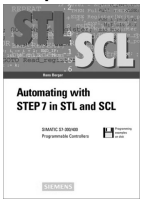
Tema 5

Temporizadores y Contadores

Bibliografía

- Título: “Step7 Avanzado”
  - Autor: José Martínez Torres
  - Descargar de la página web
- Manual Siemens “Step7-AWL para S7-300 y S7-400”
- Manual Siemens “Step7-KOP para S7-300 y S7-400”
- Manual Siemens “Step7-FUP para S7-300 y S7-400”
- Título: “Automating with Step7 in STL and SCL”
  - Autor: Hans Berger
  - ISBN: 3-89578-140-1



Bibliografía

- Título: “Comunicaciones Industriales”
  - Autores: V.Sempere, J. Silvestre, J.A. Martínez
  - Editorial : SPUPV (SPUPV-2002.213)
  - Año:2002

Índice

- Acumuladores
- Operaciones de carga y transferencia
- Temporizadores
  - Area de memoria y componentes de un temporizador
  - Programación
  - Tipos
- Contadores

Acumuladores

- Los acumuladores son registros auxiliares en la CPU que se utilizan en el intercambio de datos y para operaciones de comparación y matemáticas. El S7-300 tiene dos acumuladores de 32 bits cada uno y el S7-400 cuatro.

ACU	Bit
ACUx (x = 1 a 2)	Bits 0 a 31
ACUx-L	Bits 0 a 15
ACUx-H	Bits 16 a 31
ACUx-LL	Bits 0 a 7
ACUx-LH	Bits 8 a 15
ACUx-HL	Bits 16 a 23
ACUx-HH	Bits 24 a 31

Acumuladores

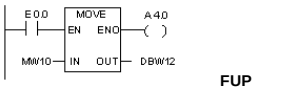
- Las siguientes instrucciones están disponibles para intercambiar y desplazar el contenido de los acumuladores:
  - **TAK** intercambia el contenido de ACCU 1 con el contenido de ACCU 2
  - **PUSH** desplaza el contenido de ACCU 1 a ACCU 2
  - **POP** desplaza el contenido de ACCU 2 a ACCU 1

Operaciones de Carga y Transferencia

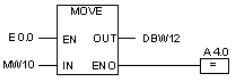
- No dependen del valor del RLO

Carga y Transferencia de Datos

KOP



FUP



EN - Habilitar entrada
ENO -Habilitar salida
IN - Valor de entrada
(todos los tipos de datos con longitud 8, 16, 32 bit)
O - Dirección del resultado
(todos los tipos de datos con longitud 8, 16, 32 bit)

AWL

L - Carga T - Transferencia
(Todos los tipos de datos con 8, 16, 32 bits)

Ejemplos:

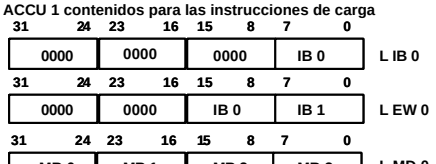
```
L +5                    // Carga un entero de 16-bit
L L#523123              // Carga un entero de 32-bit
L B#16#EF               // Carga un hexadecimal de 8-bit
L 2#0001_0110_1110_0011
                        // Carga un binario de 16-bit
L TOD#1:10:3.3
                        // Carga un tiempo de 32-bit

T MB0                   // Transferir un valor al byte de
                           marcas 0
T AD256                 // Transferir un valor a la doble
                           doble palabra de salida 256
```


Operaciones de Carga y Transferencia

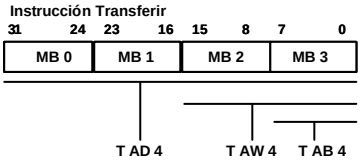
- **Operación de Carga**
  - La operación de carga siempre afecta al ACCU 1. Las posiciones no utilizadas se ponen a 0. El valor actual del ACCU 1 pasa al ACCU 2 durante la carga.
- **Operación de Transferencia**
  - Durante una transferencia, el contenido de ACCU 1 se retiene y se usa para transferir la información a varias áreas de memoria. Si sólo se transfiere un byte se usan los ocho bits de la derecha.

