# Manual de formación
# para soluciones generales en automatización
# Totally Integrated Automation (T I A)

## *Anexo III*
## Comandos básicos de programación
## KOP/FUP/AWL en STEP 7

Estos documentos están elaborados por Siemens A&D FEA (Investigación, desarrollo y formación en automatización automática) con la finalidad que su uso sea el de la formación.

Siemens no se compromete a garantizar lo que concierne al contenido.

La publicación de estos documentos, así como la utilización y el anuncio de éstos, está permitida dentro de la formación pública. Con la salvedad de que se precisa la autorización escrita por Siemens A&D FEA (Hr. Knust: e-mail: michael.knust@hvr.siemens.de).

Las infracciones serán sometidas a una indemnización. Todos los derechos de las traducciones están también condicionados, especialmente para el caso de la patentación o del registro GM.

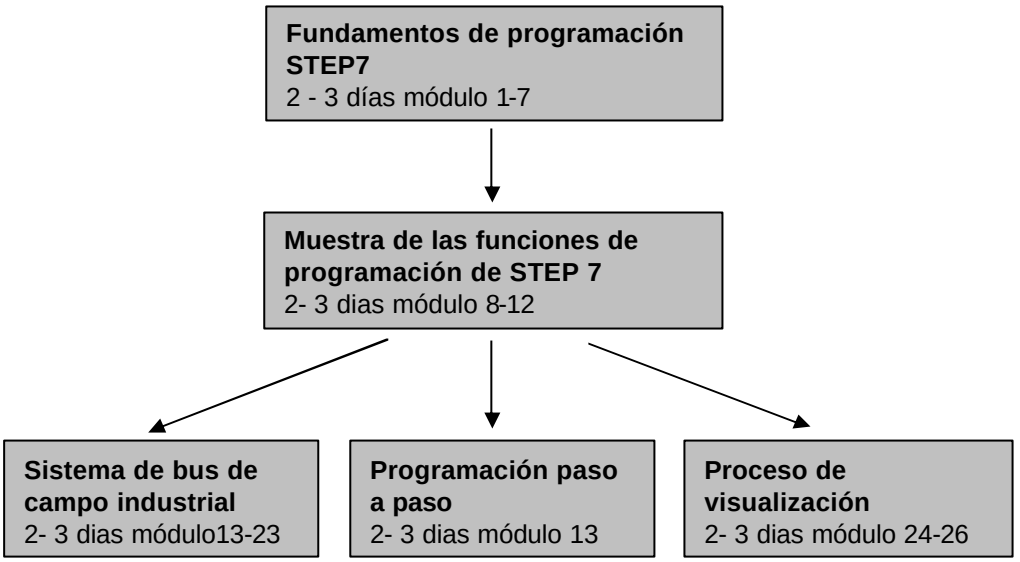
Agradecemos al ingeniero Fa. Michael Dziallas y a los profesores, así como a las personas que han apoyado la elaboración de estos documentos.

PÁGINA:

**PÁGINA:**

## 1. PREÁMBULO

El apéndice C es necesario para la elaboración de todos los módulos.




**Objetivo:**

El lector recibe con este apéndice una colección de las instrucciones más importantes de programación, las cuales serán necesarias para solucionar los programas en los módulos 1-26.

**Condiciones:**

Para la adaptación de este módulo, se suponen los siguientes conocimientos previos:

- Conocimientos básicos de programación de PLC con STEP7 (p.e. apéndice A - Conocimientos básicos de programación de PLC con SIMATIC S7-300)

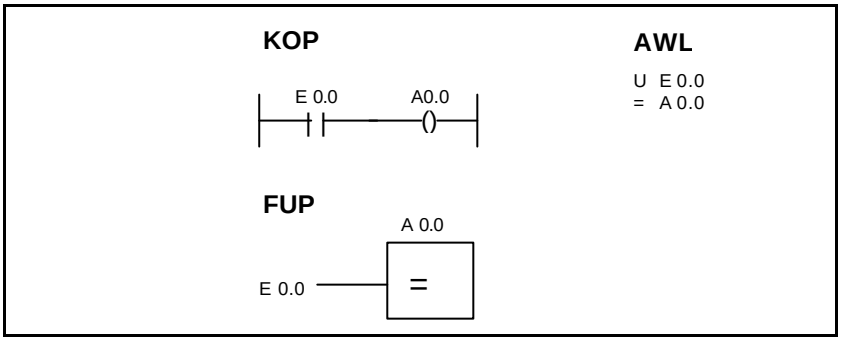
## 2. FUNDAMENTOS DE PROGRAMACIÓN

Las siguientes instrucciones de programación son suficientes para las bases de programación. No obstante, éstas no representan una lista completa de todas las instrucciones.

Para más información sobre las instrucciones en KOP/FUP/AWL puede usted buscar en el manual o aún mejor en la **ayuda Online** en el *link* **descripciones de lenguaje KOP, FUP,** así como **AWL.**

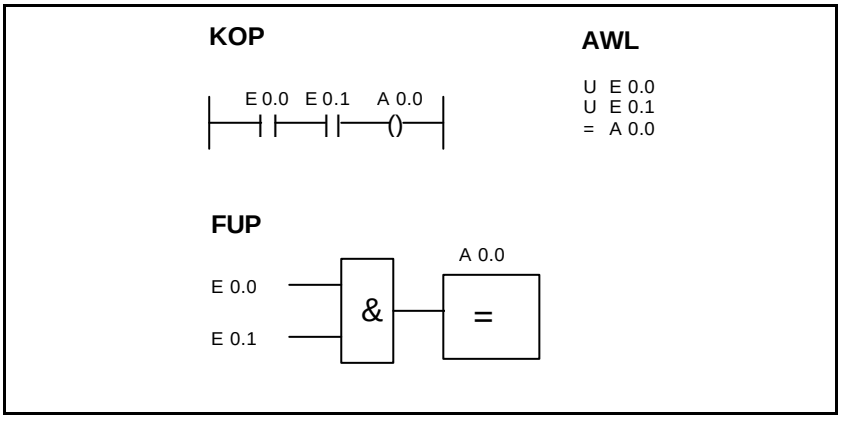
### 2.1 OPERACIÓN ASIGNAR

La operación asignar (=) copia en la siguiente operación el resultado obtenido en la operación anterior. Una unión en cadena puede ser cerrada a través de una asignación.



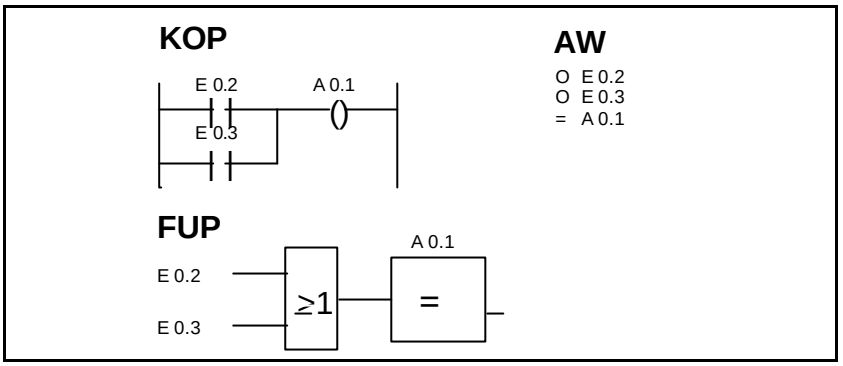

### 2.2 Unión UND

La unión UND corresponde a una conexión en serie en un diagrama de contactos. La salida A 0.0 será activa, si todas las entradas están al mismo tiempo activadas. Si una de las entradas está desactivada, la salida permace desactivada.



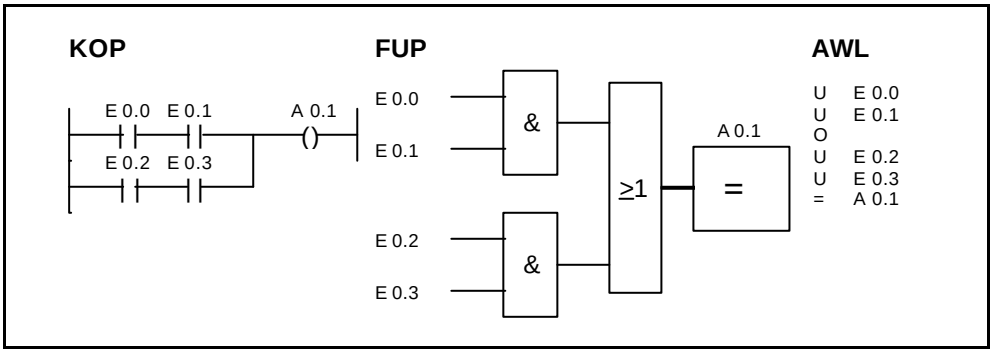
## 2.3 Unión ODER

La unión ODER corresponde a una conexión en paralelo en un diagrama de contactos. La salida A 0.1 será activa, si como mínimo está activa una de las entradas. Sólo estará la salida desactivada, en el caso que todas las entradas estén desactivadas.




