= Resultado > 32767	1	0	1	1
-1.073.709.056 <= Resultado < -32768	0	1	1	1

### Ejemplo

AWL	Explicación
L EW10	//El valor de EW10 se carga en el ACU1-L.
L MW14	//Cargar el contenido del ACU1-L en el ACU 2-L. Cargar el valor de MW14 //en el ACU1-L.
*I	//Multiplicar el ACU2-L por el ACU1-L, almacenar el resultado en el //ACU 1.
T DB1.DBD25	//El contenido del ACU 1 (resultado) se transfiere del DB1 a DBD25.

## 7.6 // Dividir ACU 2 por ACU 1 como entero

### Formato

//

### Descripción de la operación

// (Dividir enteros) divide el contenido del ACU2-L por el contenido del ACU1-L. Los contenidos de ACU1-L y ACU2-L se evalúan como enteros (de 16 bits). El resultado se almacena en el ACU 1 y contiene dos enteros (de 16 bits), el cociente y el resto de la división. El cociente se almacena en el ACU1-L y el resto de la división en el ACU1-H. La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits de la palabra de estado A1, A0, OS y OV.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU 3. El contenido del ACU 4 queda inalterado.

### Palabra de estado

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

Configuración de los bits de la palabra de estado	A1	A0	OV	OS
Cociente = 0	0	0	0	-
-32768 <= Cociente < 0	0	1	0	-
32767 >= Cociente > 0	1	0	0	-
Cociente = 32768	1	0	1	1
División por cero	1	1	1	1

### Ejemplo

AWL	Explicación
L EW10	//Cargar el valor de EW10 en el ACU1-L.
L MW14	//Cargar el contenido del ACU1-L en el ACU2-L. Cargar el valor de MW14 //en el ACU1-L.
/I	//Dividir ACU2-L por ACU1-L, almacenar el resultado en el ACU 1: ACU1-L: //Cociente, ACU1-H: resto de la división
T MD20	//El contenido del ACU 1 (resultado) se transfiere a MD20.

**Ejemplo de división con cifras: 13 / 4**

Contenido del ACU2-L antes de la operación (EW10):	"13"
Contenido del ACU1-L antes de la operación (MW14):	"4"
Operación /I (ACU2-L / ACU1-L):	"13/4"
Contenido del ACU1-L después de la operación (cociente):	"3"
Contenido del ACU1-H después de la operación (resto de la división):	"1"

## 7.7 + Sumar constante entera o entera doble

### Formato

+ <constante entera>

Operando	Tipo de datos	Descripción
<constante entera>	Constante entera (16 bits) o entera doble (32 bits)	Constante a sumar

### Descripción de la operación

+ <constante entera> suma la constante entera al contenido del ACU 1 y almacena el resultado en el ACU 1. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

+ <constante entera de 16 bits> suma una constante entera (de 16 bits; dentro del margen de -32768 a +32767) al contenido del ACU 1-L y almacena el resultado en el ACU 1-L.

+ <constante entera de 32 bits> suma una constante entera doble (dentro del margen de -2.147.483.648 a 2.147.483.647) al contenido del ACU 1 y almacena el resultado en el ACU 1.

### Palabra de estado

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

### Ejemplo 1

AWL	Explicación
L EW10	//Cargar el valor de EW10 en el ACU1-L.
L MW14	//Cargar el contenido del ACU1-L en el ACU2-L. Cargar el valor de MW14 //en el ACU1-L.
+I	//Sumar el ACU2-L y el ACU1-L, almacenar el resultado en el ACU1-L.
+ 25	//Sumar el ACU1-L y 25, almacenar el resultado en el ACU1-L.
T DB1.DBW25	//Transferir el contenido de ACU1-L (resultado) del DB 1 a DBW25.

### Ejemplo 2

AWL	Explicación
L EW12	
L EW14	
+ 100	//Sumar el ACU1-L y 100, almacenar el resultado en el ACU 1-L.
>I	//Si el ACU 2 > ACU 1, o si EW 12 > (EW14 + 100),
SPB NEXT	//saltar a la meta NEXT.

**Ejemplo 3**

AWL		Explicación
L	MD20	
L	MD24	
+D		//Sumar ACU 1 y ACU 2, almacenar el resultado en el ACU 1.
+	L#-200	//Sumar ACU1 y -200, almacenar el resultado en el ACU 1.
T	MD28	

## 7.8 +D Sumar ACU 1 y 2 como entero doble

### Formato

+D

### Descripción de la operación

**+D** (Sumar enteros dobles) suma el contenido del ACU 1 al contenido del ACU 2 y almacena el resultado en el ACU 1. Los contenidos del ACU 1 y del ACU 2 se evalúan como enteros dobles, es decir, de 32 bits. La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits de la palabra de estado A1, A0, OS y OV.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenido del ACU 3 al ACU 2 y del ACU 4 al ACU 3. El contenido del ACU 4 queda inalterado.

### Palabra de estado

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

Configuración de los bits de la palabra de estado	A1	A0	OV	OS
Suma = 0	0	0	0	-
-2.147.483.648 <= Suma < 0	0	1	0	-
2.147.483.647 >= Suma > 0	1	0	0	-
Suma = -4.294.967.296	0	0	1	1
4.294.967.294 >= Suma > 2.147.483.647	0	1	1	1
-4.294.967.295 <= Suma < -2.147.483.648	1	0	1	1

### Ejemplo

AWL	Explicación
L ED10	//El valor de ED10 se carga en el ACU 1.
L MD14	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD14 //en el ACU 1.
+D	//Sumar ACU 2 y ACU 1, almacenar el resultado en el ACU 1.
T DB1.DBD25	//El contenido del ACU 1 (resultado) se transfiere del DB1 a DBD25.

## 7.9 -D Restar ACU 1 de ACU 2 como entero doble

### Formato

-D

### Descripción de la operación

**-D** (Restar enteros dobles) resta el contenido del ACU 1 del contenido del ACU 2 y almacena el resultado en el ACU 1. Los contenidos de ACU 1 y ACU 2 se evalúan como enteros dobles, es decir, de 32 bits. La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits de la palabra de estado A1, A0, OS y OV.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU 3. El contenido del ACU 4 queda inalterado.

### Palabra de estado

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

Configuración de los bits de la palabra de estado	A1	A0	OV	OS
Diferencia = 0	0	0	0	-
-2.147.483.648 <= Diferencia < 0	0	1	0	-
2.147.483.647 >= Diferencia > 0	1	0	0	-
4.294.967.295 >= Diferencia > 2.147.483.647	0	1	1	1
-4.294.967.295 <= Diferencia < -2.147.483.648	1	0	1	1

### Ejemplo

AWL	Explicación
L ED10	//El valor de ED10 se carga en el ACU 1.
L MD14	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD14 //en el ACU 1.
-D	//Restar ACU 2 de ACU 1, almacenar el resultado en el ACU 1.
T DB1.DBD25	//El contenido del ACU 1 (resultado) se transfiere del DB1 a DBD25.

## 7.10 \*D Multiplicar ACU 1 por ACU 2 como entero doble

### Formato

\*D

### Descripción de la operación

\*D (Multiplicar enteros dobles) multiplica el contenido del ACU 1 por el contenido del ACU 2. Los contenidos de ACU 1 y ACU 2 se evalúan como enteros dobles, es decir, como enteros de 32 bits. El resultado se almacena como entero doble en el ACU 1. Si los bits de la palabra de estado OV1 = 1 y OS = 1, el resultado queda fuera del margen de un entero doble.

La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits de la palabra de estado A1, A0, OS y OV.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU 3. El contenido del ACU 4 queda inalterado.

### Palabra de estado

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

Configuración de los bits de la palabra de estado	A1	A0	OV	OS
Resultado = 0	0	0	0	-
-2.147.483.648 <= Resultado < 0	0	1	0	-
2.147.483.647 >= Resultado > 0	1	0	0	-
Resultado > 2.147.483.647	1	0	1	1
Resultado < -2.147.483.648	0	1	1	1

### Ejemplo

AWL	Explicación
L ED10	//El valor de ED10 se carga en el ACU 1.
L MD14	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD14 en //el ACU 1.
*D	//Multiplicar ACU 2 por ACU 1, almacenar el resultado en el ACU 1.
T DB1.DB25	//El contenido del ACU 1 (resultado) se transfiere del DB1 a DB25.

## 7.11 /D Dividir ACU 2 por ACU 1 como entero doble

### Formato

/D

### Descripción de la operación

**/D** (Dividir enteros dobles) divide el contenido del ACU 2 por el contenido del ACU 1. Los contenidos de ACU 1 y ACU 2 se evalúan como enteros dobles, es decir, como enteros de 32 bits. El resultado se almacena en el ACU 1. El resultado contiene sólo el cociente, pero no el resto de la división (el resto de la división se obtiene realizando la operación **MOD**).

La operación se realiza sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits de la palabra de estado A1, A0, OS y OV.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU 3. El contenido del ACU 4 queda inalterado.

### Palabra de estado

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

Configuración de los bits de la palabra de estado	A1	A0	OV	OS
Cociente = 0	0	0	0	-
-2147483648 <= Cociente < 0	0	1	0	-
2147483647 >= Cociente > 0	1	0	0	-
Cociente = 2147483648	1	0	1	1
División por cero	1	1	1	1

### Ejemplo

AWL	Explicación
L ED10	//El valor de ED10 se carga en el ACU 1.
L MD14	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD14 en //el ACU 1.
/D	//Dividir ACU 2 por ACU 1, almacenar el resultado (cociente) en el ACU 1.
T MD20	//El contenido del ACU 1 (resultado) se transfiere a MD20.

### Ejemplo: 13 / 4

Contenido del ACU 2 antes de la operación (ED10):	"13"
Contenido del ACU 1 antes de la operación (MD14):	"4"
Operación /D (ACU 2 / ACU 1):	"13/4"
Contenido del ACU 1 después de la operación (cociente):	"3"

## 7.12 MOD Resto de la división de enteros dobles

### Formato

**MOD**

### Descripción de la operación

**MOD** (Resto de división de enteros dobles) divide el contenido del ACU 2 por el contenido del ACU 1. Los contenidos de ACU 1 y ACU 2 se evalúan como enteros dobles, es decir, como enteros de 32 bits. El resultado se almacena en el ACU 1. El resultado sólo contiene el resto de la división, pero no el cociente (el cociente se obtiene realizando la operación **/D**).

La operación se realiza sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits de la palabra de estado A1, A0, OS y OV.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU 3. El contenido del ACU 4 queda inalterado.

### Palabra de estado

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

Configuración de los bits de la palabra de estado	A1	A0	OV	OS
Resto = 0	0	0	0	-
-2147483648 <= Resto < 0	0	1	0	-
2147483647 >= Resto > 0	1	0	0	-
División por cero	1	1	1	1

### Ejemplo

AWL	Explicación
L ED10	//El valor de ED10 se carga en el ACU 1.
L MD14	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD14 en //el ACU 1.
MOD	//Dividir ACU 2 por ACU 1, almacenar el resultado (resto de la división) //en el ACU 1.
T MD20	//El contenido del ACU 1 (resultado) se transfiere a MD20.

### Ejemplo: 13 / 4

Contenido del ACU 2 antes de la operación (ED10): "13"  
 Contenido del ACU 1 antes de la operación (MD14): "4"  
 Operación /D (ACU 2 / ACU 1): "13/4"  
 Contenido del ACU 1 después de la operación (resto de la división): "1"

## 8 Aritmética en coma flotante

### 8.1 Lista de operaciones aritméticas con números en coma flotante

#### Descripción

Las operaciones aritméticas combinan el resultado de los ACUs 1 y 2. El resultado se deposita en el ACU 1. El contenido del ACU 2 permanece inalterado.

En las CPU con 4 acumuladores se copia a continuación el contenido del ACU 3 en el ACU 2, y el contenido del ACU 4 en el ACU 3. El antiguo contenido del ACU 4 no varía.

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:

- +R Sumar ACU 1 y 2 como número de coma flotante (32 bits)
- -R Restar ACU 1 de ACU 2 como número de coma flotante (32 bits)
- \*R Multiplicar ACU 1 por ACU 2 como número de coma flotante (32 bits)
- /R Dividir ACU 2 por ACU 1 como número de coma flotante (32 bits)

Con las operaciones aritméticas de números en coma flotante se pueden ejecutar las siguientes funciones con **un** número en coma flotante (32 bit, IEEE 754):

- ABS Valor absoluto de un número de coma flotante (32 bits, IEEE 754)
- EXP Calcular el exponente de un número de coma flotante (32 bits)
- LN Calcular el logaritmo natural de un número de coma flotante (32 bits)
- SQR Calcular el cuadrado de un número de coma flotante (32 bits)
- SQRT Calcular la raíz cuadrada de un número de coma flotante (32 bits)
  
- SIN Calcular el seno de ángulos como números de coma flotante (32 bits)
- COS Calcular el coseno de ángulos como números de coma flotante (32 bits)
- TAN Calcular la tangente de ángulos como números de coma flotante (32 bits)
- ASIN Calcular el arcoseno de un número de coma flotante (32 bits)
- ACOS Calcular el arcocoseno de un número de coma flotante (32 bits)
- ATAN Calcular la arcotangente de un número de coma flotante (32 bits)

## 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 no válido	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 o operación no permitida (valor de entrada fuera del margen válido de valores)	1	1	1	1

## 8.3 Operaciones básicas

### 8.3.1 +R Sumar ACU 1 y 2 como número en coma flotante (32 bits)

#### Formato

+R

#### Descripción de la operación

**+R** (Sumar números en coma flotante de 32 bits, IEEE 754) suma el contenido del ACU 1 al contenido del ACU 2 y almacena el resultado en el ACU 1. Los contenidos de ACU 1 y ACU 2 se evalúan como números en coma flotante (32 bits, IEEE 754). La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits A1, A0, OS y OV de la palabra de estado.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU3. El contenido del ACU 4 queda inalterado.

#### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Desbordamiento por exceso
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por defecto
+cero	0	0	0	-	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-infinito	0	1	1	1	Desbordamiento por exceso
-qNaN	1	1	1	1	

#### Palabra de estado

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

### Ejemplo

AWL		Explicación
AUF	DB10	
L	ED10	//El valor de ED10 se carga en el ACU 1.
L	MD14	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD14 en el //ACU 1.
+R		//Sumar ACU 2 y ACU 1, almacenar el resultado en el ACU 1.
T	DBD25	//El contenido del ACU 1 (= resultado) se transfiere a DBD25 en DB10.

### 8.3.2 -R Restar ACU 1 de ACU 2 como número en coma flotante (32 bits)

#### Formato

-R

#### Descripción de la operación

**-R** (Restar números en coma flotante de 32 bits, IEEE 754) resta el contenido del ACU 1 del contenido del ACU 2 y almacena el resultado en el ACU 1. Los contenidos de ACU 1 y ACU 2 se evalúan como números en coma flotante (32 bits, IEEE 754). El resultado se almacena en el ACU 1. La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits A1, A0, OS y OV de la palabra de estado.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU3. El contenido del ACU 4 queda inalterado.

#### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Desbordamiento por exceso
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por defecto
+cero	0	0	0	-	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-infinito	0	1	1	1	Desbordamiento por exceso
-qNaN	1	1	1	1	

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
AUF DB10	
L ED10	//El valor de ED10 se carga en el ACU 1.
L MD14	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD14 en el //ACU 1.
-R	//Restar ACU 2 de ACU 1, almacenar el resultado en el ACU 1.
T DBD25	//El contenido del ACU 1 (= resultado) se transfiere a DBD25 en DB10.

### 8.3.3 \*R Multiplicar ACU 1 por ACU 2 como número en coma flotante (32 bits)

#### Formato

\*R

#### Descripción de la operación

\*R (Multiplicar números en coma flotante de 32 bits, IEEE 754) multiplica el contenido del ACU 2 por el contenido del ACU 1. Los contenidos de ACU 1 y ACU 2 se evalúan como números en coma flotante (32 bits, IEEE 754). El resultado se almacena en formato de número en coma flotante (32 bits, IEEE 754) en el ACU 1. La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits A1, A0, OS y OV de la palabra de estado.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU 3. El contenido del ACU 4 queda inalterado.

#### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Desbordamiento por exceso
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por defecto
+cero	0	0	0	-	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-infinito	0	1	1	1	Desbordamiento por exceso
-qNaN	1	1	1	1	

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
AUF DB10	
L ED10	//El valor de ED10 se carga en el ACU 1.
L MD14	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD14 en //el ACU 1.
*R	//Multiplicar ACU 2 por ACU 1, almacenar el resultado en el ACU 1.
T DBD25	//El contenido del ACU 1 (= resultado) se transfiere a DBD25 en DB10.

### 8.3.4 /R Dividir ACU 2 por ACU 1 como número en coma flotante (32 bits)

#### Formato

/R

#### Descripción de la operación

/R (Dividir números en coma flotante de 32 bits, IEEE 754) divide el contenido del ACU 2 por el contenido del ACU 1. Los contenidos de ACU 1 y ACU 2 se evalúan como números en coma flotante (32 bits, IEEE 754). La operación se ejecuta sin tener en cuenta ni afectar al RLO. Una vez realizada la operación se activan los bits A1, A0, OS y OV de la palabra de estado.

En las CPU con dos acumuladores, el contenido del ACU 2 queda inalterado.

En las CPU con cuatro acumuladores, se copian los contenidos del ACU 3 al ACU 2 y del ACU 4 al ACU3. El contenido del ACU 4 queda inalterado.

#### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Desbordamiento por exceso
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por defecto
+cero	0	0	0	-	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-infinito	0	1	1	1	Desbordamiento por exceso
-qNaN	1	1	1	1	

#### Palabra de estado

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

#### Ejemplo

AWL	Explicación
AUF DB10	
L ED10	//El valor de ED10 se carga en el ACU 1.
L MD14	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD14 en el //ACU 1.
/R	//Dividir ACU 2 por ACU 1, almacenar el resultado en el ACU 1.
T DBD20	//El contenido del ACU 1 (= resultado) se transfiere a DBD20 en DB10.

### 8.3.5 ABS Valor absoluto de un número en coma flotante (32 bits, IEEE 754)

#### Formato

**ABS**

#### Descripción de la operación

**ABS** (Valor absoluto de un número en coma flotante de 32 bits, IEEE 754) calcula el valor absoluto de un número en coma flotante (32bits, IEEE 754) en el ACU 1. El resultado se almacena en el ACU 1. La operación se ejecuta sin tener en cuenta ni afectar los bits de la palabra de estado.

#### Palabra de estado

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

#### Ejemplo

AWL		Explicación
L	ED8	//Cargar el valor en el ACU 1 (ejemplo: ED8 = -1.5E+02).
ABS		//Calcular el valor absoluto, almacenar el resultado en el ACU 1.
T	MD10	//Transferir el resultado a MD10 (ejemplo: resultado = 1.5E+02).

## 8.4 Operaciones ampliadas

### 8.4.1 SQR Calcular el cuadrado de un número en coma flotante (32 bits)

#### Formato

SQR

#### Descripción de la operación

**SQR** (Calcular el cuadrado de un número en coma flotante de 32 bits, IEEE 754) calcula el cuadrado de un número en coma flotante (de 32 bits, IEEE 754) en el ACU 1. El resultado se almacena en el ACU 1. La operación afecta a los bits A1, A0, OV y OS de la palabra de estado.

Los contenidos del ACU 2 (y en las CPU con cuatro acumuladores también los contenidos del ACU 3 y ACU 4) permanecen inalterados.

#### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Desbordamiento
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento negativo
+cero	0	0	0	-	
-qNaN	1	1	1	1	

#### Ejemplo

AWL	Explicación
AUF DB17	//Abrir bloque de datos DB17.
L DBD0	//El valor de la doble palabra de datos DBD0 se carga en el ACU 1. //(Este valor debe tener formato en coma flotante.)
SQR	//Calcular el cuadrado del número en coma flotante (32-bits, IEEE 754) //en el ACU 1. Depositar el resultado en el ACU 1.
UN OV	//Consultar si el estado de señal del bit OV es "0".
SPB OK	//Si en la operación SQR no se produjo ningún error, saltar a la marca //OK.
BEA	//Fin de bloque incondicionado, si en la operación SQR se produjo un //error.
OK: T DBD4	//Transferir el resultado del ACU 1 a la doble palabra de datos DBD4.

### 8.4.2 SQRT Calcular la raíz cuadrada de un número en coma flotante (32 bits)

#### Formato

##### SQRT

#### Descripción de la operación

**SQRT** (Calcular la raíz cuadrada de un número en coma flotante de 32 bits, IEEE 754) calcula la raíz cuadrada de un número en coma flotante (de 32 bits, IEEE 754) en el ACU 1. El resultado se almacena en el ACU 1. El valor de entrada debe ser mayor o igual que cero; el resultado es entonces positivo. Única excepción: la raíz cuadrada de -0 es -0. La operación afecta a los bits A1, A0, OV y OS de la palabra de estado.

Los contenidos del ACU 2 (y en las CPU con cuatro acumuladores también los contenidos del ACU 3 y del ACU 4) quedan inalterados.

#### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Desbordamiento por exceso
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por defecto
+cero	0	0	0	-	
-cero	0	0	0	-	
-qNaN	1	1	1	1	

#### Ejemplo

AWL			Explicación
L	MD10		//El valor de la doble palabra de marcas MD10 se carga en el ACU 1. //(Este valor debe tener formato en coma flotante.)
SQRT			//Calcular la raíz cuadrada del número en coma flotante //(32-bits, IEEE 754) en el ACU 1. Depositar el resultado en el ACU 1.
UN	OV		//Consultar si el estado de señal del bit OV es "0".
SPB	OK		//Si en la operación SQRT no se produjo ningún error, saltar a la marca //OK.
BEA			//Fin de bloque incondicionado, si en la operación SQRT se produjo un //error.
OK:	T	MD20	//Transferir el resultado del ACU 1 a la doble palabra de marcas MD20.

### 8.4.3 EXP Calcular el exponente de un número en coma flotante (32 bits)

#### Formato

**EXP**

#### Descripción de la operación

**EXP** (Calcular el exponente de un número en coma flotante de 32 bits, IEEE 754) calcula el exponente (valor exponencial con base e) de un número en coma flotante (de 32 bits, IEEE 754) en el ACU 1. El resultado se almacena en el ACU 1. La operación afecta a los bits A1, A0, OV y OS de la palabra de estado.

Los contenidos del ACU 2 (y en las CPU con cuatro acumuladores también los contenidos del ACU 3 y del ACU 4) quedan inalterados.

#### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Desbordamiento por exceso
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por defecto
+cero	0	0	0	-	
-qNaN	1	1	1	1	

#### Ejemplo

AWL			Explicación
L	MD10		//El valor de la doble palabra de marcas MD 10 se carga en el ACU 1. //(Este valor debe tener formato en coma flotante.)
EXP			//Calcular el exponente de un número en coma flotante //(de 32 bits, IEEE 754) con base e en el ACU 1. Depositar el resultado //en el ACU 1.
UN	OV		//Consultar si el estado de señal del bit OV es "0".
SPB	OK		//Si en la operación EXP no se produjo ningún error, saltar a la marca //OK.
BEA			//Fin de bloque incondicionado, si en la operación EXP se produjo un //error.
OK:	T	MD20	//Transferir el resultado del ACU 1 a la doble palabra de marcas MD20.

### 8.4.4 LN Calcular el logaritmo natural de un número en coma flotante (32 bits)

#### Formato

LN

#### Descripción de la operación

LN (Calcular el logaritmo natural de un número en coma flotante de 32 bits, IEEE 754) calcula el logaritmo natural (logaritmo con base e) de un número en coma flotante (de 32 bits, IEEE 754) en el ACU 1. El resultado se almacena en el ACU 1. El valor de entrada debe ser mayor que cero. La operación afecta a los bits A1, A0, OV y OS de la palabra de estado.

Los contenidos del ACU 2 (y en las CPU con cuatro acumuladores también los contenidos del ACU 3 y del ACU 4) quedan inalterados.

#### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Desbordamiento por exceso
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por defecto
+cero	0	0	0	-	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-infinito	0	1	1	1	Desbordamiento por exceso
-qNaN	1	1	1	1	

#### Ejemplo

AWL	Explicación		
L MD10	//El valor de la doble palabra de marcas MD 10 se carga en el ACU 1. //(Este valor debe tener formato en coma flotante.)		
LN	//Calcular el logaritmo natural del número en coma flotante //(32 bits, IEEE 754) en el ACU 1. Depositar el resultado en el ACU 1.		
UN OV	//Consultar si el estado de señal del bit OV es "0".		
SPB OK	//Si en la operación LN no se produjo ningún error, saltar a la marca //OK.		
BEA	//Fin de bloque incondicionado, si en la operación LN se produjo un //error.		
OK: T MD20	//Transferir el resultado del ACU 1 a la doble palabra de marcas MD20.		

### 8.4.5 SIN Calcular el seno de ángulos como números en coma flotante (32 bits)

#### Formato

**SIN**

#### Descripción de la operación

**SIN** (Calcular el seno de ángulos como números en coma flotante de 32 bits, IEEE 754) calcula el seno de un ángulo indicado en radianes. El ángulo debe estar representado en el ACU 1 en formato de número en coma flotante. El resultado se almacena en el ACU 1. La operación afecta a los bits A1, A0, OV y OS de la palabra de estado.

Los contenidos del ACU 2 (y en las CPU con cuatro acumuladores también los contenidos del ACU 3 y del ACU 4) quedan inalterados.

#### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por exceso
+cero	0	0	0	-	
+infinito	1	0	1	1	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-qNaN	1	1	1	1	

#### Ejemplo

AWL	Explicación
L MD10	//El valor de la doble palabra de marcas MD 10 se carga en el ACU 1. //(Este valor debe tener formato en coma flotante.)
SIN	//Calcular el seno del número en coma flotante (de 32 bits, IEEE 754) en //el ACU 1. Depositar el resultado en el ACU 1.
T MD20	//Transferir el resultado del ACU 1 a la doble palabra de marcas MD20.

### 8.4.6 COS Calcular el coseno de ángulos como números en coma flotante (32 bits)

#### Formato

**COS**

#### Descripción de la operación

**COS** (Calcular el coseno de ángulos como números en coma flotante de 32 bits, IEEE 754) calcula el coseno de un ángulo indicado en radianes. El ángulo debe estar representado en el ACU 1 en formato de número en coma flotante. El resultado se almacena en el ACU 1. La operación afecta a los bits A1, A0, OV y OS de la palabra de estado.

Los contenidos del ACU 2 (y en las CPU con cuatro acumuladores también los contenidos del ACU 3 y del ACU 4) quedan inalterados.

#### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por exceso
+cero	0	0	0	-	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-qNaN	1	1	1	1	

#### Ejemplo

AWL	Explicación
L MD10	//El valor de la doble palabra de marcas MD 10 se carga en el ACU 1. //(Este valor debe tener formato en coma flotante.)
COS	//Calcular el coseno del número en coma flotante (de 32 bits, IEEE 754) //en el ACU 1. Depositar el resultado en el ACU 1.
T MD20	//Transferir el resultado del ACU 1 a la doble palabra de marcas MD20.

### 8.4.7 TAN Calcular la tangente de ángulos como números en coma flotante (32 bits)

#### Formato

**TAN**

### Descripción de la operación

**TAN** (Calcular la tangente de ángulos como números en coma flotante de 32 bits, IEEE 754) calcula la tangente de un ángulo indicado en radianes. El ángulo debe estar representado en el ACU 1 en formato de número de coma flotante. El resultado se almacena en el ACU 1. La operación afecta a los bits A1, A0, OV y OS de la palabra de estado.

Los contenidos del ACU 2 (y en las CPU con cuatro acumuladores también los contenidos del ACU 3 y del ACU 4) quedan inalterados.

### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+infinito	1	0	1	1	Desbordamiento por exceso
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por defecto
+cero	0	0	0	-	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-infinito	0	1	1	1	Desbordamiento por exceso
-qNaN	1	1	1	1	

### Ejemplo

AWL	Explicación
L MD10	//El valor de la doble palabra de marcas MD 10 se carga en el ACU 1. //(Este valor debe tener formato en coma flotante.)
TAN	//Calcular la tangente del número en coma flotante (de 32 bits, //IEEE 754) en el ACU 1. Depositar el resultado en el ACU 1.
UN OV	//Consultar si el estado de señal del bit OV es "0".
SPB OK	//Si en la operación TAN no se produjeron ningún error, saltar a la marca //OK.
BEA	//Fin de bloque incondicionado, si en la operación TAN se produjo un //error.
OK: T MD20	//Transferir el resultado del ACU 1 a la doble palabra de marcas MD20.

## 8.4.8 ASIN Calcular el arcoseno de un número en coma flotante (32 bits)

### Formato

#### ASIN

**Descripción de la operación**

**ASIN** (Calcular el arcoseno de un número en coma flotante de 32 bits, IEEE 754) calcula el arcoseno de un número en coma flotante en el ACU 1. Margen de valores admisible para el valor de entrada:

$$-1 \leq \text{valor de entrada} \leq +1$$

El resultado es un ángulo indicado en radianes. El valor se encuentra dentro del margen siguiente:

$$-\pi / 2 \leq \text{arcoseno (ACU 1)} \leq +\pi / 2, \text{ siendo } \pi = 3,14159\dots$$

La operación afecta a los bits A1, A0, OV y OS de la palabra de estado.

Los contenidos del ACU 2 (y en las CPU con cuatro acumuladores también los contenidos del ACU 3 y del ACU 4) quedan inalterados.

**Resultado**

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por exceso
+cero	0	0	0	-	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-qNaN	1	1	1	1	

**Ejemplo**

AWL	Explicación
L MD10	//El valor de la doble palabra de marcas MD 10 se carga en el ACU 1. //(Este valor debe tener formato en coma flotante.)
ASIN	//Calcular el arcoseno de un número en coma flotante (de 32 bits, //IEEE 754) en el ACU 1. Depositar el resultado en el ACU 1.
UN OV	//Consultar si el estado de señal del bit OV es "0".
SPB OK	//Si en la operación ASIN no se produjo ningún error, saltar a la marca //OK.
BEA	//Fin de bloque incondicionado, si en la operación ASIN se produjo un //error.
OK: T MD20	//Transferir el resultado del ACU 1 a la doble palabra de marcas MD20.

**8.4.9 ACOS Calcular el arcocoseno de un número en coma flotante (32 bits)**

**Formato**

**ACOS**

### Descripción de la operación

**ACOS** (Calcular el arcocoseno de un número en coma flotante de 32 bits, IEEE 754) calcula el arcocoseno de un número en coma flotante en el ACU 1. Margen de valores admisible para el valor de entrada:

$$-1 \leq \text{valor de entrada} \leq +1$$

El resultado es un ángulo indicado en radianes. El valor se encuentra dentro del margen siguiente:

$$0 \leq \text{arcocoseno (ACU 1)} \leq \pi, \text{ siendo } \pi = 3,14159\dots$$

La operación afecta a los bits A1, A0, OV y OS de la palabra de estado.

Los contenidos del ACU 2 (y en las CPU con cuatro acumuladores también los contenidos del ACU 3 y del ACU 4) quedan inalterados.

### Resultado

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por exceso
+cero	0	0	0	-	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-qNaN	1	1	1	1	

### Ejemplo

AWL	Explicación
L MD10	//El valor de la doble palabra de marcas MD 10 se carga en el ACU 1. //(Este valor debe tener formato en coma flotante.)
ACOS	//Calcular el arcocoseno de un número en coma flotante (de 32 bits, //IEEE 754) en el ACU 1. Depositar el resultado en el ACU 1.
UN OV	//Consultar si el estado de señal del bit OV es "0".
SPB OK	//Si en la operación ACOS no se produjo ningún error, saltar a la marca //OK.
BEA	//Fin de bloque incondicionado, si en la operación ACOS se produjo un //error.
OK: T MD20	//Transferir el resultado del ACU 1 a la doble palabra de marcas MD20.

## 8.4.10 ATAN Calcular la arcotangente de un número en coma flotante (32 bits)

### Formato

**ATAN**

**Descripción de la operación**

**ATAN** (Calcular la arcotangente de un número en coma flotante de 32 bits, IEEE 754) calcula la arcotangente de un número en coma flotante en el ACU 1. El resultado es un ángulo indicado en radianes. El valor se encuentra dentro del margen siguiente:

$$-\pi / 2 \leq \text{arcotangente (ACU 1)} \leq +\pi / 2, \text{ siendo } \pi = 3,14159\dots$$

La operación afecta a los bits A1, A0, OV y OS de la palabra de estado.

Los contenidos del ACU 2 (y en las CPU con cuatro acumuladores también los contenidos del ACU 3 y del ACU 4) quedan inalterados.

**Resultado**

El resultado en el ACU 1 es	A1	A0	OV	OS	Observación
+qNaN	1	1	1	1	
+normalizado	1	0	0	-	
+desnormalizado	0	0	1	1	Desbordamiento por exceso
+cero	0	0	0	-	
-cero	0	0	0	-	
-desnormalizado	0	0	1	1	Desbordamiento por defecto
-normalizado	0	1	0	-	
-qNaN	1	1	1	1	

**Ejemplo**

AWL	Explicación
L MD10	//El valor de la doble palabra de marcas MD 10 se carga en el ACU 1. //(Este valor debe tener formato en coma flotante.)
ATAN	//Calcular la arcotangente de un número en coma flotante //(de 32 bits, IEEE 754) en el ACU 1. Depositar el resultado en el //ACU 1.
UN OV	//Consultar si el estado de señal del bit OV es "0".
SPB OK	//Si en la operación ATAN no se produjo ningún error, saltar a la marca //OK.
BEA	//Fin de bloque incondicionado, si en la operación ATAN se produjo un //error.
OK: T MD20	//Transferir el resultado del ACU 1 a la doble palabra de marcas MD20.

## 9 Operaciones de carga y transferencia

### 9.1 Lista de operaciones de cargar y transferencia

#### Descripción

Las operaciones de carga (L) y transferencia (T) permiten programar un intercambio de información entre módulos de E/S y áreas de memoria, o bien entre áreas de memoria. La CPU ejecuta estas operaciones en cada ciclo como operaciones incondicionales, es decir, independientemente del resultado lógico de la operación.

Se dispone de las operaciones de cargar y transferencia siguientes:

- L Cargar
- L STW Cargar palabra de estado en ACU 1
- LAR1 Cargar registro de direcciones 1 con contenido del ACU 1
- LAR1 <D> Cargar registro de direcciones 1 con puntero (formato de 32 bits)
- LAR1 AR2 Cargar registro de direcciones 1 con contenido del registro de direcciones 2
- LAR2 Cargar registro de direcciones 2 con contenido del ACU 1
- LAR2 <D> Cargar registro de direcciones 2 con puntero (formato de 32 bits)
  
- T Transferir
- T STW Transferir ACU 1 a la palabra de estado
- TAR Intercambiar registro de direcciones 1 y registro de direcciones 2
- TAR1 Transferir registro de direcciones 1 a ACU 1
- TAR1 AR2 Transferir registro de direcciones 1 a registro de direcciones 2
- TAR1 <D> Transferir registro de direcciones 1 a dirección de destino (puntero de 32 bits)
- TAR2 Transferir registro de direcciones 2 a ACU 1
- TAR2 <D> Transferir registro de direcciones 2 a dirección de destino (puntero de 32 bits)

## 9.2 L Cargar

### Formato

L <operando>

Operando	Tipo de datos	Área de memoria	Dirección fuente
<operando>	BYTE	E, A, PE, M, L, D,	0...65535
	WORD	puntero, parámetro	0...65534
	DWORD		0...65532

### Descripción de la operación

L <operando> carga en el ACU 1 el contenido del byte, de la palabra o de la doble palabra direccionado, después de haberse almacenado el anterior contenido del ACU 1 en el ACU 2 y de haber puesto el ACU 1 a "0".

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L EB10	//Cargar byte de entrada EB10 en el ACU1-L-L.
L MB120	//Cargar byte de marcas MB120 en el ACU1-L-L.
L DBB12	//Cargar byte de datos DBB12 en el ACU1-L-L.
L DIW15	//Cargar palabra de datos de instancia DIW15 en el ACU1-L.
L LD252	//Cargar doble palabra de datos locales LD252 en el ACU 1.
L P# E 8.7	//Cargar puntero en ACU1
L OTTO	//Cargar parámetro "OTTO" en ACU1
L P# ANNA	//Cargar puntero en el parámetro indicado en el ACU1 (Este comando carga //el offset de direcciones relativo del parámetro indicado. Para calcular //en FBs aptos para multiinstancia el offset absoluto en el bloque de //datos de instancia, se tiene que sumar a este valor el contenido del //registro AR2.

## Contenido del ACU 1

Contenido	ACU1-H-H	ACU1-H-L	ACU1-L-H	ACU1-L-L
antes de ejecutar la operación de carga	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
después de ejecutar <b>L MB10</b> (L <byte>)	00000000	00000000	00000000	<MB10>
después de ejecutar <b>L MW10</b> (L <palabra>)	00000000	00000000	<MB10>	<MB11>
después de ejecutar <b>L MD10</b> (L <doble palabra>)	<MB10>	<MB11>	<MB12>	<MB13>
después de ejecutar <b>L P# ANNA</b> (en el FB)	<86>	<Offset del bit de ANNA relativo al inicio del FB> Para calcular en FBs aptos para multiinstancia el offset absoluto en el bloque de datos de instancia, se tiene que sumar a este valor el contenido del registro AR2.		
después de ejecutar <b>L P# ANNA</b> (en la FC)	<una dirección interárea del dato que se transfiere a ANNA>			
	X = "1" o "0"			

## 9.3 L STW Cargar palabra de estado en ACU 1

### Formato

L STW

### Descripción de la operación

L STW (Operación L con el operando STW) carga el ACU 1 con el contenido de la palabra de estado. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

#### Nota

En el caso de las CPUs de la serie S7-300, los bits de la palabra de estado /ER, STA y OR no se cargan mediante la instrucción L STW. Tan sólo los bits 1, 4, 5, 6, 7 y 8 se cargan en los bits de la palabra baja contenida en el ACU 1.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L STW	//Cargar el contenido de la palabra de estado en el ACU 1.

El contenido del ACU 1 después de ejecutar L STW es el siguiente:

Bit	31-9	8	7	6	5	4	3	2	1	0
Contenido:	0	RB	A1	A0	OV	OS	OR	STA	RLO	/ER

## 9.4 LAR1 Cargar registro de direcciones 1 con contenido del ACU 1

### Formato

LAR1

### Descripción de la operación

**LAR1** carga el registro de direcciones AR 1 con el contenido del ACU 1 (puntero de 32 bits). ACU 1 y ACU 2 no se alteran. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado

### Palabra de estado

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

## 9.5 LAR1 <D> Cargar registro de direcciones 1 con puntero (formato de 32 bits)

### Formato

LAR1 &lt;D&gt;

Operando	Tipo de datos	Área de memoria	Dirección fuente
<D>	DWORD Constante de puntero	D, M, L	0...65532

### Descripción de la operación

LAR1 <D> carga el registro de direcciones AR 1 con el contenido de la doble palabra <D> direccionada o de una constante de puntero. ACU 1 y ACU 2 no se alteran. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

### Ejemplos: Direcciones directas

AWL	Explicación
LAR1 DBD 20	//Cargar AR 1 con el puntero en la doble palabra de datos DBD20.
LAR1 DID 30	//Cargar AR1 con el puntero en la doble palabra de instancia DID30.
LAR1 LD 180	//Cargar AR1 con el puntero en la doble palabra de datos locales LD180.
LAR1 MD 24	//Cargar AR1 con el puntero en la doble palabra de marcas MD24.

### Ejemplo: Constante de puntero

AWL	Explicación
LAR1 P#M100.0	//Cargar AR 1 con una constante de puntero de 32 bits.

## 9.6 LAR1 AR2 Cargar registro de direcciones 1 con contenido del registro de direcciones 2

### Formato

LAR1 AR2

### Descripción de la operación

**LAR1 AR2** (Operación LAR1 con el operando AR2) carga el registro de direcciones AR 1 con el contenido del registro de direcciones 2. ACU 1 y ACU 2 no se alteran. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

## 9.7 LAR2 Cargar registro de direcciones 2 con contenido del ACU 1

### Formato

LAR2

### Descripción de la operación

**LAR2** carga el registro de direcciones AR 2 con el contenido del ACU 1 (puntero de 32 bits).

ACU 1 y ACU 2 no se alteran. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

## 9.8 LAR2 <D> Cargar registro de direcciones 2 con puntero (formato de 32 bits)

### Formato

LAR2 <D>

Operando	Tipo de datos	Área de memoria	Dirección fuente
<D>	DWORD Constante de puntero	D, M, L	0...65532

### Descripción de la operación

LAR2 <D> carga el registro de direcciones AR 2 con el contenido de la doble palabra <D> direccionada o de una constante de puntero. ACU 1 y ACU 2 no se alteran. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

### Ejemplos: Direcciones directas

AWL	Explicación
LAR2 DBD 20	//Cargar AR 2 con el puntero en la doble palabra de datos DBD20.
LAR2 DID 30	//Cargar AR 2 con el puntero en la doble palabra de instancia DID30.
LAR2 LD 180	//Cargar AR 2 con el puntero en la doble palabra de datos locales LD180.
LAR2 MD 24	//Cargar AR 2 con el contenido de la doble palabra de marcas MD24 //direccionada directamente.

### Ejemplo: Constante de puntero

AWL	Explicación
LAR2 P#M100.0	//Cargar AR 2 con una constante de puntero de 32 bits.

## 9.9 T Transferir

### Formato

T <operando>

Operando	Tipo de datos	Área de memoria	Dirección fuente
<operando>	BYTE	E, A, PA, M, L, D	0...65535
	WORD		0...65534
	DWORD		0...65532

### Descripción de la operación

T <operando> transfiere (copia) el contenido del ACU 1 a la dirección de destino si está conectado el Master Control Relay (MCR = 1). Si el MCR es 0, en la dirección de destino se escribe el valor "0". El número de bytes que se copia del ACU 1 dependerá del tamaño indicado en la dirección de destino. El ACU 1 también almacena los datos después de la operación de transferencia. La operación de transferencia a un área de periferia directa (área de memoria PA) también transfiere el contenido del ACU 1 ó "0" (si el MCR es 0) a la dirección correspondiente en la imagen del proceso de las salidas (área de memoria A). La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
T AB10	//Transferir el contenido del ACU1-L-L al byte de salida AB10.
T MW14	//Transferir el contenido del ACU1-L a la palabra de marcas MW14.
T DBD2	//Transferir el contenido del ACU 1 a la doble palabra de datos DBD2.

## 9.10 T STW Transferir ACU 1 a la palabra de estado

### Formato

T STW

### Descripción de la operación

T STW (Operación T con el operando STW) transfiere los bits 0 a 8 del ACU 1 a la palabra de estado.

La operación se ejecuta si tener en cuenta los bits de la palabra de estado.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
T STW	//Transferir los bits 0 a 8 del ACU 1 a la palabra de estado.

Los bits del ACU 1 contienen los siguientes bits de la palabra de estado:

Bit	31-9	8	7	6	5	4	3	2	1	0
Contenido:	*)	RB	A1	A0	OV	OS	OR	STA	RLO	/ER

\*) bits que no se transfieren.

## 9.11 TAR Intercambiar registro de direcciones 1 y registro de direcciones 2

### Formato

TAR

### Descripción de la operación

**TAR** (Intercambiar registros de direcciones) intercambia los contenidos de los registros de direcciones AR 1 y AR 2. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

El contenido del registro de direcciones AR 1 se desplaza al registro de direcciones AR 2, y el contenido del registro de direcciones AR 2 se desplaza al registro de direcciones AR 1.

### Palabra de estado

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

## 9.12 TAR1 Transferir registro de direcciones 1 a ACU 1

### Formato

TAR1

### Descripción de la operación

**TAR1** transfiere el contenido de AR 1 al ACU 1 (puntero de 32 bits). El contenido del ACU 1 fue almacenado anteriormente en el ACU 2. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

## 9.13 TAR1 <D> Transferir registro de direcciones 1 a dirección de destino (puntero de 32 bits)

### Formato

TAR1 &lt;D&gt;

Operando	Tipo de datos	Área de memoria	Dirección fuente
<D>	DWORD	D, M, L	0...65532

### Descripción de la operación

**TAR1 <D>** transfiere el contenido del registro de direcciones AR 1 a la doble palabra <D> direccionada. Las áreas de destino posibles son: dobles palabras de marcas (MD), dobles palabras de datos locales (LD), dobles palabras de datos (DBD) y dobles palabras de instancia (DID).

ACU 1 y ACU 2 no se alteran. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

### Ejemplos

AWL	Explicación	
TAR1 DBD20	//Transferir el contenido de AR 1 a la doble palabra de datos DBD20.	
TAR1 DID30	//Transferir el contenido de AR 1 a la doble palabra de instancia DID30.	
TAR1 LD18	//Transferir el contenido de AR 1 a la doble palabra de datos locales LD18.	
TAR1 MD24	//Transferir el contenido de AR 1 a la doble palabra de marcas MD24.	

## 9.14 TAR1 AR2 Transferir registro de direcciones 1 a registro de direcciones 2

### Formato

TAR1 AR2

### Descripción de la operación

**TAR1 AR2** (Operación TAR1 con el operando AR2) transfiere el contenido del registro de direcciones AR 1 al registro de direcciones AR 2.

ACU 1 y ACU 2 no se alteran. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

## 9.15 TAR2 Transferir registro de direcciones 2 a ACU 1

### Formato

TAR2

### Descripción de la operación

**TAR2** transfiere el contenido del registro de direcciones AR 2 al ACU 1 (puntero de 32 bits). La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

## 9.16 TAR2 <D> Transferir registro de direcciones 2 a dirección de destino (puntero de 32 bits)

### Formato

TAR2 <D>

Operando	Tipo de datos	Área de memoria	Dirección fuente
<D>	DWORD	D, M, L	0...65532

### Descripción de la operación

**TAR2 <D>** transfiere el contenido del registro de direcciones AR 2 a la doble palabra <D> direccionada. Las áreas de destino posibles son: dobles palabras de marcas (MD), dobles palabras de datos locales (LD), dobles palabras de datos (DBD) y dobles palabras de instancia (DID).

ACU 1 y ACU 2 no se alteran. La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

### Ejemplos

AWL	Explicación	
TAR2 DBD20	//Transferir el contenido de AR 2 a la doble palabra de datos DBD20.	
TAR2 DID30	//Transferir el contenido de AR 2 a la doble palabra de instancia DID30.	
TAR2 LD18	//Transferir el contenido de AR 2 a la doble palabra de datos locales LD18.	
TAR2 MD24	//Transferir el contenido de AR 2 a la doble palabra de marcas MD24.	

# 10 Control de programa

## 10.1 Lista de operaciones de control del programa

### Descripción

Se dispone de las operaciones de control del programa siguientes:

- BE Fin de bloque
- BEB Fin de bloque condicionado
- BEA Fin de bloque incondicionado
- CALL Llamada
- CC Llamada condicionada
- UC Llamada incondicionada
- 
- Llamar a un FB
- Llamar a una FC
- Llamar a un SFB
- Llamar a una SFC
- Llamar a una multiinstancia
- Llamar a un bloque de una librería
  
- EI MCR (Master Control Relay)
- Notas importantes sobre el uso de la función MCR
- MCR( Almacenar el RLO en pila MCR, inicio área MCR
- )MCR Fin área MCR
- MCRA Activar área MCR
- MCRD Desactivar área MCR

## 10.2 BE Fin de bloque

### Formato

BE

### Descripción de la operación

**BE** (Fin de bloque) interrumpe la ejecución del programa en el bloque actual y salta al bloque que el bloque actual ha llamado. La ejecución del programa continúa con la primera instrucción después de haberse efectuado la llamada al bloque. Se libera el área de datos locales actual, y el anterior área de datos locales se convierte en el área actual. Los bloques de datos que estaban abiertos en el momento de llamar al bloque se vuelven a abrir. Adicionalmente se restablece la dependencia del bloque que efectúa la llamada con respecto al MCR y el RLO se transfiere desde el bloque actual al bloque que efectúa la llamada. Para que se realice la operación BE no hay que cumplir ningún tipo de condiciones previas. Si se omite ("se salta") la operación BE, la ejecución actual del programa no termina, sino que continúa en la meta del salto, dentro del bloque.

La operación BE de S7 no es idéntica a la del software de S5. Con el hardware de S7, la operación BE cumple la misma función que la operación BEA.

### Palabra de estado

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

### Ejemplo

AWL	Explicación	
U	E 1.0	
SPB	NEXT	//Saltar a la meta NEXT, si el RLO es 1 (E 1.0 = 1).
L	EW4	//Continuar aquí si no se ejecuta el salto.
T	EW10	
U	E 6.0	
U	E 6.1	
S	M	
	12.0	
BE		//Fin de bloque.
NEXT:	NOP 0	//Continuar aquí si se ejecuta el salto.

## 10.3 BEB Fin de bloque condicionado

### Formato

**BEB**

### Descripción de la operación

Si el RLO es 1, la operación **BEB** (Fin de bloque condicionado) interrumpe la ejecución del programa en el bloque actual y salta al bloque que el bloque actual ha llamado. La ejecución del programa continúa con la primera instrucción después de ejecutar la llamada al bloque. Se libera el área de datos locales actual, y el anterior área de datos locales se convierte en el área actual. Los bloques de datos que estaban abiertos en el momento de llamar al bloque se vuelven a abrir. Se restablece la dependencia del bloque que efectúa la llamada con respecto al MCR.

El RLO (= 1) es transferido desde el bloque que se ha terminado de ejecutar al bloque que ha efectuado la llamada. Si el RLO es 0, la operación BEB no se ejecuta. El RLO se pone entonces a "1" y la ejecución del programa continúa en la instrucción siguiente.

### Palabra de estado

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

### Ejemplo

AWL		Explicación
U	E 1.0	//Actualizar el RLO.
BEB		//Finalizar el bloque si el RLO es 1.
L	EW4	//Continuar aquí si la instrucción BEB no se ejecuta (RLO = 0).
T	MW10	

## 10.4 BEA Fin de bloque incondicionado

### Formato

BEA

### Descripción de la operación

**BEA** (Fin de bloque incondicionado) interrumpe la ejecución del programa en el bloque actual y salta al bloque que el bloque actual ha llamado. La ejecución del programa continúa con la primera instrucción después de ejecutar la llamada al bloque. Se libera el área de datos locales actual, y el anterior área de datos locales se convierte en el área actual. Los bloques de datos que estaban abiertos en el momento de llamar al bloque se vuelven a abrir. Adicionalmente se restablece la dependencia del bloque que efectúa la llamada con respecto al MCR y el RLO se transfiere desde el bloque actual al bloque que efectúa la llamada. Para que se realice la operación BEA no se tienen que cumplir previamente condiciones de ningún tipo. Si se omite ("se salta") la operación BEA, la ejecución del programa actual no finaliza, sino que continúa en la meta del salto, dentro del bloque.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
U E 1.0	
SPB NEXT	//Saltar a la meta NEXT, si el RLO es 1 (E 1.0 = 1).
L EW4	//Continuar aquí si no se ejecuta el salto.
T EW10	
U E 6.0	
U E 6.1	
S M 12.0	
BEA	//Fin de bloque incondicionado.
NEXT: NOP 0	//Continuar aquí si se ejecuta el salto.

## 10.5 CALL Llamada

### Formato

**CALL <identificación del bloque lógico>**

### Descripción de la operación

**CALL <identificación del bloque lógico>** sirve para llamar tanto a funciones (FC) y a bloques de función (FB), como para llamar a funciones de sistema (SFC) y a bloques de función del sistema (SFB) suministrados por Siemens. La operación CALL llama a la FC/SFC o al FB/SFB que se indica como operando, independientemente de cuál sea el RLO y de condiciones de cualquier otro tipo. Cuando se utiliza CALL para llamar a un FB o a un SFB hay que asignar un bloque de datos de instancia al FB/SFB. Tras editar el bloque llamado el programa continúa la ejecución con el programa del bloque que efectúa la llamada. La identificación del bloque lógico puede indicarse tanto de forma absoluta como simbólica. El contenido de los registros se vuelven a restaurar después de ejecutar la llamada al SFB/a la SFC.

### Ejemplo: CALL FB1, DB1 ó CALL FILLVAT1, RECIPE1

Bloque lógico	Tipo de bloque	Sintaxis para la llamada (dirección absoluta)
FC	Función	CALL FCn
SFC	Función de sistema	CALL SFCn
FB	Bloque de función	CALL FBn1,DBn2
SFB	Bloque de función de sistema	CALL SFBn1,DBn2

### Nota

Si se está utilizando el editor de AWL, los datos de la tabla anterior, n, n1 ó n2, deben indicar bloques válidos que ya existan. Si se quieren emplear nombres simbólicos hay que definirlos previamente.

### Transferir parámetros (utilícese para ello el editor incremental)

El bloque que efectúa la llamada puede intercambiar parámetros con el bloque llamado a través de la lista de variables. La lista de variables se agrega automáticamente al programa AWL en cuestión en cuanto se introduzca correctamente una instrucción CALL.

Si se llama a un FB/SFB o a una FC/SFC, y la tabla de declaración de variables del bloque que se ha llamado tiene declaraciones del tipo IN, OUT e IN\_OUT, dichas variables se añaden al programa del bloque que efectúa la llamada en forma de lista de parámetros formales.

Al llamar a las FC y SFC, a los parámetros formales se les tiene que asignar los parámetros actuales del bloque lógico que efectúa la llamada.

Al llamar a los FB y SFB sólo tienen que indicarse los parámetros actuales que vayan a modificarse con respecto a la última llamada, ya que los parámetros actuales se almacenan en el DB de instancia después de que se haya ejecutado el FB. Si el parámetro actual es un DB, se tiene que indicar siempre la dirección absoluta y completa, p.ej. DB1,DBW2.

Los parámetros IN pueden indicarse en forma de constantes o de direcciones, ya sean éstas absolutas o simbólicas. Los parámetros OUT e IN\_OUT tienen que indicarse en forma de direcciones absolutas o simbólicas. Hay que asegurarse de que todas las direcciones y constantes sean compatibles con los tipos de datos que se transfieren.

La operación CALL almacena en la pila BSTACK la dirección de retorno (selector y dirección relativa), los selectores de los dos bloques de datos que están abiertos y el bit MA. Además, la operación desactiva la dependencia respecto al MCR y crea el área de datos locales del bloque a llamar.

### Palabra de estado

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

### Ejemplo 1: Asignar parámetros a la llamada de la función FC6

```
CALL    FC6
        Parámetros formales  Parámetros actuales
        NO OF TOOL           := MW100
        TIME OUT             := MW110
        FOUND                 := A 0.1
        ERROR                 := A 100.0
```

### Ejemplo 2: Llamar a una SFC sin parámetros

AWL	Explicación
CALL SFC43	//Llamar a la SFC43 para volver a arrancar la vigilancia de tiempo //(sin parámetros).

### Ejemplo 3: Llamar al FB99 con el bloque de datos de instancia DB1

```
CALL    FB99,DB1
        Parámetros formales      Parámetros actuales
MAX_RPM := #RPM1_MAX
MIN_RPM := #RPM2
MAX_POWER := #POWER
MAX_TEMP := #TEMP
```

### Ejemplo 4: Llamar al FB99 con el bloque de datos de instancia DB2

```
CALL    FB99,DB2
        Parámetros formales      Parámetros actuales
MAX_RPM := #RPM3_MAX
MIN_RPM := #RPM2
MAX_POWER := #POWER1
MAX_TEMP := #TEMP
```

---

#### Nota

Antes de llamar a un FB o a un SFB tiene que existir ya el correspondiente DB de instancia. En el ejemplo anterior, los bloques DB1 y DB2 tienen que estar creados ya antes llamar al FB99.

---

## 10.6 Llamar a un FB

### Formato

**CALL FB n1, DB n1**

### Descripción

Esta operación permite llamar bloques de función creados por el usuario (FBs). La operación CALL llama el FB indicado como operando, independientemente del RLO o de cualquier otra condición. Si llama un FB con la operación CALL, tendrá que asignarle un bloque de datos de instancia. Una vez procesado el bloque invocado, el programa del bloque invocante seguirá procesándose. La identificación del bloque lógico puede indicarse de forma absoluta o simbólica.