Acumuladores

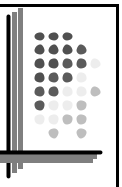
ACCU 1 contenidos para las instrucciones de carga



Instrucción Transferir

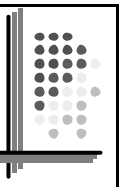


Area de memoria y componentes de un temporizador



- **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.
- Problema: de contaje de tiempo limitado a 9990 segundos como periodo máximo de contaje.
- Para periodos de tiempo mayores, es obligatorio recurrir a las OB´s de alarma horaria, o realizar un concatenamiento entre un generador de pulsos y un contador que vaya incrementando su valor.

Area de memoria y componentes de un temporizador



- Un temporizador en S7 se compone de:
  - Una palabra de 16 bits que identifica su valor actual de contaje.
    - En la palabra del temporizador es donde cargaremos el valor de contaje, junto con su base de tiempos, y podremos consultarla para conocer su estado durante el descontaje.
  - Un bit, que identifica su estado (activado o desactivado).
    - El bit nos activará acciones cuando finalice o mientras se desarrolle el proceso de contaje.

Area de memoria y componentes de un temporizador

- **Valor de temporización**
  - Los bits 0 a 9 de la palabra de temporización contienen el valor de temporización en código binario. Este valor indica un número de unidades. La actualización decrementa el valor de temporización en una unidad y en el intervalo indicado por la base de tiempo hasta alcanzar el valor 0.
  - El valor de temporización se puede cargar en los formatos binario, hexadecimal o decimal codificado en binario (BCD).
  - Para cargar un valor de temporización redefinido, se observarán las siguientes reglas sintácticas.
    - El valor de temporización se puede cargar en cualesquiera de los siguientes formatos:
      - w#16#wxyz
        - siendo: w= la base de tiempo (es decir, intervalo de tiempo o resolución)
        - xyz = el valor de temporización en formato BCD
      - S5T#**a**H_**b** M_**c**S_**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.

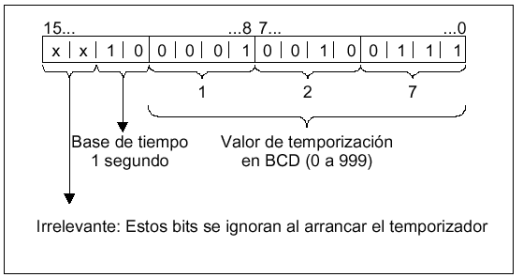
Area de memoria y componentes de un temporizador

- **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.
  - 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.

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

Area de memoria y componentes de un temporizador

- **Configuración binaria en la palabra de temporización**
  - Cuando se dispara un temporizador, el contenido de la palabra de temporización 1 se utiliza como valor de temporización. Los bits 0 a 11 de la palabra de temporización almacenan el valor de temporización en formato decimal codificado en binario (formato BCD: cada grupo de cuatro bits contiene el código binario de un valor decimal). Los bits 12 a 13 almacenan la base de tiempo en código binario.
- La figura muestra el contenido de la palabra de temporización cargado con el valor 127 y una base de tiempo de 1 segundo.



Area de memoria y componentes de un temporizador

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

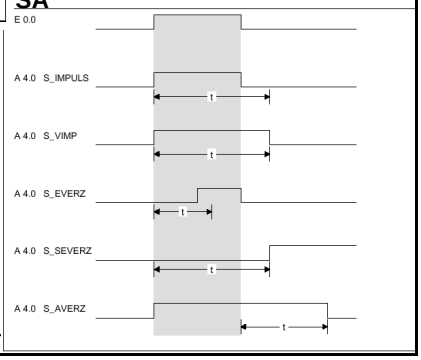
SI

SV

SE

SS

SA



Programación

- Veamos cual es la estructura de un temporizador en AWL mediante un ejemplo:

```
U E 0.0          // SI SE ACTIVA LA ENTRADA
L S5T#5S         // CARGA EN EL ACUMULADOR 5 SEGUNDOS
SI T 0           // ACTIVA EL TEMPORIZADOR 0 EN FORMATO SI CON 5
                 //SEGUNDOS
U T 0            // MIENTRAS ESTÉ ACTIVO EL TEMPORIZADOR
= A 4.0          // ACTIVA LA SALIDA
```