## 2.4 Unión UND antes de ODER

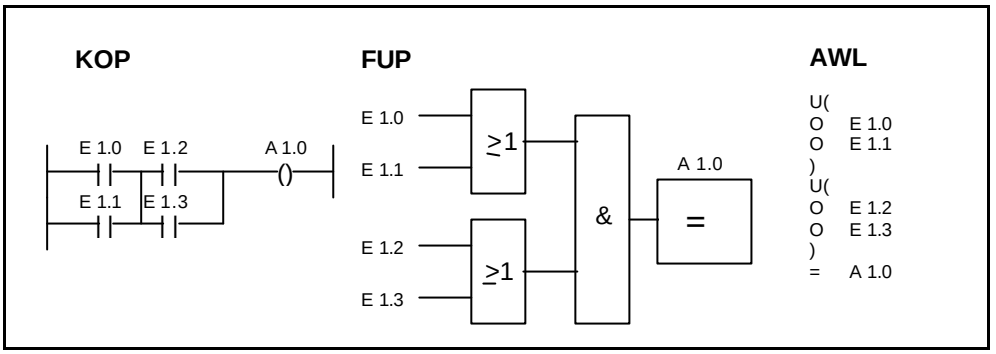
La unión UND antes de ODER corresponde a una conexión en paralelo más conexiones en serie en un diagrama de contactos.
Para que la salida 0.1 sea activa, tienen que estar como mínimo, todas las entradas de una de las conexiones en serie activas.
La unión UND antes de ODER se programa sin paréntesis en el lenguaje AWL, pero se tienen que separar las dos uniones en serie por la función ODER.
Primero se crean la funciones con UND y el resultado se une con la función ODER.
La primera unión UND (E 0.0, E 0.1) estará enlazada con la segunda unión (E 0.2, E 0.3) a través de una unión ODER.




*Las uniones UND tienen prioridad y se realizan siempre realizados antes las uniones ODER*

## 2.5 Unión ODER antes de UND

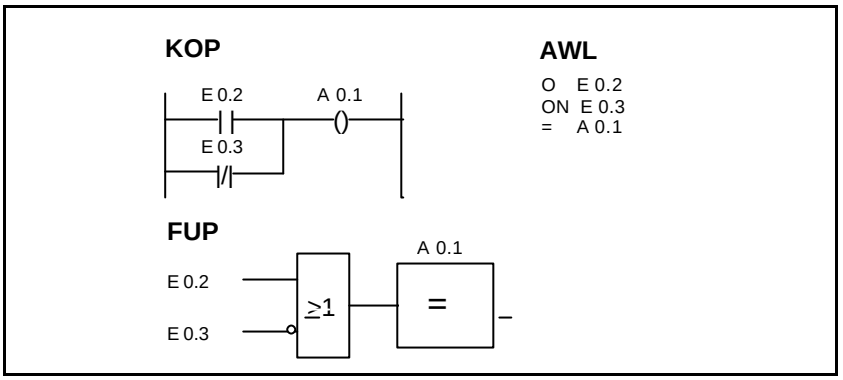
La unión ODER antes de la UND corresponde a una conexión en serie más conexiones en paralelo en un diagrama de contactos.
Para que la salida 1.0 sea activa, tienen que tener como mínimo, una entrada activa en cada una una de las ramas en paralelo.




*Con esto, la unión ODER tiene preferencia sobre la unión UND. En el lenguaje AWL hay que poner atención con los paréntesis.*

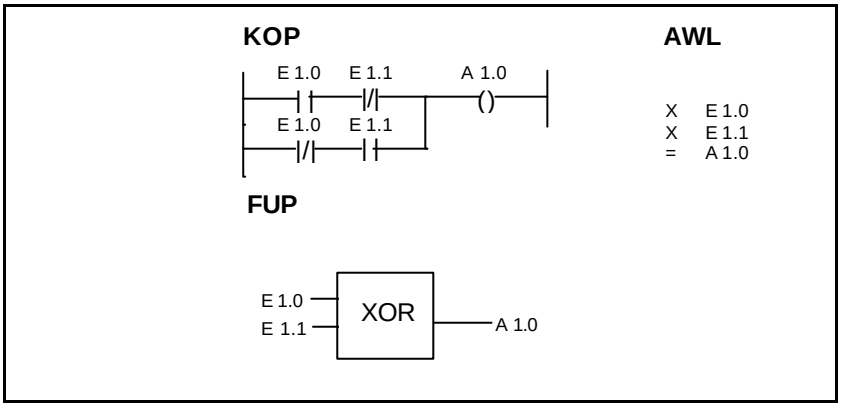
## 2.6 USO DE LA NEGACIÓN

El uso de la negación corresponde a un contacto cerrado y se realiza con las conexiones UND NICHT (UN), ODER NICHT (ON) y EXKLUSIV ODER NICHT (XN) .
Ejemplo de la unión ODER NICHT:




## 2.7 UNIÓN ODER EXCLUSIVA

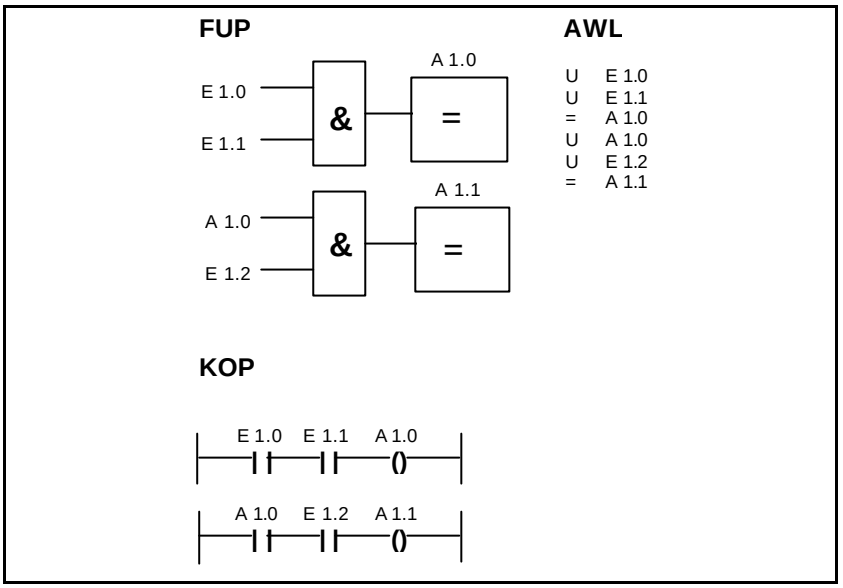
La conexión muestra una unión ODER exclusiva (X), donde la salida 1.0 sólo será activa, sólamente si una de las entradas es activa.




**Nota:** *La unión ODER exclusiva sólo puede utilizar 2 entradas.*

## 2.8 RESPUESTA A LAS SALIDAS

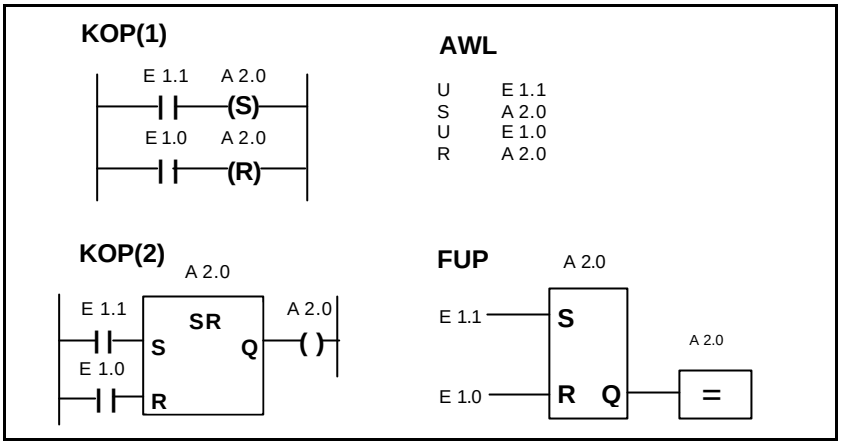
Para la conexión de las salidas A1.0 y A1.1 són válidas diferentes condiciones. En estos casos se ha de prever para cada salida, un único sentido de flujo, así como un único símbolo de enlace.
No sólo en los aparatos de automatización se puede llamar a las señales de entrada, si no que también a las de salida, como marcas, etc. Así pues, como se muestra en la figura, en el segundo esquema se llama a la salida A1.0, la cual está enlazada con una unión UND con la entrada E1.2
.