### Transferir parámetros (para ello trabaje con el modo incremental)

El bloque invocante puede intercambiar parámetros con el bloque invocado mediante la tabla de variables. Dicha tabla de variables se actualiza de forma automática en su programa AWL al introducir una instrucción CALL válida.

Si llama un FB y la tabla de declaración de variables del bloque invocado dispone de declaraciones del tipo IN, OUT e IN\_OUT, estas variables se actualizarán en el programa del bloque invocante como tabla de parámetros formales.

Al llamar los FBs sólo tiene que introducir los parámetros actuales que se hayan modificado respecto a la última llamada, ya que los parámetros actuales se han guardado en el DB de instancia una vez procesado el FB. Si el parámetro actual es un DB, se debe indicar siempre la dirección absoluta de forma completa, p.ej. DB1, DBW2.

Los parámetros IN se pueden introducir como constantes o direcciones absolutas o simbólicas. Los parámetros OUT e IN\_OUT tienen que introducirse como direcciones absolutas o simbólicas. Vigile que todas las direcciones y constantes sean compatibles con los tipos de datos que se vayan a transferir.

La operación CALL guarda la dirección de retorno (selector y dirección relativa), los selectores de los dos bloques de datos abiertos y el bit MA en la pila BSTACK. Además la operación desactiva la dependencia MCR y crea el área de datos locales del bloque que debe ser llamado.

### Palabra de estado

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

### Ejemplo 1: Llamar el FB99 con el bloque de datos de instancia DB1

```
CALL    FB99 ,DB1
        Parámetros formales      Parámetros actuales
        MAX_RPM                  := #RPM1_MAX
        MIN_RPM                   := #RPM1
        MAX_POWER                 := #POWER1
        MAX_TEMP                  := #TEMP1
```

### Ejemplo 2: Llamar el FB99 con el bloque de datos de instancia DB2

```
CALL    FB99 ,DB2
        Parámetros formales      Parámetros actuales
        MAX_RPM                  := #RPM2_MAX
        MIN_RPM                   := #RPM2
        MAX_POWER                 := #POWER2
        MAX_TEMP                  := #TEMP2
```

---

#### Nota

Toda llamada a un FB tiene que disponer de un bloque de datos de instancia. En el ejemplo anterior tienen que existir los bloques DB1 y DB2 antes de la llamada.

---

## 10.7 Llamar a una FC

### Formato

CALL FC n

---

### Nota

Si trabaja con el editor de AWL, la indicación (n) tendrá que referirse a bloques válidos ya existentes. Los nombres simbólicos también se deberán haber definido previamente.

---

### Descripción

Esta operación permite llamar funciones (FCs). La operación CALL llama la FC indicada como operando, independientemente del RLO o de cualquier otra condición. Una vez procesado el bloque invocado, el programa del bloque invocante seguirá procesándose. La identificación del bloque lógico puede indicarse de forma absoluta o simbólica.

### Transferir parámetros (para ello trabaje con el modo incremental)

El bloque invocante puede intercambiar parámetros con el bloque invocado mediante la tabla de variables. Dicha tabla de variables se actualiza de forma automática en su programa AWL al introducir una instrucción CALL válida.

Si llama una FC y la tabla de declaración de variables del bloque invocado dispone de declaraciones del tipo IN, OUT e IN\_OUT, estas variables se actualizarán en el programa del bloque invocante como tabla de parámetros formales.

Al llamar las FCs tiene que asignar parámetros actuales del bloque lógico invocante a los parámetros formales.

Los parámetros IN se pueden introducir como constantes o direcciones absolutas o simbólicas. Los parámetros OUT e IN\_OUT tienen que introducirse como direcciones absolutas o simbólicas. Vigile que todas las direcciones y constantes sean compatibles con los tipos de datos que se vayan a transferir.

La operación CALL guarda la dirección de retorno (selector y dirección relativa), los selectores de los dos bloques de datos abiertos y el bit MA en la pila BSTACK. Además la operación desactiva la dependencia MCR y crea el área de datos locales del bloque que debe ser llamado.

### Palabra de estado

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

### **Ejemplo: Asignar parámetros a la llamada de la función FC6**

<b>CALL</b>	<b>FC6</b>	
	Parámetros formales	Parámetros actuales
	<b>NO OF TOOL</b>	<b>:= MW100</b>
	<b>TIME OUT</b>	<b>:= MW110</b>
	<b>FOUND</b>	<b>:= A 0.1</b>
	<b>ERROR</b>	<b>:= A 100.0</b>

## 10.8 Llamar a un SFB

### Formato

**CALL SFB n1, DB n2**

### Descripción

Esta operación permite llamar bloques de función del sistema (SFBs) suministrados por Siemens. La operación CALL llama el SFB indicado como operando, independientemente del RLO o de cualquier otra condición. Si llama un SFB con la operación CALL, tendrá que asignarle un bloque de datos de instancia. Una vez procesado el bloque invocado, el programa del bloque invocante seguirá procesándose. La identificación del bloque lógico puede indicarse de forma absoluta o simbólica.

### Transferir parámetros (para ello trabaje con el modo incremental)

El bloque invocante puede intercambiar parámetros con el bloque invocado mediante la tabla de variables. Dicha tabla de variables se actualiza de forma automática en su programa AWL al introducir una instrucción CALL válida.

Si llama un SFB y la tabla de declaración de variables del bloque invocado dispone de declaraciones del tipo IN, OUT e IN\_OUT, estas variables se actualizarán en el programa del bloque invocante como tabla de parámetros formales.

Al llamar los SFBs sólo tiene que introducir los parámetros actuales que se hayan modificado respecto a la última llamada, ya que los parámetros actuales se han guardado en el DB de instancia una vez procesado el SFB. Si el parámetro actual es un DB, se debe indicar siempre la dirección absoluta de forma completa, p.ej. DB1, DBW2.

Los parámetros IN se pueden introducir como constantes o direcciones absolutas o simbólicas. Los parámetros OUT e IN\_OUT tienen que introducirse como direcciones absolutas o simbólicas. Vigile que todas las direcciones y constantes sean compatibles con los tipos de datos que se vayan a transferir.

La operación CALL guarda la dirección de retorno (selector y dirección relativa), los selectores de los dos bloques de datos abiertos y el bit MA en la pila BSTACK. Además la operación desactiva la dependencia MCR y crea el área de datos locales del bloque que debe ser llamado.

### Palabra de estado

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

## Ejemplo

<b>CALL</b>	<b>SFB4 ,DB4</b>	
	Parámetros formales	Parámetros actuales
	<b>IN:</b>	<b>E0 . 1</b>
	<b>PT:</b>	<b>T#20s</b>
	<b>Q:</b>	<b>M0 . 0</b>
	<b>ET:</b>	<b>MW10</b>

---

### Nota

Toda llamada a un SFB tiene que disponer de un bloque de datos de instancia. En el ejemplo anterior tienen que existir los bloques SFB4 y DB4 antes de la llamada.

---

## 10.9 Llamar a una SFC

### Formato

**CALL SFC n**

---

#### Nota

Si trabaja con el editor de AWL, la indicación (n) tendrá que referirse a bloques válidos ya existentes. Los nombres simbólicos también se deberán haber definido previamente.

---

### Descripción

Esta operación permite llamar funciones estándar creadas por el usuario (SFCs). La operación CALL llama la SFC indicada como operando, independientemente del RLO o de cualquier otra condición. Una vez procesado el bloque invocado, el programa del bloque invocante seguirá procesándose. La identificación del bloque lógico puede indicarse de forma absoluta o simbólica.

### Transferir parámetros (para ello trabaje con el modo incremental)

El bloque invocante puede intercambiar parámetros con el bloque invocado mediante la tabla de variables. Dicha tabla de variables se actualiza de forma automática en su programa AWL al introducir una instrucción CALL válida.

Si llama una SFC y la tabla de declaración de variables del bloque invocado dispone de declaraciones del tipo IN, OUT e IN\_OUT, estas variables se actualizarán en el programa del bloque invocante como tabla de parámetros formales.

Al llamar las SFCs tiene que asignar parámetros actuales del bloque lógico invocante a los parámetros formales.

Los parámetros IN se pueden introducir como constantes o direcciones absolutas o simbólicas. Los parámetros OUT e IN\_OUT tienen que introducirse como direcciones absolutas o simbólicas. Vigile que todas las direcciones y constantes sean compatibles con los tipos de datos que se vayan a transferir.

La operación CALL guarda la dirección de retorno (selector y dirección relativa), los selectores de los dos bloques de datos abiertos y el bit MA en la pila BSTACK. Además la operación desactiva la dependencia MCR y crea el área de datos locales del bloque que debe ser llamado.

### Palabra de estado

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

### Ejemplo: Llamar a una SFC sin parámetros

AWL	Explicación
CALL SFC43	//Llama la SFC43 para arrancar de nuevo la supervisión del tiempo //(sin parámetros).

## 10.10 Llamar a una multiinstancia

### Formato

**CALL # Variablenname**

### Descripción

Una multiinstancia se genera al declarar una variable estática del tipo de datos de un bloque de función. Sólo las multiinstancias ya declaradas se listarán en el catálogo de elementos de programa.

### Palabra de estado

	<b>RB</b>	<b>A1</b>	<b>A0</b>	<b>OV</b>	<b>OS</b>	<b>OR</b>	<b>STA</b>	<b>RLO</b>	<b>/ER</b>
se escribe:	-	-	-	-	0	0	X	X	X

## 10.11 Llamar a un bloque de una librería

Se le ofrecen las librerías registradas en el Administrador SIMATIC.

De estas librerías puede seleccionar:

- los bloques que estén integrados en el sistema operativo de su CPU (librería "Standard Library"),
- los bloques que usted mismo haya depositado en librerías porque desea utilizarlos repetidas veces.

## 10.12 CC Llamada condicionada

### Formato

CC <identificación del bloque lógico>

### Descripción de la operación

**CC <identificación del bloque lógico>** (Llamada condicionada) llama a un bloque lógico del tipo FC o SFC sin parámetros cuando el RLO es 1. La operación CC es prácticamente igual que la operación CALL, con la diferencia de que aquí no se pueden transferir parámetros. La operación almacena en la pila BSTACK la dirección de retorno (selector y dirección relativa), los selectores de los dos bloques de datos actuales y el bit MA; desactiva la dependencia con respecto al MCR; crea el área de datos locales del bloque a llamar y empieza a ejecutar el bloque lógico que se ha llamado. La identificación del bloque lógico puede indicarse tanto de forma absoluta como simbólica.

### Palabra de estado

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

### Ejemplo

AWL		Explicación
U	E 2.0	//Consultar el estado de señal en la entrada E 2.0.
CC	FC6	//Llamar a la función FC6, si E 2.0 es 1.
U	M3.0	//Se ejecuta: tras regresar de la función llamada, si E 2.0 = 1, ó //inmediatamente después de la instrucción U E 2.0, si E 2.0 = 0.

### Nota

Si se utiliza la operación **CALL** para llamar a un bloque de función (FB) o a un bloque de función de sistema (SFB), en la instrucción hay que especificar un bloque de datos de instancia (n.º de DB). El uso de una variable del tipo "BlockFB" o "BlockFC" no está permitido en combinación con la operación **CC**. Puesto que no puede asignar ningún bloque de datos a una llamada con la operación **CC** en el operando de la instrucción, sólo tiene la posibilidad de utilizar esta operación para bloques sin parámetros de bloque ni datos locales estáticos.

Dependiendo de cuál sea el segmento con el que se esté trabajando, durante la compilación del lenguaje de programación Esquema de contactos al lenguaje de programación Lista de instrucciones "KOP/AWL: Programar bloques" genera en parte la operación **UC** y en parte la operación **CC**. Por ello, se recomienda utilizar por regla general la instrucción **CALL**, con el fin de que no se produzcan errores en los programas que el usuario haya creado.

## 10.13 UC Llamada incondicionada

### Formato

UC <identificación del bloque lógico>

### Descripción de la operación

**UC <identificación del bloque lógico>** (Llamada incondicionada) llama a un bloque lógico del tipo FC o SFC. La operación UC es prácticamente igual a la operación CALL, con la diferencia de que no se pueden transmitir parámetros. La operación almacena en la pila BSTACK la dirección de retorno (selector y dirección relativa), los selectores de los dos bloques de datos actuales y el bit MA; desactiva la dependencia con respecto al MCR, crea el área de datos locales del bloque a llamar y empieza a ejecutar el bloque lógico que se ha llamado.

### Palabra de estado

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

### Ejemplo 1

AWL	Explicación	
UC	FC6	//Llamar a la función FC6 (sin parámetros).

### Ejemplo 2

AWL	Explicación	
UC	SFC43	//Llamar a la función de sistema SFC43 (sin parámetros).

### Nota

Si se utiliza la operación **CALL** para llamar a un bloque de función (FB) o a un bloque de función de sistema (SFB), en la instrucción hay que especificar un bloque de datos de instancia (n.º de DB). El uso de una variable del tipo "BlockFB" o "BlockFC" no está permitido en combinación con la operación **UC**. Puesto que no puede asignar ningún bloque de datos a una llamada con la operación **UC** en el operando de la instrucción, sólo tiene la posibilidad de utilizar esta operación para bloques sin parámetros de bloque ni datos locales estáticos..

Según cuál sea el segmento con el que se trabaja, "KOP/AWL: Programar bloques" genera, durante la compilación del lenguaje de programación Esquema de contactos al lenguaje de programación Lista de instrucciones, en parte la operación **UC** y en parte la operación **CC**. Por lo general, utilice la operación **CALL** para que no se produzcan errores en los programas que usted haya creado.

## 10.14 El MCR (Master Control Relay)

Notas importantes sobre el uso de la función MCR



### Atención

Para eliminar la posibilidad de que haya peligro para las personas y/o las máquinas, no utilizar nunca el MCR para sustituir a un Master Control Relay mecánico que esté conectado permanentemente y que sirva como dispositivo de emergencia.

### Descripción

El Master Control Relay (MCR) se utiliza en los esquemas de relés para activar y desactivar el flujo de señales. Dependen del MCR las operaciones que son activadas por las siguientes operaciones lógicas con bits y de transferencia:

- = <bit>
- S <bit>
- R <bit>
- T <byte>, T <palabra>, T <palabra doble>

La operación T, que se utiliza con byte, palabra o palabra doble, escribe un "0" en la memoria si el MCR es "0". Las operaciones S y R no cambian el valor ya existente. La operación = escribe un "0" en el bit direccionado.

### Operaciones que dependen del MCR y su reacción ante el estado de señal de MCR

Estado de señal del MCR	= <bit>	S <bit>, R <bit>	T <byte>, T <palabra> T <palabra doble>
0 ("OFF")	Escribe "0". (Imita a un relé que pasa al estado de reposo en caso de fallar la alimentación.)	No escribe. (Imita a un relé que permanece en su estado actual en caso de fallar la alimentación.)	Escribe "0". (Imita a un componente que da el valor "0" en caso de fallar la alimentación.)
1 ("ON")	Ejecución normal.	Ejecución normal.	Ejecución normal.

### MCR( - Inicio área MCR / )MCR - Fin área MCR

El MCR es controlado por una pila que tiene una anchura de un bit y una profundidad de ocho bits. El MCR permanece conectado mientras las ocho entradas tengan el estado de señal "1". La operación MCR( copia el bit RLO en la pila MCR. La operación )MCR borra la última entrada de la pila y pone la posición vacante a "1". Las operaciones MCR( y )MCR siempre deben utilizarse en pareja. Si hay más de ocho operaciones MCR( consecutivas, o se intenta ejecutar una operación )MCR estando la pila vacía, se provoca un mensaje de error MCRF.

### **MCRA - Activar área MCR / MCRD - Desactivar área MCR**

Las operaciones MCRA y MCRD siempre deben utilizarse en pareja. Las instrucciones que están programadas entre MCRA y MCRD dependen del estado del bit MCR. Las operaciones que se encuentran fuera de una serie MCRA-MCRD no dependen del estado del bit MCR.

La dependencia de las funciones (FCs) y de los bloques de función (FBs) con respecto al MCR tiene que programarse dentro de las respectivas funciones/bloques. Utilícese a tal fin la operación MCRA en el bloque que se ha llamado.

## 10.15 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.
- 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:

- 1º **Desactive** el Master Control Relay con el comando MCRD **antes** de la instrucción correspondiente o antes del segmento involucrado.
- 2º **Active** nuevamente el Master Control Relay con el comando MCRA **después de** la instrucción correspondiente o después del segmento involucrado.

## 10.16 MCR( Almacenar el RLO en pila MCR, inicio área MCR

Notas importantes sobre el uso de la función MCR

### Formato

MCR(

### Descripción de la operación

**MCR(** (Abrir un área MCR) almacena el RLO en la pila MCR y abre un área MCR. Área MCR = instrucciones entre la operación **MCR(** y la correspondiente operación **)MCR**. Las operaciones **MCR(** y **)MCR** siempre tienen que utilizarse en pareja.

Si el RLO es 1, el MCR está "conectado". Las instrucciones que dependen del MCR dentro de esta área MCR se ejecutan de forma normal.

Si el RLO es 0, el MCR está "desconectado". Las instrucciones que dependen del MCR dentro de esta área MCR se ejecutan conforme a la tabla siguiente.

### Operaciones dependientes del estado del bit MCR

Estado de señal del MCR	= <bit>	S <bit>, R <bit>	T <byte>, T <palabra> T <palabra doble>
0 ("OFF")	Escribe "0". (Imita a un relé que pasa al estado de reposo en caso de fallar la alimentación.)	No escribe. (Imita a un relé que permanece en su estado actual en caso de fallar la alimentación.)	Escribe "0". (Imita a un componente que da el valor "0" en caso de fallar la alimentación.)
1 ("ON")	Ejecución normal.	Ejecución normal.	Ejecución normal.

Las operaciones MCR( y )MCR se pueden anidar. La profundidad máxima de anidamiento es de ocho operaciones; por tanto, la pila puede contener como máximo ocho entradas. Si se ejecuta la operación MCR( estando la pila llena se provoca un error de pila MCR (MCRF).

### Palabra de estado

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

**Ejemplo**

AWL	Explicación
MCRA	//Activar área MCR.
U E 1.0	
MCR(	//Almacenar el RLO en la pila MCR, abrir un área MCR. El MCR está "ON", //si el RLO es 1 (E 1.0 = 1). El MCR está "OFF", si el RLO es 0 //(E 1.0 = 0).
U E 4.0	
= A 8.0	//Si el MCR está "OFF", A 8.0 se pone a 0 sin considerar E 4.0.
L MW20	
T AW10	//Si el MCR está "OFF", el valor "0" se transfiere a AW10.
)MCR	//Finalizar el área MCR.
MCRD	//Desactivar área MCR.
U E 1.1	
= A 8.1	//Estas instrucciones están fuera del área MCR y no dependen del bit MCR.

## 10.17 )MCR Fin área MCR

Notas importantes sobre el uso de la función MCR

### Formato

)MCR

### Descripción de la operación

)MCR (Finalizar un área MCR) borra una entrada de la pila MCR y finaliza un área MCR. La última entrada de la pila MCR queda libre y se pone a "1". Las operaciones MCR( y )MCR siempre deben utilizarse en pareja. Si se ejecuta la operación )MCR estando la pila vacía se provoca un error de pila MCR (MCRF).

### Palabra de estado

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

### Ejemplo

AWL		Explicación
MCRA		//Activar área MCR.
U	E 1.0	
MCR(		//Almacenar el RLO en la pila MCR, abrir un área MCR. El MCR está "ON", //si el RLO es 1 (E 1.0 = 1). El MCR está "OFF", si el RLO es 0 //(E 1.0 = 0).
U	E 4.0	
=	A 8.0	//Si el MCR está "OFF", A 8.0 se pone a "0" sin considerar E 4.0.
L	MW20	
T	AW10	//Si el MCR está "OFF", se transfiere "0" a AW10.
)MCR		//Finalizar el área MCR.
MCRD		//Desactivar área MCR.
U	E 1.1	
=	A 8.1	//Estas instrucciones están fuera del área MCR y no dependen del bit MCR.

## 10.18 MCRA Activar área MCR

Notas importantes sobre el uso de la función MCR

### Formato

**MCRA**

### Descripción de la operación

**MCRA** (Activar el Master Control Relay) conecta la dependencia con respecto al MCR para las instrucciones que siguen a esta operación. Las operaciones MCRA (Activar el Master Control Relay) y MCRD (Desactivar el Master Control Relay) siempre deben utilizarse en pareja. Las instrucciones que estén programadas entre MCRA y MCRD dependen del estado de señal del bit MCR.

La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado

### Palabra de estado

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

### Ejemplo

AWL	Explicación
MCRA	//Activar área MCR.
U E 1.0	
MCR(	//Almacenar el RLO en la pila MCR, abrir un área MCR. El MCR está "ON", //si el RLO es 1 (E 1.0 = 1). El MCR está "OFF", si el RLO es 0 //(E 1.0 = 0).
U E 4.0	
= A 8.0	//Si el MCR está "OFF", A 8.0 se pone a "0" sin considerar E 4.0.
L MW20	
T AW10	//Si el MCR está "OFF", se transfiere "0" a AW10.
)MCR	//Finalizar el área MCR.
MCRD	//Desactivar área MCR.
U E 1.1	
= A 8.1	//Estas instrucciones están fuera del área MCR y no dependen del bit MCR.

## 10.19 MCRD Desactivar área MCR

Notas importantes sobre el uso de la función MCR

### Formato

**MCRD**

### Descripción de la operación

**MCRD** (Desactivar el Master Control Relay) desconecta la dependencia MCR para las instrucciones que siguen a esta operación. Las operaciones MCRD (Desactivar el Master Control Relay) y MCRA (Activar el Master Control Relay) siempre deben utilizarse en pareja. Las instrucciones que estén programadas entre MCRA y MCRD dependen del estado de señal del bit MCR.

La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado

### Palabra de estado

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

### Ejemplo

AWL	Explicación
MCRA	//Activar área MCR.
U E 1.0	
MCR(	//Almacenar el RLO en la pila MCR, abrir un área MCR. El MCR está "ON",
	//si el RLO es 1 (E 1.0 = 1). El MCR está "OFF", si el RLO es 0
	//(E 1.0 = 0).
U E 4.0	
= A 8.0	//Si el MCR está "OFF", A 8.0 se pone a "0" sin considerar E 4.0.
L MW20	
T AW10	//Si el MCR está "OFF", se transfiere "0" a AW10.
)MCR	//Finalizar el área MCR.
MCRD	//Desactivar área MCR.
U E 1.1	
= A 8.1	//Estas instrucciones están fuera del área MCR y no dependen del bit MCR.



# 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 el contenido de la palabra baja del ACU 1 o de todo el acumulador bit por bit a la izquierda o a la derecha (v. Registros de la CPU). Un desplazamiento a la izquierda en  $n$  bits multiplica el contenido del acumulador por  $2^n$ , mientras que un desplazamiento a la derecha en  $n$  bits divide el contenido del acumulador por  $2^n$ . Desplazando, por ejemplo, el equivalente binario del valor decimal "3" tres bits a la izquierda, en el acumulador resulta el equivalente binario del valor decimal 24. Desplazando el equivalente binario del valor decimal "16" dos bits a la derecha, resulta el equivalente binario del valor decimal 4 en el acumulador.

El número que sigue a la operación de desplazamiento o bien el valor depositado en el byte bajo de la palabra baja del ACU 2 indica el número de bits desplazados, es decir, el número de posiciones en que se desplaza a la izquierda o derecha. Las posiciones que quedan vacantes como consecuencia de la operación de desplazamiento se rellenan con ceros o con el estado de señal del bit que indica el signo ("0" = positivo y "1" = negativo). El 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 evaluar el bit A1 se utilizan las operaciones de salto.

Las operaciones de desplazamiento son absolutas, es decir, que su ejecución no depende de ninguna condición especial. Además no afectan al resultado lógico.

Se dispone de las operaciones de desplazamiento siguientes:

- SSI Desplazar signo de número entero a la derecha (16 bits)
- SSD Desplazar signo de número entero a la derecha (32 bits)
- SLW Desplazar palabra a la izquierda (16 bits)
- SRW Desplazar palabra a la derecha (16 bits)
- SLD Desplazar doble palabra a la izquierda (32 bits)
- SRD Desplazar doble palabra a la derecha (32 bits)

### 11.1.2 SSI Desplazar signo de número entero a la derecha (16 bits)

#### Formato

**SSI**

**SSI <número>**

Operando	Tipo de datos	Descripción
<número>	Entero, sin signo	Número de posiciones de bit a desplazar; margen de 0 a

#### Descripción de la operación

**SSI** (Desplazar signo de número entero a la derecha) sólo desplaza el contenido del ACU1-L bit por bit a la derecha. En las posiciones de bit que quedan libres por el desplazamiento se escribe el estado de señal del bit de signo (bit 15). El último bit desplazado se carga en el bit A1 de la palabra de estado. El número de las posiciones de bit a desplazar viene indicado por el operando <número> o por un valor en el ACU2-L-L.

**SSI <número>**: El operando <número> indica el número de desplazamiento. Se admiten valores entre 0 y 15. Los bits A0 y OV de la palabra de estado se ponen a "0", si <número> es mayor que cero. Si <número> es igual a "0", la operación de rotación se procesa igual que una operación NOP.

**SSI**: El número de desplazamiento viene indicado por el valor en el ACU2-L-L. Se admiten valores entre 0 y 255. Un número de desplazamiento > 16 siempre provoca el mismo resultado: (ACU 1 = 16#0000, A1 = 0 ó ACU 1 = 16#FFFF, A1 = 1. Si el número de desplazamiento > 0, los bits A0 y OV de la palabra de estado se ponen a "0". Si el número de desplazamiento es "0", la operación de desplazamiento se procesa igual que una operación NOP.

#### Palabra de estado

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

#### Ejemplos

Contenido	ACU1-H				ACU1-L			
Bit	31 ...	..	..	... 16	15 ...	..	..	... 0
antes de ejecutar <b>SSD 6</b>	0101	1111	0110	0100	1001	1101	0011	1011
después de ejecutar <b>SSD 6</b>	0101	1111	0110	0100	1111	1110	0111	0100

### Ejemplo 1

AWL		Explicación
L	MW4	//Cargar el valor en el ACU 1.
SSI	6	//Desplazar los bits en el ACU 1 con el signo correcto 6 posiciones a la //derecha.
T	MW8	//Transferir el resultado a MW8.

### Ejemplo 2

AWL		Explicación
L	+3	//Cargar el valor +3 en el ACU 1.
L	MW20	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MW20 en //el ACU 1.
SSI		//El número de desplazamiento es el valor del ACU2-L-L. => Desplazar //los bits en el ACU 1-L con el signo correcto, 3 posiciones a la //derecha, poner las posiciones libres al estado de señal del bit de //signo.
SPP	NEXT	//Saltar a la meta NEXT, si el último bit desplazado (A1) es 1.

### 11.1.3 SSD Desplazar signo de número entero a la derecha (32 bits)

#### Formato

**SSD**

**SSD <número>**

Operando	Tipo de datos	Descripción
<número>	Entero, sin signo	Número de posiciones de bit a desplazar; margen de 0 a 32

#### Descripción de la operación

**SSD** (Desplazar signo de número entero de 32 bits a la derecha) desplaza el contenido completo del ACU 1 bit por bit a la derecha. En las posiciones de bit que quedan libres por el desplazamiento se escribe el estado de señal del bit de signo. El último bit desplazado se carga en el bit A1 de la palabra de estado. El número de las posiciones de bit a desplazar viene indicado por el operando <número> o por un valor en el ACU2-L-L.

**SSD <número>**: El operando <número> indica el número de desplazamiento. Se admiten valores entre 0 y 32. Los bits A0 y OV de la palabra de estado se ponen a "0" si <número> es mayor que cero. Si <número> es igual a "0", la operación de rotación se procesa igual que una operación NOP.

