Ejemplos de programación para PLC S5 100

(29 Problemas Resueltos)

INDICE

Ejemplo 1
Combinación AND

Realizar A2.2 = E0.0 AND E0.1

La salida A2.2 debe activarse tan sólo si los dos interruptores conectados a las entradas E0.0 y E0.1 están cerrados.
La solución ladder se obtiene poniendo en serie dos contactos, con operandos E0.0 y E0.1, y la bobina A2.2. De hecho, la combinación lógica AND, traducida al lenguaje ladder, equivale a la serie de dos contactos: en la disposición en serie 'se lee' el cierre del circuito sólo cuando ambos contactos están cerrados; de manera que esta es la única condición que activa la bobina.

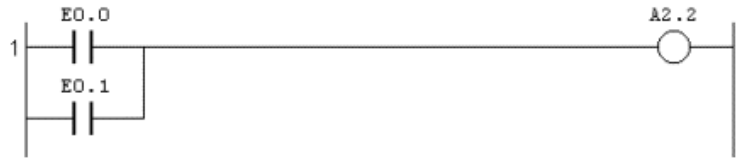
LADDER	AWL
	: U E0.0 : U E0.1 : = A2.2 : BE

La solución en AWL se obtiene cargando primero el estado de E0.0 en el registro RLC (U E0.0). De hecho, la operación puramente dicha es una AND pero aquí, al ser la primera de una secuencia, se interpreta como una operación de carga de bit y por lo tanto el estado del operando se copia en RLC. A continuación, se efectúa una AND entre este último y el estado de la entrada E0.1 (U E0.1) y el resultado se deposita de nuevo en RLC. Esta instrucción no es la primera de una secuencia y por lo tanto, la operación se interpreta realmente como una AND. La última instrucción (= A2.2) se encarga de transferir el contenido de RLC, que en ese momento representa la combinación lógica E0.0 AND E0.1, a la salida A2.2.

Ejemplo 2
Combinación OR

Realizar A2.2 = E0.0 OR E0.1

La salida A2.2 debe activarse si al menos uno de los interruptores conectados a las entradas E0.0 o E0.1 está cerrado.
La solución ladder se obtiene poniendo en serie con la bobina A2.2 el paralelo de dos contactos, con operandos E0.0 y E0.1. De hecho, la combinación lógica OR, traducida a un esquema de contactos, equivale al paralelo de dos contactos: a las cabezas del paralelo 'se lee' el cierre del circuito cuando al menos uno de los contactos está cerrado. Esta es pues la condición que conduce a la activación de la bobina.

LADDER	AWL
	: O E0.0 : O E0.1 : = A2.2 : BE

La solución en AWL se obtiene cargando primero el estado de E0.0 en el registro RLC (O E0.0). De hecho, la operación puramente dicha es una OR pero aquí, al ser la primera de una secuencia, se interpreta como una operación de carga de bit y por tanto, el estado del operando se copia en RLC. En este caso la instrucción equivale perfectamente a la U E0.0, que puede ser sustituida por ella, obteniéndose un programa con idéntico funcionamiento. A continuación se efectúa una OR entre el RLC y la entrada E0.1 (O E0.1), que deposita de nuevo el resultado en el RLC. Esta instrucción no es la primera de una secuencia y por tanto, la operación se interpreta en realidad como una OR. La última instrucción (= A2.2) se encarga de transferir el contenido de RLC, que en ese momento punto representa la combinación lógica E0.0 OR E0.1, a la salida A2.2.

Ejemplo 3
Combinación AND de OR

Realizar A2.0 = (E0.0 OR E0.1) AND (E0.2 OR E0.3)

Después de haber realizado los ejercicios anteriores, la solución ladder debería de ser intuitiva: se disponen en serie (AND) dos paralelos (OR) de contactos, conectando adecuadamente los operandos en correspondencia con estos y con la bobina.

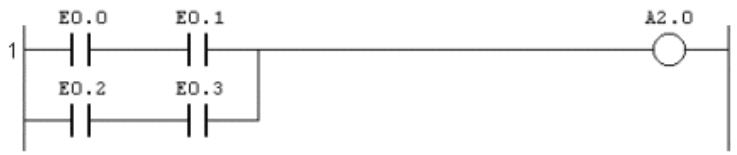
LADDER	AWL
	: O E0.0 : O E0.1 : U(: O E0.2 : O E0.3 :) : = A2.0 : BE

Por el contrario, la solución AWL requiere alguna clarificación ya que se han introducido dos nuevas operaciones. En primer lugar, observamos que, después de la ejecución de la segunda instrucción, RLC contiene el resultado de la combinación lógica OR entre E0.0 y E0.1 (ver Ejemplo 2). La siguiente operación es una apertura de paréntesis; el RLC actual se deja de lado por el momento para ser combinado sucesivamente en AND con el resultado de la expresión del interior del paréntesis. La operación U(es delimitadora de RLC y por tanto la siguiente instrucción será la primera de una nueva secuencia. Así, la cuarta instrucción (O E0.2) será interpretada como una carga en RLC del estado de E0.2 y, después de la ejecución de la siguiente instrucción, RLC contendrá el resultado de la combinación lógica E0.2 OR E0.3. La siguiente instrucción de cierre de paréntesis hará ejecutar al PLC la combinación AND (la tercera era U() entre el RLC actual, es decir, el resultado de la OR entre paréntesis, con el RLC que antes se había dejado a un lado, o sea, el resultado de la primera OR. La última instrucción (= A2.0) se encarga de transferir el contenido del RLC, que en ese momento representa la combinación lógica (E0.0 OR E0.1) AND (E0.2 OR E0.3), a la salida A2.0.

Ejemplo 4
Combinación OR de AND

Realizar A2.0 = (E0.0 AND E0.1) OR (E0.2 AND E0.3).). Donde los paréntesis, si bien no son necesarios dado que la operación AND tiene preferencia sobre la OR, se han añadido para mayor claridad.

La solución ladder pone en paralelo (OR) dos serie (AND) de contactos, conduciendo oportunamente los operandos en correspondencia con estos y con la bobina.