## 2.9 OPERACIONES ACTIVAR (S) Y DESACTIVAR (R)

Después de las normas DIN 40900 y DIN 19239 se representan las operaciones Activar (S) y Desactivar (R), con S forzar y R desforzar. La operación S (Activar) puede utilizarse para activar el estado de señal de un bit direccionado, es decir, para ponerlo a ‘1’. La operación R (Desactivar) puede utilizarse para desactivar el estado de señal de un bit direccionado, es decir, para ponerlo a ’0’.

### 2.9.1 PRIORIDADES DE DESFORZAR

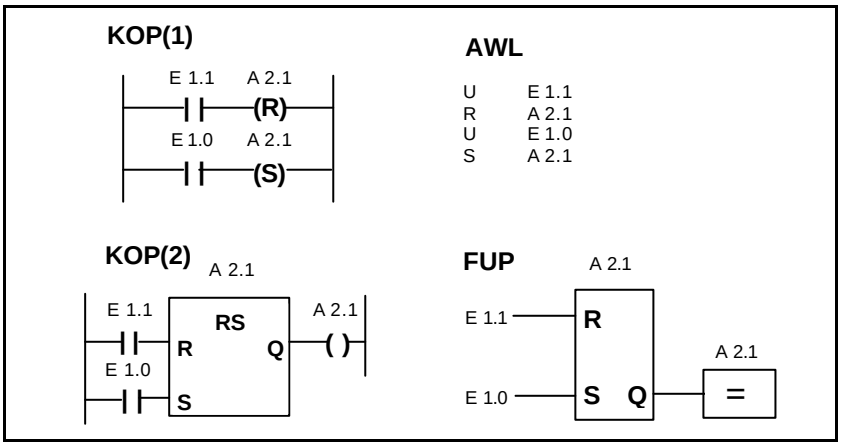



Por último, las aplicaciones programables serán programadas con preferencia desde el control. En el ejemplo, se ejecuta primero la operación forzar; la salida A 2.0 será de nuevo desforzada y permacerá en ese estado hasta que se vuelva a forzar.

*Este tiempo pequeño de forzar la salida se ejecuta sólo en el transcurso del proceso.*
*Mientras este programa sea ejecutado, este señal no tendrá ninguna influencia en los periféricos.*

### 2.9.2 PRIORIDADES DE FORZAR

Conforme al apartado 4.10.1., en este ejemplo se fuerza con prioridad la salida A 2.1.

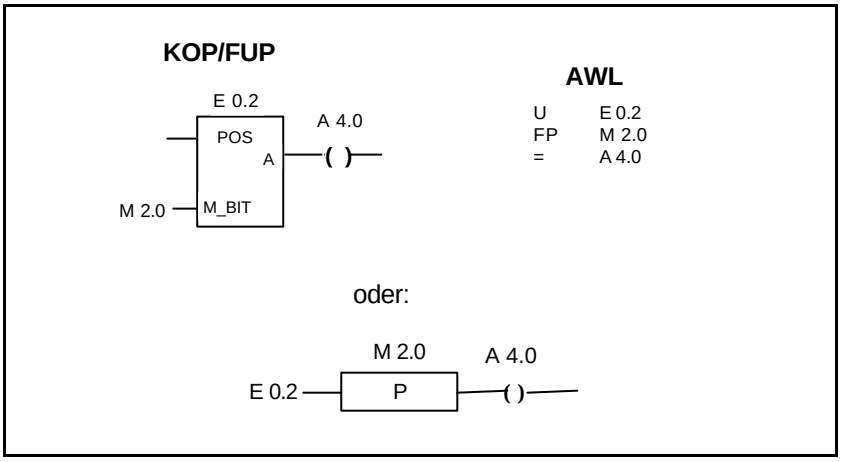


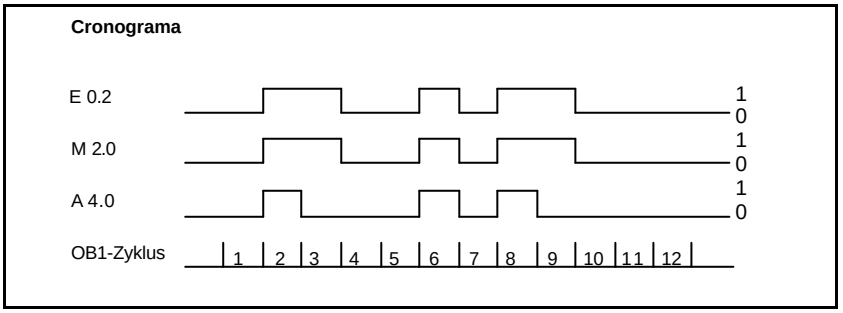
## 2.10 OPERACIONES CON FLANCOS

Las operaciones FP (flanco positivo) y FN (flanco negativo) pueden utilizarse como contactos detectores de cambio de flanco en un circuito de relé. Estas combinaciones detectan cambios en el resultado lógico y reaccionan correspondientemente.

### 2.10.1 FLANCO DE SUBIDA (FP)

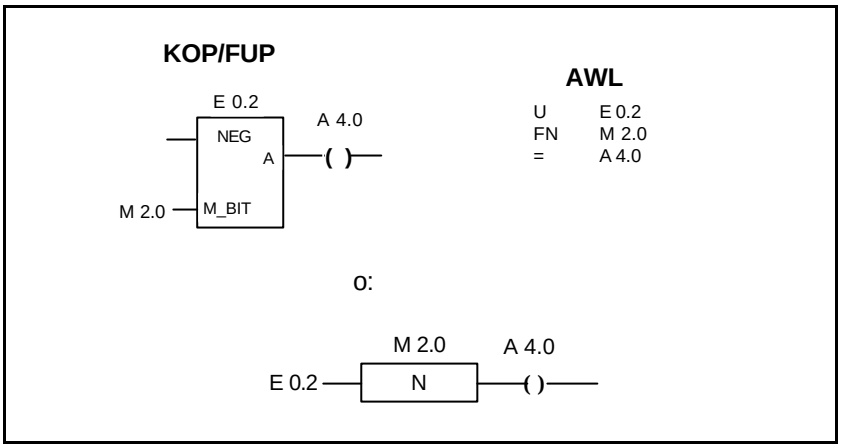
El cambio de ‘0’ a ‘1’ se denomina “flanco ascendente” (positivo). Cuando la entrada E 0.2 realiza este cambio, la salida A 4.0 será activa ‘1’ durante un ciclo OB1. Esta salida puede ser de nuevo utilizada, para p.e. forzar una marca.
La ventaja de este segundo tipo de representación en KOP/FUP es que como entrada de operación, se puede poner también como unión.

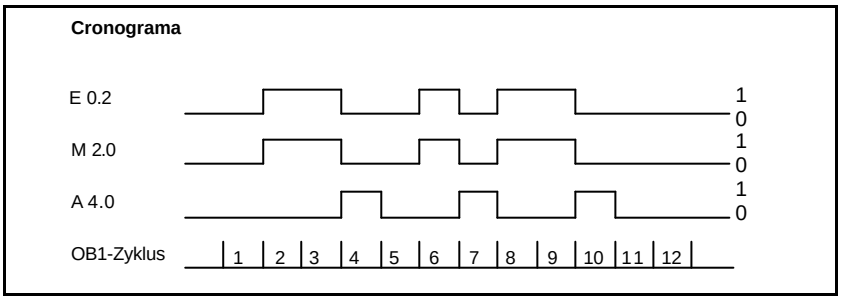


### 2.10.2 FLANCO DE BAJADA (FN)

El cambio de ’1’ a ‘0’ se denomina “flanco descendente” (negativo). Cuando la entrada E 0.2 realiza este cambio, la salida A 4.0 será activa ‘1’ durante un ciclo OB1. Esta salida puede ser de nuevo utilizada, para p.e. forzar una marca.
La ventaja de este segundo tipo de representación en KOP/FUP es que como entrada de operación, se puede también poner como unión.



## 2.11 OPERACIONES DE TEMPORIZACION

Para la realización de tareas de control se han de utilizar a menudo diferentes operaciones con temporizadores. Las operaciones de temporizadores están integradas en el grupo central del autómata. El ajuste de la duración de un periodo de tiempo deseado y el comienzo de las operaciones temporizadas ha de efectuarse en el programa de usuario.
Los Autómatas SIMATIC disponen de un determinado número de temporizadores ( dependiente de la CPU ) con diferentes operaciones de temporizadores. Cada temporizador esta clasificado en una palabra de 16 BITs.

La operaciones de temporización permiten al programa ejecutar las siguientes funciones:

### 2.11.1 HABILITAR EL TEMPORIZADOR (FR) SÓLO EN AWL

Cuando el resultado lógico cambia de ‘0’ a ‘1’ delante de una operación, habilitar temporizador (FR), se habilita el temporizador.

Para arrancar un temporizador o para ejecutar una operación de temporización normal no hace falta habilitarlo. Esta función se utiliza sólamente para redisparar un temporizador que está en marcha, es decir, para rearrancarlo. Este rearranque sólo puede efectuarse cuando la operación de arranque continúa procesándose con un VKE de ‘1’.