**SSD**: El número de desplazamiento viene indicado por el valor en el ACU2-L-L. Se admiten valores entre 0 y 255. Un número de desplazamiento > 32 siempre provoca el mismo resultado: ACU 1 = 32#00000000, A1 = 0 ó ACU 1 = 32#FFFFFF, A1 = 1. Si el número de desplazamiento > 0, los bits A0 y OV de la palabra de estado se ponen a "0". Si el número de desplazamiento es "0", la operación de desplazamiento se procesa igual que una operación NOP.

#### Palabra de estado

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

## Ejemplos

Contenido	ACU1-H				ACU1-L			
	31 ...	..	..	... 16	15 ...	..	..	... 0
antes de ejecutar <b>SSD 7</b>	1000	1111	0110	0100	0101	1101	0011	1011
después de ejecutar <b>SSD 7</b>	1111	1111	0001	1110	1100	1000	1011	1010

### Ejemplo 1

AWL		Explicación
L	MD4	//Cargar el valor en el ACU 1.
SSD	7	//Desplazar los bits en el ACU 1 con el signo correcto, 7 posiciones a la derecha.
T	MD8	//Transferir el resultado a MD8.

### Ejemplo 2

AWL		Explicación
L	+3	//Cargar el valor +3 en el ACU 1.
L	MD20	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD20 en //el ACU 1.
SSD		//El número de desplazamiento es el valor del ACU2-L-L. => Desplazar //los bits en el ACU 1 con el signo correcto, 3 posiciones a la derecha, //poner las posiciones libres al estado de señal del bit de signo.
SPP	NEXT	//Saltar a la meta NEXT, si el último bit desplazado (A1) es 1.

### 11.1.4 SLW Desplazar palabra a la izquierda (16 bits)

#### Formato

**SLW**

**SLW <número>**

Operando	Tipo de datos	Descripción
<número>	Entero, sin signo	Número de posiciones de bit a desplazar; margen de 0 a 15

#### Descripción de la operación

**SLW** (Desplazar palabra a la izquierda) sólo desplaza el contenido del ACU1-L bit por bit a la izquierda. En las posiciones de bit que quedan libres por el desplazamiento se escriben ceros. El último bit desplazado se carga en el bit A1 de la palabra de estado. El número de las posiciones de bit a desplazar viene indicado por el operando <número> o por un valor en el ACU2-L-L.

**SLW <número>**: El operando <número> indica el número de desplazamiento. Se admiten valores entre 0 y 15. Los bits A0 y OV de la palabra de estado se ponen a "0" si <número> es mayor que cero. Si <número> es igual a "0" la operación de rotación se procesa igual que una operación NOP.

**SLW**: El número de desplazamiento viene indicado por el valor en el ACU2-L-L. Un número de desplazamiento > 16 siempre provoca el mismo resultado: ACU 1-L = 0, A1 = 0, A0 = 0, OV = 0. Si 0 < número de desplazamiento <= 16, los bits A0 y OV de la palabra de estado se ponen a "0". Si el número de desplazamiento es "0", la operación de desplazamiento se procesa igual que una operación NOP.

#### Palabra de estado

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

## Ejemplos

Contenido	ACU1-H				ACU1-L			
	31 ...	..	..	... 16	15 ...	..	..	... 0
antes de ejecutar <b>SLW 5</b>	0101	1111	0110	0100	0101	1101	0011	1011
después de ejecutar <b>SLW 5</b>	1111	0110	0100	0101	1101	0011	1011	0101

### Ejemplo 1

AWL		Explicación
L	MW4	//Cargar el valor en el ACU 1.
SLW	5	//Desplazar los bits en el ACU 1, 5 posiciones a la izquierda.
T	MW8	//Transferir el resultado a MW8.

### Ejemplo 2

AWL		Explicación
L	+3	//Cargar el valor +3 en el ACU 1.
L	MW20	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MW20 en //el ACU 1.
SLW		//El número de desplazamiento es el valor del ACU2-L-L. => Desplazar //los bits en el ACU 1-L, 3 posiciones a la izquierda.
SPP	NEXT	//Saltar a la meta NEXT, si el último bit desplazado (A1) es 1.

### 11.1.5 SRW Desplazar palabra a la derecha (16 bits)

#### Formato

**SRW**

**SRW <número>**

Operando	Tipo de datos	Descripción
<número>	Entero, sin signo	Número de posiciones de bit a desplazar; margen de 0 a 15

#### Descripción de la operación

**SRW** (Desplazar palabra a la derecha) sólo desplaza el contenido del ACU1-L bit por bit a la derecha. En las posiciones de bit que quedan libres por el desplazamiento se escriben ceros. El último bit desplazado se carga en el bit A1 de la palabra de estado. El número de las posiciones de bit a desplazar viene indicado por el operando <número> o por un valor en el ACU2-L-L.

**SRW <número>**: El operando <número> indica el número de desplazamiento. Se admiten valores entre 0 y 15. Los bits A0 y OV de la palabra de estado se ponen a "0", si <número> es mayor que cero. Si <número> es igual a "0", la operación de rotación se procesa igual que una operación NOP.

**SRW**: El número de desplazamiento viene indicado por el valor en el ACU2-L-L. Se admiten valores entre 0 y 255. Un número de desplazamiento > 16 siempre provoca el mismo resultado: ACU 1-L = 0, A1 = 0, A0 = 0, OV = 0. Si 0 < número de desplazamiento <= 16, los bits A0 y OV de la palabra de estado se ponen a "0". Si el número de desplazamiento es "0", la operación de desplazamiento se procesa igual que una operación NOP.

#### Palabra de estado

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

## Ejemplos

Contenido	ACU1-H				ACU1-L			
	31 ...	..	..	... 16	15 ...	..	..	... 0
antes de ejecutar <b>SRW 6</b>	0101	1111	0110	0100	0101	1101	0011	1011
después de ejecutar <b>SRW 6</b>	1111	0110	0100	0101	1101	0011	1011	0101

### Ejemplo 1

AWL	Explicación	
L    MW4	//Cargar el valor en el ACU 1.	
SRW    6	//Desplazar los bits en el ACU 1, 6 posiciones a la derecha.	
T    MW8	//Transferir el resultado a MW8.	

### Ejemplo 2

AWL	Explicación	
L    +3	//Cargar el valor +3 en el ACU 1.	
L    MW20	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MW20 en //el ACU 1.	
SRW	//El número de desplazamiento es el valor del ACU2-L-L. => Desplazar //los bits en el ACU 1-L, 3 posiciones a la derecha.	
SPP    NEXT	//Saltar a la meta NEXT, si el último bit desplazado (A1) es 1.	

### 11.1.6 SLD Desplazar doble palabra a la izquierda (32 bits)

#### Formato

**SLD**

**SLD <número>**

Operando	Tipo de datos	Descripción
<número>	Entero, sin signo	Número de posiciones de bit a desplazar; margen de 0 a 32

#### Descripción de la operación

**SLD** (Desplazar doble palabra a la izquierda) desplaza el contenido completo del ACU 1 bit por bit a la izquierda. En las posiciones de bit que quedan libres por el desplazamiento se escriben ceros. El último bit desplazado se carga en el bit A1 de la palabra de estado. El número de las posiciones de bit a desplazar viene indicado por el operando <número> o por un valor en el ACU2-L-L.

**SLD <número>**: El operando <número> indica el número de desplazamiento. Se admiten valores entre 0 y 32. Los bits A0 y OV de la palabra de estado se ponen a "0", si <número> es mayor que cero. Si <número> es igual a "0", la operación de rotación de procesa igual que una operación NOP.

**SLD**: El número de desplazamiento viene indicado por el valor en el ACU2-L-L. Se admiten valores entre 0 y 255. Un número de desplazamiento > 32 siempre provoca el mismo resultado: ACU 1 = 0, A1 = 0, A0 = 0. Si 0 < número de desplazamiento <= 32, los bits A0 y OV de la palabra de estado se ponen a "0". Si el número de desplazamiento es "0", la operación de desplazamiento se procesa igual que una operación NOP.

#### Palabra de estado

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

## Ejemplos

Contenido	ACU1-H				ACU1-L			
	31 ...	..	..	... 16	15 ...	..	..	... 0
ACU 1 antes de ejecutar <b>SLD 5</b>	0101	1111	0110	0100	0101	1101	0011	1011
ACU 1 después de ejecutar <b>SLD 5</b>	1111	0110	0100	0101	1101	0011	1011	0101

### Ejemplo 1

AWL		Explicación
L	MD4	//Cargar el valor en el ACU 1.
SLD	5	//Desplazar los bits en el ACU 1, 5 posiciones a la izquierda.
T	MD8	//Transferir el resultado a MD8.

### Ejemplo 2

AWL		Explicación
L	+3	//Cargar el valor +3 en el ACU 1.
L	MD20	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD20 en //el ACU 1.
SLD		//El número de desplazamiento es el valor del ACU2-L-L. => Desplazar //los bits en el ACU 1, 3 posiciones a la izquierda.
SPP	NEXT	//Saltar a la meta NEXT, si el último bit desplazado (A1) es 1.

### 11.1.7 SRD Desplazar doble palabra a la derecha (32 bits)

#### Formato

**SRD**

**SRD <número>**

Operando	Tipo de datos	Descripción
<número>	Entero, sin signo	Número de posiciones de bit a desplazar; margen de 0 a 32

#### Descripción de la operación

**SRD** (Desplazar doble palabra a la derecha) desplaza el contenido completo del ACU 1 bit por bit a la derecha. En las posiciones de bit que quedan libres por el desplazamiento se escriben ceros. El último bit desplazado se carga en el bit A1 de la palabra de estado. El número de las posiciones de bit a desplazar viene indicado por el operando <número> o por un valor en el ACU2-L-L.

**SRD <número>**: El operando <número> indica el número de desplazamiento. Se admiten valores entre 0 y 32. Los bits A0 y OV de la palabra de estado se ponen a "0" si <número> es mayor que cero. Si <número> es igual a "0" la operación de rotación se procesa igual que una operación NOP.

**SRD**: El número de desplazamiento viene indicado por el valor en el ACU2-L-L. Se admiten valores entre 0 y 255. Un número de desplazamiento > 32 siempre provoca el mismo resultado: ACU 1 = 0, A1 = 0, A0 = 0, OV = 0. Si 0 < número de desplazamiento <= 32, los bits A0 y OV de la palabra de estado se ponen a "0". Si el número de desplazamiento es "0", la operación de desplazamiento se procesa igual que una operación NOP.

#### Palabra de estado

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

## Ejemplos

Contenido	ACU1-H				ACU1-L			
	31 ...	..	..	... 16	15 ...	..	..	... 0
antes de ejecutar <b>SRD 7</b>	0101	1111	0110	0100	0101	1101	0011	1011
después de ejecutar <b>SRD 7</b>	1111	0110	0100	0101	1101	0011	1011	0101

### Ejemplo 1

AWL		Explicación
L	MD4	//Cargar el valor en el ACU 1.
SRD	7	//Desplazar los bits en el ACU 1, 7 posiciones a la derecha.
T	MD8	//Transferir el resultado a MD8.

### Ejemplo 2

AWL		Explicación
L	+3	//Cargar el valor +3 en el ACU 1.
L	MD20	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD20 en //el ACU 1.
SRD		//El número de desplazamiento es el valor del ACU2-L-L. => Desplazar //los bits en el ACU 1, 3 posiciones a la derecha.
SPP	NEXT	//Saltar a la meta NEXT, si el último bit desplazado (A1) es 1.

## 11.2 Operaciones de rotación

### 11.2.1 Lista de operaciones de rotación

#### Descripción

Las operaciones de rotación hacen circular todo el contenido del ACU 1 bit por bit a la izquierda o a la derecha (v. Registros de la CPU). Las operaciones de rotación activan funciones similares a las funciones de desplazamiento. Sin embargo, las posiciones vacantes se rellenan con los estados de señal de los bits que se desplazan fuera del acumulador.

El número que sigue a la operación de desplazamiento, o bien un valor del byte bajo de la palabra baja del ACU 2 indica el número de bits que se rotarán.

Dependiendo de la operación, la rotación tendrá lugar a través del bit de la palabra de estado. El bit de estado A0 se pone a "0".

Se dispone de las siguientes operaciones de rotación:

- RLD Rotar doble palabra a la izquierda (32 bits)
- RRD Rotar doble palabra a la derecha (32 bits)
- RLDA Rotar ACU 1 a la izquierda vía A1 (32 bits)
- RRDA Rotar ACU 1 a la derecha vía A1 (32 bits)

## 11.2.2 RLD Rotar doble palabra a la izquierda (32 bits)

### Formato

**RLD**

**RLD <número>**

Operando	Tipo de datos	Descripción
<número>	Entero, sin signo	Número de posiciones de bit a rotar; margen de 0 a 32

### Descripción de la operación

**RLD** (Rotar doble palabra a la izquierda) rota el contenido completo del ACU 1 bit por bit a la izquierda. En las posiciones de bit que quedan libres por la rotación se escriben los estados de señal de los bits que se desplazan fuera del ACU 1. El último bit rotado se carga en el bit A1 de la palabra de estado. El número de las posiciones de bit a rotar viene indicado por el operando <número> o un valor en el ACU2-L-L.

**RLD <número>**: El operando <número> indica el número de rotación. Se admiten valores entre 0 y 32. Los bits A0 y OV de la palabra de estado se ponen a "0" si <número> es mayor que 0. Si <número> es igual a "0", la operación de rotación se procesa igual que una operación NOP.

**RLD**: El número de rotación viene indicado por el valor en el ACU2-L-L. Se admiten valores entre 0 y 255. Los bits A0 y OV de la palabra de estado se ponen a "0" si el contenido del ACU2-L-L es mayor que cero. Si el número de rotación es "0" la operación de rotación se procesa igual que una operación NOP.

### Palabra de estado

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

Ejemplos

Contenido	ACU1-H				ACU1-L			
Bit	31 ...	..	..	... 16	15 ...	..	..	... 0
antes de ejecutar <b>RLD 4</b>	0101	1111	0110	0100	0101	1101	0011	1011
después de ejecutar <b>RLD 4</b>	1111	0110	0100	0101	1101	0011	1011	0101

Ejemplo 1

AWL	Explicación	
L MD2	//Cargar el valor en el ACU 1.	
RLD 4	//Rotar los bits en el ACU 1, 4 posiciones a la izquierda.	
T MD8	//Transferir el resultado a MD8.	

Ejemplo 2

AWL	Explicación	
L +3	//Cargar el valor +3 en el ACU 1.	
L MD20	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD20 en //el ACU 1.	
RLD	//El número de rotación es el valor del ACU2-L-L. => Rotar los bits en el //ACU 1, 3 posiciones a la izquierda.	
SPP NEXT	//Saltar a la meta NEXT, si el último bit rotado (A1) es 1.	

### 11.2.3 RRD Rotar doble palabra a la derecha (32 bits)

#### Formato

**RRD**

**RRD <número>**

Operando	Tipo de datos	Descripción
<número>	Entero, sin signo	Número de posiciones de bit a rotar; margen de 0 a 32

#### Descripción de la operación

**RRD** (Rotar doble palabra a la derecha) rota el contenido completo del ACU 1 bit por bit a la derecha. En las posiciones de bit que quedan libres por la rotación se escriben los estados de señal de los bits que se rotan fuera del ACU 1. El último bit rotado se carga en el bit A1 de la palabra de estado. El número de las posiciones de bit a rotar viene indicado por el operando <número> o por un valor en el ACU2-L-L.

**RRD <número>**: El operando <número> indica el número de rotación. Se admiten valores entre 0 y 32. Los bits A0 y OV de la palabra de estado se ponen a "0" si <número> es mayor que cero. Si <número> es igual a cero la operación de rotación se procesa igual que una operación NOP.

**RRD**: El número de rotación viene indicado por el valor en el ACU2-L-L. Se admiten valores entre 0 y 255. Los bits A0 y OV de la palabra de estado se ponen a "0" si el contenido del AKKU2-L-L es mayor que cero. Si el número de rotación es "0", la operación de rotación se procesa igual que una operación NOP.

#### Palabra de estado

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

#### Ejemplos

Contenido	ACU1-H				ACU1-L			
	31 ...	..	..	... 16	15 ...	..	..	... 0
antes de ejecutar <b>RRD 4</b>	0101	1111	0110	0100	0101	1101	0011	1011
después de ejecutar <b>RRD 4</b>	1111	0110	0100	0101	1101	0011	1011	0101

### Ejemplo 1

AWL		Explicación
L	MD2	//Cargar el valor en el ACU 1.
RRD	4	//Desplazar los bits en el ACU 1, 4 posiciones a la derecha.
T	MD8	//Transferir el resultado a MD8.

### Ejemplo 2

AWL		Explicación
L	+3	//Cargar el valor +3 en el ACU 1.
L	MD20	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el valor de MD20 en //el ACU 1.
RRD		//El número de rotación es el valor del ACU2-L-L. => Rotar los bits en el //ACU 1, 3 posiciones a la derecha.
SPP	NEXT	//Saltar a la meta NEXT, si el último bit rotado (A1) es 1.

### 11.2.4 RLDA Rotar ACU 1 a la izquierda vía A1 (32 bits)

#### Formato

**RLDA**

#### Descripción de la operación

**RLDA** (Rotar doble palabra a la izquierda vía A1) rota el contenido completo del ACU 1 una posición de bit a la izquierda mediante el código de condición A1. Los bits A0 y OV de la palabra de estado se ponen a "0".

#### Palabra de estado

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

#### Ejemplos

Contenido	BI1	ACU1-H				ACU1-L			
Bit		31 ...	..	..	... 16	15 ...	..	..	... 0
Antes de ejecutar <b>RLDA</b>	<b>X</b>	0101	1111	0110	0100	0101	1101	0011	1011
Después de ejecutar <b>RLDA</b>	<b>0</b>	1011	1110	1100	1000	1011	1010	0111	011 <b>X</b>
(X = 0 ó 1, estado de señal antiguo de A1)									

AWL	Explicación	
L MD2	//Cargar el valor de MD2 en el ACU 1.	
RLDA	//Rotar los bits en el ACU 1 una posición a la izquierda vía A1.	
SPP NEXT	//Saltar a la meta NEXT, si el último bit rotado (A1) es 1.	

### 11.2.5 RRDA Rotar ACU 1 a la derecha vía A1 (32 bits)

#### Formato

RRDA

#### Descripción de la operación

**RRDA** (Rotar doble palabra a la derecha vía A1) rota el contenido completo del ACU 1 una posición de bit a la derecha mediante el código de condición A1. Los bits A0 y OV de la palabra de estado se ponen a "0".

#### Palabra de estado

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

#### Ejemplos

Contenido	BI1	ACU1-H				ACU1-L			
Bit		31 ...	..	..	... 16	15 ...	..	..	... 0
Antes de ejecutar <b>RRDA</b>	<b>X</b>	0101	1111	0110	0100	0101	1101	0011	1011
Después de ejecutar <b>RRDA</b>	<b>1</b>	<b>X</b> 010	1111	1011	0010	0010	1110	1001	1101
(X = 0 ó 1, estado de señal antiguo de A1)									

AWL	Explicación	
L MD2	//Cargar el valor deMD2 en el ACU 1.	
RRDA	//Rotar los bits en el ACU 1 una posición a la derecha vía A1.	
SPP NEXT	//Saltar a la meta NEXT, si el último bit rotado (A1) es 1.	

## 12 Operaciones de temporización

### 12.1 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:

- FR Habilitar temporizador
- L Cargar valor actual del temporizador en ACU 1 como entero
- LC Cargar el valor actual de temporización en ACU 1 como número BCD
- R Desactivar temporizador
- SI Temporizador como impulso
- SV Temporizador como impulso prolongado
- SE Temporizador como retardo a la conexión
- SS Temporizador como retardo a la conexión con memoria
- SA Temporizador como retardo a la desconexión

## 12.2 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 AWL 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 decreta 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 decreta 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#txyz**
  - siendo: t= la base de tiempo (es decir, intervalo de tiempo o resolución)
  - xyz = el valor de temporización en formato BCD
- **S5T#aH\_bM\_cS\_dMS**
  - 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.

## 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

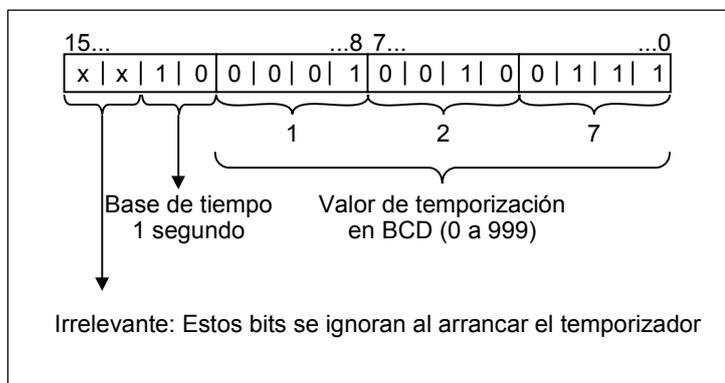
Los valores no deben exceder 2H\_46M\_30S. Los valores con un margen o una resolución demasiado grandes se redondean. El formato general para el tipo de datos S5TIME tiene los siguientes valores límite:

Resolución	Margen
0,01 segundos	10MS a 9S_990MS
0,1 segundos	100MS a 1M_39S_900MS
1 segundo	1S a 16M_39S
10 segundos	10S a 2H_46M_30S

## Configuración binaria en ACU1

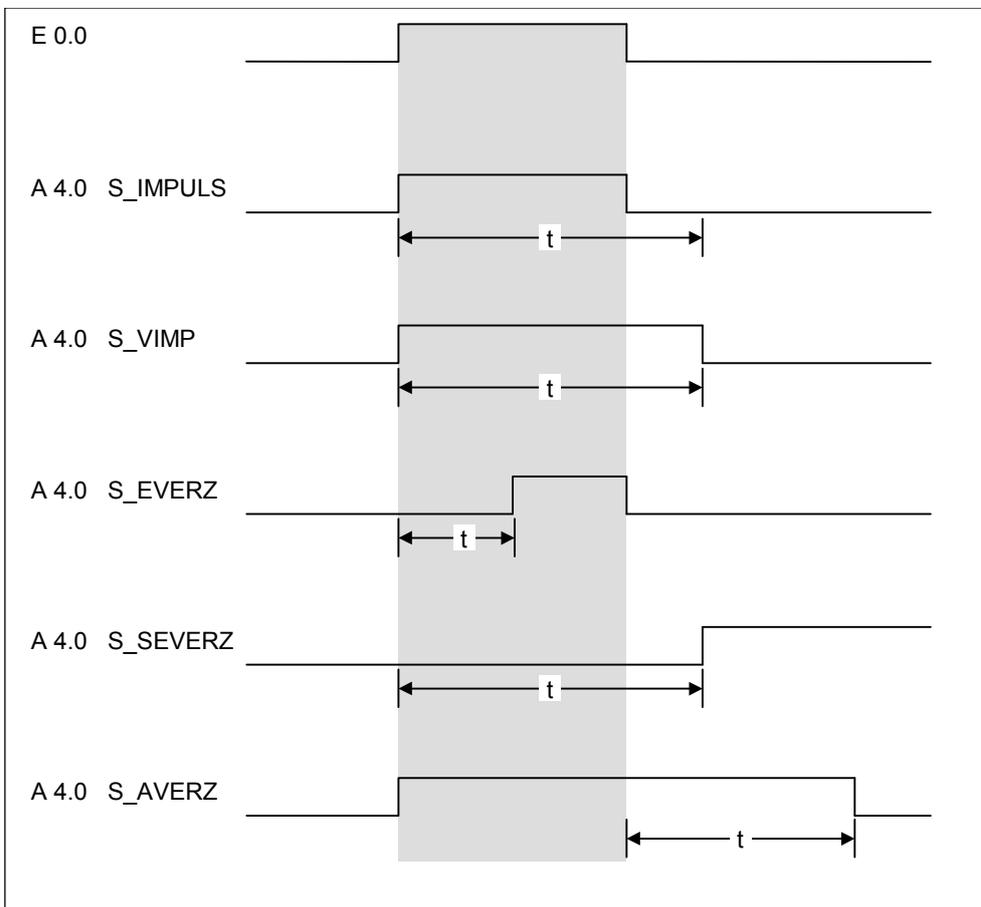
Cuando se dispara un temporizador, se utiliza el contenido del ACU1 como valor de temporización. Los bits 0 a 11 del ACU1-L contienen el valor de temporización en formato decimal codificado en binario (formato BCD: cada grupo de cuatro bits contiene el código binario de un valor decimal). Los bits 12 a 13 contienen la base de tiempo en código binario.

La figura muestra el contenido del ACU1-L una vez cargado el valor 127 con una base de tiempo de 1 segundo.



### 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
<b>S_IMPULS</b> 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.
<b>S_VIMP</b> 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.
<b>S_EVERZ</b> 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.
<b>S_SEVERZ</b> 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.
<b>S_AVERZ</b> 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.

## 12.3 FR Habilitar temporizador

### Formato

FR <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

### Descripción de la operación

**FR <temporizador>** borra la marca de flancos que se utiliza para arrancar el temporizador direccionado, si el RLO cambia de "0" a "1". Si el bit RLO cambia de "0" a "1" antes de una operación Habilitar temporizador (FR) se habilita un temporizador.

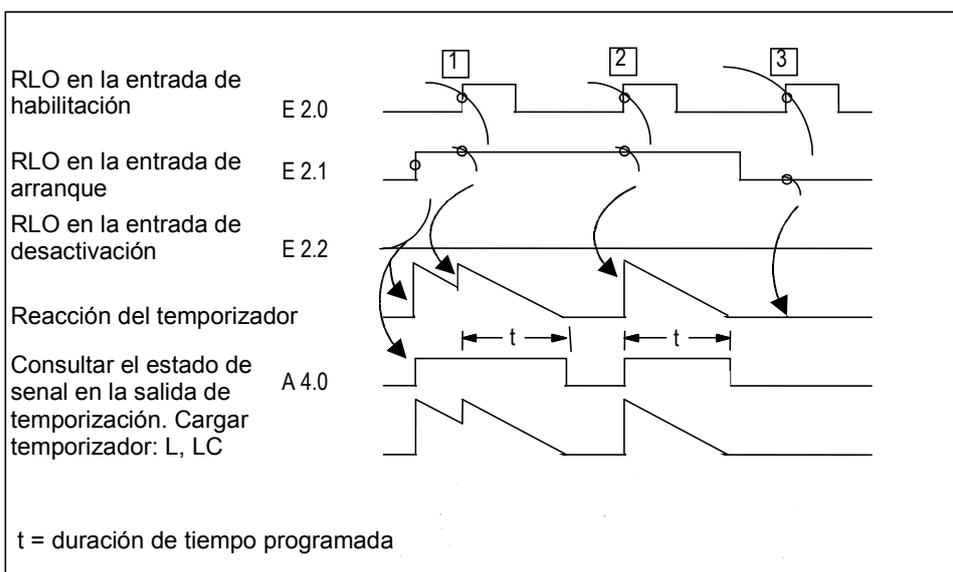
La operación Habilitar temporizador no es necesaria efectuar el arranque normal de un temporizador; sólo se utiliza para volver a arrancar un temporizador que está en funcionamiento. Ello sólo es posible si la operación de arranque se sigue procesando con el RLO = 1.

### Palabra de estado

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

**Ejemplo**

AWL	Explicación
U E 2.0	
FR T1	//Habilitar el temporizador T1.
U E 2.1	
L S5T#10s	//Ajustar una preselección de 10 segundos en el ACU 1.
SI T1	//Arrancar el temporizador T1 como impulso.
U E 2.2	
R T1	//Poner el temporizador T1 a 0.
U T1	//Consultar el estado de señal del temporizador T1.
= A 4.0	
L T1	//Cargar el valor de temporización actual del temporizador T1 como número //binario.
T MW10	



- (1) Si el RLO en la entrada de habilitación cambia de "0" a "1"; durante el funcionamiento del temporizador, éste volverá a arrancarse. El temporizador programado es el temporizador actual para el nuevo arranque. Si el RLO en la entrada de habilitación cambia de "1" a "0", esto no tiene influencia.
- (2) Si el RLO en la entrada de habilitación cambia de "0" a "1", y el temporizador no está en funcionamiento, mientras hay un RLO de "1" en la entrada de arranque, el temporizador se arranca con el valor de temporización programado.
- (3) Si el RLO en la entrada de habilitación cambia de "0" a "1"; mientras hay un RLO de "0" en la entrada de arranque, esto no tiene influencia sobre el temporizador.