LADDER	AWL
	: U E0.0 : U E0.1 : O(: U E0.2 : U E0.3 :) : = A2.0 : BE

Por lo que se refiere a la solución AWL, observamos en primer lugar que, después de la ejecución de la segunda instrucción, RLC contiene el resultado de la combinación lógica AND entre E0.0 y E0.1 (ver Ejemplo 1). La siguiente operación es una apertura de paréntesis; el RLC actual se deja a un lado por el momento para combinarse sucesivamente en AND con el resultado de la expresión del interior del paréntesis. La operación O(es delimitadora del RLC y por tanto la siguiente instrucción será la primera de una nueva secuencia. Así, la cuarta instrucción (U E0.2) se interpretará como una carga en RLC del estado de E0.2 y, después de la ejecución de la siguiente instrucción, RLC contendrá el resultado de la combinación lógica E0.2 AND E0.3. La siguiente instrucción de cierre de paréntesis hará ejecutar al PLC la combinación OR (la tercera era O() entre el RLC actual, es decir, el resultado de la AND entre paréntesis, con el RLC que antes se había dejado de lado, o sea, el resultado de la primera AND. La última instrucción (= A2.0) se encarga de transferir el contenido del RLC, que en ese momento representa la combinación lógica (E0.0 AND E0.1) OR (E0.2 AND E0.3), a la salida A2.0.

Ejemplo 5
Combinación XOR

Realizar A2.2 = E0.1 XOR E0.2

La operación lógica XOR aplicada a dos variables booleanas da un resultado cierto cuando una y sólo una de las dos variables es cierta.
La primera serie de contactos del programa ladder está cerrada únicamente cuando E0.1 está cerrado y E0.2 está abierto. La segunda serie, por el contrario, está cerrada tan sólo cuando E0.1 está abierto y E0.2 está cerrado. Realizando el paralelo de las dos se obtiene la función deseada. Es decir, la bobina se activa tan sólo cuando una entrada está cerrada y la otra está abierta.

LADDER	AWL
	: U E0.1 : UN E0.2 : O(: UN E0.1 : U E0.2 :) : = A2.2 : BE

La solución AWL no es más que la traducción del programa ladder y su funcionamiento es muy similar al ejemplo anterior salvo en que en correspondencia con los contactos NC, se programan operaciones UN, un test sobre el estado negado del operando.

Ejemplo 6
Autorretención

Un pulsador conectado a la entrada I0.0 debe activar la salida Q0.15 y un segundo pulsador conectado a la entrada I0.1 debe desactivarla.

En el programa ladder propuesto como solución se realiza un circuito con autorretención. Accionando el pulsador conectado a E0.0 la bobina A3.7 se activa y entonces, el contacto con el mismo operando de la segunda línea se cierra (imaginen que el contacto y la bobina son parte de un mismo relé A3.7) y continua manteniendo activada la bobina incluso después de la apertura de E0.0. El cierre del pulsador en la entrada E0.1 provoca la apertura del contacto, normalmente cerrado en el esquema, desactivando la bobina y cortando la autorretención.

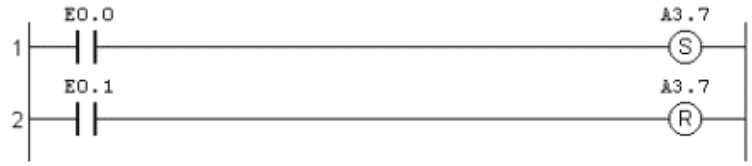
LADDER	AWL
	: U E0.0 : O A3.7 : UN E0.1 : = A3.7 : BE

El programa AWL propone la conversión de lo anteriormente descrito. El valor de la salida A3.7 en la última instrucción se calcula cargando el estado de E0.0, valorando luego la OR con A3.7 y, poniendo a continuación en AND el resultado con el complemento de E0.1.

Ejemplo 7
Set y reset

Un pulsador conectado a la entrada E0.0 debe activar la salida A3.7; un segundo pulsador conectado a la entrada E0.1 debe desactivarla

El ejercicio es idéntico al anterior pero, en esta ocasión, en la solución se usan bobinas de set y reset.

LADDER	AWL
	: U E0.0 : S A3.7 : U E0.1 : R A3.7 : BE

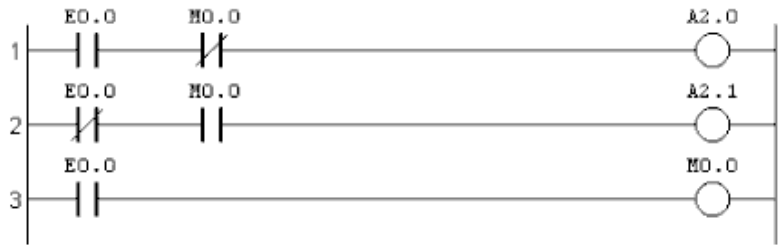
Haciendo funcionar el programa, observamos que si presionamos simultáneamente sobre dos pulsadores, se produce la puesta a cero de la salida. Efectivamente, en ambos lenguajes, y siendo válidas las condiciones de test, el operando A3.7 se activa primero en el recorrido 1 o con las dos primeras instrucciones y luego se desactiva en el recorrido 2 o con la tercera y cuarta instrucción. Pero recordemos que A3.7 no representa efectivamente la salida física del PLC, sino el correspondiente bit en el interior de la memoria de la imagen de proceso. Dicho bit de memoria es llevado efectivamente a 1 y luego a 0 pero, tan solo al final de la elaboración del programa utilizado, el valor cargado para ello se transfiere al canal físico de la salida correspondiente, que por consiguiente se mantiene constantemente en el valor bajo cuando ambas entradas están cerradas.
Así pues, con esta escritura de programas hemos hecho prevalecer el reset (desactivación) respecto del set (activación). Si desea obtener lo contrario, le bastará con invertir la posición de los recorridos en el esquema de contactos o bien el primer grupo de dos instrucciones con el segundo en el programa AWL.

Ejemplo 8
Activación por flanco

Activar las salidas A2.0 y A2.1 respectivamente con los flancos ascendente y descendente de la entrada E0.0.

