Programación en STEP 7 en lenguajes KOP y AWL.

Elementos e instrucciones básicas

1. Operaciones Combinacionales lógicas

Dan lugar a la asignación de una salida o marca o a la ejecución de una instrucción determinada en función de la combinación de datos binarios.

Lenguaje KOP

Se tienen diversos segmentos y cada uno de ellos debe terminar en una asignación de un valor a una bobina (salida) o marca (variable auxiliar), ya sea de igualdad o a través de otras funciones, como Set y Reset. También puede terminar dando lugar a la ejecución de instrucciones dependientes del estado lógico al final (a la derecha) del segmento.

El valor que finalmente llega a esta asignación o instrucción será 1/0 si el resultado de la combinación lógica de las entradas y marcas desde el inicio (a la izquierda) hasta el final (a la derecha) da como resultado un 1/0.

Las operaciones lógicas en KOP son muy parecidas a la representación en el esquema cableado.

En un esquema cableado (se suele disponer en vertical) se alimenta la bobina que se encuentra al final del circuito si se encuentra un camino para la corriente desde la parte superior (tensión de 220 V o 24 V) hasta la bobina. Para ello debe existir un camino donde todos los contactos estén cerrados. En el lenguaje KOP, empezando por la izquierda se va realizando una consulta al estado de las entradas y se combina lógicamente esa entrada con el estado anterior.

Si la entrada está conectada a un contacto normalmente abierto, en condiciones de reposo la entrada estará a ‘0’ lógico. Si el contacto es activado, la entrada se pondrá a ‘1’.

Inversamente, si la entrada está asociada a un contacto normalmente abierto, en reposo la entrada estará a ‘1’ lógico. Si el contacto es activado, la entrada se pondrá a ‘0’.

Independientemente del tipo de contacto conectado a una entrada concreta[1], se puede operar con el valor lógico de esa entrada o con su valor negado. El primero caso sería el de la consulta al estado de la entrada EX.Y (–| |–), en el que se emplea el valor lógico de dicha entrada. Para operar con el valor negado, se hará una consulta negada al estado de la entrada EX.Y (–| / |–) .

[1] Aunque en todo momento, al hablar de consultas se refieren a entradas, también existen y son muy empleadas las consultas (negadas o no) a salidas, marcas o direcciones de memoria, estado de temporizadores o contadores ...

A modo de resumen, el valor lógico con el que se opera en función del tipo de contacto, de su activación o no, y del tipo de consulta, será:

Pulsador o contacto	**Estado**	**Consulta**	**Valor con que se opera**
N O	En reposo	–| |–	'0'
N O	En reposo	–| / |–	'1'
N O	Activado	–| |–	'1'
N O	Activado	–| / |–	'0'
N C	En reposo	–| |–	'1'
N C	En reposo	–| / |–	'0'
N C	Activado	–| |–	'0'
N C	Activado	–| / |–	'1'

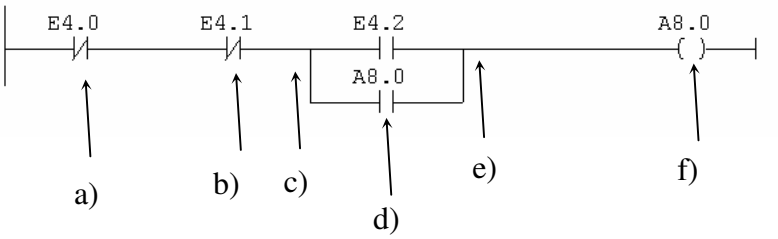
En este sentido, si dos entradas (o datos binarios en general) están en serie, ambas deben corresponderse con un nivel alto para que la combinación de ambas dé lugar a '1' o TRUE. Es una operación lógica AND (&) o Y. Es directamente asimilable a la combinación serie de dos contactos eléctricos, donde si uno de ellos está abierto, se impide el paso de corriente.

Si dos entradas se encuentran en paralelo, basta con que una de ellas sea cierta para que la combinación de ambas dé lugar a '1'. Se trata de una operación lógica OR u O. Se corresponde con la combinación paralelo de dos contactos, en los que basta con que uno de ellos esté cerrado para que se permita el paso de corriente.

Recorriendo el segmento de izquierda a derecha se van analizando las combinaciones de datos binarios (entradas, salidas, marcas, estado de temporizadores...), y tras cada operación lógica se renueva lo que se llama el RLO (resultado de la operación lógica) que es el valor lógico que se tiene en cuenta al combinar con el elemento siguiente.

Cuando aparece una bifurcación (dos elementos en paralelo) el RLO se guarda en una pila de manera que se pueda realizar una nueva combinación lógica entre los contactos que se hallan en paralelo. El resultado de la combinación OR de todos estos elementos en paralelo es un nuevo RLO que se combina con el RLO anterior (que se saca de la pila).