## 12.4 L Cargar valor actual del temporizador en ACU 1 como entero

### Formato

L <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

### Descripción de la operación

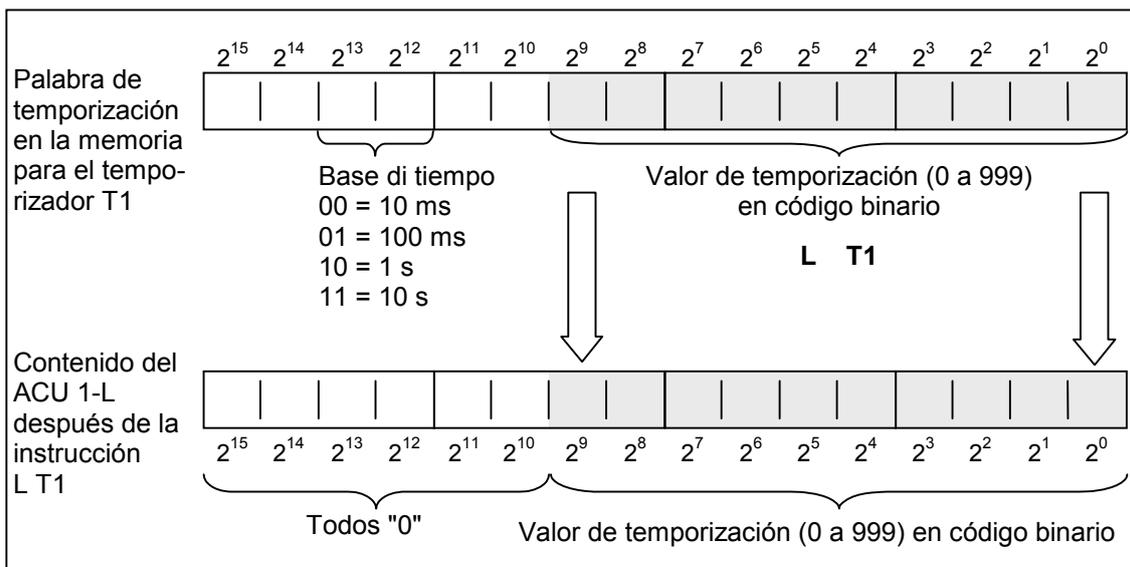
L <temporizador> carga en el ACU1-L el valor de temporización actual de la palabra de temporización sin base de tiempo como entero binario, después de que se haya cargado el contenido del ACU 1 en el ACU 2.

### Palabra de estado

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

**Ejemplo**

AWL	Explicación
L T1	//Cargar el ACU1-L con el valor de temporización actual del temporizador //T1 en código binario.



**Nota**

L <temporizador> sólo carga el código binario del valor de temporización actual en el ACU 1-L, pero no la base de tiempo. El valor de temporización que se carga es el valor inicial del tiempo menos el tiempo que ha transcurrido desde el arranque de la función de temporización.

## 12.5 LC Cargar el valor actual de temporización en ACU 1 como número BCD

### Formato

LC <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

### Descripción de la operación

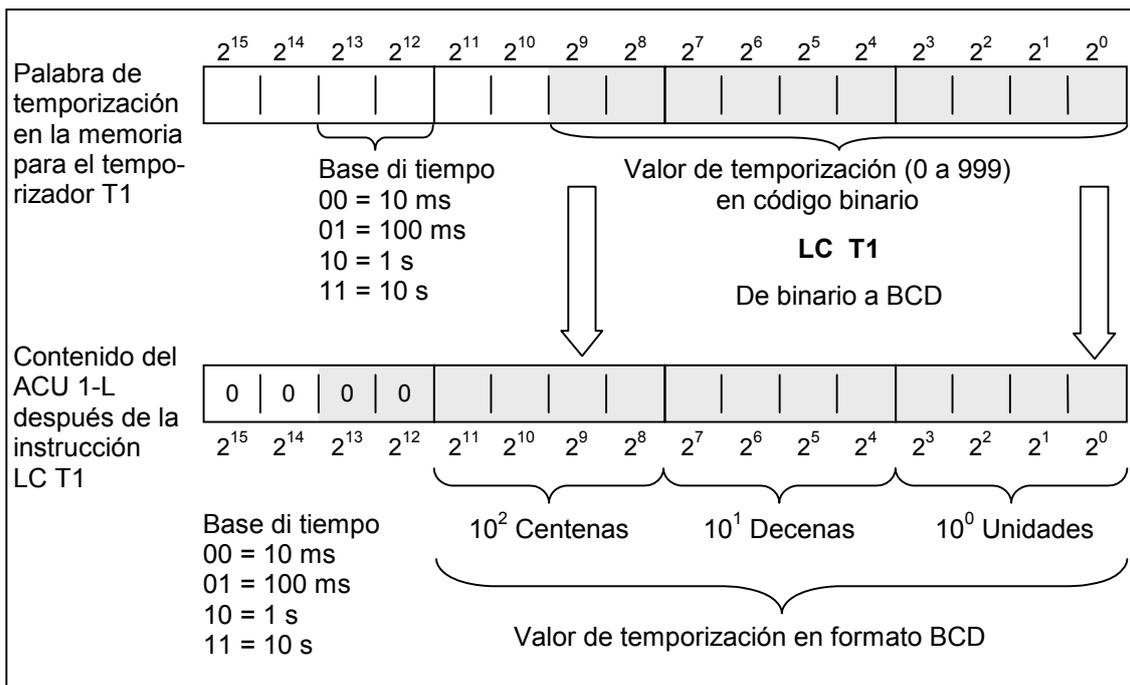
LC <temporizador> carga en el ACU 1 el valor de temporización actual y la base de tiempo de la palabra de temporización direccionada como número en formato decimal codificado en binario (BCD), después de que se haya cargado el contenido del ACU 1 en el ACU 2.

### Palabra de estado

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

Ejemplo

AWL	Explicación
LC T1	//Cargar ACU1-L con la base de tiempo y el valor de temporización actual //del temporizador T1 en formato BCD en el ACU1-L.



## 12.6 R Desactivar temporizador

### Formato

R <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

### Descripción de la operación

R <temporizador> finaliza la función de temporización actual y borra el valor de temporización y la base de tiempo de la palabra de temporización direccionada, si el RLO cambia de "0" a "1".

### Palabra de estado

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

### Ejemplo

AWL	Explicación
U E 2.1	
R T1	//Consultar el estado de señal en la entrada E 2.1. Si el RLO cambia de // "0" a "1", desactivar el temporizador T1.

## 12.7 SI Temporizador como impulso

### Formato

SI <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

### Descripción de la operación

**SI <temporizador>** arranca el temporizador direccionado si el RLO cambia de "0" a "1". El intervalo programado transcurre mientras el RLO sea 1. Si el RLO cambia a "0" antes de que haya transcurrido el intervalo programado, el temporizador se para. Para esta operación (Arrancar temporizador) tienen que estar almacenados el valor de temporización y la base de tiempo en formato BCD en el ACU1-L.

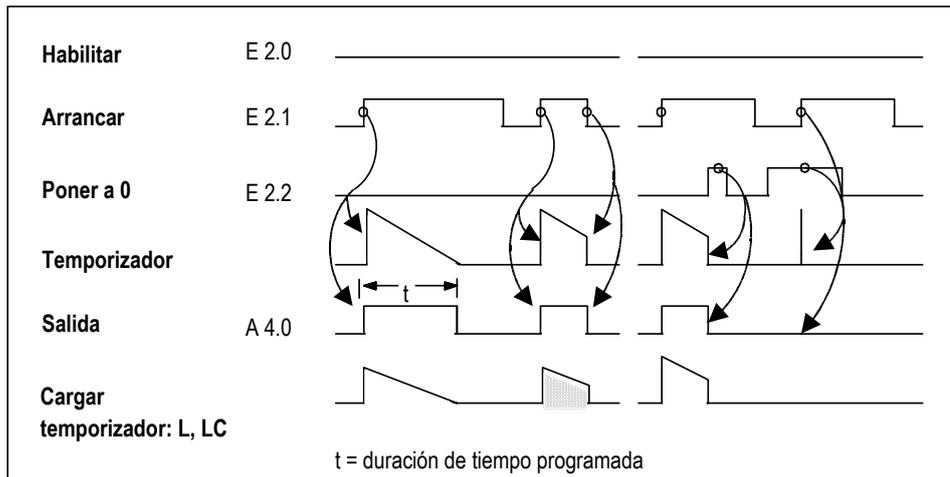
Consulte también Area de memoria y componentes de un temporizador.

### Palabra de estado

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

## Ejemplo

AWL	Explicación
U E 2.0	
FR T1	//Habilitar el temporizador T1.
U E 2.1	
L S5T#10s	//Ajustar una preselección de 10 segundos en el ACU 1.
SI T1	//Arrancar el temporizador T1 como impulso.
U E 2.2	
R T1	//Poner el temporizador T1 a 0.
U T1	//Consultar el estado de señal del temporizador T1.
= A 4.0	
L T1	//Cargar el valor de temporización actual del temporizador T1 como número //binario.
T MW10	
LC T1	//Cargar el valor de temporización actual del temporizador T1 en formato //BCD.
T MW12	



## 12.8 SV Temporizador como impulso prolongado

### Formato

SV <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

### Descripción de la operación

**SV <temporizador>** arranca el temporizador direccionado si el RLO cambia de "0" a "1". El intervalo programado transcurre aunque el RLO cambie mientras tanto a "0". Si el RLO cambia de "0" a "1" antes de que haya transcurrido el intervalo programado, se vuelve a arrancar el intervalo programado. Para que se ejecute esta orden de arrancar el temporizador tienen que estar almacenados en el ACU1-L el valor de temporización y la base de tiempo en formato BCD.

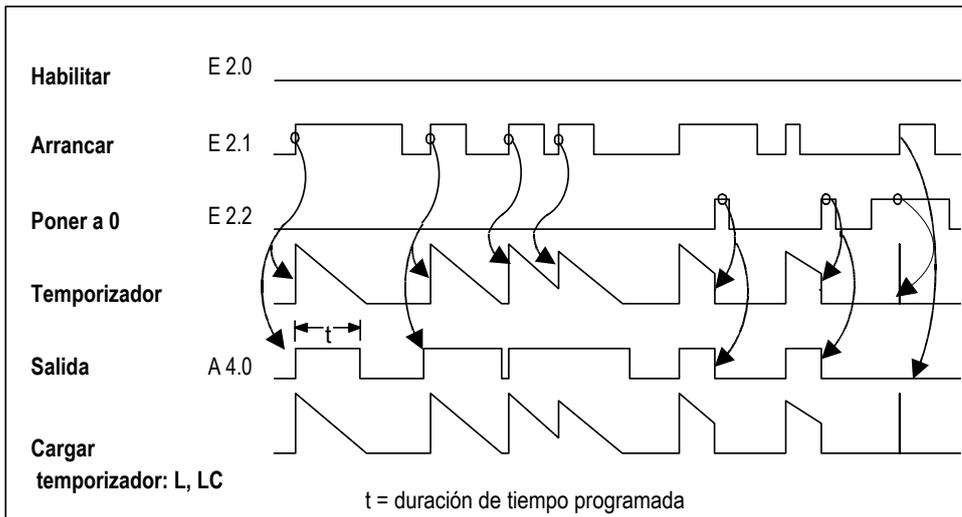
Consulte también Area de memoria y componentes de un temporizador.

### Palabra de estado

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

## Ejemplo

AWL	Explicación
U E 2.0	
FR T1	//Habilitar el temporizador T1.
U E 2.1	
L S5T#10s	//Ajustar una preselección de 10 segundos en el ACU 1.
SV T1	//Arrancar el temporizador T1 como impulso prolongado.
U E 2.2	
R T1	//Poner el temporizador T1 a 0.
U T1	//Consultar el estado de señal del temporizador T1.
= A 4.0	
L T1	//Cargar el valor de temporización actual del temporizador T1 como número //binario.
T MW10	
LC T1	//Cargar el valor de temporización actual del temporizador T1 en formato //BCD.
T MW12	



## 12.9 SE Temporizador como retardo a la conexión

### Formato

SE <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

### Descripción de la operación

**SE <temporizador>** arranca el temporizador direccionado si el RLO cambia de "0" a "1". El intervalo programado transcurre mientras el RLO sea 1. Si el RLO cambia a "0" antes de haber transcurrido el intervalo programado, el temporizador se detiene. Para que se realice esta orden de arrancar el temporizador tienen que estar almacenados en el ACU1-L el valor de temporización y la base de tiempo en formato BCD.

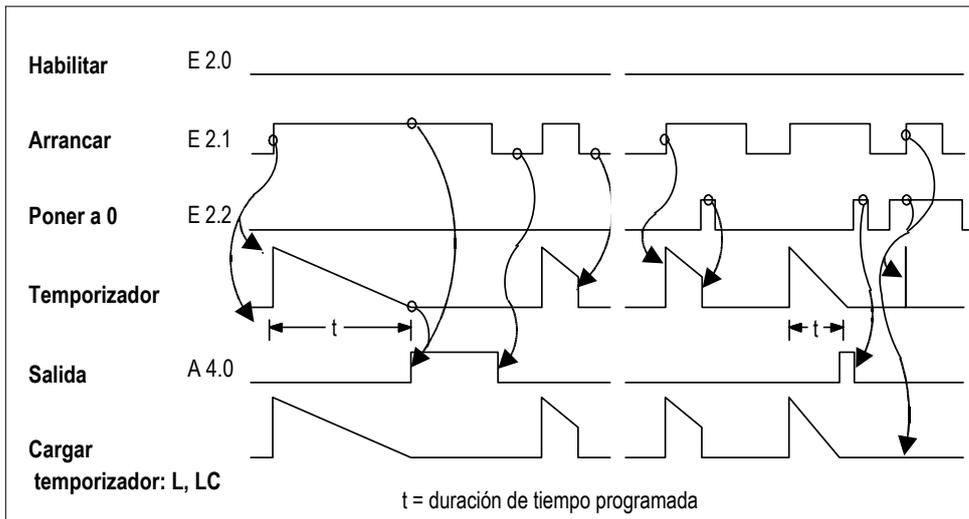
Consulte también Area de memoria y componentes de un temporizador.

### Palabra de estado

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

## Ejemplo

AWL	Explicación
U E 2.0	
FR T1	//Habilitar el temporizador T1.
U E 2.1	
L S5T#10s	//Ajustar una preselección de 10 segundos en el ACU 1.
SE T1	//Arrancar el temporizador T1 como retardo a la conexión.
U E 2.2	
R T1	//Poner el temporizador T1 a 0.
U T1	//Consultar el estado de señal del temporizador T1.
= A 4.0	
L T1	//Cargar el valor de temporización actual del temporizador T1 como número //binario.
T MW10	
LC T1	//Cargar el valor de temporización actual del temporizador T1 en formato //BCD.
T MW12	



## 12.10 SS Temporizador como retardo a la conexión con memoria

### Formato

SS <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

### Descripción de la operación

SS <temporizador> (Arrancar temporizador como retardo a la conexión con memoria) arranca el temporizador direccionado si el RLO cambia de "0" a "1". El intervalo programado transcurre aunque el RLO cambie mientras tanto a "0". Si el RLO cambia de "0" a "1" antes de que haya transcurrido el intervalo programado, se vuelve a arrancar el intervalo programado. Para que se ejecute esta operación de arrancar el temporizador tienen que estar almacenados en el ACU1-L el valor de temporización y la base de tiempo en formato BCD.

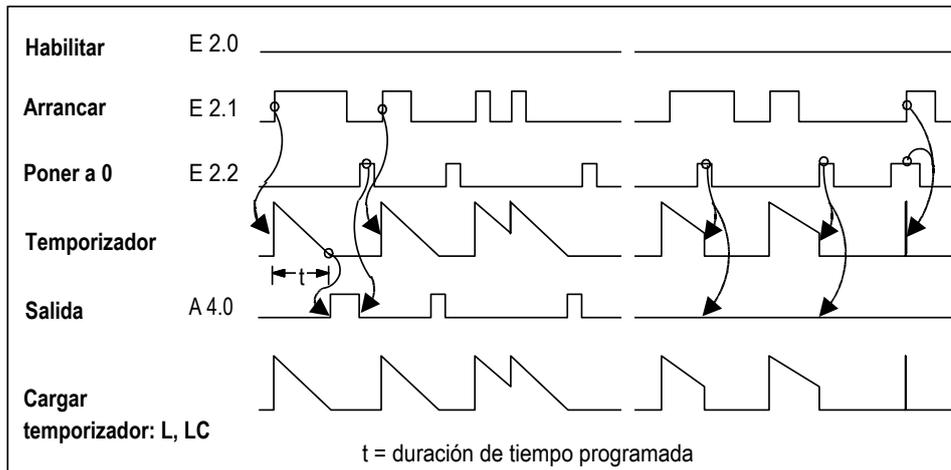
Consulte también Area de memoria y componentes de un temporizador.

### Palabra de estado

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

## Ejemplo

AWL		Explicación
U	E 2.0	
FR	T1	//Habilitar el temporizador T1.
U	E 2.1	
L	S5T#10s	//Ajustar una preselección de 10 segundos en el ACU 1.
SS	T1	//Arrancar el temporizador T1 como retardo a la conexión con memoria.
U	E 2.2	
R	T1	//Poner el temporizador T1 a 0.
U	T1	//Consultar el estado de señal del temporizador T1.
=	A 4.0	
L	T1	//Cargar el valor de temporización actual del temporizador T1 como número //binario.
T	MW10	
LC	T1	//Cargar el valor de temporización actual del temporizador T1 en formato //BCD.
T	MW12	



## 12.11 SA Temporizador como retardo a la desconexión

### Formato

SA <temporizador>

Operando	Tipo de datos	Area de memoria	Descripción
<temporizador>	TIMER	T	Número del temporizador; el área varía según la CPU utilizada

### Descripción de la operación

**SA <temporizador>** arranca el temporizador direccionado si el RLO cambia de "1" a "0". El intervalo programado transcurre mientras el RLO sea 0. Si el RLO cambia a "1" antes de que haya transcurrido el intervalo programado, el temporizador se para. Para esta instrucción Arrancar temporizador tienen que estar almacenados el valor de temporización y la base de tiempo en formato BCD en el ACU1-L.

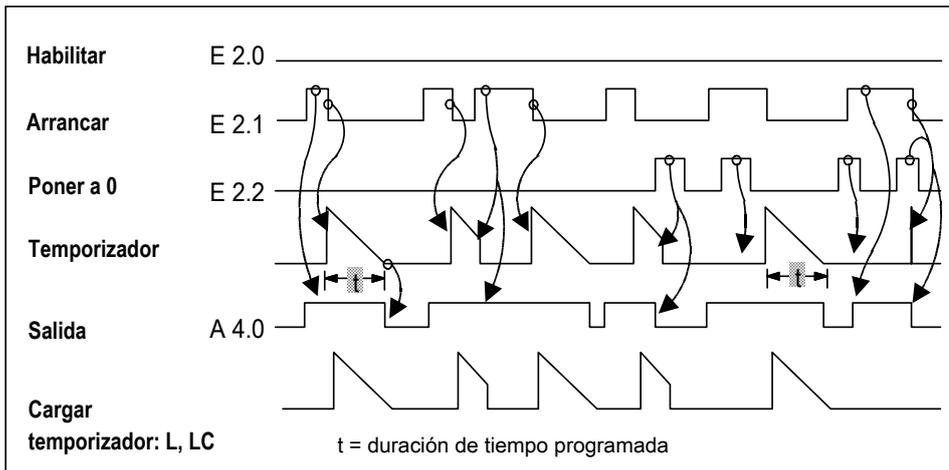
Consulte también Area de memoria y componentes de un temporizador.

### Palabra de estado

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

## Ejemplo

AWL	Explicación
U E 2.0	
FR T1	//Habilitar el temporizador T1.
U E 2.1	
L S5T#10s	//Ajustar una preselección de 10 segundos en el ACU 1.
SA T1	//Arrancar el temporizador T1 como retardo a la desconexión.
U E 2.2	
R T1	//Poner el temporizador T1 a 0.
U T1	//Consultar el estado de señal del temporizador T1.
= A 4.0	
L T1	//Cargar el valor de temporización actual del temporizador T1 como número //binario.
T MW10	
LC T1	//Cargar el valor de temporización actual del temporizador T1 en formato //BCD.
T MW12	





# 13 Operaciones lógicas con palabras

## 13.1 Lista de operaciones lógicas con palabras

### Descripción

Las operaciones lógicas con palabras combinan pares de palabras (16 bits) o palabras dobles (32 bits) bit por bit de acuerdo con la lógica de Boole. Cada palabra o palabra doble debe encontrarse en uno de ambos acumuladores.

En las operaciones con palabras se combina el contenido de la palabra baja del ACU 2 con el contenido de la palabra baja del ACU 1. El resultado lógico se almacena en la palabra baja del ACU 1, sobrescribiendo el antiguo contenido.

En las operaciones con palabras dobles se combina el contenido del ACU 2 con el contenido del ACU 1. El resultado lógico se almacena en el ACU 1, sobrescribiendo el antiguo contenido.

Si el resultado lógico es "0", el bit A1 de la palabra de estado se pone a "0". Si el resultado no es igual a "0", el bit A1 se pone a "1". Los bits A0 y OV de la palabra de estado se ponen en ambos casos a "0".

Se dispone de las operaciones lógicas con palabras siguientes:

- UW Y con palabra (16 bits)
- OW O con palabra (16 bits)
- XOW O-exclusiva con palabra (16 bits)
- UD Y con doble palabra (32 bits)
- OD O con doble palabra (32 bits)
- XOD O-exclusiva con doble palabra (32 bits)

## 13.2 UW Y con palabra (16 bits)

### Formato

**UW**

**UW <constante>**

Operando	Tipo de datos	Descripción
<constante>	WORD, constante (16 bits)	Configuración binaria a combinar por medio de Y con el ACU1-L.

### Descripción de la operación

**UW** (Y con palabra) combina el contenido del ACU1-L con el ACU2-L ó con una constante (de 16 bits) bit por bit realizando una operación lógica Y. Solamente cuando los bits correspondientes de ambas palabras a combinar son "1" será también "1" el bit respectivo de la doble palabra resultante. El resultado se almacena en el ACU1-L. El ACU1-H y el ACU 2 (y en las CPU con cuatro acumuladores, el ACU 3 y el ACU 4) no se alteran. El bit de la palabra de estado A1 se activa como resultado de la operación (A1 = 1, si el resultado es diferente de cero). Los bits A0 y OV de la palabra de estado se ponen a "0".

**UW:** combina el ACU1-L con el ACU2-L.

**UW <constante>:** combina el ACU1-L con una constante (16 bits).

### Palabra de estado

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

## Ejemplos

Bit	15..	..	..	...0
ACU1-L antes de ejecutar <b>UW</b>	0101	1001	0011	1011
ACU2-L ó constante (16 bits)	1111	0110	1011	0101
Resultado (ACU1-L) después de ejecutar <b>UW</b>	0101	0000	0011	0001

### Ejemplo 1

AWL		Explicación
L	EW20	//Cargar el contenido de EW20 en el ACU1-L.
L	EW22	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el contenido de EW22 //en el ACU 1-L.
UW		//Combinar los bits del ACU1-L con los bits del ACU2-L realizando una //operación Y lógica, almacenar el resultado en el ACU1-L.
T	MW 8	//Transferir el resultado a MW8.

### Ejemplo 2

AWL		Explicación
L	EW20	//Cargar el contenido de EW20 en el ACU1-L.
UW	W#16#0FF F	//Combinar los bits del ACU1-L con la configuración binaria de la //constante (de 16 bits) (0000_1111_1111_1111) realizando una operación Y //lógica, almacenar el resultado en el ACU1-L.
SPP	NEXT	//Saltar a la meta NEXT si el resultado es diferente de cero (A1 = 1).

## 13.3 OW O con palabra (16 bits)

### Formato

**OW**

**OW <constante>**

Operando	Tipo de datos	Descripción
<constante>	WORD, constante (16 bits)	Configuración binaria a combinar con el ACU1-L realizando una operación O lógica.

### Descripción de la operación

**OW** (O con palabra) combina el contenido del ACU1-L con el ACU2-L ó con una constante (de 16 bits) bit por bit realizando una operación lógica O. Si al menos uno los bits correspondientes de ambas dobles palabras a combinar es "1", el bit respectivo de la palabra resultante también será "1". El resultado se almacena en el ACU1-L. El ACU1-H y el ACU 2 (y en las CPU con cuatro acumuladores, el ACU 3 y el ACU 4) no se alteran. El bit de la palabra de estado A1 se activa como resultado de la operación (A1 = 1, si el resultado es diferente de cero). Los bits A0 y OV de la palabra de estado se ponen a "0".

**OW:** combina el ACU1-L con el ACU2-L.

**OW <constante>:** combina el ACU1-L con una constante (16 bits).

### Palabra de estado

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

## Ejemplos

Bit	15..	..	..	...0
ACU1-L antes de ejecutar <b>OW</b>	0101	0101	0011	1011
ACU2-L ó constante (16 bits)	1111	0110	1011	0101
Resultado (ACU1-L) después de ejecutar <b>OW</b>	1111	0111	1011	1111

### Ejemplo 1

AWL	Explicación
L EW20	//Cargar el contenido de EW20 en el ACU1-L.
L EW22	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el contenido de EW22 //en el ACU 1-L.
OW	//Combinar los bits del ACU1-L con los bits del ACU2-L realizando una //operación O lógica, almacenar el resultado en el ACU1-L.
T MW8	//Transferir el resultado a MW 8.

### Ejemplo 2

AWL	Explicación
L EW20	//Cargar el contenido de EW20 en el ACU1-L.
OW W#16#0FF F	//Combinar los bits del ACU1-L con la configuración binaria de la //constante (16 bits) (0000_1111_1111_1111) realizando una operación O //lógica, almacenar el resultado en el ACU1- L.
SPP NEXT	//Saltar a la meta NEXT si el resultado es diferente de cero (A1 = 1).

## 13.4 XOW O-exclusiva con palabra (16 bits)

### Formato

**XOW**

**XOW <constante>**

Operando	Tipo de datos	Descripción
<constante>	WORD, constante (16 bits)	Configuración binaria a combinar lógicamente con el ACU1-L realizando una operación O-exclusiva.

### Descripción de la operación

**XOW** (O-exclusiva con palabra) combina el contenido del ACU1-L con el ACU2-L o una constante (de 16 bits) bit por bit realizando una operación lógica O-exclusiva. Si uno -y solamente uno- de los bits correspondientes de ambas palabras a combinar es "1", el bit respectivo de la palabra resultante también es "1". El resultado se almacena en el ACU1-L. El ACU1-H y el ACU 2 (y en las CPU con cuatro acumuladores, el ACU 3 y el ACU 4) no se alteran. El bit de la palabra de estado A1 se activa como resultado de la operación (A1 = 1, si el resultado es diferente de cero). Los bits A0 y OV de la palabra de estado se ponen a "0".

**XOW**: combina el ACU1-L con el ACU2-L.

**XOW <constante>**: combina el ACU1-L con una constante (16 bits).

### Palabra de estado

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

## Ejemplos

Bit	15 ..	..	..	... 0
ACU 1 antes de ejecutar <b>XOW</b>	0101	0101	0011	1011
ACU2-L ó constante (16 bits):	1111	0110	1011	0101
Resultado (ACU 1) después de ejecutar <b>XOW</b>	1010	0011	1000	1110

### Ejemplo 1

AWL		Explicación
L	EW20	//Cargar el contenido de EW20 en el ACU1-L.
L	EW22	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el contenido de ED24 //en el ACU 1-L.
XOW		//Combinar lógicamente los bits del ACU1-L con los bits del ACU2-L //realizando una operación O-exclusiva, almacenar el resultado en el //ACU1-L.
T	MW8	//Transferir el resultado a MW8.