Observamos que el último recorrido del esquema ladder y las dos últimas instrucciones del programa AWL imponen, al final de la ejecución del programa, la igualdad del merker bit M0.0 al estado de la entrada E0.0. Sin embargo, en correspondencia con los flancos y para los recorridos o las instrucciones anteriores, se da el hecho que el estado de las dos variables es opuesto y que únicamente al final de la carga del programa utilizado son iguales. Todo ello queda representado en las dos primeras líneas del diagrama con un retraso temporal de M0.0 respecto a E0.0, que vale un ciclo de ejecución.
La parte inicial del programa activa la bobina A2.0, para un ciclo de ejecución, cuando E0.0 está a 1 y M0.0 está a 0, es decir, en correspondencia con el flanco ascendente de E0.0, tal como aparece indicado en la tercera línea del diagrama. En cambio, la bobina A2.1 se activará para un ciclo de ejecución, cuando E0.0 esté a 0 y M0.0 esté a 1, es decir, en correspondencia con el flanco descendente de E0.0, tal como aparece indicado en la cuarta línea del diagrama.

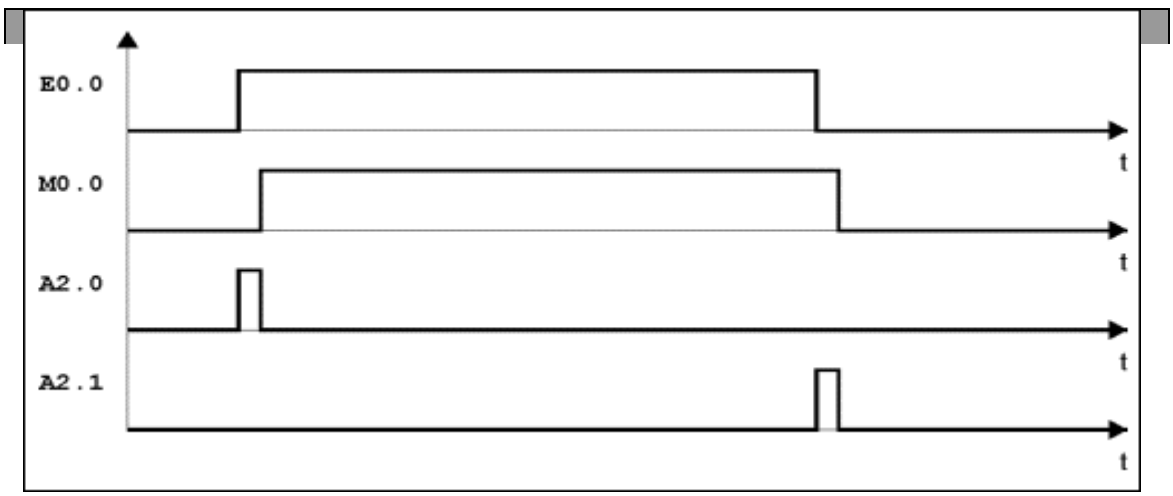
LADDER



AWL

```
: U   E0.0                  'elevación flanco
: UN  M0.0                  'ascendente
: =   A2.0

: UN  E0.0                  'elevación flanco
: U   M0.0                  'descendente
: =   A2.1

: U   E0.0                  'memoria para la elevación
: =   M0.0                  'de flancos

: BE
```

Diagrama temporal de un generador de flancos



Ejemplo 9
Temporizador a impulso

La salida A2.5 se activa al cierre de la entrada E0.0 y se desactiva 5 segundos después. Si la entrada vuelve a abrirse durante ese periodo, la salida se desactiva inmediatamente.

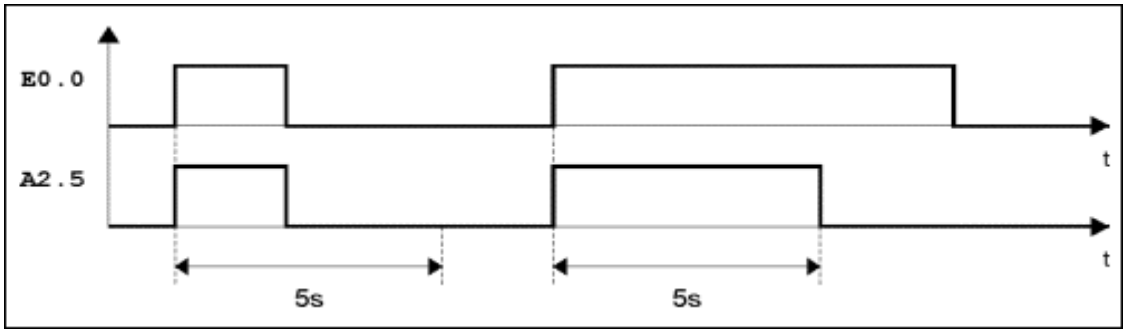
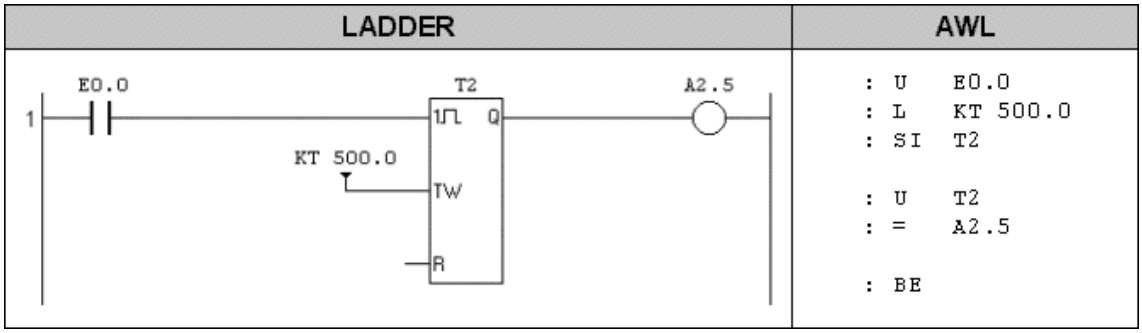


Diagrama temporal del temporizador a impulso

El funcionamiento del temporizador está ilustrado en el diagrama temporal situado aquí arriba. La primera línea representa la señal de entrada, la segunda su salida.
El programa ladder se ha realizado utilizando un temporizador SI (impulso) activado por el contacto NA de E0.0, con una constante de tiempo igual a 500 centésimas de segundo, y con la salida conectada a la bobina de A2.5.