Considerando el ejemplo del arranque de un motor de inducción, y suponiendo que los contactos físicos del térmico y del pulsador de paro son normalmente abiertos (también el del pulsador de marcha), el programa en KOP sería



a) Inicialmente se parte de un RLO igual a ‘1’. Se hace una consulta al estado negado de E4.0. En condiciones normales, el contacto NO al que está conectado no será accionado pues se trata del contacto NO del relé térmico, y este estado abierto del contacto da lugar a un nivel bajo de tensión en la entrada E4.0. Al ser la consulta al estado negado, en condiciones normales (sin actuar el relé térmico) esta consulta da lugar a un 1 lógico, que se combina en serie (operación Y) con el RLO anterior y da como resultado un RLO nuevo igual a '1'. Sólo en el caso de que exista una sobrecarga y el contacto NO del térmico se cierre, se tendrá un nivel alto de tensión en la entrada E4.0, que supone un nivel bajo en la *consulta negada de E4.0*, que impide que se pueda asignar un '1' a la salida.
b) De nuevo el RLO fruto de la operación entre el estado negado de E4.0 y el RLO de partida (este último siempre es 1) se combina en serie (Y) con el estado negado de E4.1. El nuevo RLO será 0 si alguna de las entradas E4.0 o E4.1 está a nivel alto de tensión (consulta negada igual a '1')
c) Al encontrarse una derivación paralelo, el RLO se almacena en una pila de resultados. Le llamaremos RLO_1 .
d) Se inicia una nueva secuencia de comprobaciones de estados. En principio, se halla la combinación paralelo (OR) de los estados de E4.2 y de A8.0. Este estado es el de la salida A8.0 de manera que cuando la salida esté activada, el dato de memoria A8.0 estará a 1 (en principio se supondrá que esta salida vale ‘0’). Si alguna de estas variables está a 1, RLO_2 valdrá 1.
e) Cuando halla la combinación paralelo y tiene el nuevo RLO (que es RLO_2), saca de la pila el valor anterior de RLO (que era RLO_1) y los combina en serie, valor que sustituye al anterior RLO (que valía RLO_1)
f) Este valor de RLO será asignado a la salida A8.0 y en el siguiente ciclo, la consulta al estado A8.0 del punto e) tomará dicho valor.

Lenguaje AWL

Al editar un módulo en lenguaje AWL se pide al programador que escriba cada una de las instrucciones correspondientes a las operaciones lógicas que haya que realizar indicando con qué parámetros se trabaja.

De forma análoga a lo que ocurre en lenguaje KOP también se puede dividir el programa en segmentos, aunque no es necesario.

El anterior esquema, traducido a lenguaje AWL sería:

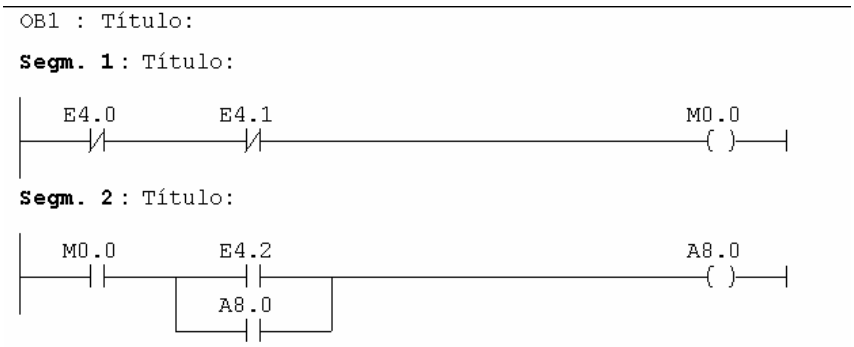
```
UN   E    4.0
UN   E    4.1
U(
O    E    4.2
O    A    8.0
)
=    A    8.0
```

1. **UN E 4.0** Es una consulta al estado negado de la entrada 4.0. El RLO vale después de esta instrucción el valor negado de E 4.0. La primera instrucción del tipo U, O o X (operación XOR) al comienzo de un segmento o después de una asignación de igualdad u operación SET o RESET siempre es una consulta, y por tanto, no se combina con otro RLO.
2. **UN E 4.1.** El RLO anterior se combina según un producto lógico con el estado negado de E 4.1 y el resultado de dicha operación lógica se convierte en el nuevo RLO
3. **U(** El RLO debería multiplicarse lógicamente con el dato que viniera después de U. Sin embargo, al aparecer el paréntesis, el RLO que se tenía después del paso 2 es guardado en una pila de RLO (llamada pila MCR), a la espera de que termine la operación entre paréntesis.
4. **O E 4.2** También la primera operación U, O o X después de abrir un paréntesis es una operación de consulta, por lo que simplemente se consulta el estado de E4.2 y ese valor será el nuevo RLO
5. **O A 8.0** El valor de RLO del paso 4 se combina según una suma lógica con el estado de la salida A8.0 y el resultado será el nuevo RLO en ese nivel
6.) Se vuelve al nivel superior con lo que se hace la combinación lógica expresada en 3 (producto lógico) entre el RLO que se encontraba en la pila (de la que sale) y el resultado del paréntesis
7. **= A 8.0** El valor resultante es asignado a la salida A8.0

2. Marcas

Se denominan marcas a las variables auxiliares que pueden retener un determinado resultado (no necesariamente lógico). Se reconocen por la letra M. Son asignadas de igual forma que las salidas, y su valor también puede ser empleado de igual forma.