- Las tres primeras líneas realizan la carga del valor de tiempos en el temporizador, y además activan su arranque. A partir de ese instante comienza a descontar el valor actual del temporizador cada x tiempo especificado en la base de tiempos del temporizador, hasta llegar a 0, donde finaliza su contaje.
- Dependiendo del tipo de temporizador que hayamos seleccionado en la instrucción Sx T0 (siendo x el tipo de temporizador) se comportará su bit de estado de una manera u otra.

Programación

- También es posible resetear el temporizador mediante una entrada, con lo cual el valor del temporizador pasa a 0 y el bit del mismo se deshabilita automáticamente.

```
U E 0.1          // SI ESTA LA ENTRADA
R T 0            // EL TEMPORIZADOR SE RESETEA
```

- Otra posibilidad es relanzar el contaje del temporizador, mediante la función FR de liberación de temporización. Cuando se active la entrada, el contador comienza de nuevo su proceso de contaje desde el último valor que se le había asignado como valor preseleccionado.

```
U  E 0.3         // SI ESTA LA ENTRADA
FR T 0           // COMIENZA DE NUEVO EL CONTAJE
```

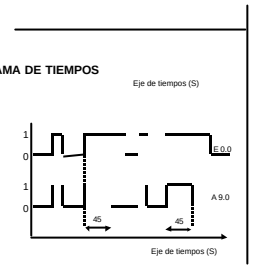
- Por último nos puede ser interesante conocer el estado actual del temporizador (cuanto tiempo le resta por contar). Para ello, únicamente debemos de cargar el valor de la palabra del temporizador. Esta carga se puede realizar de dos modos: normal en formato decimal (para comparaciones), o codificada en formato BCD (utilizada en displays).

```
L T 0            // CARGA EL VALOR ACTUAL DEL TEMPORIZADOR
T MW 0           // TRANSFIERELO EN DECIMAL

LC T 0           // CARGA CODIFICADO EL VALOR EL TEMPORIZADOR
T MW 2           // TRANSFIERELO EN FORMATO BCD
```

Tipos de Temporizadores

- **Temporizador de Impulso SI**
  - Al arrancar un temporizador SI, se obtiene una respuesta de duración idéntica al impulso producido o, como máximo, igual al valor de tiempo prefijado en nuestra constante de tiempo.
  - Si la entrada S cambia de 0 a 1, el temporizador arranca. El temporizador **SI** marcha con el valor de tiempo indicado en TW. Si la entrada S cambia de 1 a 0 antes de terminar el tiempo, el temporizador se detiene. Si la entrada R cambia de 0 a 1 en cualquier momento, el temporizador se resetea. La salida entrega un 1 mientras el temporizador cuenta.

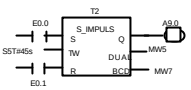
Temporizador de impulso S_IMPULS



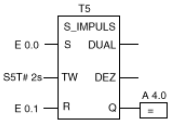
AWL

```
S_IMPULS
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
```

KOP



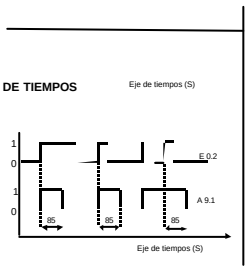
FUP



Tipos de Temporizadores

- **Temporizador de Impuso Prolongado 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.