El primer grupo de instrucciones del programa AWL conduce a la activación del temporizador T2 como impulso (SI T2) a continuación de un flanco ascendente de la entrada (U E0.0), con la constante de tiempo de 5 segundos cargada en el acumulador 1 (L KT500.0). El segundo grupo de instrucciones copia el estado del temporizador (U T2) en la salida del PLC (= A2.5).

Ejemplo 10
Temporizador a impulso prolongado

La salida A2.5 se activa al cierre de la entrada E0.0 y se desactiva 5 segundos después, independientemente de si la entrada se reabre o no durante dicho periodo.

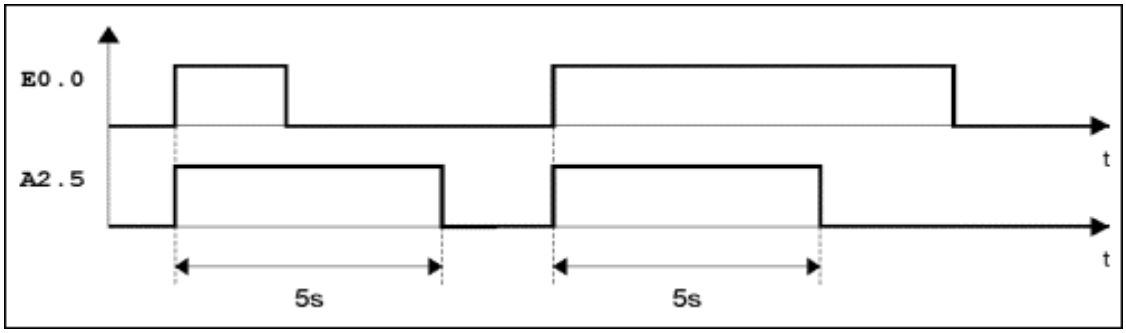
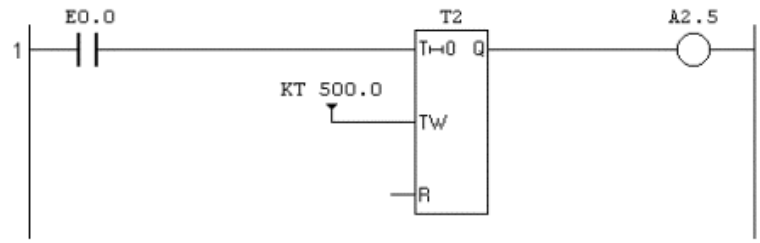


Diagrama temporal del temporizador a impulso prolongado

El funcionamiento del temporizador queda ilustrado en el diagrama temporal que aparece aquí arriba. La primera línea representa la señal de entrada y la , segunda, su salida
El programa ladder se ha realizado utilizando un temporizador SV (impulso prolongado) activado por el contacto NA de E0.0, con una constante de tiempo igual a 500 centésimas de segundo, y con la salida conectada a la bobina de A2.5.

LADDER	AWL
	: U E0.0 : L KT 500.0 : SV T2 : U T2 : = A2.5 : BE

El primer grupo de instrucciones del programa AWL produce la activación del temporizador T2 como impulso prolongado (SV T2) a continuación de un flanco ascendente de la entrada (U E0.0), con la constante de tiempo de 5 segundos cargada en el acumulador 1 (L KT500.0). El segundo grupo de instrucciones copia el estado del temporizador (U T2) en la salida del PLC (= A2.5).

Ejemplo 11
Temporizador con retardo a la activación

La salida A2.5 se activa 5 segundos después del cierre de la entrada E0.0. Cuando la entrada se reabre, la salida se desactiva.

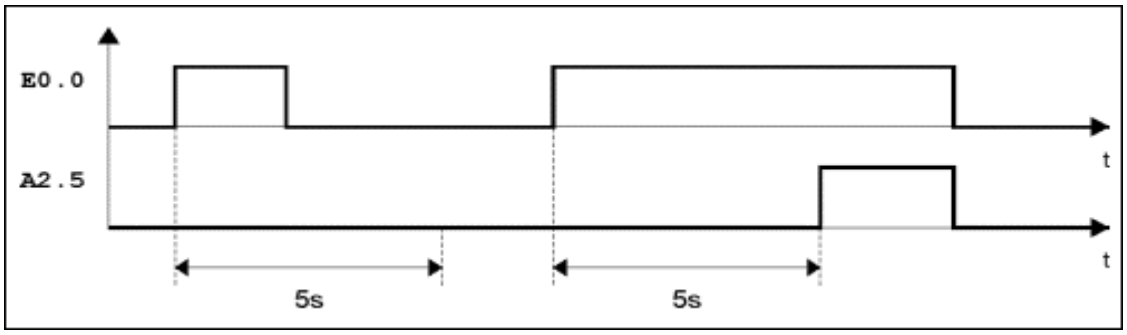
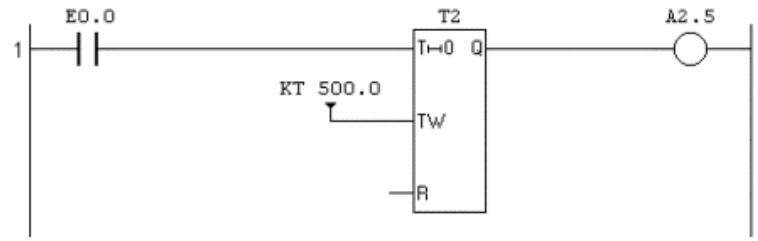


Diagrama del temporizador con retardo a la activación

El funcionamiento del temporizador queda ilustrado en el anterior diagrama temporal. La primera línea representa la señal de entrada y la segunda, la salida que se obtiene.
El programa ladder se ha realizado utilizando un temporizador SE (retardo a la activación) activado por el contacto NA de E0.0, con una constante de tiempo igual a 500 centésimas de segundo y con la salida conectada a la bobina de A2.5.

LADDER	AWL
	: U E0.0 : L KT 500.0 : SE T2 : U T2 : = A2.5 : BE

El primer grupo de instrucciones del programa AWL conduce a la activación del temporizador T2 como retardo a la activación (SE T2) a continuación de un flanco ascendente de la entrada (U E0.0), con la constante de tiempo de 5 segundos cargada en el acumulador 1 (L KT500.0). El segundo grupo de instrucciones copia el estado del temporizador (U T2) en la salida del PLC (= A2.5).