***La habilitación del temporizador (FR) existe sólo en la programación en AWL.***

### 2.11.2 ARRANCAR EL TEMPORIZADOR (SI/SV/SE/SS/SA)

El cambio del señal donde empieza la entrada ( flanco positivo) es la activación del temporizador. Para activar un tiempo, se han de insertar en el programa AWL, 3 instrucciones:

- *Consultar el estado de señal de la entrada*
- *Cargar el tiempo de arranque en el ACU 1*
- *Arrancar el temporizador (a elección SI, SV, SE, SS ó SA)*

```
p.e.:
U    E 0.0
L    S5T#2S
SE   T5
```

### 2.11.3 BASE DE TIEMPO DEL TEMPORIZADOR (TW)

Un temporizador debe avanzar un determinado tiempo. La duración del tiempo TW puede ser como una constante predefinida en el programa o como una palabra de entrada EW, como una palabra de salida AW, como una palabra de datos DBW/DIW, como una palabra de datos local LW, o como una palabra de marcas MW. La actualización del temporizador decrementa el valor de temporización en un intervalo dictado por la base de tiempo.

Para cargar un valor de temporización predefinido se utiliza la siguiente sintaxis:

- *L W#16#abcd*
  - siendo: a = la base de tiempo ( intervalo o resolución)
  - bcd = valor de temporización en formato BCD
- *L S5T#aH_bbM_ccS_dddMS*
  - siendo: a = horas, bb = minutos, cc = segundos y ddd = milisegundos
  - La base de tiempo se selecciona automáticamente

**Base de tiempo:**

La base de tiempo define el intervalo en que decrementa el valor de temporización.
Debido a que los valores de temporización se almacenan en un sólo intervalo de tiempo, los valores que no son exactamente múltiplos de un intervalo quedan truncados. Los valores cuya resolución es demasiado alta para el margen deseado se redondean por defecto, alcanzando el margen, pero no la resolución deseada.

| Base de tiempo | Código binario | Margen de la base de tiempo |
|---|---|---|
| 10ms | 00 | 10MS bis 9S_990MS |
| 100ms | 01 | 100MS bis 1M_39S_900MS |
| 1s | 10 | 1S bis 16M_39S |
| 10s | 11 | 10S bis 2H_46M_30S |

### 2.11.4 PONER EL TEMPORIZADOR A 0 (R)

El temporizador se borra con la operación R (Poner a 0). La CPU pone un temporizador a 0 si el resultado lógico es ‘1’ inmediatamente antes de que el programa ejecute la operación R.
Cuando se borra un temporizador, éste deja de funcionar y el valor de temporización es ‘0’.

### 2.11.5 CARGAR EL TEMPORIZADOR (L/LC)

En una palabra de contaje se encuentra almacenado un valor en código binario. En la palabra de contaje se puede cargar en el acumulador un valor en formato binario o un valor en formato BCD.

Para la programación en AWL existe la posibilidad de cargar un valor en binario *L T1* y la de cargar un valor en formato BCD *LC T1* .

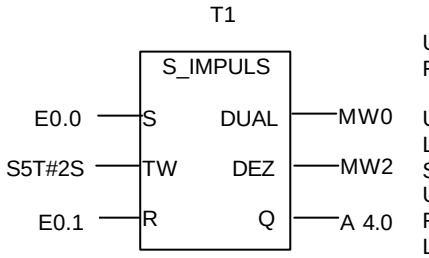
### 2.11.6 SOLICITUD DEL VALOR DE TIEMPO DEL SEÑAL (Q)

Un valor de tiempo de su señal (‘0’ o ‘1’) puede ser preguntado.
Los estados de los señales pueden ser solicitados, como de costumbre, con U T1, UN T1, ON T1,etc... y así relacionarse con más enlaces.
Existes 5 tipos diferentes de temporizadores:

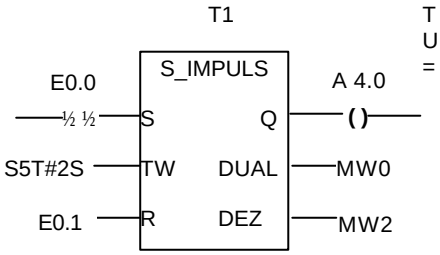
### 2.11.7 IMPULSO (SI)

La salida de un elemento temporizador, arranca con un impulso y después se ejecutar el señal 1 (1). La salida es desforzada, si la duración del tiempo programada es (2), si el señal de salida es forzada a ‘0’ o si se desactiva la entrada del elemento temporizador(4).
Un cambio positivo en el flanco ( de ‘0’ a ‘1’) en el resultado de la operación habilita la ejecución de un nuevo tiempo (5). Este nuevo intento sólo es posible, si la operación de empezar es ejecutada con el VKE ‘1’.

**FUP**



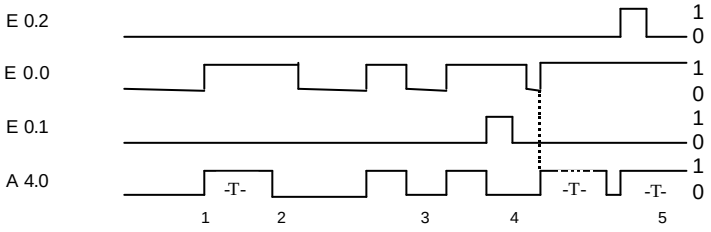

**KOP**




**AWL**

```
U   E 0.2
FR  T1      Habilitar temporizador T1 (sólo en AWL)
U   E 0.0
L   S5T#2S  Cargar el tiempo (2s) en el ACU 1
SI  T1      Arrancar temporizador T1 como impulso
U   E 0.1
R   T1      Poner temporizador T1 a 0
L   T1      Cargar el tiempo T1 en código binario
T   MW0
LC  T1      Cargar el tiempo T1 en código BCD
T   MW2
U   T1      Llamar al temporizador T1
=   A 4.0
```

**Cronograma con habilitación (FR)**

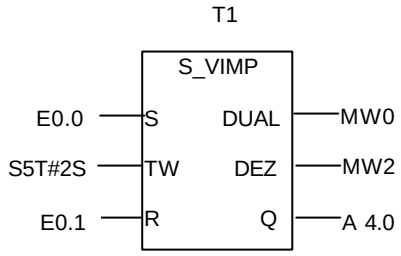


## 2.11.8 IMPULSO PROLONGADO (SV)

La salida de un elemento temporizador, arranca con un impulso prolongado y después se ejecutar el señal 1 (1). La salida es desforzada, si la duración del tiempo simulado es (2), o si se llama a la función de desforzar la entrada (5).

Una desconexión de la activación de la entrada no acarrea, mientras el tiempo transcurre, ninguna desactivación de la salida (3). La presencia de un nuevo cambio de señal, mientras el tiempo aún está transcurre, provoca que el temporizador se active de nuevo ( después del trigger) (4).

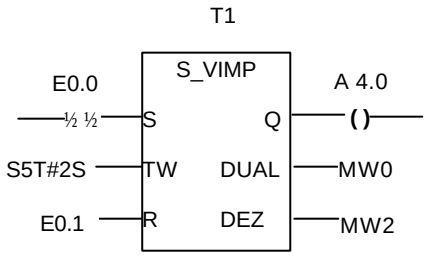
**FUP**



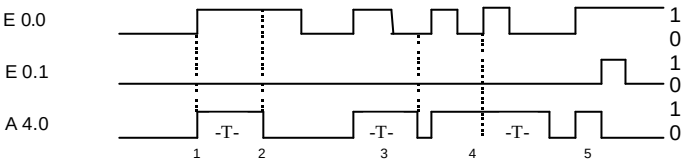

**AWL**

```
U   E 0.0
L   S5T#2S   Carga el tiempo (2s) en el ACU1
SV  T1       Arrancar temporizador T1 como impulso
U   E 0.1
R   T1       Poner temporizador T1a 0
L   T1       Cargar el tiempo T1 en código binario
T   MW0
LC  T1       Cargar el tiempo T1 en código BCD
T   MW2
U   T1       Llamar al temporizador T1
=   A 4.0
```

**KOP**




**Cronograma**

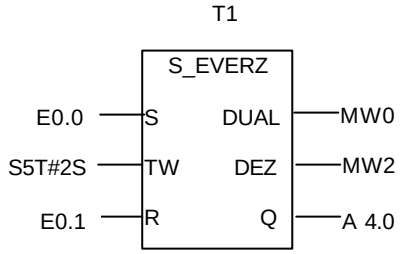


## 2.11.9 RETARDO A LA CONEXIÓN (SE)

La salida de un elemento temporizador, arranca con un retardo a partir de la ejecución del señal de una entrada 1, si el tiempo programado se está ejecutando y el VKE 1 está a la espera de la activación de la entrada (1). La habilitación de la activación de la entrada acarrea con esto un retraso en en activación de la salida Q. La salida será desforzada, cuando el señal de entrada sea desconectado o cuando el temporizador sea desactivado a través de otra entrada (3).