Un ejemplo, un tanto artificioso, del empleo de las marcas nos lo daría este otro esquema correspondiente al mismo arranque directo de un motor.



3. Puesta a 1 (S) y Puesta a 0 (R) del operando

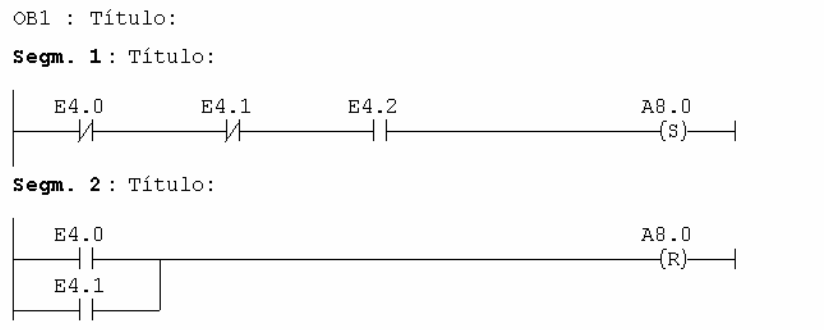
Estos son dos elementos (KOP) o instrucciones (AWL) dependientes en cualquier caso del RLO y que se aplican a un operando (típicamente salidas y marcas de memoria).

Se identifican como –(S) y –(R) en KOP y como S y R en AWL.

1. Puesta a 1. Si el RLO es '1', pone el operando al que se aplica a '1'. Si el RLO es '0' no produce ningún efecto y mantiene el operando en su valor anterior, ya sea '1' o '0'.
2. Puesta a 0. Si el RLO es '1', pone el operando al que se aplica a '0'. Si el RLO es '0' no produce ningún efecto y mantiene el operando en su valor anterior, ya sea '1' o '0'.

Después de una operación Set o Reset en AWL, la siguiente instrucción U, O o X será simplemente una consulta al estado del operando de la instrucción.

El manido ejemplo del arranque del motor de inducción quedaría como:



El equivalente en AWL es

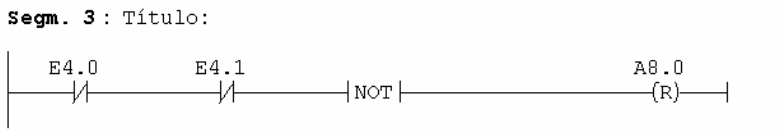
```
UN   E    4.0
UN   E    4.1
U    E    4.2
S    A    8.0

O    E    4.0      ← Tras el Set, la siguiente operación es simple consulta a E4.0
O    E    4.1
R    A    8.0
```

La salida A8.0 se activará cuando concurra que no ha disparado el térmico (el contacto NO no se ha cerrado), que no se ha actuado sobre el pulsador de paro NO, y que se ha actuado sobre el pulsador de marcha NO. En ese momento se activa la salida A8.0 y no se desactivará hasta que se resetee o se le asigne el valor cero.

Un reseteo se producirá cuando, o bien dispare el térmico y el contacto NO se cierre (nivel alto en la entrada E4.0) o cuando se pulse el pulsador de paro (su contacto normalmente abierto se cierra y da un nivel alto en E4.1).

El segundo segmento también se puede escribir como



o bien

```
UN   E    4.0
UN   E    4.1
NOT
R    A    8.0
```

donde se emplea la instrucción NOT que invierte el valor del anterior RLO (el negado de un producto es la suma de los negados).

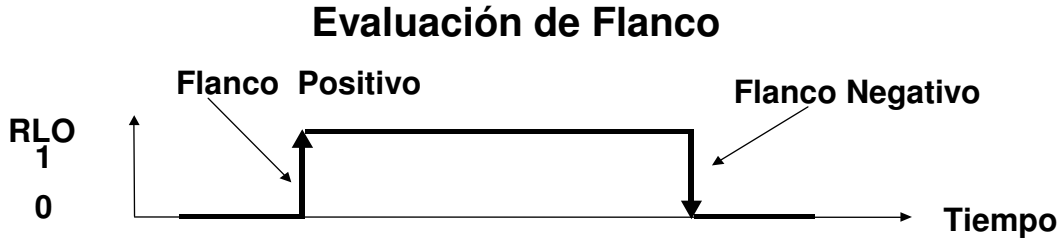
4. Puesta a 1 (SET) y puesta a 0 del RLO (CLR)

Las anteriores instrucciones habían de aplicarse al operando que acompaña a la instrucción S o R o las equivalentes en KOP. Las instrucciones SET y CLR (de AWL) se aplican únicamente al RLO poniéndolo a 1 o a 0 respectivamente, independientemente del estado anterior del RLO. Estas funciones se suelen utilizar en el módulo OB100 para inicializar el estado del proceso.