Ejemplo 12
Temporizador con retardo a la activación con memoria y reset

La salida A2.5 se activa 5 segundos después del cierre de la entrada E0.0 (aunque este último se vuelva a abrir durante ese período) y se desactiva en correspondencia con el cierre de la entrada E0.1.

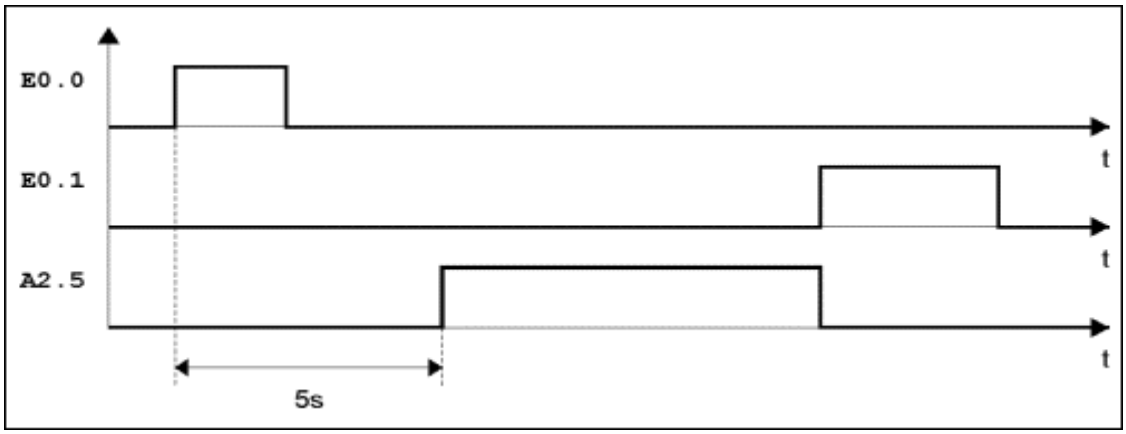
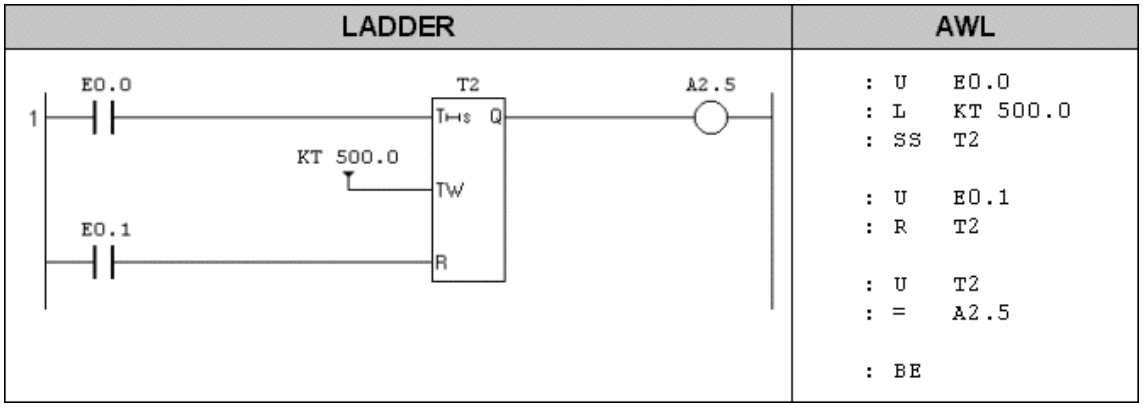


Diagrama de un temporizador con retardo a la activación con memoria y reset

El funcionamiento del temporizador con retardo a la activación con memoria y reset se obtiene comparando las dos primeras líneas y la última del diagrama temporal precedente. Las dos primeras líneas representan las señales de entrada y la última, la correspondiente salida.
El programa ladder se ha realizado utilizando un temporizador SS (retardo a la activación con memoria) activado por el contacto NA de E0.0, con una constante de tiempo igual a 500 centésimas de segundo, el reset conectado a un contacto NA de E0.1 y la salida conectada a la bobina de A2.5.



El primer grupo de instrucciones del programa AWL conduce a la activación del temporizador T2 como retardo a la activación con memoria (SS T2) a continuación de un flanco ascendente en la entrada E0.0 (U E0.0), con una constante de tiempo de 5 segundos cargada en el acumulador 1 (L KT500.0). El segundo grupo de instrucciones se ocupa del reset del temporizador (R T2) en correspondencia con el estado alto de E0.1 (U E0.1). El último grupo copia el estado del temporizador (U T2) en la salida del PLC (= A2.5).

Ejemplo 13
Temporizador con retardo a la desactivación

La salida A2.5 debe activarse al cierre de la entrada E0.0 y desactivarse 5 segundos después de su reapertura.

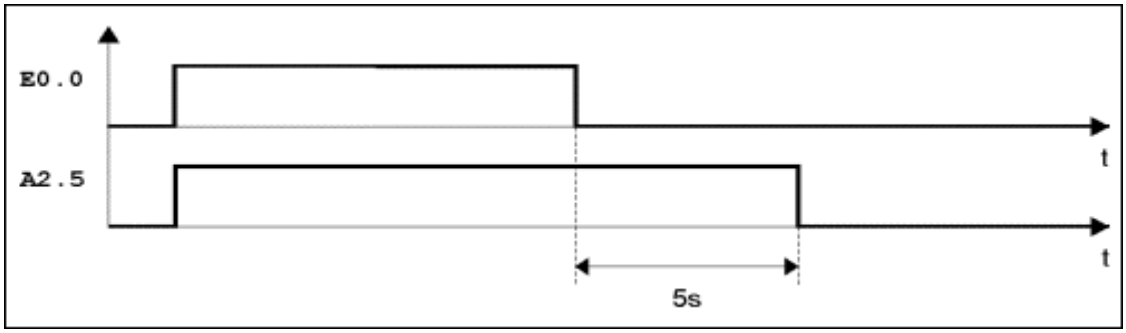
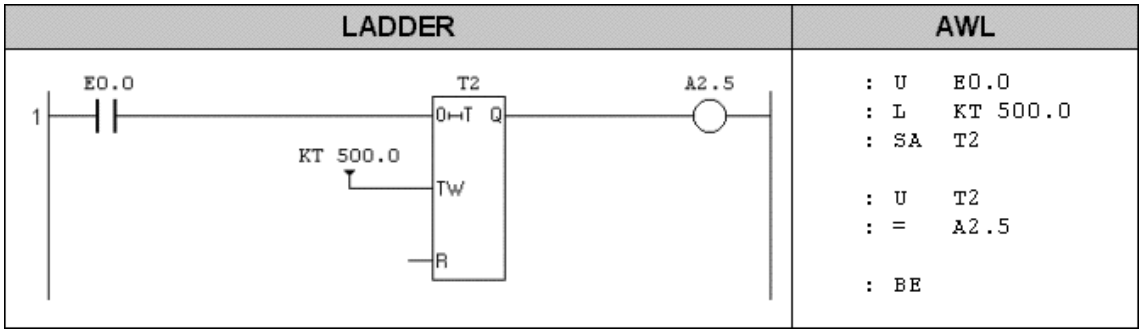