### Ejemplo 2

AWL		Explicación
L	EW20	//Cargar el contenido de EW20 en el ACU1-L.
XOW	16#0FFF	//Combinar lógicamente los bits del ACU1-L con la configuración binaria //de la constante (de 16 bits) (0000_1111_1111_1111) realizando una //operación O-exclusiva, almacenar el resultado en el ACU1-L.
SPP	NEXT	//Saltar a la meta NEXT si el resultado es diferente de cero (A1 = 1).

## 13.5 UD Y con doble palabra (32 bits)

### Formato

UD

UD <constante>

Operando	Tipo de datos	Descripción
<constante>	DWORD, constante (32 bits)	Configuración binaria a combinar lógicamente por medio de Y con el ACU 1.

### Descripción de la operación

**UD** (Y con doble palabra) combina el contenido del ACU 1 con el ACU 2 ó con una constante (32 bits) bit por bit realizando una operación lógica Y. Solamente cuando los bits correspondientes de ambas dobles palabras a combinar son "1" será también "1" el bit respectivo de la doble palabra resultante. El resultado se almacena en el ACU 1. El ACU 2 (y en las CPU con cuatro acumuladores, ACU 3 y ACU 4) no se altera. El bit de la palabra de estado A1 se activa como resultado de la operación (A1 = 1, si el resultado es diferente de cero). Los bits A0 y OV de la palabra de estado se ponen a "0".

**UD:** combina el ACU 1 con el ACU 2.

**UD <constante>:** combina el ACU 1 con una constante (32 bits).

### Palabra de estado

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

## Ejemplos

Bit	31 ..	..	..	..	..	..	..	... 0
ACU 1 antes de ejecutar UD	0101	0000	1111	1100	1000	1001	0011	1011
ACU 2 ó constante (32 bits)	1111	0011	1000	0101	0111	0110	1011	0101
Resultado (ACU 1) después de ejecutar UD	0101	0000	1000	0100	0000	0000	0011	0001

### Ejemplo 1

AWL	Explicación
L ED20	//Cargar el contenido de ED20 en el ACU 1.
L ED24	//Cargar el contenido del ACU 1 en el ACU 2. Cargar el contenido de ED24 //en el ACU 1.
UD	//Combinar los bits del ACU 1 con los bits del ACU 2 realizando una //operación Y lógica, almacenar el resultado en el ACU 1.
T MD8	//Transferir el resultado a MD8.

### Ejemplo 2

AWL	Explicación
L ED 20	//Cargar el contenido de ED20 en el ACU 1.
UD DW#16#0F FF_EF21	//Combinar los bits del ACU 1 con la configuración binaria de la //constante (32 bits) (0000_1111_1111_1111_1110_1111_0010_0001) //realizando una operación Y lógica, almacenar el resultado en el ACU 1.
JP NEXT	//Saltar a la meta NEXT si el resultado es diferente de cero (A1 = 1).

## 13.6 OD O con doble palabra (32 bits)

### Formato

OD

OD <constante>

Operando	Tipo de datos	Descripción
<constante>	DWORD, constante (32 bits)	Configuración binaria a combinar con el ACU 1 realizando una O lógica.

### Descripción de la operación

**OD** (O con doble palabra) combina el contenido del ACU 1 con ACU 2 ó con una constante (32 bits) bit por bit realizando una operación lógica O. Si al menos uno los bits correspondientes de ambas dobles palabras a combinar es "1", el bit respectivo de la doble palabra resultante también será "1". El resultado se almacena en el ACU 1. El ACU 2 (en las CPU con cuatro acumuladores también el ACU 3 y el ACU4) no se altera. El bit de la palabra de estado A1 se activa como resultado de la operación (A1 = 1, si el resultado es diferente de cero). Los bits A0 y OV de la palabra de estado se ponen a "0".

**OD:** combina el ACU 1 con el ACU 2.

**OD <constante>:** combina el ACU 1 con una constante (32 bits).

### Palabra de estado

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

## Ejemplos

Bit	31	..	..	..	..	..	..	0
ACU 1 antes de ejecutar OD	0101	0000	1111	1100	1000	0101	0011	1011
ACU 2 ó constante (32 bits)	1111	0011	1000	0101	0111	0110	1011	0101
Resultado (ACU 1) después de ejecutar OD	1111	0011	1111	1101	1111	0111	1011	1111

### Ejemplo 1

AWL		Explicación
L	ED20	//Cargar el contenido de ED20 en el ACU 1.
L	ED24	//Cargar el contenido de ACU 1 en el ACU 2. Cargar el contenido de ED24 //en el ACU 1.
OD		//Combinar los bits del ACU 1 con los bits del ACU 2 realizando una //operación O lógica, almacenar el resultado en el ACU 1.
T	MD8	//Transferir el resultado a MD8.

### Ejemplo 2

AWL		Explicación
L	ED20	//Cargar el contenido de ED20 en el ACU 1.
OD	DW#16#0F FF_EF21	//Combinar los bits del ACU 1 con la configuración binaria de la //constante (32 bits) (0000_1111_1111_1111_1110_1111_0010_0001) por //realizando una operación O lógica, almacenar el resultado en el ACU 1.
SPP	NEXT	//Saltar a la meta NEXT si el resultado es diferente de cero (A1 = 1).

## 13.7 XOD O-exclusiva con doble palabra (32 bits)

### Formato

XOD

XOD <constante>

Operando	Tipo de datos	Descripción
<constante>	DWORD, constante (32 bits)	Configuración binaria a combinar lógicamente con el ACU 1 realizando una operación O-exclusiva

### Descripción de la operación

**XOD** (O-exclusiva con doble palabra) combina el contenido del ACU 1 con ACU 2 ó con una constante (32 bits) bit por bit realizando una operación lógica O-exclusiva. Si uno -y solamente uno- de los bits correspondientes de ambas dobles palabras a combinar es "1", el bit respectivo de la doble palabra resultante también es "1". El resultado se almacena en el ACU 1. El ACU 2 (y en las CPU con cuatro acumuladores, el ACU 3 y el ACU4) no se altera. El bit de la palabra de estado A1 se activa como resultado de la operación (A1 = 1, si el resultado es diferente de cero). Los bits A0 y OV de la palabra de estado se ponen a "0".

La función O-exclusiva también se puede aplicar varias veces consecutivas. Entonces el resultado lógico común será "1" si un número impar de los operandos consultados da el resultado "1".

**XOD**: combina el ACU 1 con el ACU 2.

**XOD <constante>**: combina el ACU 1 con una constante (32 bits).

### Palabra de estado

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

## Ejemplos

Bit	31 ..	..	..	..	..	..	..	... 0
ACU 1 antes de ejecutar XOD	0101	0000	1111	1100	1000	0101	0011	1011
ACU 2 ó constante (32 bits)	1111	0011	1000	0101	0111	0110	1011	0101
Resultado (ACU 1) después de ejecuta XOD	1010	0011	0111	1001	1111	0011	1000	1110

## Ejemplo 1

AWL	Explicación
L ED20	//Cargar el contenido de ED20 en el ACU 1.
L ED24	//Cargar el contenido de ACU 1 en el ACU 2. Cargar el contenido de ED24 //en el ACU 1.
XOD	//Combinar lógicamente los bits del ACU 1 con los bits del ACU 2 //realizando una operación O-exclusiva, almacenar el resultado en el //ACU 1.
T MD8	//Transferir el resultado a MD8.

## Ejemplo 2

AWL	Explicación
L ED20	//Cargar el contenido de ED20 en el ACU 1.
XOD DW#16#0F FF_EF21	//Combinar lógicamente los bits del ACU 1 con la configuración binaria de //la constante (32 bits) (0000_1111_1111_1111_1111_1110_0010_0001) //realizando una operación O-exclusiva, almacenar el resultado en el //ACU 1.
SPP NEXT	//Saltar a la meta NEXT si el resultado es diferente de cero (A1 = 1).



# 14 Operaciones con acumuladores

## 14.1 Lista de operaciones con acumuladores

### Descripción

Para operar con el contenido de uno o varios acumuladores o registros de direcciones se dispone de las siguientes operaciones:

- TAK Intercambiar ACU 1 y ACU 2
- PUSH CPU con dos acumuladores
- PUSH CPU con cuatro acumuladores
- POP CPU con dos acumuladores
- POP CPU con cuatro acumuladores
  
- ENT Introducir pila de ACU
- LEAVE Salir de la pila de ACU
- INC Incrementar ACU 1-L-L
- DEC Decrementar ACU 1-L-L
  
- +AR1 Sumar el ACU 1 al registro de direcciones 1
- +AR2 Sumar el ACU1 al registro de direcciones 2
- BLD Estructuración de imagen (operación nula)
- NOP 0 Operación nula 0
- NOP 1 Operación nula 1

## 14.2 TAK Intercambiar ACU 1 y ACU 2

### Formato

TAK

### Descripción de la operación

**TAK** (Intercambiar ACU 1 y ACU 2) intercambia el contenido del ACU 1 con el contenido del ACU 2. La operación se ejecuta sin considerar ni afectar a los bits de la palabra de estado. Los contenidos del ACU 3 y del ACU 4 quedan inalterados (en las CPU con cuatro acumuladores).

### Palabra de estado

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

### Ejemplo: Restar el valor menor del valor mayor

AWL	Explicación
L MW10	//Cargar el contenido de MW10 en ACU1-L.
L MW12	//Cargar el contenido de ACU1-L en ACU2-L.
	//Cargar el contenido de MW12 en ACU1-L.
>I	//Comprobar si ACU2-L (MW10) es mayor que ACU1-L (MW12).
SPB NEXT	//Salta a la meta NEXT, si el ACU 2 (MW10) es mayor que el ACU 1 (MW12).
TAK	//Intercambiar los contenidos de ACU 1 y ACU 2.
NEXT: -I	//Restar el contenido de ACU1-L del contenido de ACU2-L.
T MW14	//Transferir el resultado (= valor mayor menos el valor menor) a MW14.

### Contenidos de ACU 1 y ACU 2

Contenidos	ACU 1	ACU 2
Antes de ejecutar la operación <b>TAK</b>	<MW12>	<MW10>
Después de ejecutar la operación <b>TAK</b>	<MW10>	<MW12>

## 14.3 PUSH CPU con dos acumuladores

### Formato

**PUSH**

### Descripción de la operación

**PUSH** (ACU 1 a ACU 2) copia el contenido completo del ACU 1 al ACU 2. El ACU 1 no se altera. La operación se ejecuta sin considerar ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L      MW10	//Cargar el contenido de MW10 en el ACU 1.
PUSH	//Copiar el contenido completo del ACU 1 al ACU 2.

### Contenidos de ACU 1 y ACU 2

Contenidos	ACU 1	ACU 2
Antes de ejecutar la operación <b>PUSH</b>	<MW10>	<X>
Después de ejecutar la operación <b>PUSH</b>	<MW10>	<MW10>

## 14.4 PUSH CPU con cuatro acumuladores

### Formato

PUSH

### Descripción de la operación

**PUSH** (CPU con cuatro acumuladores) copia el contenido del ACU 3 al ACU 4, el contenido del ACU 2 al ACU 3 y el contenido del ACU 1 al ACU 2. El ACU 1 no se altera. La operación se ejecuta sin considerar ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

### Ejemplo

AWL	Explicación
L MW10	//Cargar el contenido del MW10 en el ACU 1.
PUSH	//Copiar el contenido completo del ACU 1 al ACU 2, el contenido del ACU 2 //al ACU 3 y el contenido del ACU 3 al ACU 4.

### Contenidos de ACU 1 hasta ACU 4

Contenidos	ACU 1	ACU 2	ACU 3	ACU 4
Antes de ejecutar la operación <b>PUSH</b>	Valor A	Valor B	Valor C	Valor D
Después de ejecutar la operación <b>PUSH</b>	Valor A	Valor A	Valor B	Valor C

## 14.5 POP CPU con dos acumuladores

### Formato

POP

### Descripción de la operación

**POP** (CPU con dos acumuladores) copia el contenido completo del ACU 2 al ACU 1. El ACU 2 no se altera. La operación se ejecuta sin considerar ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

### Ejemplo

AWL		Explicación
//	MD10	//Transferir el contenido del ACU 1 (= valor A) a MD10.
POP		//Copiar el contenido completo del ACU 2 al ACU 1.
T	MD14	//Transferir el contenido del ACU 1 (= valor B) a MD14.

### Contenidos de ACU 1 y ACU 2

Contenidos	ACU 1	ACU 2
Antes de ejecutar la operación <b>POP</b>	Valor A	Valor B
Después de ejecutar la operación <b>POP</b>	Valor B	Valor B

## 14.6 POP CPU con cuatro acumuladores

### Formato

POP

### Descripción de la operación

**POP** (CPU con cuatro acumuladores) copia el contenido del ACU 2 al ACU 1, el contenido del ACU 3 al ACU 2 y el contenido del ACU 4 al ACU 3. El ACU 4 no se altera. La operación se ejecuta sin considerar ni afectar a los bits de la palabra de estado.

### Palabra de estado

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

### Ejemplo

AWL		Explicación
//	MD10	//Transferir el contenido del ACU 1 (= valor A) a MD10.
POP		//Copiar el contenido del ACU 2 al ACU1, el contenido del ACU 3 al ACU 2 y //el contenido del ACU 4 al ACU 3.
T	MD14	//Transferir el contenido del ACU 1 (= valor B) a MD14.

### Contenidos de ACU 1 hasta ACU 4

Contenidos	ACU 1	ACU 2	ACU 3	ACU 4
Avant exécution de l'opération <b>POP</b>	Valor A	Valor B	Valor C	Valor D
Después de ejecutar la operación <b>POP</b>	Valor B	Valor C	Valor D	Valor D

## 14.7 ENT Introducir pila de ACU

### Formato

ENT

### Descripción de la operación

**ENT** (Introducir pila de ACU) copia el contenido del ACU 3 al ACU 4 y el contenido del ACU 2 al ACU 3. Si se programa la operación ENT directamente antes de una operación de carga, se puede salvar con ello un resultado parcial en el ACU 3.

### Ejemplo

AWL		Explicación
//	DBD0	//Cargar el valor de la palabra doble de datos DBD0 en el ACU 1. (Este valor debe tener formato en coma flotante.)
L	DBD4	//Copiar el valor del ACU 1 al ACU 2. Cargar el valor de la palabra doble de datos DBD4 en el ACU 1. (Este valor debe tener formato en coma flotante.)
+R		//Sumar los contenidos del ACU 1 y el ACU 2 como números en coma flotante (32 bits, IEEE 754) y almacenar el resultado en el ACU 1.
L	DBD8	//Copiar el valor del ACU 1 al ACU 2. Cargar el valor de la palabra doble de datos DBD8 en el ACU 1.
ENT		//Copiar el contenido del ACU 3 al ACU 4. Copiar el contenido del ACU 2 (resultado parcial) al ACU 3.
L	DBD12	//Cargar el valor de la palabra doble de datos DBD12 en el ACU 1.
-R		//Restar el contenido del ACU 1 del contenido del ACU 2 y almacenar el resultado en el ACU 1. Copiar el contenido del ACU 3 en el ACU 2 y el contenido del ACU 4 al ACU 3.
/R		//Dividir el contenido del ACU 2 (DBD0 + DBD4) por el contenido del ACU 1 (DBD8 - DBD12) y almacenar el resultado en el ACU 1.
T	DBD16	//Transferir el resultado (ACU 1) a la palabra doble de datos DBD16

## 14.8 LEAVE Salir de la pila de ACU

### Formato

LEAVE

### Descripción de la operación

**LEAVE** (Salir de la pila de ACU) copia el contenido del ACU 3 al ACU 2 y el contenido del ACU 4 al ACU 3. Si se programa la operación LEAVE directamente antes de una operación de desplazamiento y rotación que combina acumuladores, entonces la operación LEAVE funciona como una operación aritmética. Los contenidos del ACU 1 y del ACU 4 quedan inalterados.

## 14.9 INC Incrementar ACU 1-L-L

### Formato

INC <entero de 8 bits>

Operando	Tipo de datos	Descripción
<entero de 8 bits>	Constante (entero de 8 bits)	Constante a sumar; margen de 0 hasta 255

### Descripción de la operación

**INC <entero de 8 bits>** (Incrementar ACU1-L-L) suma el número entero (8 bits) al contenido del ACU1-L-L y almacena el resultado en ACU1-L-L. ACU1-L-H, ACU1-H y ACU 2 no se alteran. La operación se ejecuta sin considerar ni afectar a los bits de la palabra de estado.

### Nota

Esta operación no sirve para las operaciones aritméticas de 16 ó 32 bits, puesto que no se transfiere nada desde el byte bajo de la palabra baja del ACU 1 al byte alto de la palabra baja del ACU 1. Para las operaciones aritméticas de 16 ó 32 bits hay que utilizar la operación +I o +D, respectivamente.

### Palabra de estado

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

### Ejemplo

AWL		Explicación
L	MB22	//Cargar el valor de MB22.
INC	1	//Incrementar en 1 ACU 1 (MB 22), almacenar el resultado en ACU1-L-L.
T	MB22	//Transferir el contenido de ACU1-L-L (resultado) de vuelta al MB22.

## 14.10 DEC Decrementar ACU 1-L-L

### Formato

**DEC <entero de 8 bits>**

Operando	Tipo de datos	Descripción
<entero de 8 bits>	Constante (entero de 8 bits)	Constante a restar; margen de 0 hasta 255

### Descripción de la operación

**DEC <entero de 8 bits>** (Decrementar ACU1-L-L) resta el número entero (8 bits) del contenido del ACU1-L-L y almacena el resultado en ACU1-L-L. ACU1-L-H, ACU1-H y ACU 2 no se alteran. La operación se ejecuta sin considerar ni afectar a los bits de la palabra de estado.

### Nota

Esta operación no sirve para las operaciones aritméticas de 16 ó 32 bits, puesto que no se transfiere nada desde el byte bajo de la palabra baja del ACU 1 al byte alto de la palabra baja del ACU 1. Para las operaciones aritméticas de 16 ó 32 bits hay que utilizar la operación +I o +D, respectivamente.

### Palabra de estado

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

### Ejemplo

AWL		Explicación
L	MB250	//Cargar el valor de MB250.
DEC	1	//Decrementar en 1 el ACU1-L-L, almacenar el resultado en ACU1-L-L.
T	MB250	//Transferir el contenido de ACU1-L- L (resultado) de vuelta a MB250.

## 14.11 +AR1 Sumar el ACU 1 al registro de direcciones 1

### Formatos

**+AR1**

**+AR1 <P#Byte.Bit>**

Operando	Tipo de datos	Descripción
<P#Byte.Bit>	Constante de puntero	Dirección que se suma a AR 1.

### Descripción de la operación

**+AR1** (Sumar a AR1) suma el desplazamiento que se haya indicado en la instrucción o en el ACU 1-L, al contenido de AR1. El entero (16 bits) es ampliado primero con el signo correcto a 24 bits, y luego se suma a los 24 bits menos significativos de AR1 (parte de la dirección relativa en AR1). La parte de la identificación del área en AR 1 (bits 24, 25 y 26) no se altera. La operación se ejecuta sin considerar ni afectar a los bits de la palabra de estado.

**+AR1**: El entero (16 bits) que se va a sumar al contenido de AR1 lo indica el valor de ACU1-L. Se admiten valores de -32768 hasta +32767.

**+AR1 <P#Byte.Bit>**: El desplazamiento a sumar lo indica el operando **<P#Byte.Bit>** .

### Palabra de estado

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

### Ejemplo 1

AWL	Explicación
L +300	//Cargar el valor en el ACU1-L.
+AR1	//Sumar el ACU1-L (entero) a AR 1.

### Ejemplo 2

AWL	Explicación
+AR1 P#300.0	//Sumar el desplazamiento 300.0 a AR 1.

## 14.12 +AR2 Sumar el ACU1 al registro de direcciones 2

### Formatos

**+AR2**

**+AR2 <P#Byte.Bit>**

Operando	Tipo de datos	Descripción
<P#Byte.Bit>	Constante de puntero	Dirección que se suma a AR 2.

### Descripción de la operación

**+AR2** (Sumar a AR2) suma el desplazamiento que se haya indicado en la instrucción o en el ACU1-L, al contenido de AR2. El entero (16 bits) es ampliado primero con el signo correcto a 24 bits, y luego se suma a los 24 bits menos significativos de AR 2 (parte de la dirección relativa en AR2). La parte de la identificación de área en AR2 (bits 24, 25 y 26) no se altera. La operación se ejecuta sin considerar ni afectar a los bits de la palabra de estado.

**+AR2:** El entero (16 bits) a sumar al contenido de AR2 lo indica el valor del ACU1-L. Está permitido utilizar valores entre -32768 y +32767.

**+AR2 <P#Byte.Bit>:** El desplazamiento a sumarse es indicado por el operando <P#Byte.Bit>.

### Palabra de estado

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

### Ejemplo 1

AWL	Explicación
L +300	//Cargar el valor en el ACU1-L.
+AR2	//Sumar el ACU1-L (entero) a AR 2.

### Ejemplo 2

AWL	Explicación
+AR2 P#300.0	//Sumar el desplazamiento 300.0 a AR 2.

## 14.13 BLD Estructuración de imagen (operación nula)

### Formato

BLD <número>

Operando	Descripción
<número>	Número de identificación de la operación BLD; margen de 0 hasta 255

### Descripción de la operación

**BLD <número>** (Estructuración de imagen; operación nula) no ejecuta ninguna función y tampoco afecta a los bits de la palabra de estado. La operación sirve de ayuda a la unidad de programación (PG) para estructurar la imagen de forma gráfica, la cual se genera de forma automática cuando se visualiza un programa KOP o FUP en AWL. El operando <número> es el número de identificación de la operación BLD y es generado por la unidad de programación.

### Palabra de estado

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

## 14.14 NOP 0 Operación nula 0

### Formato

NOP 0

### Descripción de la operación

**NOP 0** (Operación NOP con el operando "0") no ejecuta ninguna función y tampoco afecta a los bits de la palabra de estado. El código de operación contiene una configuración binaria con 16 ceros. La operación sólo tiene importancia para la unidad de programación (PG) cuando se visualiza un programa.

### Palabra de estado

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

## 14.15 NOP 1 Operación nula 1

### Formato

NOP 1

### Descripción de la operación

**NOP 1** (Operación NOP con el operando "1") no ejecuta ninguna función y tampoco afecta a los bits de la palabra de estado. El código de operación contiene una configuración binaria con 16 unos. La operación sólo tiene importancia para la unidad de programación (PG) cuando se visualiza un programa.

### Palabra de estado

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



## A Sinopsis de las operaciones AWL

### A.1 Operaciones AWL ordenadas según la nemotécnica alemana (SIMATIC)

Nemo-técnica alemana	Nemo-técnica inglesa	Catálogo de elementos del programa	Descripción
=	=	Operaciones lógicas con bits	Asignar
)	)	Operaciones lógicas con bits	Cerrar paréntesis
*D	*D	Función en coma fija	Multiplicar ACU 1 por ACU 2 como entero doble
*I	*I	Función en coma fija	Multiplicar ACU 1 por ACU 2 como entero
*R	*R	Función en coma flotante	Multiplicar ACU 1 por ACU 2 como número de coma flotante (32 bits)
/D	/D	Función en coma fija	Dividir ACU 2 por ACU 1 como entero doble
/I	/I	Función en coma fija	Dividir ACU 2 por ACU 1 como entero
/R	/R	Función en coma flotante	Dividir ACU 2 por ACU 1 como número de coma flotante (32 bits)
? D	? D	Comparador	Comparar enteros dobles ==, <>, >, <, >=, <=
? I	? I	Comparador	Comparar enteros ==, <>, >, <, >=, <=
? R	? R	Comparador	Comparar números en coma flotante (32 bits) ==, <>, >, <, >=, <=
+	+	Función en coma fija	Sumar constante entera o entera doble
+AR1	+AR1	Acumulador	Sumar el ACU 1 al registro de direcciones 1
+AR2	+AR2	Acumulador	Sumar el ACU1 al registro de direcciones 2
+D	+D	Función en coma fija	Sumar ACU 1 y 2 como entero doble
+I	+I	Función en coma fija	Sumar ACU 1 y 2 como entero
+R	+R	Función en coma flotante	Sumar ACU 1 y 2 como número de coma flotante (32 bits)
ABS	ABS	Función en coma flotante	Valor absoluto de un número de coma flotante (32 bits, IEEE 754)
ACOS	ACOS	Función en coma flotante	Calcular el arcocoseno de un número de coma flotante (32 bits)
ASIN	ASIN	Función en coma flotante	Calcular el arcoseno de un número de coma flotante (32 bits)
ATAN	ATAN	Función en coma flotante	Calcular la arcotangente de un número de coma flotante (32 bits)
AUF	OPN	Bloque de datos	Abrir bloque de datos
BE	BE	Control del programa	Fin de bloque
BEA	BEU	Control del programa	Fin de bloque incondicionado
BEB	BEC	Control del programa	Fin de bloque condicionado
BLD	BLD	Acumulador	Estructuración de imagen (operación nula)
BTD	BTD	Convertidor	Convertir número BCD a entero doble
BTI	BTI	Convertidor	Convertir BCD a entero
CALL	CALL	Control del programa	Llamada
CALL	CALL	Control del programa	Llamar a una multiinstancia

A.1 Operaciones AWL ordenadas según la nemotécnica alemana (SIMATIC) Operaciones con acumuladores

Nemo-técnica alemana	Nemo-técnica inglesa	Catálogo de elementos del programa	Descripción
CALL	CALL	Control del programa	Llamar a un bloque de una librería
CC	CC	Control del programa	Llamada condicionada
CLR	CLR	Operaciones lógicas con bits	Desactivar RLO (=0)
COS	COS	Función en coma flotante	Calcular el coseno de ángulos como números de coma flotante (32 bits)
-D	-D	Función en coma fija	Restar ACU 1 de ACU 2 como entero doble
DEC	DEC	Acumulador	Decrementar ACU 1-L-L
DTB	DTB	Convertidor	Convertir entero doble en BCD
DTR	DTR	Convertidor	Convertir entero doble en número en coma flotante (32 bits, IEEE 754)
ENT	ENT	Acumulador	Introducir pila de ACU
EXP	EXP	Función en coma flotante	Calcular el exponente de un número de coma flotante (32 bits)
FN	FN	Operaciones lógicas con bits	Flanco negativo
FP	FP	Operaciones lógicas con bits	Flanco positivo
FR	FR	Contadores	Habilitar contador (Frei, FR Z 0 zu Z 255)
FR	FR	Temporizadores	Habilitar temporizador
-I	-I	Función en coma fija	Restar ACU 1 de ACU 2 como entero
INC	INC	Acumulador	Incrementar ACU 1-L-L
INVD	INVD	Convertidor	Complemento a uno de un entero doble
INVI	INVI	Convertidor	Complemento a uno de un entero
ITB	ITB	Convertidor	Convertir entero en BCD
ITD	ITD	Convertidor	Convertir entero en entero doble
L	L	Cargar/Transferir	Cargar
L STW	L STW	Cargar/Transferir	Cargar palabra de estado en ACU 1
L	L	Temporizadores	Cargar valor actual del temporizador en ACU 1 como entero (el valor de temporización actual puede ser un valor comprendido en el margen de 0 a 255, p. ej., L T 32)
L	L	Contadores	Cargar valor actual del contador en ACU 1 como número BCD (el valor de contaje actual puede ser un valor comprendido en el margen de 0 a 255, p. ej., L Z 15)
L DBLG	L DBLG	Bloque de datos	Cargar la longitud del DB global en el ACU 1
L DBNO	L DBNO	Bloque de datos	Cargar número del bloque de datos global en ACU 1
L DILG	L DILG	Bloque de datos	Cargar longitud del bloque de datos de instancia en ACU 1
L DINO	L DINO	Bloque de datos	Cargar número del bloque de datos de instancia en ACU 1
LAR1	LAR1	Cargar/Transferir	Cargar registro de direcciones 1 con contenido del ACU 1
LAR1	LAR1	Cargar/Transferir	Cargar registro de direcciones 1 con puntero (formato de 32 bits)
LAR1	LAR1	Cargar/Transferir	Cargar registro de direcciones 1 con contenido del registro de direcciones 2
LAR2	LAR2	Cargar/Transferir	Cargar registro de direcciones 2 con contenido del ACU 1
LAR2	LAR2	Cargar/Transferir	Cargar registro de direcciones 2 con puntero (formato de 32 bits)
LC	LC	Contadores	Cargar valor actual del contador en ACU 1 como número BCD (el valor de contaje actual puede ser un número comprendido en el margen de 0 a 255, p. ej. LC T 32)