No existen funciones iguales en KOP, pero se puede conseguir la misma función que:

- SET si se disponen en paralelo dos consultas, una negada y otra no negada, a la misma dirección (ya sea marca, entrada o salida)
- CLR si se disponen en serie dos consultas, una negada y otra no negada, a la misma dirección

5. Detección de flancos

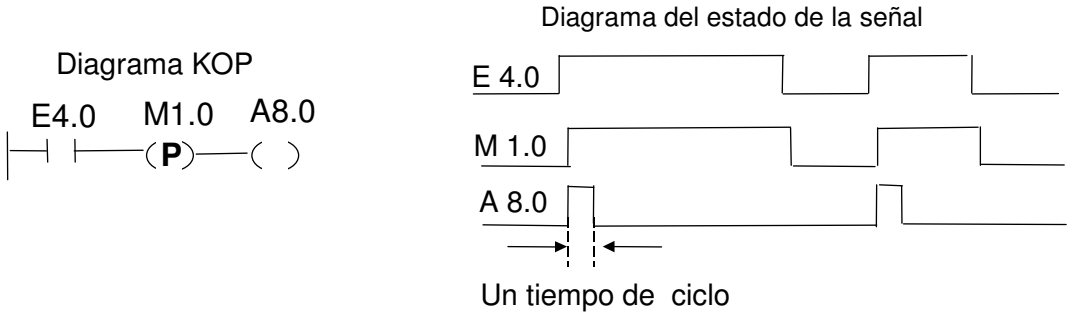


El estado de señal del bit RLO se compara durante cada ciclo del programa con el estado de señal del bit RLO del ciclo anterior para determinar los cambios de estado. Para poder ejecutar la comparación hay que almacenar el estado del bit RLO anterior en alguna marca libre (<bit>). Si el estado de señal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

Puede detectar tanto flancos positivos (instrucción **FP**) como negativos (instrucción **FN**).

Ejemplo:

```
U E 4.0
FP M 1.0   ← Aquí es donde se almacena el valor del RLO del ciclo anterior.
= A 8.0
```



El detector de flancos negativos FN o –(N)— sería similar, sólo que la salida sería activada durante un tiempo de ciclo tras el flanco negativo de E4.0

6. Carga y transferencia de datos

Acumuladores

Los dos acumuladores de 32 bits son registros universales para el procesamiento de bytes, palabras y dobles palabras. Se puede cargar las constantes y valores de los operadores desde la memoria en el ACU 1 (Carga de datos L) y ejecutar aquí las operaciones lógicas. También se puede transferir el resultado lógico del ACU 1 a una dirección de operando (transferencia de datos T).

El mecanismo de pila para gestionar los acumuladores funciona de la siguiente manera:

- Una operación de carga siempre tiene efecto únicamente en el ACU 1 y almacena el contenido antiguo de ACU 1 en el ACU 2.
- Una operación de transferencia (operación de copia) no cambia el contenido de los acumuladores.
- La operación TAK permuta los contenidos de ACU 1 y ACU 2.
- El resultado de las operaciones lógicas entre el ACU 1 y el ACU 2 (operaciones aritméticas, de comparación, Y, O, ...) siempre se almacena en el ACU 1.

Atendiendo a su tamaño, existen diversos tipos de datos con los que el STEP7 trabaja: Todos los tipos de datos simples definen el tamaño y formato de un objeto de datos. Los principales son:
BOOL (1 bit)
BYTE (1 byte, 8 bits)
WORD o palabra (2 bytes, 16 bits)
DWORD o doble palabra (4 bytes, 32 bits)

Por ejemplo para designar al dato booleano correspondiente a la entrada 4.0, se emplea E 4.0. Para designar al byte de entrada 4 completo, se emplea EB 4. La palabra de entrada 2 (EW 2) está compuesta por los bytes de entrada 2 y 3 (EB 2 y EB 3). La doble palabra de entrada ED 0 está compuesta por EW0 y EW2 o también por EB0, EB1, EB2 y EB3. Para una salida, sería AB, AW o AD. Para una marca, MB, MW o MD.