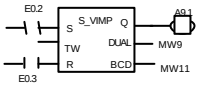
Temporizador Impulso prolongado S_VIMP



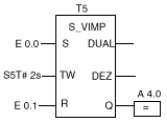
AWL

```
S_VIMP
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
```

KOP

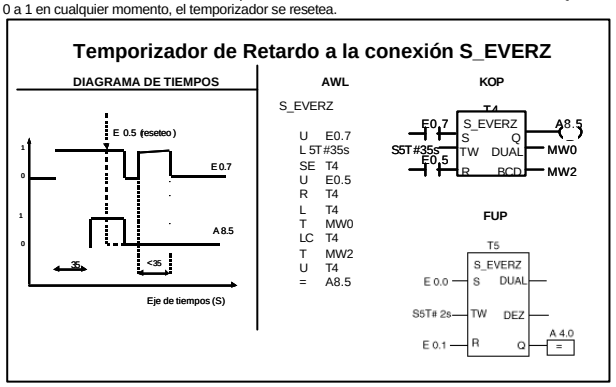


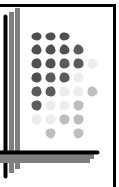
FUP



Tipos de Temporizadores

- **Retardo a la Conexión 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 se ñal 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.

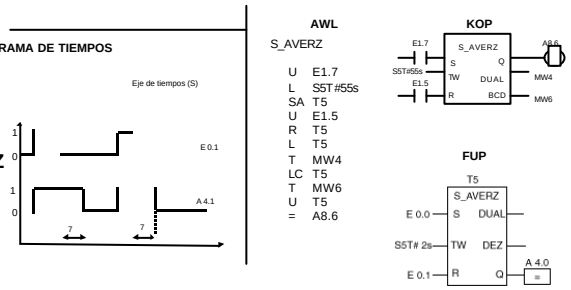




Tipos de Temporizadores

- **Retardo a la desconexión 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.

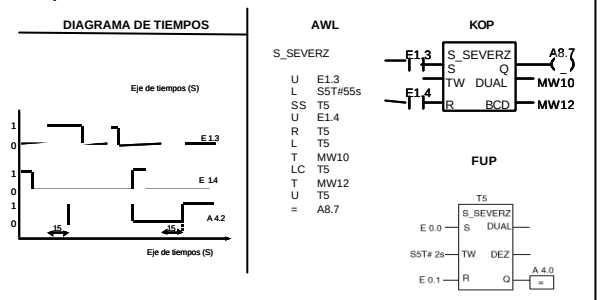
Temporizador de Retardo a la desconexión S_AVERZ



Tipos de Temporizadores

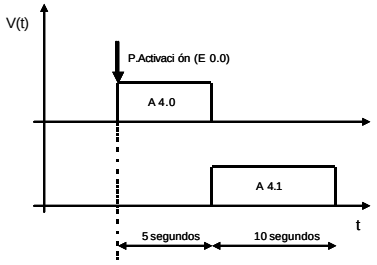
- **Retardo a la Conexión Memorizada SS**
  - Un temporizador SS es idéntico al SE, excepto en un aspecto: este temporizador se mantiene activo a no ser que se produzca la orden de reseteo.
  - Podemos decir, que se trata de una memoria retardada el tiempo prefijado en nuestra constante.
  - Si la entrada S cambia de 0 a 1, el temporizador arranca y conti núa 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.

Temporizador de Retardo a la conexión memorizada S_SEVERZ



Ejercicio

Diagrama de tiempos (1)



Se dispone de un Pulsador de Activación E 0.0.

Se desea automatizar un sistema que debe cumplir el diagrama de tiempos definido en la figura adjunta.

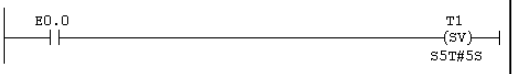
Diseñar el programa de automatización, teniendo en cuenta que durante el tiempo que dure el ciclo deberá evitarse cualquier rearme de tiempos.

El ejercicio se utilizará utilizando únicamente uno de 5 tipos de temporizadores.