La salida Q no se activará, en el caso en que el señal de entrada sea desactivado mientras el tiempo aún está transcurriendo o que un señal 1 solicitado desactive la entrada del temporizador.

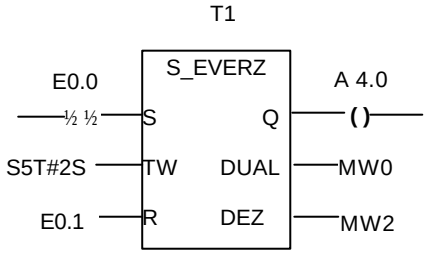
**FUP**



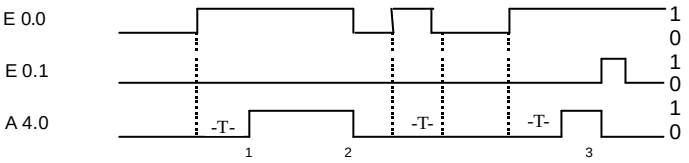

**AWL**

```
U   E 0.0
L   S5T#2S    Cargar el tiempo (2s) en el ACU 1
SE  T1        Arrancar temporizador T1 como impulso
U   E 0.1
R   T1        Poner temporizador T1 a 0
L   T1        Cargar el tiempo T1en código binario
T   MW0
LC  T1        Cargar el tiempo T1 en código BCD
T   MW2
U   T1        Llamar al temporizador T1
=   A 4.0
```

**KOP**



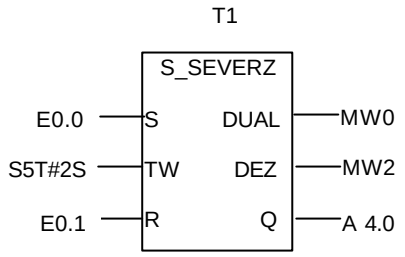

**Cronograma**



## 2.11.10 RETARDO A LA CONEXIÓN CON MEMORIA (SS)

La salida de un elemento temporizador, arranca con un retardo a la conexión con memoria a partir de la ejecución del señal de una entrada 1, si el tiempo programado se está ejecutando (1). Después de la activación de la entrada, la función no necesita ningún otro VKE 1, por lo tanto, éste puede ser desconectado (automáticamente) (3).
La salida será entonces sólo desconectada, si se activa la desconexión de la entrada del temporizador (2). Una nueva activación o desactivación de la entrada, mientras el tiempo está transcurriendo, provoca que el temporizador se active de nuevo (después del trigger) (4).

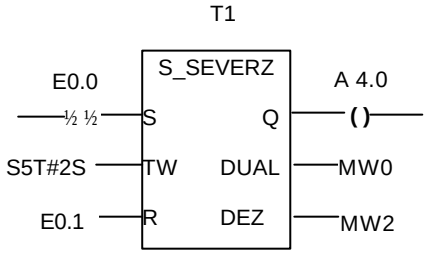
**FUP**



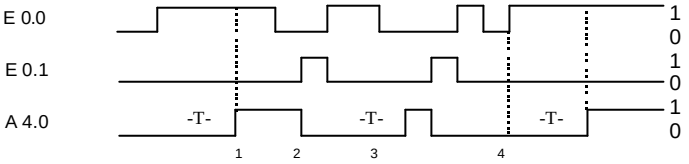

**AWL**

```
U   E 0.0
L   S5T#2S    Cargar el tiempo (2s) en el ACU 1
SS  T1        Arrancar temporizador T1 como impulso
U   E 0.1
R   T1        Poner temporizador T1 a 0
L   T1        Cargar el tiempo T1en código binario
T   MW0
LC  T1        Cargar el tiempo T1 en código BCD
T   MW2
U   T1        Llamar al temporizador T1
=   A 4.0
```

**KOP**

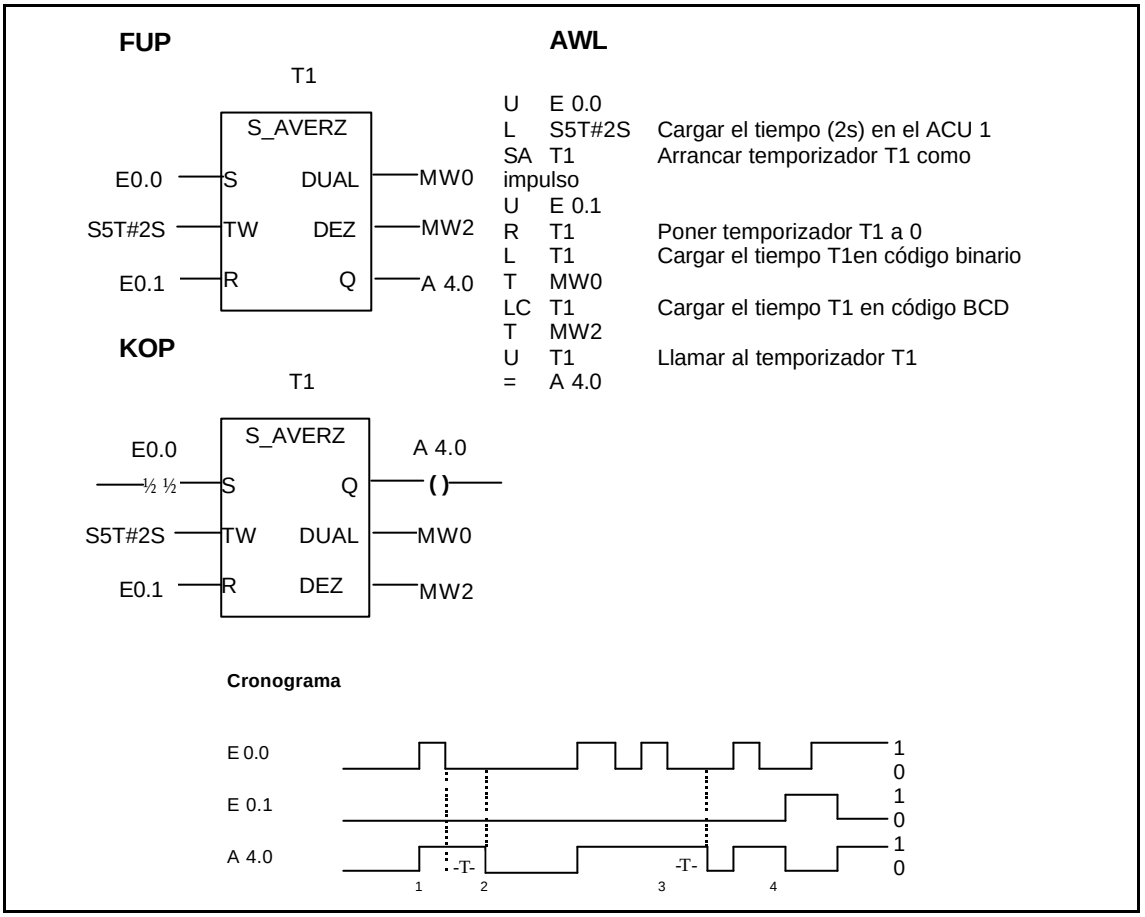



**Cronograma**



## 2.11.11 RETARDO A LA DESCONEXIÓN (SA)

En un cambio de señal ( flanco positivo ) en la activación del temporizador, se activa el retardo a la conexión y la salida Q del temporizador (1). Cuando la entrada se desconecta, la salida sigue en estado activo hasta que el tiempo programado haya transcurrido (2). La desactivación de la entrada ( flanco negativo ), provoca que la salida siga activa durante el tiempo que el temporizador esté programado.
La salida del temporizador será también desconectada, cuando una entrada provoque un reset a la salida (4). Si se produce una desactivación de la entrada (flanco negativo), mientras el tiempo está transcurriendo, se activará de nuevo el temporizador, hasta que éste haya consumido su tiempo programado (3).



## 2.12 GENERADOR DE IMPULSOS DE RELOJ

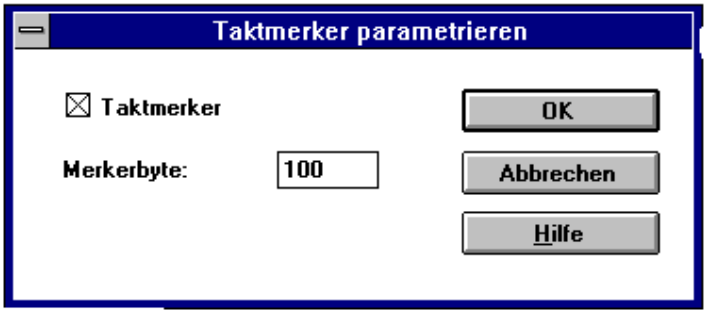
Los contadores se emplean para diferentes instalaciones de control, regulación y vigilancia.
En la tecnología digital se utilizan en circuitos biestables.
El empleo de generadores de impulsos de reloj es habitual en los sistemas de señalización que controlan la intermitencia de las lámparas de indicación.