Existen otros tipos de datos que reconoce el software de programación:

1 bit	1 byte (8 bits)	1 palabra(2 bytes)	2 palabras(4 bytes)(32 bits)
BOOL	BYTE	WORD	DWORD
	CHAR		
		INT	DINT
		DATE	
		S5TIME	
			REAL
			TIME
			TIME_OF_DAY

Carga de datos

L <operando> carga en el ACU 1 el contenido del byte, de la palabra o de la doble palabra direccionado. El antiguo valor de ACU 1 pasa a ACU 2. No acepta carga de bits por separado.
Ejemplo:

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	PEPE	Cargar parámetro "PEPE" en ACU1

Contenido del ACU 1 antes de ejecutar la operación de carga:

ACU1-H-H ACU1-H-L ACU1-L-H ACU1-L-L
XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX X = "1" ó "0"
Contenido del ACU 1 después de ejecutar la instrucción L MB10 (L <byte>):

00000000 00000000 00000000 <MB10>
Contenido del ACU 1 después de ejecutar la instrucción L MW10 (L <palabra>):

00000000 00000000 <MB10> <MB11>
Contenido del ACU 1 después de ejecutar la instrucción L MD10 (L <doble palabra>):

<MB10> <MB11> <MB12> <MB13>

Trasferencia de datos

Descripción de la operación:

T <operando> transfiere (copia) el contenido del ACU 1 a la dirección de destino. 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 (no se borra de ACU 1). La operación se ejecuta sin tener en cuenta ni afectar a los bits de la palabra de estado.

Ejemplos:

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.

7. Temporizadores

Existe un área de memoria reservada a los temporizadores. Por cada temporizador se reserva una palabra (16 bits). El valor de tiempo puede estar en un rango de 10 ms a 9990s (2 horas, 46 minutos y 30 segundos).

Valores de temporización predeterminados:
Tipo de datos S5TIME => unidades disponibles: h (horas), m (minutos), s (segundos), ms (milisegundos)
Formato general ===> s5t#_xh_ym_zs_xxms (las variables definidas por el usuario son x,y,z,xx,)

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

Margen			Resolución
10MS	a	9S_990MS	0,01 segundos
100MS	a	1M_39S_900MS	0,1 segundos
1S	a	16M_39S	1 segundo
10S	a	2H_46M_30S	10 segundos

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

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

El S7-300 ofrece tres opciones para el retardo de tiempo:
- S_EVERZ Temporizador de retardo a la conexión
- S_AVERZ Temporizador de retardo a la desconexión
- S-SEVERZ Temporizador de retardo a la conexión memorizado

así como dos opciones de temporización por impulso:
- S_IMPULS Temporizador de impulso
- S_VIMP Temporizador de impulso prolongado

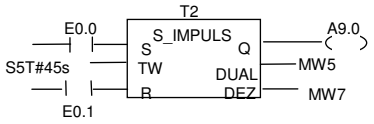
Temporizador de impulso S_IMPULS (SI)

La salida acompaña a la entrada pero hasta un tiempo máximo que es el valor que se carga en el temporizador.

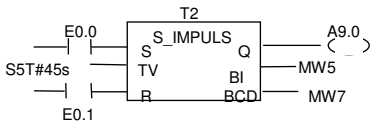
Ejemplo:

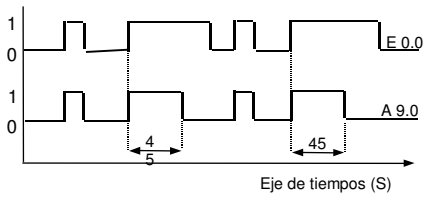
```
U     E0.0
L     S5T#45S
SI    T2
U     E0.1
R     T2
L     T2
T     MW5
LC    T2
T     MW7
U     T2
=     A9.0
```

Símbolo SIMATIC



Símbolo Internacional





Parámetro Internacional	Parámetro SIMATIC	Tipo de datos	Area de memoria	Descripción
N.° de T	N.º de T	TIMER	T	Número de identificación del temporizador, el área varía según la CPU que se utilice
S	S	BOOL	E, A, M, L, D	Entrada de arranque
TV	TW	S5TIME	E, A, M, L, D	Valor de temporización predeterminado
R	R	BOOL	E, A, M, L, D	Entrada de puesta a 0
BI	DUAL	WORD	E, A, M, L, D	Valor de temporización actual, codificado en binario
BCD	DEZ	WORD	E, A, M, L, D	Tiempo restante, formato BCD
Q	Q	BOOL	E, A, M, L, D	Estado del temporizador

S_IMPULS (Parametrizar y arrancar temporizador como impulso) arranca el temporizador indicado cuando hay un flanco creciente en la entrada de arranque S. Para arrancar un temporizador tiene que producirse necesariamente un cambio de señal. El temporizador funciona mientras el estado de señal en la entrada S sea "1", pero como máximo durante el tiempo indicado por el valor de temporización en la entrada TV/TW. El estado de señal en la salida Q es "1" mientras que funcione el temporizador. Si el estado de señal en la entrada S cambia de "1" a "0" antes de transcurrir el intervalo de tiempo, el temporizador se para. En este caso el estado de señal en la salida Q es "0". El temporizador se pone a 0 si la entrada de desactivación R del temporizador se pone a "1" mientras funciona el temporizador. El valor de temporización actual y la base de tiempo también se ponen a 0. Un "1" en la entrada R del temporizador no tiene efecto alguno si el temporizador no está en marcha.