Ejercicio: solución

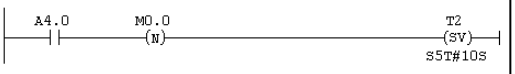
OB1 : Diagrama de tiempos (I)

Segm. 1: Activación T1

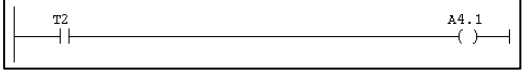


Segm. 2 : Transferencia a salida 1

Segm. 3 : Activación T2



Segm. 4 : Transferencia a salida 2



OB1 : Diagrama de tiempos (I)

Segm. 1: Activación T1

```
      U     E      0.0
      L     S5T#5S
      SV    T      1
```

Segm. 2 : Transferencia a salida 1

```
      U     T      1
      =     A      4.0
```

Segm. 3 : Activación T2

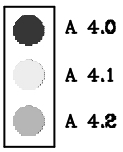
```
      U     A      4.0
      FN    M      0.0
      L     S5T#10S
      SV    T      2
```

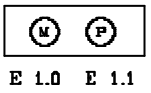
Segm. 4 : Transferencia a salida 2

```
      U     T      2
      =     A      4.1
```


Ejercicio

- Tenemos un semáforo con las tres luces verde, amarillo y rojo. Tenemos dos pulsadores de mando: un pulsador de marcha y un pulsador de paro.
- Con el pulsador de marcha quiero que comience el ciclo. El ciclo de funcionamiento es el siguiente:
- 1º/ Verde durante 5 seg.
- 2º/ Verde + Amarillo durante 2 seg.
- 3º/ Rojo durante 6 seg.
- El ciclo es repetitivo hasta que se pulse el pulsador





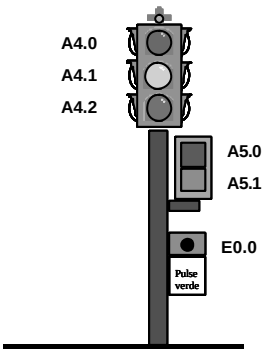
Ejercicio: solución

SOLUCIÓN EN AWL

```
U   E   0.0   //Al activar el pulsador de marcha
S   A   4.2   //Encender el verde
U   A   4.2   //Si se ha encendido el verde
L   S5T#5S    //Cuenta 5 segundos
SE  T   1     //Con el temporizador 1
U   T   1     //Y cuando acabes de contar
S   A   4.1   //Enciende el amarillo
U   A   4.1   //Si se ha encendido el amarillo
L   S5T#2S    //Cuenta 2 segundos
SE  T   2     //Con el temporizador 2
U   T   2     //Y cuando acabes de contar
S   A   4.0   //Enciende el rojo
R   A   4.1   //Apaga el amarillo
R   A   4.2   //Y apaga el verde
U   A   4.0   //Si se ha encendido el rojo
L   S5T#6S    //Cuenta 6 segundos
SE  T   3     //Con el temporizador 3
U   T   3     //Cuando acabes de contar
S   A   4.2   //Enciende el verde
R   A   4.0   //Y apaga el rojo
U   E   0.1   //Si se activa el pulsador de paro
R   A   4.0   //Apaga el rojo
R   A   4.1   //Apaga el amarillo
R   A   4.2   //Apaga el verde
```


Ejercicio

Control de un Semáforo



Se dispone de un semáforo, el cual en condiciones normales se encuentra del modo siguiente:

- Verde vehículos
- Rojo Peatones.

En el mismo instante que un peatón accione sobre el pulsador situado en el semáforo, éste pasará a amarillo para vehículos, estado que durará durante 3 ‘‘. Finalizado este, pasara a estado rojo para vehículos y verde para peatones.

El tiempo de duración fijado para rojo vehículos: 6 ‘‘.

Finalizado el proceso, el semáforo regresará al estado normal.

Durante el tiempo de duración del ciclo, deberá evitarse que cualquier nueva activación sobre el pulsador verde, rearme el ciclo.