*En caso de utilizar la S7-300, es posible implementar la función de generador de impulsos de reloj utilizando un procesamiento controlado por tiempo en bloques especiales.de organización.*

**Configurar el tiempo de marca:**

El tiempo de marca son marcas dentro de un „ tiempo de marca byte“.

Si se activa el tiempo marca ( la cruz en la casilla de control es visible ), entonces se han de confirmar también el número de marcas de byte. La marca de byte distinguida, no puede ser utilizada para guardar los datos.




**Periodo de duración de los tactos:**

Cada bit del tiempo de marca byte se clasifica por duración del período / frecuencia. Existe la siguiente tabla:

| **Bit:** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| Duración del periodo (s): | 2 | 1,6 | 1 | 0,8 | 0,5 | 0,4 | 0,2 | 0,1 |
| Frecuencia (Hz): | 0,5 | 0,625 | 1 | 1,25 | 2 | 2,5 | 5 | 10 |

## 2.13 OPERACIONES DE CONTAJE

En el control técnico son necesarios el registro del número de impulso, la utilización de reglones y la utilización de operaciones de contaje a distancia. SIMATIC S7 dispone de contadores integrados en el grupo central de trabajo. Los contadores tienen una área de memoria reservada en la propia CPU. El valor de contaje puede estar comprendido entre 0 y 999.

La siguientes funciones pueden ser programadas en un contador:

### 2.13.1 HABILITAR CONTADOR (FR) SÓLO EN AWL

Un flanco positivo ( de ‘0’ a ‘1’ ) da como resultado la habilitación contador (FR).
No es necesario habilitar el contador ni para ponerlo a un valor determinado ni para operaciones de contaje normales. La habilitación se utiliza sólamente para activar un contador o para contar adelante o atrás sin necesidad de que se produzca un flanco positivo (cambio de ‘0’ a ‘1’) delante de la instrucción de contaje correspondiente( ZV, ZR o S ). La habilitación sólo puede efectuarse si el bit VKE de la operación correspondiente está a ”1”.

***La habilitación del temporizador (FR) existe sólo en la programación en AWL.***

### 2.13.2 INCREMENTAR CONTADOR (ZV)

Cuando en el programa AWL el resultado lógico cambia de ‘0’ a ‘1’ antes de una instrucción incrementar contador (ZV), se incrementa el valor del contador. Cada vez que el VKE cambia de ‘0’ a ‘1’ inmediatamente antes de una operación Incrementar contador, el valor del contador se incrementa en 1 unidad. Cuando el contador alcanza el límite superior de 999, se detiene y los cambios posteriores del estado de señal no tienen efecto alguno sobre la entrada de contaje ade-lante.

### 2.13.3 DECREMENTAR CONTADOR (ZR)

Cuando en el programa AWL el resultado lógico cambia de ‘0’ a ‘1’ antes de una instrucción decrementar contador (ZR), decrementa el valor del contador. Cada vez que el VKE cambia de ‘0’ a ‘1’ se decrementa en 1 unidad. Cuando el contador alcanza el límite inferior de ‘0’, se detiene. Los cambios posteriores del estado de señal no tienen efecto alguno sobre la entrada de contaje atrás. El contador no funciona con valores negativos.

### 2.13.4 INIZIALIZAR CONTADOR (S)

Para inizializar un contador, se han de insertar en el programa AWL tres instrucciones:

- *Consultar el estado de señal de la entrada*
- *Cargar el tiempo de arranque en el ACU 1*
- *Arrancar el temporizador con la carga del valor en el contador.*
  Esta función sólo será realizada en los cambio de flanco positivo.

```
p.e.:
U    E 2.3
L    C#5
S    Z1
```

### 2.13.5 BASE DE TIEMPOS DE CONTADOR (ZW)

Los contadores tienen la posibilidad de cargar un valor ya sea en código binario o en código BCD. Los operandos posibles son:

- *Palabra de entrada* *EW ..*
- *Palabra de salida* *AW ..*
- *Palabra de marcas* *MW ..*
- *Palabra de datos* *DBW/DIW ..*
- *Palabra de datos local* *LW ..*
- *Constante* *C#5, 2#...etc.*

### 2.13.6 PONER CONTADOR A 0 (R)

Un VKE 1 pone el contador a 0 (reset). Un VKE 0 no tiene tiene ninguna influencia sobre el contador. La desactivación de un contador provoca estabilidad.

### 2.13.7 CARGAR CONTADOR (L/LC)

En una palabra de contaje se encuentra almacenado un valor en código binario. En la palabra de contaje se puede cargar en el acumulador un valor en formato binario o un valor en formato BCD.

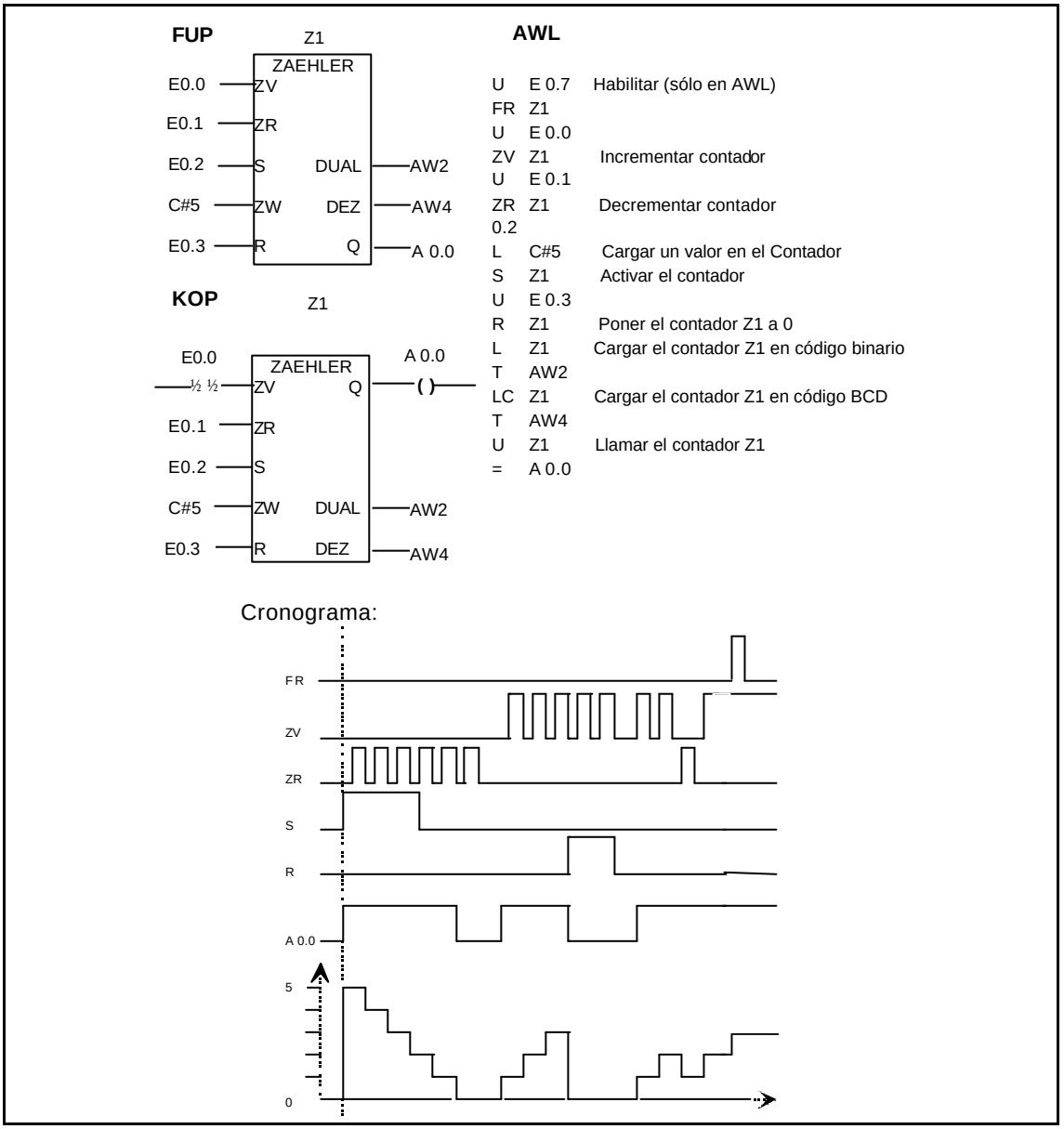
Para la programación en AWL existe la posibilidad decargar un valor en binario *L Z1* y la de cargar un valor en formato BCD *LC T1* .