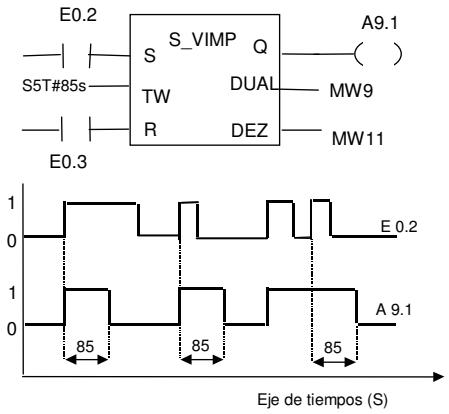
El valor de temporización actual queda depositado en las salidas BI/DUAL y BCD/DEZ. El valor de temporización en la salida BI/DUAL está en código binario y se obtiene con la instrucción **L** (L <contador> carga el valor de contaje del temporizador direccionado en forma de número entero en ACU1-L, después de que se haya

almacenado el contenido del ACU 1 en el ACU 2). El valor de temporización en la salida BCD/DEZ está en formato decimal codificado en binario y se obtiene con la instrucción LC (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). El valor de temporización actual equivale al valor inicial de TV/TW menos el valor de temporización que ha transcurrido desde el arranque del temporizador.

Temporizador de impulso prolongado S_VIMP (SV)

Al arrancar un temporizador SV, se obtiene una respuesta de duración igual al valor de tiempo prefijado en nuestra constante de tiempo, independientemente de la duración del impulso de entrada. Cualquier nueva actuación sobre la señal de activación del temporizador supone el rearme del temporizador.
Si la entrada S cambia de 0 a 1, el temporizador arranca y continúa en marcha incluso si la entrada S cambia a 0 antes de que el temporizador termine de contar. Mientras el tiempo está corriendo, la salida Q=1. Si la entrada R cambia de 0 a 1 en cualquier momento, el temporizador se resetea.

```
U    E0.2
L    S5T#85S
SV   T9
U    E0.3
R    T9
L    T9
T    MW9
LC   T9
T    MW11
U    T9
=    A9.1
```



Temporizador de retardo a la conexión S_EVERZ (SE)

Al arrancar un temporizador SE, se obtiene un impulso igual al de entrada menos el valor prefijado en la constante de tiempo. La resta se produce al inicio del impulso de la señal de entrada.
El temporizador arranca cuando hay un flanco ascendente en la entrada S. El temporizador continúa en marcha con el valor de temporización indicado en la entrada TW mientras sea positivo el estado de señal en la entrada S. El estado de señal en la salida Q es "1" si el tiempo ha transcurrido sin errores y si el estado de señal en la entrada S es "1". Si el estado de señal en la entrada S cambia de "1" a "0" mientras está en marcha el temporizador, éste cambia el estado de señal en la salida Q a "0". Si la entrada R cambia de 0 a 1 en cualquier momento, el temporizador se resetea.

```
U     E0.7
L     S5T#35S
SE    T4
U     E0.5
R     T4
L     T4
T     MW0
LC    T4
T     MW2
U     T4
=     A8.5
```

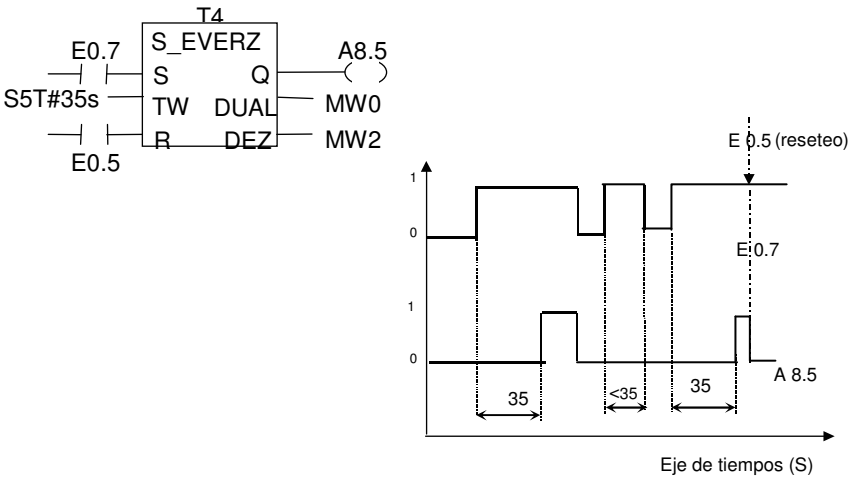
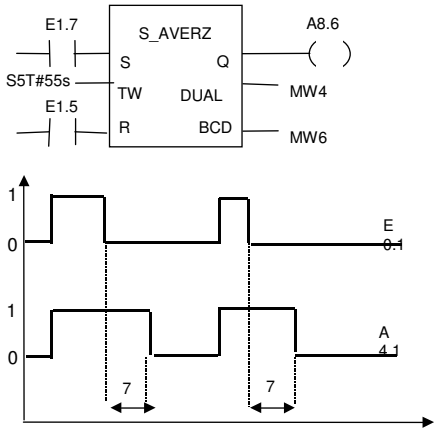