## A.1 Operaciones AWL ordenadas según la nemotécnica alemana (SIMATIC) Operaciones con acumuladores

Nemo-técnica alemana	Nemo-técnica inglesa	Catálogo de elementos del programa	Descripción
LC	LC	Temporizadores	Cargar el valor actual de temporización en ACU 1 como número BCD (el valor de temporización actual puede ser un número comprendido en el margen de 0 a 255, p. ej.: LC T 32)
LEAVE	LEAVE	Acumulador	Salir de la pila de ACU
LN	LN	Función en coma flotante	Calcular el logaritmo natural de un número de coma flotante (32 bits)
LOOP	LOOP	Saltos	Bucle
MCR(	MCR(	Control del programa	Almacenar el RLO en pila MCR, inicio área MCR
)MCR	)MCR	Control del programa	Fin área MCR
MCRA	MCRA	Control del programa	Activar área MCR
MCRD	MCRD	Control del programa	Desactivar área MCR
MOD	MOD	Función en coma fija	Resto de la división de enteros dobles
NEGD	NEGD	Convertidor	Complemento a dos de un entero doble
NEGI	NEGI	Convertidor	Complemento a dos de un entero
NEGR	NEGR	Convertidor	Invertir un número en coma flotante (32 bits, IEEE 754)
NOP 0	NOP 0	Acumulador	Operación nula 0
NOP 1	NOP 1	Acumulador	Operación nula 1
NOT	NOT	Operaciones lógicas con bits	Negar el RLO
O	O	Operaciones lógicas con bits	O
O(	O(	Operaciones lógicas con bits	O con abrir paréntesis
OD	OD	Bits Operaciones lógicas con palabras	O con doble palabra (32 bits)
ON	ON	Operaciones lógicas con bits	O-No
ON(	ON(	Operaciones lógicas con bits	O-No con abrir paréntesis
OW	OW	Bits Operaciones lógicas con palabras	O con palabra (16 bits)
POP	POP	Acumulador	CPU con dos acumuladores
POP	POP	Acumulador	CPU con cuatro acumuladores
PUSH	PUSH	Acumulador	CPU con dos acumuladores
PUSH	PUSH	Acumulador	CPU con cuatro acumuladores
R	R	Operaciones lógicas con bits	Desactivar
R	R	Contadores	Desactivar contador (el valor de temporización actual puede ser un número comprendido en el margen de 0 a 255, p. ej.: R Z 15)
R	R	Temporizadores	Desactivar temporizador (el temporizador actual puede ser un número comprendido en el margen de 0 a 255, p. ej.: R T 32)
-R	-R	Función en coma flotante	Restar ACU 1 de ACU 2 como número de coma flotante (32 bits)
RLD	RLD	Desplazar/Rotar	Rotar doble palabra a la izquierda (32 bits)
RLDA	RLDA	Desplazar/Rotar	Rotar ACU 1 a la izquierda vía A1 (32 bits)
RND	RND	Convertidor	Redondear un número en coma flotante a entero
RND-	RND-	Convertidor	Redondear un número real al próximo entero inferior
RND+	RND+	Convertidor	Redondear un número real al próximo entero superior
RRD	RRD	Desplazar/Rotar	Rotar doble palabra a la derecha (32 bits)

A.1 Operaciones AWL ordenadas según la nemotécnica alemana (SIMATIC) Operaciones con acumuladores

Nemo-técnica alemana	Nemo-técnica inglesa	Catálogo de elementos del programa	Descripción
RRDA	RRDA	Desplazar/Rotar	Rotar ACU 1 a la derecha vía A1 (32 bits)
S	S	Operaciones lógicas con bits	Activar
S	S	Contadores	Poner contador al valor inicial (el contador actual puede ser un número comprendido en el margen de 0 a 255, p. ej.: S Z 15)
SA	SF	Temporizadores	Temporizador como retardo a la desconexión
SAVE	SAVE	Operaciones lógicas con bits	Memorizar el RLO en el registro RB
SE	SD	Temporizadores	Temporizador como retardo a la conexión
SET	SET	Operaciones lógicas con bits	Activar
SI	SP	Temporizadores	Temporizador como impulso
SIN	SIN	Función en coma flotante	Calcular el seno de ángulos como números de coma flotante (32 bits)
SLD	SLD	Desplazar/Rotar	Desplazar doble palabra a la izquierda (32 bits)
SLW	SLW	Desplazar/Rotar	Desplazar palabra a la izquierda (16 bits)
SPA	JU	Salto	Salto incondicionado
SPB	JC	Salto	Saltar si RLO = 1
SPBB	JCB	Salto	Saltar si RLO = 1 y salvaguardar RLO en RB
SPBI	JB	Salto	Saltar si RB = 1
SPBIN	JNB	Salto	Saltar si RB = 0
SPBN	JCN	Salto	Saltar si RLO = 0
SPBNB	JNB	Salto	Saltar si RLO = 0 y salvar RLO en RB
SPL	JL	Salto	Saltar utilizando una lista de metas
SPM	JM	Salto	Saltar si resultado < 0
SPMZ	JMZ	Salto	Saltar si el resultado <= 0
SPN	JN	Salto	Saltar si resultado <> 0
SPO	JO	Salto	Saltar si OV = 1
SPP	JP	Salto	Saltar si el resultado > 0
SPPZ	JPZ	Salto	Saltar si el resultado >= 0
SPS	JOS	Salto	Saltar si OS = 1
SPU	JUO	Salto	Saltar si el resultado no es válido
SPZ	JZ	Salto	Saltar si el resultado = 0
SQR	SQR	Función en coma flotante	Calcular el cuadrado de un número de coma flotante (32 bits)
SQRT	SQRT	Función en coma flotante	Calcular la raíz cuadrada de un número de coma flotante (32 bits)
SRD	SRD	Desplazar/Rotar	Desplazar doble palabra a la derecha (32 bits)
SRW	SRW	Desplazar/Rotar	Desplazar palabra a la derecha (16 bits)
SS	SS	Temporizadores	Temporizador como retardo a la conexión con memoria
SSD	SSD	Desplazar/Rotar	Desplazar signo de número entero a la derecha (32 bits)
SSI	SSI	Desplazar/Rotar	Desplazar signo de número entero a la derecha (16 bits)
SV	SE	Temporizadores	Temporizador como impulso prolongado
T	T	Cargar/Transferir	Transferir
T STW	T STW	Cargar/Transferir	Transferir ACU 1 a la palabra de estado
TAD	CAD	Convertidor	Invertir el orden de los bytes en el ACU 1 (32 bits)

## A.1 Operaciones AWL ordenadas según la nemotécnica alemana (SIMATIC) Operaciones con acumuladores

Nemo-técnica alemana	Nemo-técnica inglesa	Catálogo de elementos del programa	Descripción
TAK	TAK	Acumulador	Intercambiar ACU 1 y ACU 2
TAN	TAN	Función en coma flotante	Calcular la tangente de ángulos como números de coma flotante (32 bits)
TAR	CAR	Cargar/Transferir	Intercambiar registro de direcciones 1 y registro de direcciones 2
TAR1	TAR1	Cargar/Transferir	Transferir registro de direcciones 1 a ACU 1
TAR1	TAR1	Cargar/Transferir	Transferir registro de direcciones 1 a dirección de destino (puntero de 32 bits)
TAR1	TAR1	Cargar/Transferir	Transferir registro de direcciones 1 a registro de direcciones 2
TAR2	TAR2	Cargar/Transferir	Transferir registro de direcciones 2 a ACU 1
TAR2	TAR2	Cargar/Transferir	Transferir registro de direcciones 2 a dirección de destino (puntero de 32 bits)
TAW	CAW	Convertidor	Cambiar el orden de los bytes en el ACU 1-L (16 bits)
TDB	CDB	Bloque de datos	Intercambiar bloque de datos global y bloque de datos de instancia
TRUNC	TRUNC	Convertidor	Truncar
U	A	Operaciones lógicas con bits	Y
U(	A(	Operaciones lógicas con bits	Y con abrir paréntesis
UC	UC	Control del programa	Llamada incondicionada
UD	AD	Bits Operaciones lógicas con palabras	Y con doble palabra (32 bits)
UN	AN	Operaciones lógicas con bits	Y-No
UN(	AN(	Operaciones lógicas con bits	Y-No con abrir paréntesis
UW	AW	Bits Operaciones lógicas con palabras	Y con palabra (16 bits)
X	X	Operaciones lógicas con bits	O-exclusiva
X(	X(	Operaciones lógicas con bits	O-exclusiva con abrir paréntesis
XN	XN	Operaciones lógicas con bits	O-exclusiva-NO
XN(	XN(	Operaciones lógicas con bits	O-exclusiva-NO con abrir paréntesis
XOD	XOD	Bits Operaciones lógicas con palabras	O-exclusiva con doble palabra (32 bits)
XOW	XOW	Bits Operaciones lógicas con palabras	O-exclusiva con palabra (16 bits)
ZR	CD	Contadores	Decrementar contador
ZV	CU	Contadores	Incrementar contador

## A.2 Operaciones AWL ordenadas según la nemotécnica inglesa (internacional)

Nemo-técnica inglesa	Nemo-técnica alemana	Catálogo de elementos del programa	Descripción
=	=	Operaciones lógicas con bits	Asignar
)	)	Operaciones lógicas con bits	Cerrar paréntesis
+	+	Función en coma fija	Sumar constante entera o entera doble
*D	*D	Función en coma fija	Multiplicar ACU 1 por ACU 2 como entero doble
*I	*I	Función en coma fija	Multiplicar ACU 1 por ACU 2 como entero
*R	*R	Función en coma flotante	Multiplicar ACU 1 por ACU 2 como número de coma flotante (32 bits)
/D	/D	Función en coma fija	Dividir ACU 2 por ACU 1 como entero doble
/I	/I	Función en coma fija	Dividir ACU 2 por ACU 1 como entero
/R	/R	Función en coma flotante	Dividir ACU 2 por ACU 1 como número de coma flotante (32 bits)
? D	? D	Comparador	Comparar enteros dobles ==, <>, >, <, >=, <=
? I	? I	Comparador	Comparar enteros ==, <>, >, <, >=, <=
? R	? R	Comparador	Comparar números en coma flotante (32 bits) ==, <>, >, <, >=, <=
+AR1	+AR1	Acumulador	Sumar el ACU 1 al registro de direcciones 1
+AR2	+AR2	Acumulador	Sumar el ACU1 al registro de direcciones 2
+D	+D	Función en coma fija	Sumar ACU 1 y 2 como entero doble
+I	+I	Función en coma fija	Sumar ACU 1 y 2 como entero
+R	+R	Función en coma flotante	Sumar ACU 1 y 2 como número de coma flotante (32 bits)
A	U	Operaciones lógicas con bits	Y
A(	U(	Operaciones lógicas con bits	Y con abrir paréntesis
ABS	ABS	Función en coma flotante	Valor absoluto de un número de coma flotante (32 bits, IEEE 754)
ACOS	ACOS	Función en coma flotante	Calcular el arcocoseno de un número de coma flotante (32 bits)
AD	UD	Bits Operaciones lógicas con palabras	Y con doble palabra (32 bits)
AN	UN	Operaciones lógicas con bits	Y-No
AN(	UN(	Operaciones lógicas con bits	Y-No con abrir paréntesis
ASIN	ASIN	Función en coma flotante	Calcular el arcoseno de un número de coma flotante (32 bits)
ATAN	ATAN	Función en coma flotante	Calcular la arcotangente de un número de coma flotante (32 bits)
AW	UW	Bits Operaciones lógicas con palabras	Y con palabra (16 bits)
BE	BE	Control del programa	Fin de bloque
BEC	BEB	Control del programa	Fin de bloque condicionado
BEU	BEA	Control del programa	Fin de bloque incondicionado
BLD	BLD	Acumulador	Estructuración de imagen (operación nula)
BTD	BTD	Convertidor	Convertir número BCD a entero doble
BTI	BTI	Convertidor	Convertir BCD a entero
CAD	TAD	Convertidor	Invertir el orden de los bytes en el ACU 1 (32 bits)

## A.2 Operaciones AWL ordenadas según la nemotécnica inglesa (internacional) Operaciones con acumuladores

Nemo-técnica inglesa	Nemo-técnica alemana	Catálogo de elementos del programa	Descripción
CALL	CALL	Control del programa	Llamada
CALL	CALL	Control del programa	Llamar a una multiinstancia
CALL	CALL	Control del programa	Llamar a un bloque de una librería
CAR	TAR	Cargar/Transferir	Intercambiar registro de direcciones 1 y registro de direcciones 2
CAW	TAW	Convertidor	Cambiar el orden de los bytes en el ACU 1-L (16 bits)
CC	CC	Control del programa	Llamada condicionada
CD	ZR	Contadores	Decrementar contador
CDB	TDB	Bloque de datos	Intercambiar bloque de datos global y bloque de datos de instancia
CLR	CLR	Operaciones lógicas con bits	Desactivar RLO (=0)
COS	COS	Función en coma flotante	Calcular el coseno de ángulos como números de coma flotante (32 bits)
CU	ZV	Contadores	Incrementar contador
-D	-D	Función en coma fija	Restar ACU 1 de ACU 2 como entero doble
DEC	DEC	Acumulador	Decrementar ACU 1-L-L
DTB	DTB	Convertidor	Convertir entero doble en BCD
DTR	DTR	Convertidor	Convertir entero doble en número en coma flotante (32 bits, IEEE 754)
ENT	ENT	Acumulador	Introducir pila de ACU
EXP	EXP	Función en coma flotante	Calcular el exponente de un número de coma flotante (32 bits)
FN	FN	Operaciones lógicas con bits	Flanco negativo
FP	FP	Operaciones lógicas con bits	Flanco positivo
FR	FR	Contadores	Habilitar contador (Frei, FR Z 0 zu Z 255)
FR	FR	Temporizadores	Habilitar temporizador
-I	-I	Función en coma fija	Restar ACU 1 de ACU 2 como entero
INC	INC	Acumulador	Incrementar ACU 1-L-L
INVD	INVD	Convertidor	Complemento a uno de un entero doble
INVI	INVI	Convertidor	Complemento a uno de un entero
ITB	ITB	Convertidor	Convertir entero en BCD
ITD	ITD	Convertidor	Convertir entero en entero doble
JBI	SPBI	Saltos	Saltar si RB = 1
JC	SPB	Saltos	Saltar si RLO = 1
JCB	SPBB	Saltos	Saltar si RLO = 1 y salvaguardar RLO en RB
JCN	SPBN	Saltos	Saltar si RLO = 0
JL	SPL	Saltos	Saltar utilizando una lista de metas
JM	SPM	Saltos	Saltar si resultado < 0
JMZ	SPMZ	Saltos	Saltar si el resultado <= 0
JN	SPN	Saltos	Saltar si resultado <> 0
JNB	SPBNB	Saltos	Saltar si RLO = 0 y salvar RLO en RB
JNBI	SPBIN	Saltos	Saltar si RB = 0
JO	SPO	Saltos	Saltar si OV = 1

Nemo-técnica inglesa	Nemo-técnica alemana	Catálogo de elementos del programa	Descripción
JOS	SPS	Salto	Saltar si OS = 1
JP	SPP	Salto	Saltar si el resultado > 0
JPZ	SPPZ	Salto	Saltar si el resultado >= 0
JU	SPA	Salto	Salto incondicionado
JUO	SPU	Salto	Saltar si el resultado no es válido
JZ	SPZ	Salto	Saltar si el resultado = 0
L	L	Cargar/Transferir	Cargar
L STW	L STW	Cargar/Transferir	Cargar palabra de estado en ACU 1
L	L	Temporizadores	Cargar valor actual del temporizador en ACU 1 como entero (el valor de temporización actual puede ser un valor comprendido en el margen de 0 a 255, p. ej., L T 32)
L	L	Contadores	Cargar valor actual del contador en ACU 1 como número BCD (el valor de contaje actual puede ser un valor comprendido en el margen de 0 a 255, p. ej., L Z 15)
L DBLG	L DBLG	Bloque de datos	Cargar la longitud del DB global en el ACU 1
L DBNO	L DBNO	Bloque de datos	Cargar número del bloque de datos global en ACU 1
L DILG	L DILG	Bloque de datos	Cargar longitud del bloque de datos de instancia en ACU 1
L DINO	L DINO	Bloque de datos	Cargar número del bloque de datos de instancia en ACU 1
LAR1	LAR1	Cargar/Transferir	Cargar registro de direcciones 1 con contenido del ACU 1
LAR1	LAR1	Cargar/Transferir	Cargar registro de direcciones 1 con puntero (formato de 32 bits)
LAR1	LAR1	Cargar/Transferir	Cargar registro de direcciones 1 con contenido del registro de direcciones 2
LAR2	LAR2	Cargar/Transferir	Cargar registro de direcciones 2 con contenido del ACU 1
LAR2	LAR2	Cargar/Transferir	Cargar registro de direcciones 2 con puntero (formato de 32 bits)
LC	LC	Contadores	Cargar valor actual del contador en ACU 1 como número BCD (el valor de contaje actual puede ser un número comprendido en el margen de 0 a 255, p. ej. LC T 32)
LC	LC	Temporizadores	Cargar el valor actual de temporización en ACU 1 como número BCD (el valor de temporización actual puede ser un número comprendido en el margen de 0 a 255, p. ej.: LC T 32)
LEAVE	LEAVE	Acumulador	Salir de la pila de ACU
LN	LN	Función en coma flotante	Calcular el logaritmo natural de un número de coma flotante (32 bits)
LOOP	LOOP	Salto	Bucle
MCR(	MCR(	Control del programa	Almacenar el RLO en pila MCR, inicio área MCR
)MCR	)MCR	Control del programa	Fin área MCR
MCRA	MCRA	Control del programa	Activar área MCR
MCRD	MCRD	Control del programa	Desactivar área MCR
MOD	MOD	Función en coma fija	Resto de la división de enteros dobles
NEGD	NEGD	Convertidor	Complemento a dos de un entero doble
NEGI	NEGI	Convertidor	Complemento a dos de un entero
NEGR	NEGR	Convertidor	Invertir un número en coma flotante (32 bits, IEEE 754)
NOP 0	NOP 0	Acumulador	Operación nula 0

## A.2 Operaciones AWL ordenadas según la nemotécnica inglesa (internacional) Operaciones con acumuladores

Nemo-técnica inglesa	Nemo-técnica alemana	Catálogo de elementos del programa	Descripción
NOP 1	NOP 1	Acumulador	Operación nula 1
NOT	NOT	Operaciones lógicas con bits	Negar el RLO
O	O	Operaciones lógicas con bits	O
O(	O(	Operaciones lógicas con bits	O con abrir paréntesis
OD	OD	Bits Operaciones lógicas con palabras	O con doble palabra (32 bits)
ON	ON	Operaciones lógicas con bits	O-No
ON(	ON(	Operaciones lógicas con bits	O-No con abrir paréntesis
OPN	AUF	Bloque de datos	Abrir bloque de datos
OW	OW	Bits Operaciones lógicas con palabras	O con palabra (16 bits)
POP	POP	Acumulador	CPU con dos acumuladores
POP	POP	Acumulador	CPU con cuatro acumuladores
PUSH	PUSH	Acumulador	CPU con dos acumuladores
PUSH	PUSH	Acumulador	CPU con cuatro acumuladores
R	R	Operaciones lógicas con bits	Desactivar
R	R	Contadores	Desactivar contador (el valor de temporización actual puede ser un número comprendido en el margen de 0 a 255, p. ej.: R Z 15)
R	R	Temporizadores	Desactivar temporizador (el temporizador actual puede ser un número comprendido en el margen de 0 a 255, p. ej.: R T 32)
-R	-R	Función en coma flotante	Restar ACU 1 de ACU 2 como número de coma flotante (32 bits)
RLD	RLD	Desplazar/Rotar	Rotar doble palabra a la izquierda (32 bits)
RLDA	RLDA	Desplazar/Rotar	Rotar ACU 1 a la izquierda vía A1 (32 bits)
RND	RND	Convertidor	Redondear un número en coma flotante a entero
RND-	RND-	Convertidor	Redondear un número real al próximo entero inferior
RND+	RND+	Convertidor	Redondear un número real al próximo entero superior
RRD	RRD	Desplazar/Rotar	Rotar doble palabra a la derecha (32 bits)
RRDA	RRDA	Desplazar/Rotar	Rotar ACU 1 a la derecha vía A1 (32 bits)
S	S	Operaciones lógicas con bits	Activar
S	S	Contadores	Poner contador al valor inicial (el contador actual puede ser un número comprendido en el margen de 0 a 255, p. ej.: S Z 15)
SAVE	SAVE	Operaciones lógicas con bits	Memorizar el RLO en el registro RB
SD	SE	Temporizadores	Temporizador como retardo a la conexión
SE	SV	Temporizadores	Temporizador como impulso prolongado
SET	SET	Operaciones lógicas con bits	Activar
SF	SA	Temporizadores	Temporizador como retardo a la desconexión
SIN	SIN	Función en coma flotante	Calcular el seno de ángulos como números de coma flotante (32 bits)
SLD	SLD	Desplazar/Rotar	Desplazar doble palabra a la izquierda (32 bits)
SLW	SLW	Desplazar/Rotar	Desplazar palabra a la izquierda (16 bits)
SP	SI	Temporizadores	Temporizador como impulso
SQR	SQR	Función en coma flotante	Calcular el cuadrado de un número de coma flotante (32 bits)

Nemo-técnica inglesa	Nemo-técnica alemana	Catálogo de elementos del programa	Descripción
SQRT	SQRT	Función en coma flotante	Calcular la raíz cuadrada de un número de coma flotante (32 bits)
SRD	SRD	Desplazar/Rotar	Desplazar doble palabra a la derecha (32 bits)
SRW	SRW	Desplazar/Rotar	Desplazar palabra a la derecha (16 bits)
SS	SS	Temporizadores	Temporizador como retardo a la conexión con memoria
SSD	SSD	Desplazar/Rotar	Desplazar signo de número entero a la derecha (32 bits)
SSI	SSI	Desplazar/Rotar	Desplazar signo de número entero a la derecha (16 bits)
T	T	Cargar/Transferir	Transferir
T STW	T STW	Cargar/Transferir	Transferir ACU 1 a la palabra de estado
TAK	TAK	Acumulador	Intercambiar ACU 1 y ACU 2
TAN	TAN	Función en coma flotante	Calcular la tangente de ángulos como números de coma flotante (32 bits)
TAR1	TAR1	Cargar/Transferir	Transferir registro de direcciones 1 a ACU 1
TAR1	TAR1	Cargar/Transferir	Transferir registro de direcciones 1 a dirección de destino (puntero de 32 bits)
TAR1	TAR1	Cargar/Transferir	Transferir registro de direcciones 1 a registro de direcciones 2
TAR2	TAR2	Cargar/Transferir	Transferir registro de direcciones 2 a ACU 1
TAR2	TAR2	Cargar/Transferir	Transferir registro de direcciones 2 a dirección de destino (puntero de 32 bits)
TRUNC	TRUNC	Convertidor	Truncar
UC	UC	Control del programa	Llamada incondicionada
X	X	Operaciones lógicas con bits	O-exclusiva
X(	X(	Operaciones lógicas con bits	O-exclusiva con abrir paréntesis
XN	XN	Operaciones lógicas con bits	O-exclusiva-NO
XN(	XN(	Operaciones lógicas con bits	O-exclusiva-NO con abrir paréntesis
XOD	XOD	Bits Operaciones lógicas con palabras	O-exclusiva con doble palabra (32 bits)
XOW	XOW	Bits Operaciones lógicas con palabras	O-exclusiva con palabra (16 bits)

## B Ejemplos de programación

### B.1 Lista de ejemplos de programación

#### Aplicaciones prácticas

Todas las instrucciones AWL 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

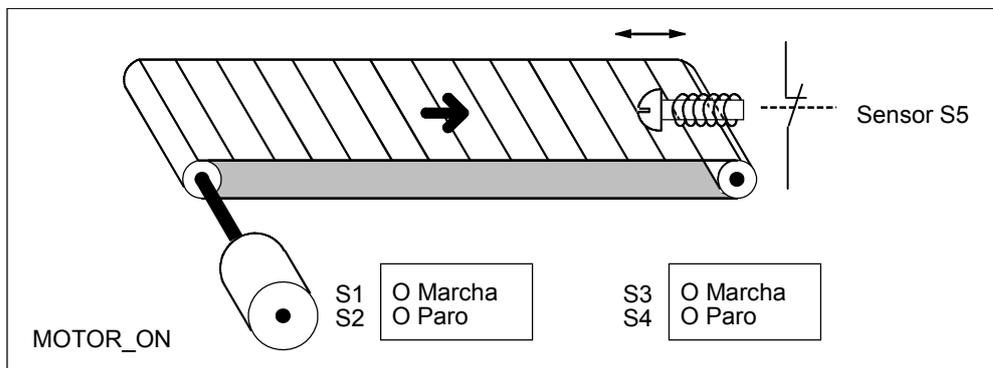
#### Operaciones utilizadas

Nemotécnica alemana	Operación	Descripción
UW	Lógica de palabras	Y con palabra
OW	Lógica de palabras	O con palabra
ZV, ZR	Contadores	Decrementar contador, Incrementar contador
S, R	Operaciones lógicas con bits	Activar, Desactivar
NOT	Operaciones lógicas con bits	Negar RLO
FP	Operaciones lógicas con bits	Flanco positivo
+I	Función en coma fija	Sumar ACU 1 y ACU 2 como entero
/I	Función en coma fija	Dividir ACU 2 por ACU 1 como entero
*I	Función en coma fija	Multiplicar ACU 1 y ACU 2 como entero
>=I, <=I	Comparadores	Comparar enteros
U, UN	Operaciones lógicas con bits	Y, Y no
O, ON	Operaciones lógicas con bits	O, O no
=	Operaciones lógicas con bits	Asignar
INC	Acumuladores	Incrementar ACU 1
BE, BEB	Control del programa	Fin de bloque, Fin de bloque condicionado
L, T	Carga/Transferencia	Cargar, Transferir
SV	Temporizadores	Arrancar temporizador como 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

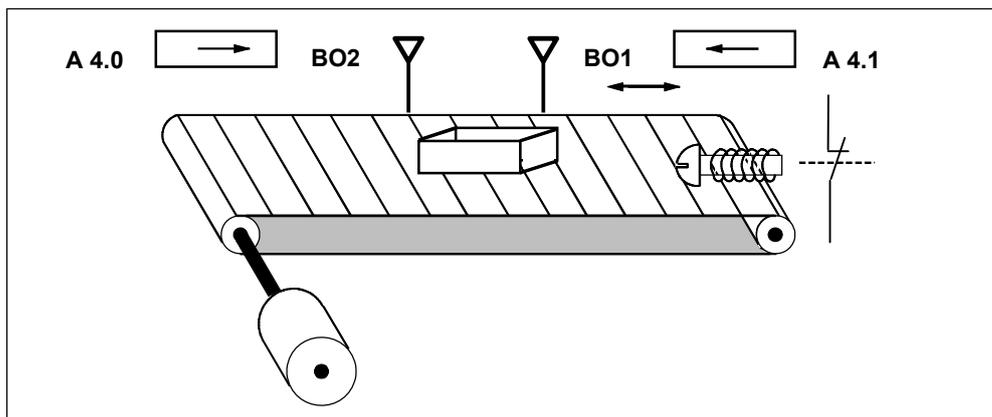
Programación absoluta	Programación simbólica
O E 1.1	O S1
O E 1.3	O S3
S A 4.0	S MOTOR_ON
O E 1.2	O S2
O E 1.4	O S4
ON E 1.5	ON S5
R A 4.0	R MOTOR_ON

### Operación AWL para controlar una cinta transportadora

AWL	Explicación
O E 1.1	//Accionando cualquiera de los pulsadores start arranca el motor.
O E 1.3	
S A 4.0	
O E 1.2	//Accionando cualquiera de los pulsadores stop o abriendo el contacto //normalmente cerrado al final de la cinta separa el motor.
O E 1.4	
ON E 1.5	
R A 4.0	

### 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

Programación absoluta	Programación simbólica
U E 0.0	U BO1
FP M 0.0	FP MI1
UN E 0.1	UN MI 2
S A 4.1	S IZQ
U E 0.1	U BO1
FP M 0.1	FP MI 2
UN E 0.0	UN BO1
S A 4.0	S DER
UN E 0.0	UN BO1
UN E 0.1	UN BO2
R A 4.0	R DER
R A 4.1	R IZQ

### Operación AWL para detectar el sentido de marcha de una cinta transportadora

AWL	Explicación
U E 0.0	//Si el estado de señal cambia de 0 a 1 (flanco positivo) en la entrada E 0.0 //y, al mismo tiempo, el estado de señal de la entrada E 0.1 es 0, significa //que el paquete que transporta la cinta se está moviendo a la izquierda.
FP M 0.0	
UN E 0.1	
S A 4.1	
U E 0.1	//Si el estado de señal cambia de 0 a 1 (flanco positivo) en la entrada E 1.0 //y, al mismo tiempo, el estado de señal de la entrada E 0.0 es 0, significa //que el paquete que transporta la cinta se está moviendo a la derecha. Si una //de las barreras ópticas se interrumpe, significa que hay un paquete entre //las barreras.
FP M 0.1	
UN E 0.0	
S A 4.0	
UN E 0.0	//Si ninguna de las barreras ópticas está interrumpida, significa que no hay //ningún paquete entre las barreras. El puntero del sentido de transporte se //desconecta.
UN E 0.1	
R A 4.0	
R A 4.1	

## 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.

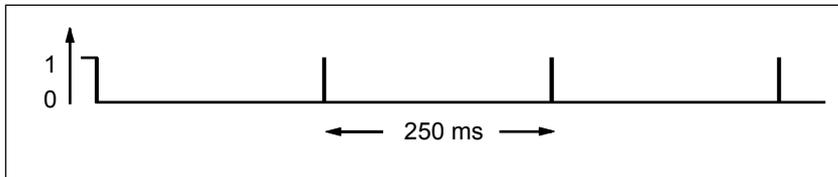
En el S7-300 se puede implementar la función Reloj usando un procesamiento temporizado en bloques de organización especiales. El ejemplo siguiente de un programa AWL muestra el uso de funciones temporizadas para generar un reloj.