## 2.13.8 SOLICITUD DEL VALOR DE TIEMPO DEL SEÑAL (Q)

El contador puede ser solicitado por su señal. Con esto significa:

*Señal 0 = El contador permanece con el valor 0;*
*Señal 1 = El contador funciona, es decir, está contando.*

Los estados de las señales pueden ser solicitados, como de costumbre, con U Z1, UN Z1, ON Z1,etc... y así relacionarse con más enlaces.

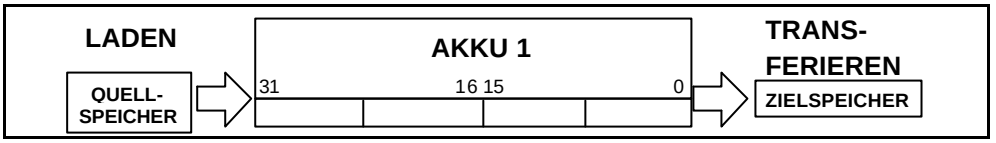


## 2.14 OPERACION DE CARGA Y DE TRANSFERENCIA (L/T) SOLO EN AWL

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.
Este intercambio de información no se produce directamente, siempre a través del Acumulador 1 ( ACU 1). El ACU 1 es un registro de la CPU que sirve de memoria intermedia.

El flujo de la información tiene este sentido:

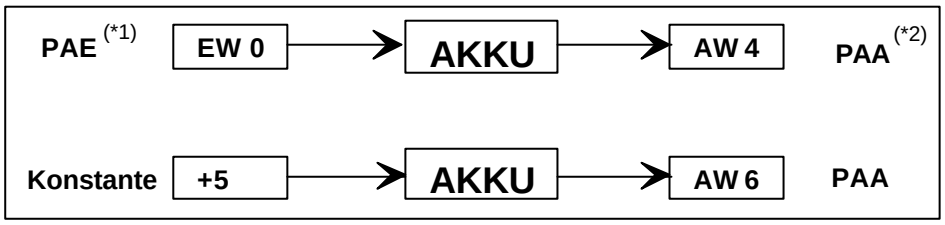
*CARGAR:* *Se guarda de la memoria fuente al acumulador*
*TRANSFERIR:* *del acumulador a la memoria destino*




Para cargar el contenido de la memoria fuente, se copia el contenido de éste y se escribe en el ACU 1.
Para transferir el contenido del acumulador, se copia el contenido de éste y se escribe en la memoria de destino.
Allí sólo será copiado el contenido del acumulador, donde éste está esperando para realizar otras operaciones de transferencia.

**AWL:**

```
: L   EW   0
: T   AW   4
: L   +5
: T   AW   6
: BE
```




*1: Imagen del proceso de las entradas *2: Imagen del proceso de las salidas

Cargar y transferir son operaciones incondicionales, la independencia del resultado se efectua en cada ciclo.

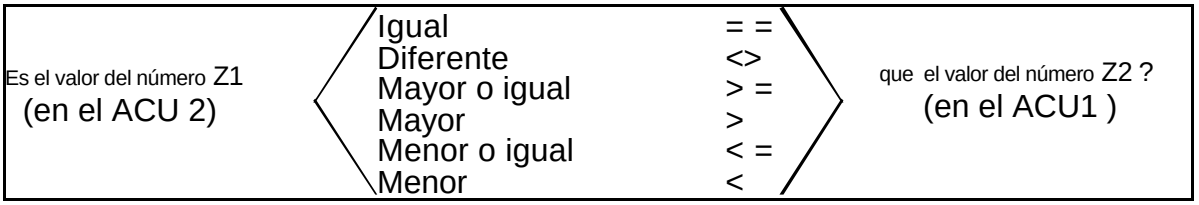
## 2.15 OPERACIONES DE COMPARACION

El lenguaje de programación STEP 7 ofrece la posibilidad, de comparar directamente dos valores y entregar directamente el resultado de la operación (VKE). La condición es que ambos valores tienen que tener el mismo formato.

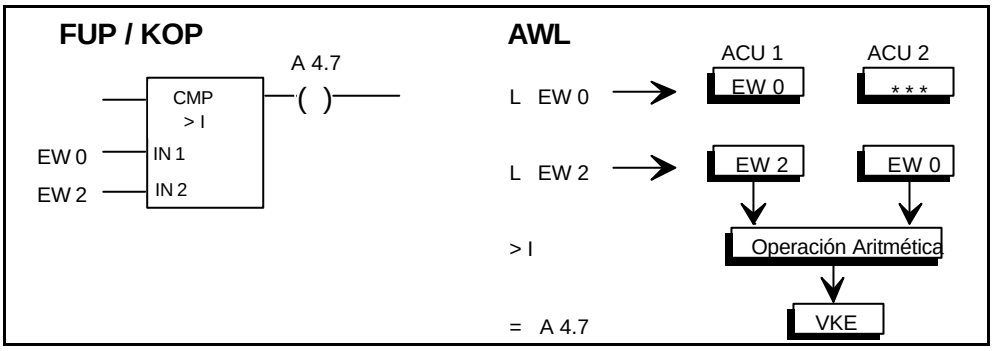
Los siguientes tipos de número pueden ser comparados:

- *dos enteros ( 16 Bit Símbolo: I )*
- *dos enteros ( 32 Bit Símbolo: D )*
- *dos reales ( dos números en coma flotante 32 Bit, Símbolo: R )*

Se puede elegir entre 6 diferentes comparaciones:




Con las funciones de comparación se comparan dos valores uno con el otro, los cuales están en el ACU 1 y el ACU 2. Con la primera operación se carga el primer operando (p.e. EW 0) en el ACU 1. Con la segunda operación de carga se transfiere el primer operando del ACU 1 al ACU 2 y después se carga el segundo operando (p.e. EW 2) en el ACU 1. A continuación los dos valores que están en los acumuladores serán comparados. El resultado de la comparación es un valor binario. Si el resultado de la comparción ha sido satisfactorio, el resultado será 1. Si el resultado de la comparación no ha sido satisfactorio, el resultado del VKE será 0.



## 2.16 OPERACIONES DE CONTROL DEL PROGRAMA

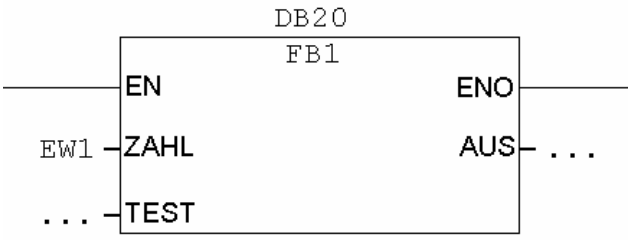
### 2.16.1 LLAMAR A FUNCIONES Y A BLOQUES DE FUNCIÓN CON (CALL)

La operación de llamada CALL se utiliza para llamar a funciones (FCs) y al bloque de función (FBs) así como a la función del sistema (SFCs) y al bloque de función del sistema (SFBs).
La operación CALL llama a la función FC o al bloque FB indicado como operando, independientemente del resultado lógico o de cualquier otra condición.

**AWL**

| | | |
|---|---|---|
| CALL | FB1, DB20 | |
| ZAHL := | EW 1 | ZAHL (Parámetro formal) es asignado a EW (Parámetro actual) |
| AUS := | | AUS (Parámetro formal) no es asignado ningún parámetro. |
| TEST := | | TEST (Parámetro formal) no es asignado ningún parámetro. |

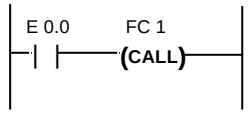
**KOP/FUP**




### 2.16.2 LLAMADA CONDICIONADA (CC)

La operación de llamada CC se utiliza para llamar funciones (FCs) y a bloques de función (FBs) así como a la función del sistema (SFCs) y a los bloques de función del sistema (SFBs). Sin embargo no pueden transferir ningún parámetro así como describir ninguna variables.
La llamado sólo será ejecuta, si el resultado de la operación es '1'.

**KOP/FUP**

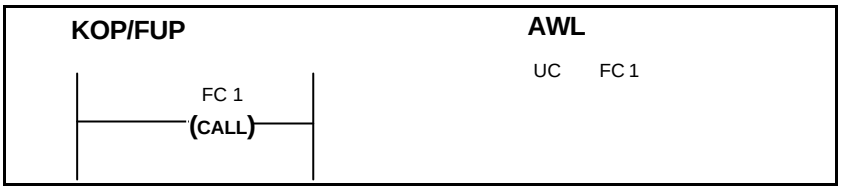



**AWL**

```
U     E 0.0
CC    FC 1
```

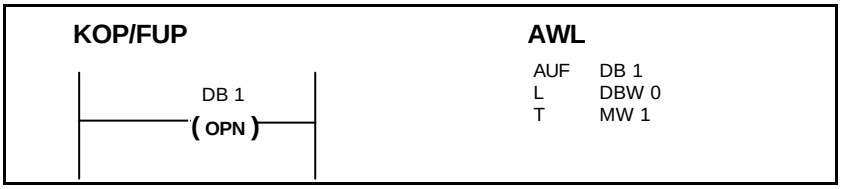
### 2.16.3 LLAMADA INCONDICIONADA (UC)

La operación de llamada CC se utiliza para llamar funciones (FCs) y a bloques de función (FBs) así como a la función del sistema (SFCs) y a los bloque de función del sistema (SFBs). Sin embargo no pueden transferir ningún parámetro así como describir ninguna variables.
La llamada se ejecuta independientemente del resultado de la operación.