Figura 1 ***e retardo a la desconexión S_AVERZ (SA)***

Al arrancar un temporizador SA, se obtiene una respuesta igual a la de entrada más el tiempo prefijado en la constante de tiempo.
Si la entrada S cambia de 1 a 0, el temporizador arranca y continua corriendo. Si la entrada S cambia a 1 antes de que el temporizador termine de contar, se redispara el temporizador. Mientras el tiempo está corriendo, la salida Q=1. Si la entrada R cambia de 0 a 1 en cualquier momento, el temporizador se resetea.

```
U     E 1.7
L     S5T#55s
SA    T5
U     E1.5
R     T5
L     T5
T     MW4
LC    T5
T     MW6
U     T5
=     A8.6
```



Temporizador de Retardo a la conexión memorizada S_SEVERZ (SS)

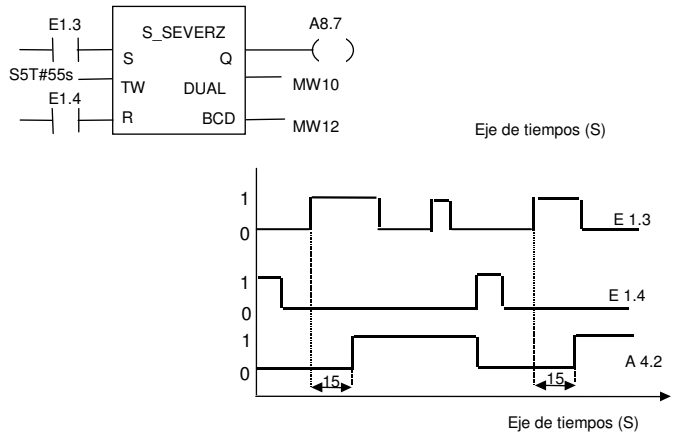
Un temporizador SS es idéntico al SE, excepto en un aspecto: este temporizador se mantiene activo a no ser que se produzca la orden de reseteo.

Se puede decir que se trata de una memoria retardada el tiempo prefijado en nuestra constante.

Si la entrada S cambia de 0 a 1, el temporizador arranca y continúa corriendo incluso si la entrada S cambia a 0 antes de que el temporizador termine de contar. Si el tiempo ha concluido la salida Q continúa =1 independientemente del estado de S. Si la entrada R

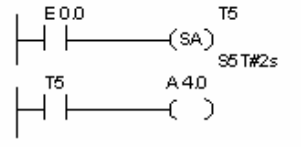
cambia de 0 a 1 en cualquier momento, el temporizador se resetea. El temporizador vuelve a arrancar con el valor de temporización indicado si el estado de señal en la entrada S cambia de "0" a "1" mientras el temporizador está en marcha.

```
U     E 1.3
L     S5T#55s
SS    T5
U     E1.4
R     T5
L     T5
T     MW10
LC    T5
T     MW12
U     T5
=     A8.7
```



Instrucciones de Bit para Temporizadores

Con el fin de ahorrar instrucciones, es posible utilizar los temporizadores en forma de bobina si no se desea utilizar todas las entradas y salidas disponibles en la cajas de temporización. De la misma forma, se puede consultar el valor binario de cualquiera de ellos como si de una entrada se tratara.



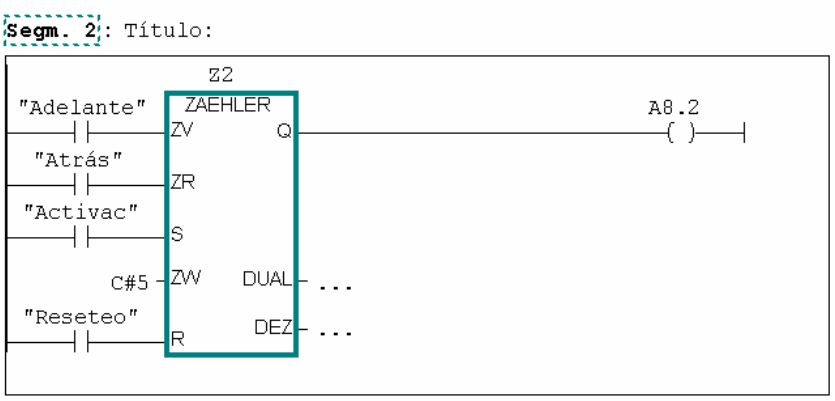

```
U     E0.0
L     S5T#25S
SA    T5

U     T5
=     A4.0
```

8. Contadores

La forma de proceder con los contadores es muy similar a la de los temporizadores. De igual forma, existe en KOP, dos formas de emplearlos. Una con un elemento general, y otra forma, accediendo a cada una de sus funciones.