### Operación AWL para generar un impulso de reloj (relación impulso-pausa 1:1)

AWL		Explicación
UN	T1	//Si se ha vencido el tiempo T1,
L	S5T#250ms	//cargar el valor de temporización 250 ms en T1
SV	T1	//y arrancar T1 como temporizador de impulso prolongado.
NOT		//Negar (invertir) el resultado lógico.
BEB		//Finalizar el bloque actual cuando el tiempo transcurra.
L	MB100	//Si se ha terminado el tiempo, cargar el contenido del byte de marcas //MB100,
INC	1	//incrementar su contenido en "1" y
T	MB100	//transferir el resultado al byte de marcas MB100.

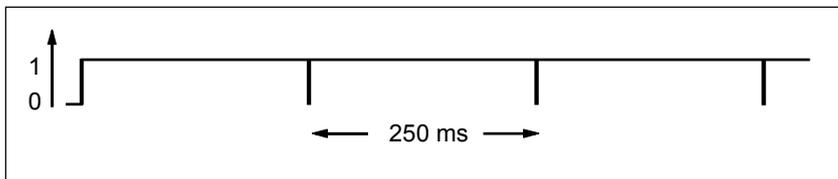
### Consulta del estado de señal

La consulta del estado de señal del temporizador T1 da el resultado lógico.



El temporizador vuelve a arrancar tan pronto como haya transcurrido el tiempo programado. Por este motivo, la consulta efectuada por la instrucción UN T1 produce el estado de señal "1" sólo brevemente.

La figura muestra el aspecto de un bit RLO negado (invertido):



Cada 250 ms el bit RLO negado es 0. En este caso la operación BEB no finaliza el bloque, sino que se incrementa en "1" el contenido del byte de marcas MB100.

El contenido del byte de marcas MB100 cambia cada 250 de la forma siguiente:

0 -> 1 -> 2 -> 3 -> ... -> 254 -> 255 -> 0 -> 1 ...

### Programar una frecuencia determinada

Con los bits de los bytes de marca MB100 se consiguen las frecuencias siguientes:

MB100	Frecuencia en hertzios	Duración
M 100.0	2.0	0.5 s (250 ms on / 250 ms off)
M 100.1	1.0	1 s (0.5 s on / 0.5 s off)
M 100.2	0.5	2 s (1 s on / 1 s off)
M 100.3	0.25	4 s (2 s on / 2 s off)
M 100.4	0.125	8 s (4 s on / 4 s off)
M 100.5	0.0625	16 s (8 s on / 8 s off)
M 100.6	0.03125	32 s (16 s on / 16 s off)
M 100.7	0.015625	64 s (32 s on / 32 s off)

### Operación AWL

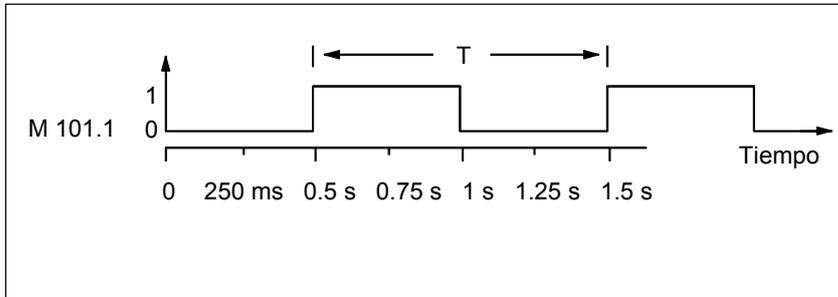
AWL	Explicación
U M10.0	//M10.0 = 1, cuando se produce un fallo. La lámpara indicadora de fallos
U M100.1	//luce intermitentemente a una frecuencia de 1 Hz cuando ocurre un fallo.
= A 4.0	

### 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)**

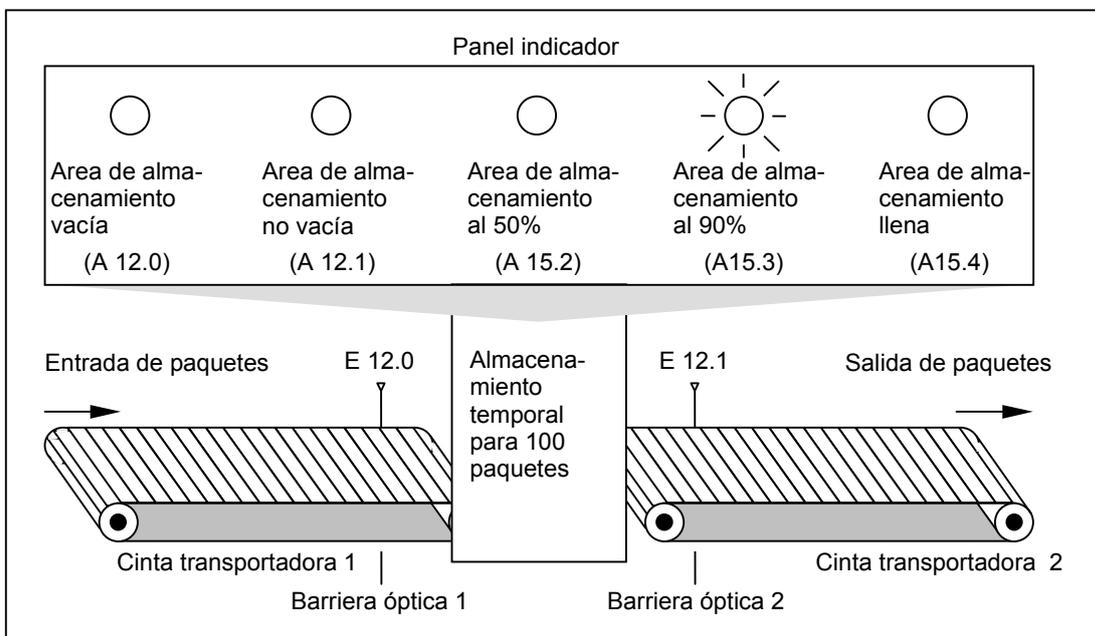
Frecuencia =  $1/T = 1/1 \text{ s} = 1 \text{ Hz}$



## 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.



## Operación AWL para activar las lámparas del panel indicador

AWL	Explicación
U	E 0.0 //Cada impulso generado por la barrera óptica 1
ZV	Z1 //aumenta el valor del contador Z1 en una unidad, contando así el número de //paquetes transportados al área de almacenamiento. //
U	E 0.1 //Cada impulso generado por la barrera óptica 2
ZR	Z1 //disminuye el valor del contador Z1 en una unidad, contando así los paquetes //que salen del área de almacenamiento. //
UN	Z1 //Si el valor de contaje es 0,
=	A 4.0 //se enciende la lámpara indicadora "Area de almacenamiento vacía". //
U	Z1 //Si el valor de contaje no es 0,
=	A 4.1 //se enciende la lámpara indicadora "Area de almacenamiento no vacía". //
L	50
L	Z1
<=I	//Si el valor de contaje es menor o igual a 50,
=	A 4.2 //se enciende la lámpara indicadora "Area de almacenamiento al 50%". //
L	90
>=I	//Si el valor de contaje es mayor o igual a 90,
=	A 4.3 //se enciende la lámpara indicadora "Area de almacenamiento al 90%". //
L	Z1
L	100
>=I	//Si el valor de contaje es mayor o igual a 100,
=	A 4.4 //se enciende la lámpara indicadora "Area de almacenamiento llena". (También //se puede utilizar la salida A 4.4 para bloquear la cinta transportadora 1).

## B.5 Ejemplo: Operaciones de aritmética 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:

$$MD4 = ((EW0 + DBW3) \times 15) / MW2$$

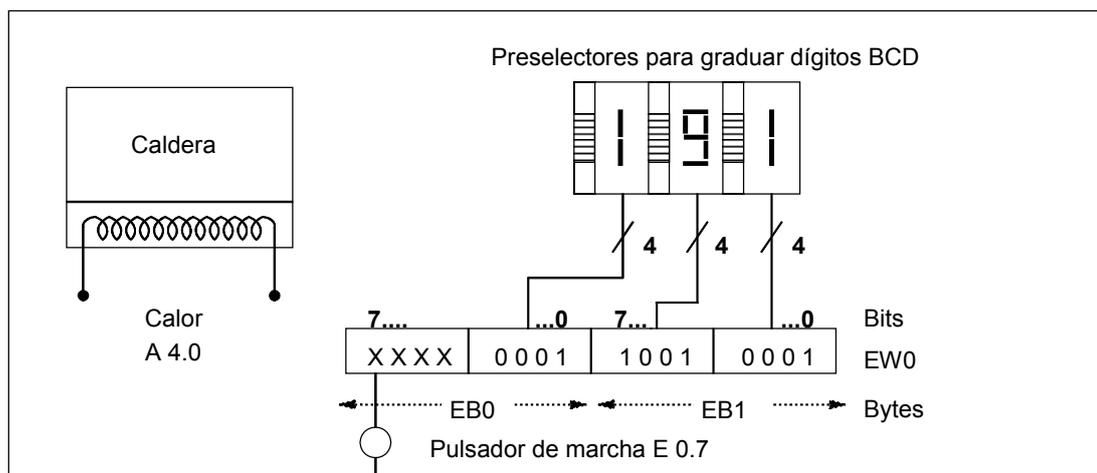
### Operación AWL

AWL	Explicación
L EW0	//Cargar el valor de la palabra de entrada EW0 en el ACU 1.
L DB5.DBW3	//Cargar el valor de la palabra de datos global DBW3 del DB5 en el ACU 1. //El antiguo contenido del ACU 1 se desplaza al ACU 2.
+I E 0.1	//Sumar el contenido de las palabras bajas de los ACUs 1 y 2. El resultado //se deposita en la palabra baja del ACU 1. El contenido del ACU 2 y la //palabra alta del ACU 1 permanecen inalterados.
L +15	//Cargar el valor constante +15 en el ACU 1. El antiguo contenido del //ACU 1 se desplaza al ACU 2.
*I	//Multiplicar el contenido de la palabra baja del ACU 2 por el contenido //de la palabra baja del ACU 1. El resultado se deposita en el ACU 1. El //contenido del ACU 2 permanece inalterado.
L MW2	//Cargar el valor de la palabra de marcas MW2 en el ACU 1. El antiguo //contenido del ACU 1 se desplaza al ACU 2.
/I	//Dividir el contenido de la palabra baja del ACU 2 por el contenido de la //palabra baja del ACU 1. El resultado se deposita en el ACU 1. El //contenido del ACU 2 permanece inalterado.
T MD4	//Transferir el resultado final a la palabra doble de marcas MD4. El //contenido de ambos acumuladores permanece inalterado.

## 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

### Operación AWL

AWL	Explicación
U T1	//Si el temporizador está en marcha,
= A 4.0	//poner en marcha la calefacción.
BEB	//Si el temporizador está en marcha, finalizar aquí el procesamiento. Esto //impide que el temporizador T1 vuelva a arrancar al accionar el pulsador.
L EW0	
UW W#16#0FFF	//Enmascarar los bits de entrada E 0.4 a E 0.7 (es decir, volverlos a //poner a 0). Este valor de temporización en segundos se encuentra en la //palabra baja del ACU 1 en formato decimal codificado en binario.
OW W#16#2000	//Asignar la base de tiempo en segundos en los bits 12 y 13 de la palabra //baja del ACU 1.
U E 0.7	
SV T1	//Arrancar el temporizador T1 como temporizador de impulso prolongado //cuando se accione el pulsador.



## C Transferencia 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.

---



---

### 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 del DB de instancia el valor 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 dentro del 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

)

) 25  
)MCR 177

\*

\*D 116  
\*I 109  
\*R 126

/

/D 117  
/I 110, 111  
/R 127

?

? D 41  
? I 40  
? R 42

+

+ 112  
+AR1 241, 242  
+AR2 242, 243  
+D 114  
+I 107  
+R 123, 124

=

= 27

**A**

Abrir bloque de datos 76  
ABS 128  
ACOS 137, 138  
Activar 29  
Activar área MCR 178  
Activar el RLO (=1) 30  
Almacenar RLO en pila MCR  
  inicio área MCR 175  
Aplicaciones prácticas 257  
Area de memoria 65, 200  
Area MCR 178  
Asignar 27  
ASIN 136, 137

ATAN 138, 139  
AUF 76  
Ayuda en pantalla 5

**B**

Base de tiempo 201  
BE 156  
BEA 158  
BEB 157  
BLD 243  
BTD 48  
BTI 46  
Bucle 102, 103

**C**

Calcular la raíz cuadrada de un número de coma flotante  
  (32 bits) 130  
Calcular el arcocoseno de un número de coma flotante  
  (32 bits) 137  
Calcular el coseno de un número de coma flotante  
  (32 bits) 136  
Calcular el coseno de ángulos como números de coma  
  flotante (32 bits) 134  
Calcular el cuadrado de un número de coma flotante  
  (32 bits) 129  
Calcular el logaritmo natural de un número de coma  
  flotante (32 bits) 132  
Calcular el seno de ángulos como números de coma  
  flotante (32 bits) 133  
Calcular la arcotangente de un número de coma flotante  
  (32 bits) 138  
Calcular la tangente de ángulos como números de coma  
  flotante (32 bits) 135  
CALL 159, 160, 161  
Cambiar el orden de los bytes en el ACU 1-L (16 bits) 58  
Cargar 142  
Cargar el valor actual de temporización en ACU 1 como  
  número BCD 207  
Cargar la longitud del DB global en el ACU 1 77  
Cargar longitud del bloque de datos de instancia en  
  ACU 1 78  
Cargar número del bloque de datos de instancia en  
  ACU 1 79  
Cargar número del bloque de datos global en ACU 1 78  
Cargar palabra de estado en ACU 1 144  
Cargar registro de direcciones 1 con contenido del  
  ACU 1 145  
Cargar registro de direcciones 1 con contenido del  
  registro de direcciones 2 147  
Cargar registro de direcciones 2 con contenido del  
  ACU 1 147

Cargar registro de direcciones 2 con puntero (formato de 32 bits) 148  
 Cargar valor actual del contador en ACU 1 como número BCD 68  
 Cargar valor actual del contador en ACU 1 en forma de entero 67  
 Cargar valor actual del temporizador en ACU 1 como entero 205  
 CC 170  
 Cerrar paréntesis 25  
 CLR 32  
 Comparar enteros ( $== <> > < >= <=$ ) 40  
 Comparar enteros dobles ( $== <> > < >= <=$ ) 41  
 Comparar números en coma flotante (32 bits) ( $== <> > < >= <=$ ) 42  
 Complemento a dos de un entero 55  
 Complemento a dos de un entero doble 56  
 Complemento a uno de un entero 53  
 Complemento a uno de un entero doble 54  
 Componentes de un temporizador 200  
 Configuración binaria en el contador 65  
 Convertir BCD a entero 46  
 Convertir entero doble en BCD 51  
 Convertir entero doble en número en coma flotante (32 bits - IEEE 754) 52  
 Convertir entero en BCD 47  
 Convertir entero en entero doble 49  
 Convertir número BCD a entero doble 48  
 COS 134

## D

-D 115  
 DEC 240, 241  
 Decrementar ACU 1-L-L 240  
 Decrementar contador 73  
 Desactivar 28  
 Desactivar área MCR 179  
 Desactivar contador 69  
 Desactivar RLO (=0) 32  
 Desactivar temporizador 208  
 Desplazar doble palabra a la derecha (32 bits) 192  
 Desplazar doble palabra a la izquierda (32 bits) 190  
 Desplazar palabra a la derecha (16 bits) 188  
 Desplazar palabra a la izquierda (16 bits) 186  
 Desplazar signo de número entero a la derecha (16 bits) 182  
 Desplazar signo de número entero doble a la derecha (32 bits) 184  
 Dividir ACU 2 por ACU 1 como entero 110  
 Dividir ACU 2 por ACU 1 como entero doble 117  
 Dividir ACU 2 por ACU 1 como número de coma flotante (32 bits) 127  
 DTB 51  
 DTR 52

## E

Ejemplo  
 Operaciones de aritmética con enteros 267  
 Operaciones de contaje y comparación 265  
 Operaciones de temporización 262  
 Operaciones lógicas con bits 258  
 Operaciones lógicas con palabras 268  
 Ejemplos de programación 257  
 El MCR (Master Control Relay) 172  
 ENT 239  
 Estructuración de imagen (operación nula) 243  
 Evaluar bits de la palabra de estado en operaciones en coma fija 106  
 Evaluar los bits de la palabra de estado (operaciones de coma flotante) 122  
 EXP Calcular el exponente de un número de coma flotante (32 bits) 131

## F

Fin área MCR 177  
 Fin de bloque 156  
 Fin de bloque condicionado 157  
 Fin de bloque incondicionado 158  
 Flanco negativo 34, 35  
 Flanco positivo 36  
 FN 34  
 FP 36  
 FR 66, 204

## H

Habilitar contador 66  
 Habilitar temporizador 204

## I

-I 108  
 INC 240  
 Incrementar ACU 1-L-L 240  
 Incrementar contador 72  
 Intercambiar ACU 1 y ACU 2 234  
 Intercambiar bloque de datos global y bloque de datos de instancia 77  
 Intercambiar registro de direcciones 1 y registro de direcciones 2 151  
 Introducir pila de ACU 239  
 INVD 54  
 Invertir el orden de los bytes en el ACU 1 (32 bits) 59  
 Invertir un número en coma flotante (32 bits - IEEE 754) 57  
 INVI 53  
 ITB 47  
 ITD 49

**L**

L 67, 142  
 L DBLG 77  
 L DBNO 78  
 L DILG 78  
 L DINO 79  
 L STW 144  
 LAR1 145  
 LAR1 <D> Cargar registro de direcciones 1 con puntero  
 (formato de 32 bits) 146  
 LAR1 AR2 147  
 LAR2 147  
 LAR2 <D> 148  
 LC 68, 207  
 LEAVE 239  
 Lista de ejemplos de programación 257  
 Lista de operaciones aritméticas con enteros 105  
 Lista de operaciones aritméticas con números en coma  
 flotante 121  
 Lista de operaciones con bloques 75  
 Lista de operaciones de acumuladores 233  
 Lista de operaciones de cargar y transferencia 141  
 Lista de operaciones de comparación 39  
 Lista de operaciones de contaje 65  
 Lista de operaciones de control del programa 155  
 Lista de operaciones de conversión 45  
 Lista de operaciones de desplazamiento 181  
 Lista de operaciones de rotación 194  
 Lista de operaciones de salto 81  
 Lista de operaciones de temporización 200  
 Lista de operaciones lógicas con bits 13  
 Lista de operaciones lógicas con palabras 219  
 Llamada 159, 160  
 Llamada a un FB 163  
 Llamada a un SFB 167  
 Llamada a una FC 164  
 Llamada a una SFC 168  
 Llamada condicionada 170  
 Llamada incondicionada 171  
 Llamar a un bloque de una librería 169  
 Llamar a un FB 162  
 Llamar a un SFB 166  
 Llamar a una FC 164  
 Llamar a una multiinstancia 169  
 Llamar a una SFC 168  
 LN 132  
 LOOP 102, 103

**M**

MCR 175, 176, 177, 178, 179  
 MCR( 175, 176  
 MCRA 178  
 MCRD 179  
 Memorizar el RLO en el registro RB 33  
 MOD 118  
 Multiplicar ACU 1 por ACU 2 como entero doble 116  
 Multiplicar ACU 1 por ACU 2 como de coma flotante  
 (32 bits) 126  
 Multiplicar ACU 1 por ACU 2 como entero 109

**N**

Negar el RLO 30  
 NEGD 56  
 NEGI 55  
 NEGR 57  
 Nemotécnica alemana/SIMATIC 247  
 Nemotécnica inglesa 252  
 Nociones básicas 3  
 NOP 0 245  
 NOP 1 245  
 NOT 30  
 Notas importantes sobre el uso de la función MCR 174

**O**

O 17, 21  
 O con abrir paréntesis 23  
 O con doble palabra (32 bits) 228  
 O con palabra (16 bits) 222  
 O( 23  
 OD 228, 229  
 O-exclusiva 19  
 O-exclusiva con abrir paréntesis 24  
 O-exclusiva con doble palabra (32 bits) 230  
 O-exclusiva con palabra (16 bits) 224  
 O-exclusiva-NO 20  
 O-exclusiva-NO con abrir paréntesis 25  
 ON 18  
 ON( 24  
 O-No 18  
 O-No con abrir paréntesis 24  
 Operación nula 0 245  
 Operación nula 1 245  
 Operaciones AWL ordenadas según la nemotécnica  
 alemana (SIMATIC) 247  
 Operaciones AWL ordenadas según la nemotécnica  
 inglesa (internacional) 252  
 OW 222, 223

**P**

Poner contador al valor inicial 71  
 POP 237, 238  
 CPU con cuatro acumuladores 238  
 CPU con dos acumuladores 237  
 PUSH 235, 236  
 CPU con cuatro acumuladores 236  
 CPU con dos acumuladores 235

**R**

R 28, 69, 208  
 -R 125  
 -R 125  
 Redondear número real al próximo entero inferior 63  
 Redondear un número en coma flotante a entero 60  
 Redondear un número real al próximo entero superior 62  
 Restar ACU 1 de ACU 2 como entero 108  
 Restar ACU 1 de ACU 2 como entero doble 115  
 Restar ACU 1 de ACU 2 como número de coma flotante  
 (32 bits) 125

Resto de la división de enteros dobles 118  
 RLD 195, 196  
 RLDA 198, 199  
 RND 60  
 RND- 63  
 RND- 63  
 RND- 63  
 RND- 63  
 RND+ 62  
 Roar ACU 1 a la derecha vía A1 (32 bits) 199  
 Rotar ACU 1 a la izquierda vía A1 (32 bits) 198  
 Rotar doble palabra a la derecha (32 bits) 197  
 Rotar doble palabra a la izquierda (32 bits) 195  
 RRD 197, 198  
 RRDA 199

## S

S 29, 71  
 SA 215, 216  
 Salir de la pila de ACU 239  
 Saltar si el resultado  $\leq 0$  100  
 Saltar si el resultado = 0 95  
 Saltar si el resultado  $> 0$  97  
 Saltar si el resultado  $\geq 0$  99  
 Saltar si el resultado no es válido 101  
 Saltar si OS = 1 93  
 Saltar si OV = 1 92  
 Saltar si RB = 0 91  
 Saltar si RB = 1 90  
 Saltar si resultado  $< 0$  98  
 Saltar si resultado  $\leq 0$  96  
 Saltar si RLO = 0 87  
 Saltar si RLO = 0 y salvar RLO en RB 89  
 Saltar si RLO = 1 86  
 Saltar si RLO = 1 y salvaguardar RLO en RB 88  
 Saltar utilizando una lista de metas 84  
 Salto incondicionado 83  
 SAVE 33  
 SE 211, 212  
 SET 30  
 SI 209  
 SIN 133  
 SLD 190, 191  
 SLW 186, 187  
 SPA 83  
 SPB 86  
 SPBB 88  
 SPBI 90  
 SPBIN 91  
 SPBN 87  
 SPBNB 89  
 SPL 84, 85  
 SPM 98  
 SPMZ 100  
 SPN 96  
 SPO 92  
 SPP 97  
 SPPZ 99  
 SPS 93, 94  
 SPU 101  
 SPZ 95

SQR 129  
 SQRT 130  
 SRD 192, 193  
 SRW 188, 189  
 SS 213, 214  
 SSD 184, 185  
 SSI 182, 183  
 Sumar ACU 1 y 2 como entero 107  
 Sumar ACU 1 y 2 como entero doble 114  
 Sumar ACU 1 y 2 como número de coma flotante (32 bits) 123  
 Sumar constante entera o entera doble 112  
 Sumar el ACU 1 al registro de direcciones 1 241  
 Sumar el ACU1 al registro de direcciones 2 242  
 SV 210, 211

## T

T 149  
 T STW 150  
 TAD 59  
 TAK 234  
 TAN 135  
 TAR 151  
 TAR1 151  
 TAR1 <D> 152  
 TAR1 AR2 153  
 TAR2 153  
 TAR2 <D> 154  
 TAW 58  
 TDB 77  
 Temporizador como impulso 209  
 Temporizador como impulso prolongado 210  
 Temporizador como retardo a la conexión 211  
 Temporizador como retardo a la conexión con memoria 213  
 Temporizador como retardo a la desconexión 215  
 Transferencia de parámetros 269  
 Transferir 149  
 Transferir ACU 1 a la palabra de estado 150  
 Transferir registro de direcciones 1 a ACU 1 151  
 Transferir registro de direcciones 1 a dirección de destino (puntero de 32 bits) 152  
 Transferir registro de direcciones 1 a registro de direcciones 2 153  
 Transferir registro de direcciones 2 a ACU 1 153  
 Transferir registro de direcciones 2 a dirección de destino (puntero de 32 bits) 154  
 TRUNC 61  
 Truncar 61

## U

U 15  
 U( 22  
 UC 171  
 UD 226, 227  
 UN 16  
 UN( 23  
 UW 220, 221

**V**

Valor absoluto de un número de coma flotante (32 bits - IEEE 754) 128

Valor de contaje 65

Valor de temporización 200, 201, 202, 203

**X**

X 19

X( 24

XN 20

XN( 25

XOD 230, 231

XOW 224, 225

**Y**

Y 15

Y antes de O 21

Y con abrir paréntesis 22

Y con doble palabra (32 bits) 226

Y con palabra (16 bits) 220

Y-No 16

Y-No con abrir paréntesis 23

**Z**

ZR 73

ZV 72