Diagrama del temporizador con retardo a la desactivación

El funcionamiento del temporizador queda ilustrado en el diagrama temporal. La primera línea representa la señal de entrada y la segunda, la correspondiente salida.
El programa ladder se ha realizado utilizando un temporizador SA (retardo a la desactivación) activado por el contacto NA de E0.0, con una constante de tiempo igual a 500 centésimas de segundo y con la salida conectada a la bobina de A2.5.



El primer grupo de instrucciones del programa AWL conduce a la activación del temporizador T2 como retardo a la desactivación (SA T2), a continuación de un flanco descendente de la entrada (U E0.0), con la constante de tiempo de 5 segundos cargada en el acumulador 1 (L KT500.0). El segundo grupo de instrucciones copia el estado del temporizador (U T2) en la salida del PLC (= A2.5).

Ejemplo 14
Temporizador con retardo a la activación y a la desactivación

La salida A3.3 se activa 3 segundos después del cierre de la entrada E0.1 y se desactiva 7 segundos después de su reapertura.

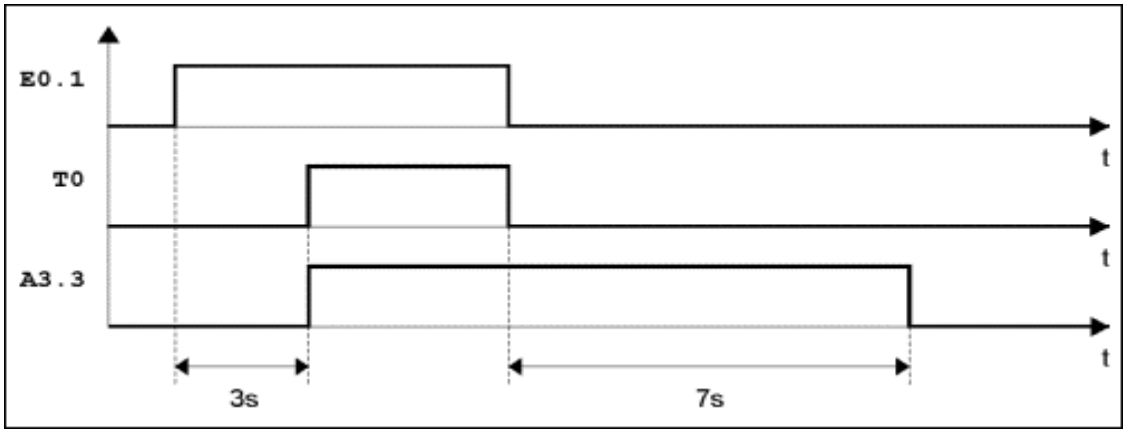
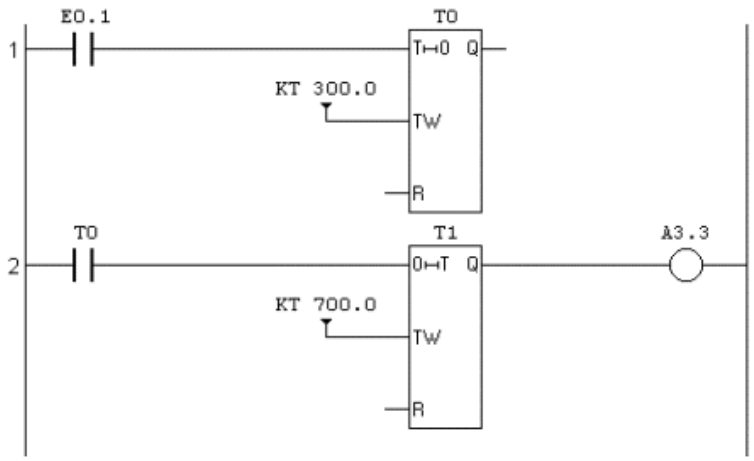


Diagrama temporizador con retardo a la activación y a la desactivación

La figura precedente ilustra, en la primera y la última línea del diagrama, el desarrollo de las dos señales descritas en el trazado. En la línea intermedia se ha diseñando el desarrollo de un temporizador con retardo a la activación cuya entrada es precisamente E0.1. Observemos que el desarrollo de la salida A3.3, respecto al desarrollo de T0, representa un retardo a la desactivación. ¡Hagan juego, señores!. Se trata pues de escribir un programa que contenga dos temporizadores: el primero, un retardo a la activación de 3 seg., tiene como entrada E0.1; el segundo, un retardo a la desactivación de 7 seg., tiene como entrada el estado del primer temporizador y como salida A3.3.
Los programas ladder y AWL tratados representan precisamente esto.

LADDER



AWL

```
: U   E0.1
: L   KT 300.0
: SE  T0
: U   T0
: L   KT 700.0
: SA  T1

: U   T1
: =   A3.3

: BE
```

Esempio Ejemplo 15
Impulso retardado

La salida A2.7 se activa 2 segundos después de la apertura de la entrada E1.5 por un período de 1 segundo.