Contadores

- A continuación se describen tres opciones de contador para el S7-300. Existe un área en la memoria de la CPU reservada para los contadores. En éste área hay una palabra (16 bits) reservada para cada contador. El máximo valor especificado es 999 (BCD).
- **Contador Ascendente Z_VORW**
  - Si la entrada S cambia de 0 a 1, el valor que hay en ZW se fija como valor del contador especificado. Comenzando con 0, el contador cuenta ascendentemente cada vez que la entrada ZV cambia de 0 a 1. La salida Q es siempre 1 si el valor binario del contador (DUAL) no es =0. Si la entrada de reset R cambia de 0 a 1, el contador es fijado a 0.
- **Contador Descendente Z_RUECK**
  - Si la entrada S cambia de 0 a 1, el valor en ZW se fija como valor de contador especificado. Cada vez que la entrada ZR pasa de 0 a 1, el contador se decrementa en una unidad. La salida Q es siempre 1 si el valor binario del contador (DUAL) no es =0. Si la entrada de reset R cambia de 0 a 1, el contador es fijado a 0.
- **Contador Ascendente/ Descendente ZAEHLER**
  - Este contador es una combinación de los dos anteriores.

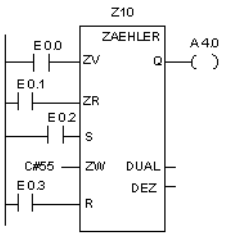
Contadores

```
U   E   124.0
ZV  Z   0          // CONTAR HACIA ADELANTE

U   E   124.1
ZR  Z   0          // CONTAR HACIA ATRAS

U   E   124.2
L   C#10
S   Z   0          // SETEAR EL CONTADOR A UN VALOR PRESELECCIONADO

U   E   124.3
R   Z   0          // RESETAR EL CONTADOR

L   Z   0
T   MW  0          // CARGAR EL VALOR ACTUAL EN DECIMAL

LC  Z   0
T   MW  2          // CARGAR EL VALOR ACTUAL EN BCD
```

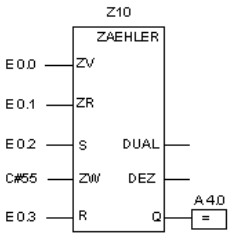
Contadores

Contadores

KOP



FUP



AWL

```
U   E0.4
ZV  Z5
U   E0.5
ZR  Z5
U   E0.3
L   C#20
S   Z5
U   E0.7
R   Z5
L   Z5
T   MW4
LC  Z5
T   MW7
U   Z5
=   A8.3
```


Contadores

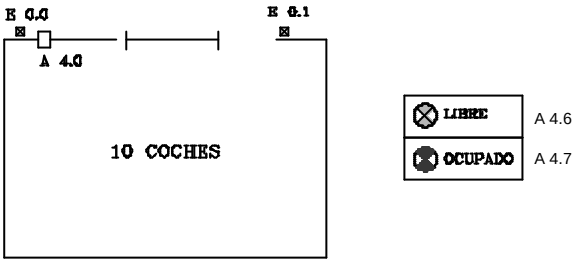
- Para meter los valores en los acumuladores, tenemos la instrucción de carga. (L).
  - Cuando cargamos un valor, siempre se carga en el acumulador 1. Cuando volvemos a cargar otro valor, también se guarda en acumulador 1. Lo que tenía en el acumulador 1 pasa al acumulador 2, y lo que tenía en el acumulador 2 lo pierde.
- En nuestro caso, cargaremos el valor de Z1 y a continuación cargaremos el valor con el que queremos comparar.
- Una vez tengamos los valores en el acumulador, tendremos que compararlos. Para ello tenemos las siguientes instrucciones:

>	>	>=	<=	==	<>
Mayor	Menor	Mayor o igual	Menor o igual	Igua	Dist.

- A continuación del símbolo de comparación pondremos una I si lo que estamos comparando son dos números enteros. Pondremos una R si lo que estamos comparando son números reales.

Ejercicio

- El funcionamiento que queremos es el siguiente:
  - Cuando llega un coche y el parking esté libre, queremos que se abra la barrera. A la salida no tenemos barrera. Cuando sale un coche simplemente sabemos que ha salido.
  - En el parking caben 10 coches. Cuando el parking tenga menos de 10 coches queremos que esté encendida la luz de libre. Cuando en el parking haya 10 coches queremos que esté encendida la luz de ocupado.
  - Además queremos que si el parking está ocupado y llega un coche que no se le abra la barrera.