En KOP, existen contadores ascendentes, descendentes y ascendentes/descendentes. Se verá sólo el tercero al ser los otros dos una particularización de éste.



El valor al que se inicializa el contador viene dado por el dato asociado a la entrada ZW.

El contador se inicializa a este valor y estará preparado para contar cuando se produzca un flanco de subida en el valor lógico asociado a la entrada S.

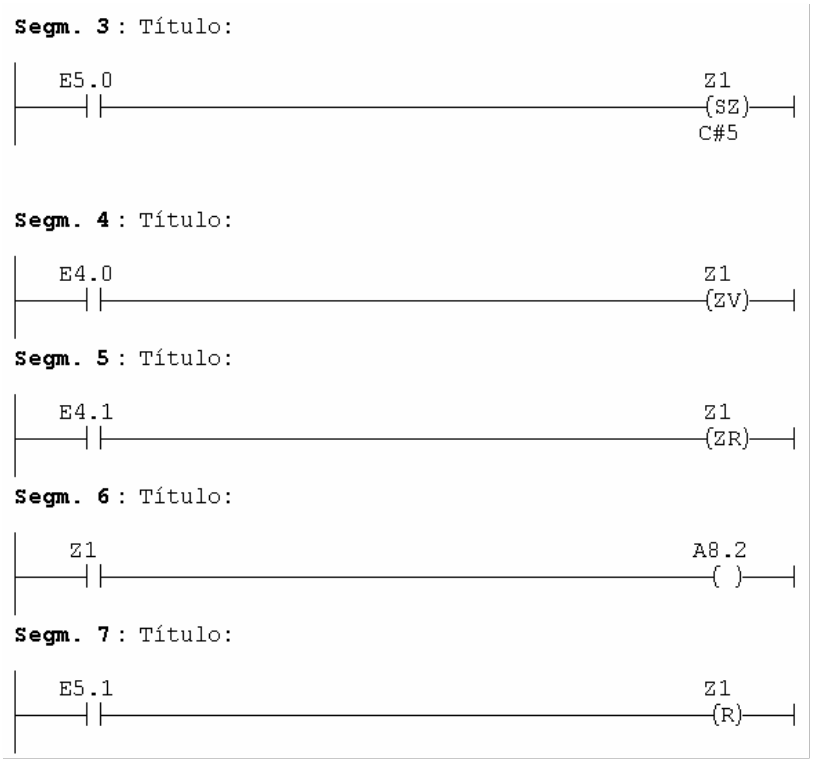
Una vez activado el contador, ZV y ZR incrementan y decrementan, respectivamente, la cuenta cuando el valor lógico conectado a su entrada pasa de 0 a 1.

La puesta a cero del contador se puede forzar anticipadamente con un flanco de subida en la entrada R.

Con las salidas DUAL y DEZ se puede conocer el valor actual de la cuenta, tanto como número entero como en código BCD.

Finalmente la salida Q estará a 1 desde que se activa el contador y mientras el valor de la cuenta sea distinto de 0, y estará a 0 antes de activarlo, después de resetearlo, o desde el momento en que la cuenta llega a cero. En este caso, para activar de nuevo el contador, será necesario un flanco de subida en S.

A cada una de estas posibilidades puede accederse sin necesidad de usar el elemento contador Zähler. Así, lo anterior sería equivalente a :



o bien

```
U   E5.0
L   C#5
S   Z1
U   E4.0
ZV  Z1
U   E4.1
ZR  Z1
U   Z1
=   A8.2
U   E5.1
R   Z1
```

9. Saltos

En un modulo de organización, módulo de función o función, se puede alterar la ejecución secuencial de un programa, dirigiendo el puntero que apunta a la siguiente instrucción a ejecutar a otro punto del programa. Este punto al que se salta se llama etiqueta y habrá de ser una palabra de como mucho 4 caracteres.

En el lugar desde el que se produce el salto, se referencia la etiqueta precedida de la orden de salto correspondiente. En el punto al que se produce el salto, delante de la primera instrucción a ejecutar tras el salto, se situará el nombre de la etiqueta (en lenguaje AWL irá seguida de dos puntos ":").

El salto puede ser incondicional (se salta siempre) o condicional (se produce el salto sólo si el RLO es cierto).

Ejemplo en lenguaje KOP y AWL

Se quiere que si la entrada E4.1 está a '1', que asigne el valor de la entrada E4.7 a la salida A5.1. Si la entrada E4.1 está a 0, que asigne la entrada E4.7 a la salida A5.2, y que además resetee la salida A5.0 en función de E4.3

```
        U     E4.1
        SPB   SAL1
        U     E4.7
        =     A5.2
        U     E4.3
        R     A5.0
        SPA   SAL2
SAL1:   U     E4.7
        =     A5.1
SAL2:   // A partir de aquí  ya vendría el resto de instrucciones, p.ej.
        U M0.0
        = M0.0
```