El diagrama siguiente ilustra en la primera línea el desarrollo de la entrada y en la última, el de la salida que se desea obtener. La segunda y la tercera línea representan el desarrollo de dos temporizadores con retardo a la desactivación T10 y T11, de 2 y 3 segundos respectivamente, que tienen como señal de entrada, precisamente E1.5.
Observamos entonces que la salida debe ser cierta cuando se dan simultáneamente las condiciones: T11 cierto y T10 falso. Es decir, en términos de expresión booleana: A2.7 = T11 · NOT(T10)

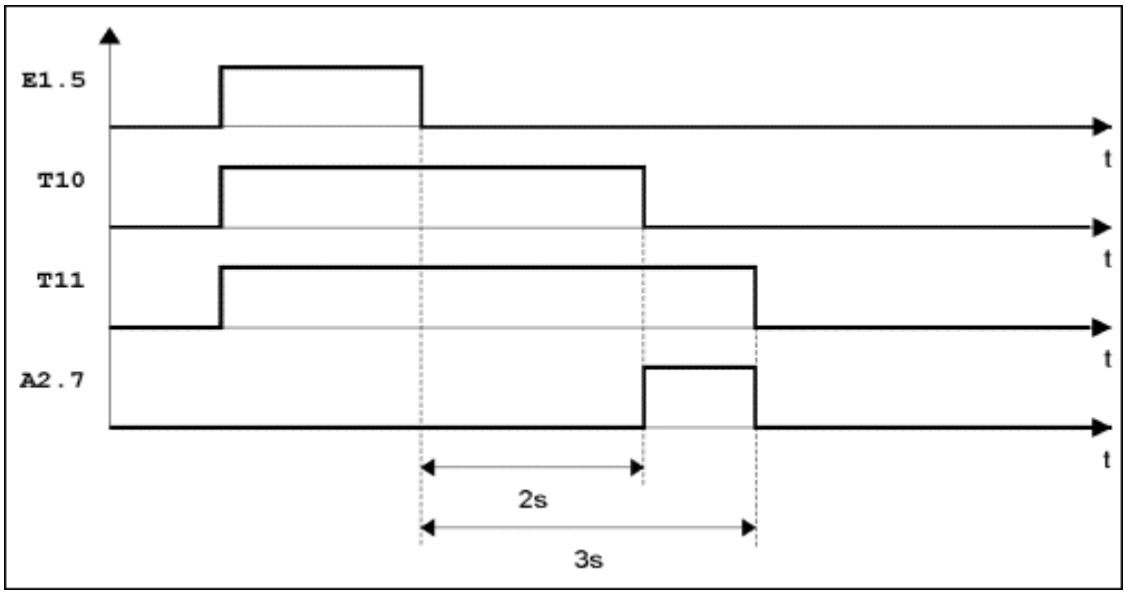
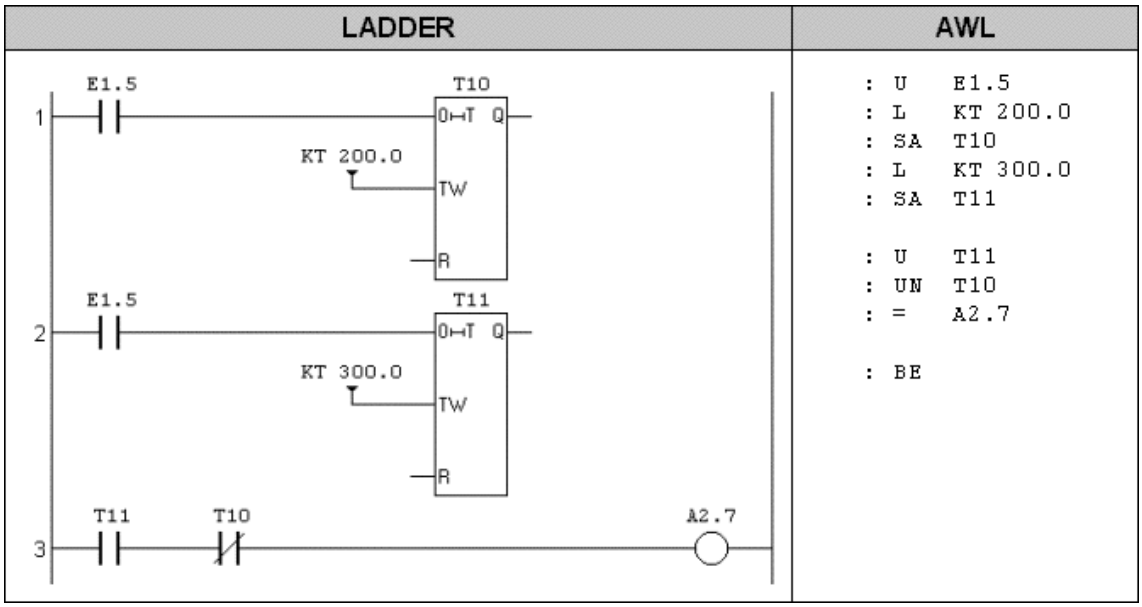


Diagrama de tiempos del impulso retardado



En el programa ladder los dos primeros recorridos están destinados a la activación de los dos temporizadores. Ambos tiene por entrada la señal E1.5. El tercer recorrido implementa la expresión lógica

recién obtenida. La serie de los dos contactos representa la AND y la utilización del tipo NC para el segundo equivale a la negación del su operando.
Análogamente, para el programa AWL, el primer grupo de instrucciones activa los dos temporizadores sobre la señal de entrada. El segundo grupo calcula el valor de la expresión booleana y lo asigna a la salida.

Ejemplo 16
Tren de impulsos

La salida A2.4 debe activarse un instante a cada segundo.