Ejercicio: solución

SOLUCIÓN EN AWL

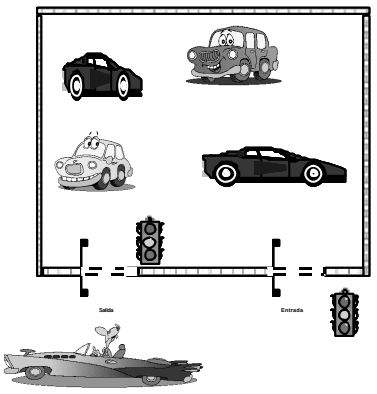
```
U   E     0.0     //Si llega un coche
U   A     4.6     //Y está libre
=   A     4.0     //Abre la barrera
U   A     4.0     //Si se he abierto la barrera
ZV  Z     1       //Cuenta uno con el contador 1
U   E     0.1     //Si sale un coche
ZR  Z     1       //Descuenta 1 con el contador 1
L   Z     1       //Carga el contador 1
L   10            //Carga un 10
<I                //Si en el contador hay menos de 10
S   A     4.6     //Enciende la luz de libre
R   A     4.7     //Y apaga la de ocupado
==I               //Si el contador de coches vale 10
R   A     4.6     //Apaga la luz de libre
S   A     4.7     //Y enciende la luz de ocupado
```

Ejercicio: solución

UTILIZANDO EL BIT DEL CONTADOR
SOLUCIÓN AWL

```
U  E    0.7      //Si activamos la entrada 0.7
L  C#10          //Carga un 10
S  Z    1        //Mete el 10 en el contador
U  E    0.0      //Si llega un coche
U  A    4.6      //Y está libre
=  A    4.0      //Abre la barrera
U  A    4.0      //Si se ha abierto la barrera
ZR Z    1        //Descuenta 1 en el contador 1.1 plaza libre
   menos
U  E    0.1      //Si sale un coche
ZV Z    1        //Cuenta 1 en el contador 1. 1 plaza libre mas.
UN Z    1        //Si en el contador 1 hay un 0
=  A    4.7      //Enciende la luz de ocupado
UN A    4.7      //Si no está ocupado
=  A    4.6      //Enciende la luz de libre
```


Ejercicio



Control de Acceso de garaje

Automatizar un garaje de 5 plazas de tal forma que si éste se encuentra lleno se encienda una luz indicándolo y no suba la barrera. En caso contrario deberá estar encendida otra luz indicando "LIBRE".

El garaje consta de 5 plazas

Disponemos de una célula fotoeléctrica y una barrera en la entrada y lo mismo en la salida.

Asignación de variables	
E0.0	Célula fotoeléctrica de entrada
E0.1	Célula fotoeléctrica de salida
A4.0	Barrera de entrada
A4.1	Barrera de salida
A4.2	Luz de señalización de "LIBRE"
A4.3	Luz de señalización de "LLENO"

Ejercicio: solución

```
OB1 : Control de Garaje
Segm. 1: Título:
      UN    M    130.0                  //Inicializar contador a 5
      L     C#5
      S     Z      1
      UN    M    130.0
      S     M    130.0

      U     Z      1                    //Si hay sitio y llega un coche,
      U     E      0.0                  //abrir barrera
      =     A      4.0

      U     A      4.0                  //Al subir la barrera
      ZR    Z      1                    //un sitio menos

      U     E      0.1                  //Si hay un coche a la salida
      =     A      4.1                  //abrir la barrera

      U     A      4.1                  //Al subir la barrera
      ZV    Z      1                    //un sitio más

      UN    Z      1                    //Encender la luz "LIBRE"
      =     A      4.3

      U     Z      1                    //Encender la luz "LLENO"
      =     A      4.2
```