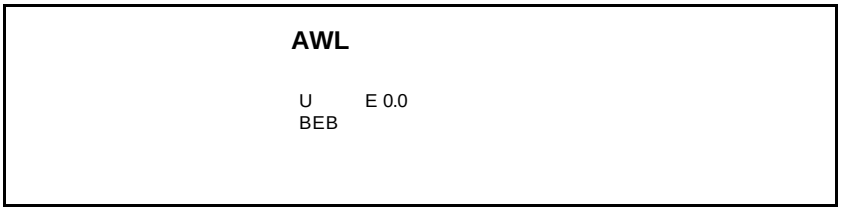

### 2.16.4 BLOQUE DE DATOS (AUF)

La operación de bloque de datos (AUF) sirve para abrir un bloque de datos global (DB) o un bloque de datos de instancia (DI). En el programa pueden estar simultáneamente abiertos, un bloque de global de datos y un bloque de instancia (p.e. con las operaciones de carga y transferencia)



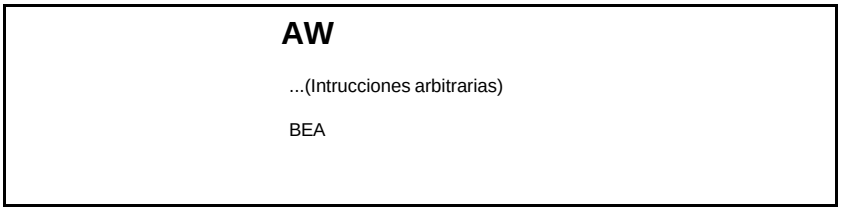

### 2.16.5 FIN DE BLOQUE CONDICIONAL (BEB) SOLO EN AWL

Esta operación finaliza la ejecución del bloque actual y devuelve el control al bloque que llamó al que acaba de ser ejecutado. Cuando el programa encuentra una operación BEB sólamente finaliza el bloque actual si el resultado lógico es ‘1’.



### 2.16.6 FIN DE BLOQUE INCONDICIONAL (BEA) SOLO EN AWL

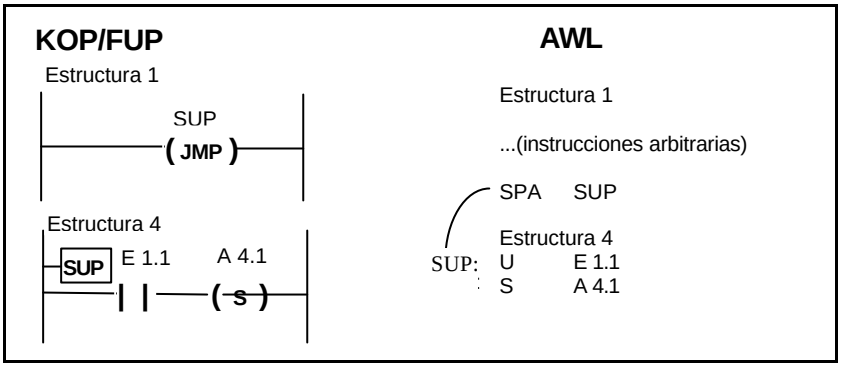
Esta operación finaliza la ejecución del bloque actual y devuelve el control al bloque que a su vez ha llamado al bloque que acaba de ser ejecutado. Cuando el programa encuentra una operación BEA finaliza el bloque actual, independientemente del resultado lógico.



## 2.17 OPERACIONES DE SALTO

### 2.17.1 SALTO ABSOLUTO (SPA)

Las operaciones SPA interrumpen el desarrollo normal del programa, haciendo que el programa salte a un punto determinado. El salto se efectua independientemente de las condiciones.

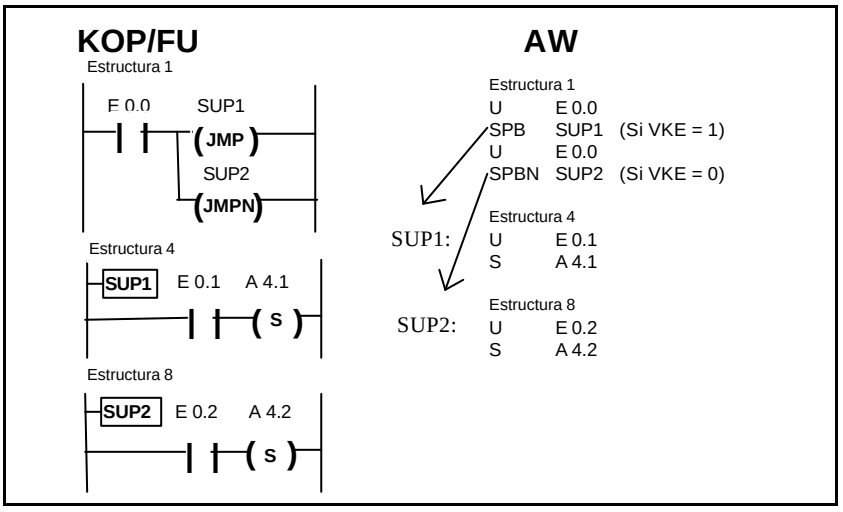



### 2.17.2 SALTO CONDICIONADO (SPB/SPBN)

La operación del salto condicionado interrumpe la ejecución normal del programa y inicializa un salto hacia otro operando solicitado.
La ejecutación del salto es dependiente del resultado de la operación.

Las siguientes operaciones de salto condicionado se pueden ejecutar:

- *SPB : Salto, si VKE = 1*
- *SPBN : Salto, si VKE = 0*



### 2.17.3 BUCLE (LOOP) SÓLO EN AWL

La operación LOOP (bucle) sirve para llamar varias veces un segmento del Programa.
La operación LOOP decrementa la palabra baja del ACU 1 en '1'. Después se comprueba el valor depositado en la palabra baja del ACU 1. Si no es igual a '0', se ejecuta un salto a la meta indicada en la operación LOOP. En caso contrario, se ejecuta la siguiente operación.

```
      L     5
NEXT: T     MB 10
      :
      :
      L     MB 10
      LOOP  NEXT
```

***El bucle (LOOP) existe sólo en la programación en AWL.***

## 2.18 OPERACIONES NULAS

### 2.18.1 OPERACIÓN NULO 0 / 1 (NOP0/NOP1) SÓLO EN AWL

Estas operaciones no ejecutan ninguna función ni varian el contenido del bit de estado. Las operaciones nulas son necesarias para decompilar, p.e. de AWL a KOP

## 2.19 ELABORACION DE VKE

En STEP 7 hay operaciones en las que se puede cambiar el resultado de la operación (VKE). Puesto que además el VKE tiene una influencia directa, las operaciones no poseen ningún operando.

### 2.19.1 NEGACION DE VKE (NOT) SOLO EN AWL

La operación NOT puede utilizarse en el programa para invertir el valor actual del VKE. Si el valor actual es VKE '0', entonces la operación NOT lo convierte en '1'; Si el valor actual es VKE '1', entonces la operación NOT lo convierte en '0'.

### 2.19.2 FORZAR VKE (SET) SÓLO EN AWL

La operación SET puede utilizarse en el programa para forzar el VKE incondicionado a '1'.

### 2.19.3 DESFORZAR VKE (CLR) SÓLO EN AWL

La operación CLR puede utilizarse en el programa para forzar el VKE incondicionado a '0'.

### 2.19.4 GUARDAR VKE (SAVE) SÓLO AWL

La operación SAVE puede utilizarse en el programa para almacenar el VKE para usarlo posteriormente.

Estructura de la palabra de estado:

| | BIE | A1 | A0 | OV | OS | OR | STA | VKE | /ER |
|---|---|---|---|---|---|---|---|---|---|
| p.e.: | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |

| **Lista de instrucciones:** | **Estado del señal:** | **Resultado(VKE):** |
|---|---|---|
| SET | | 1 |
| = M 1.0 | 1 | |
| = E 0.0 | 1 | |
| CLR | | 0 |
| = M 1.0 | 0 | |
| = E 0.0 | 0 | |
| NOT | 1 | |
| SAVE | 1 | guardar el Bit BIE en la palabra de estado |