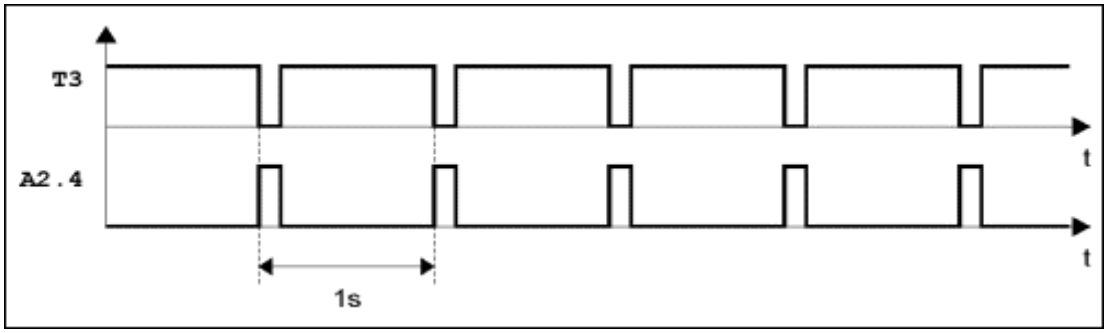
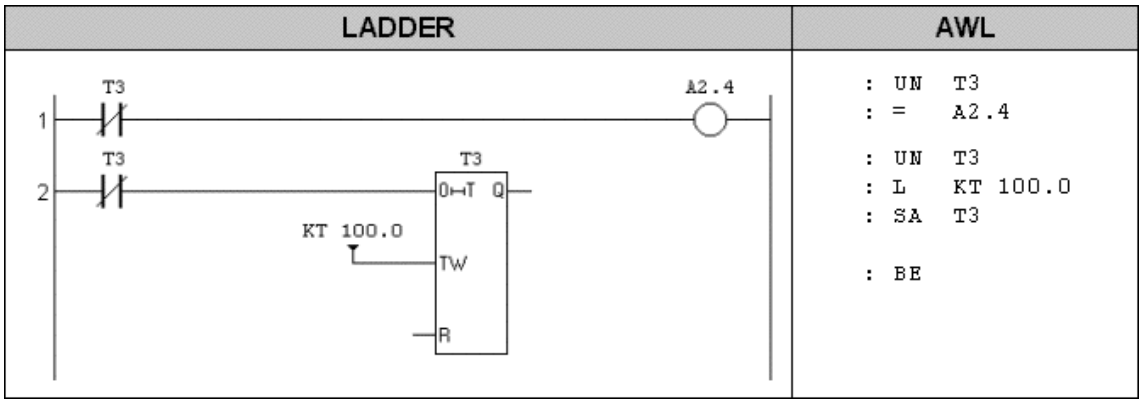


Diagrama de tiempos del tren de impulsos

El ejercicio se resuelve utilizando un temporizador que se autoarranca cíclicamente. Concentrémonos primero en el recorrido 2 del ladder y en el segundo grupo de instrucciones del AWL. La primera ejecución del programa encuentra el temporizador parado; por tanto, su contacto NC en el esquema ladder está cerrado y la interrogación sobre el estado bajo en el programa AWL da resultado cierto. En ambos casos la salida del temporizador con retardo a la desactivación se lleva al estado alto. A causa de ello, al ciclo siguiente, las interrogaciones antes descritas ya no se verificarán. Así pues, la entrada del temporizador se ha llevado a cero y el tiempo empieza a avanzar. Durante todo este período, la salida permanece alta. Finalizado el tiempo, ésta se pone a cero. A continuación, las condiciones de interrogación sobre la entrada del temporizador vuelven a verificarse y todo se repite tal como se ha descrito. El diagrama precedente describe, en la primera línea, el desarrollo de la salida del temporizador que resulta ser la señal opuesta a la requerida por el trazado. Ya tan solo queda invertir esta señal y asignarla a la salida A2.4. El primer recorrido del esquema de contactos y el primer grupo de instrucciones del programa AWL se encarga de esta tarea.



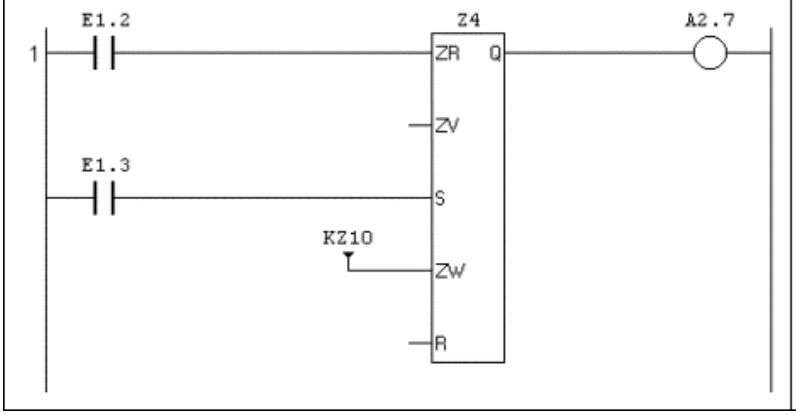
Queda por aclarar que el recorrido de asignación de A2.4 debe preceder necesariamente al de la activación del temporizador. De hecho, si no fuera así, el tiempo se reiniciaría antes que A2.4 pueda ser programado a 1. A2.4 se mantendría constantemente en estado bajo ya que el estado de T3, visto en este punto de la elaboración del programa, aparecería siempre alto. Todo lo anterior es válido también para el programa AWL, en el que el grupo de instrucciones de asignación de A2.4 debe preceder, por las mismas razones, al del arranque de T3. Estas consideraciones deberán recordarse cada vez que, en los ejemplos siguientes, se utilice un tren de impulsos.

Ejemplo 17
Conteo hacia atrás

El contador Z4 está programado a un valor 10 en correspondencia con el cierre de la entrada E1.3 y va decreciendo a cada cierre de la entrada E1.2. La salida A2.7 se desactiva al final del conteo (valor 0).

El programa ladder es muy sencillo. La constante KZ10, asignada a la entrada ZW, facilitará el valor de preset 10 al contador. Su entrada de set (S) se ha conectado a un contacto de E1.3, mientras que la de decremento (ZR) a un contacto de E1.2. En el flanco de cierre de E1.3 se produce la programación del valor de conteo a 10. A cada flanco de cierre de E1.2 el conteo decrece en 1. La salida del contador (Q) es de potencial alto cuando el valor de conteo es distinto de 0. Por tanto, será suficiente pilotar con ella la salida A2.7 del PLC.

LADDER



AWL

```
: U   E1.2
: ZR  Z4

: U   E1.3
: L   KZ 10
: S   Z4

: U  Z4
: =   A2.7

: BE
```

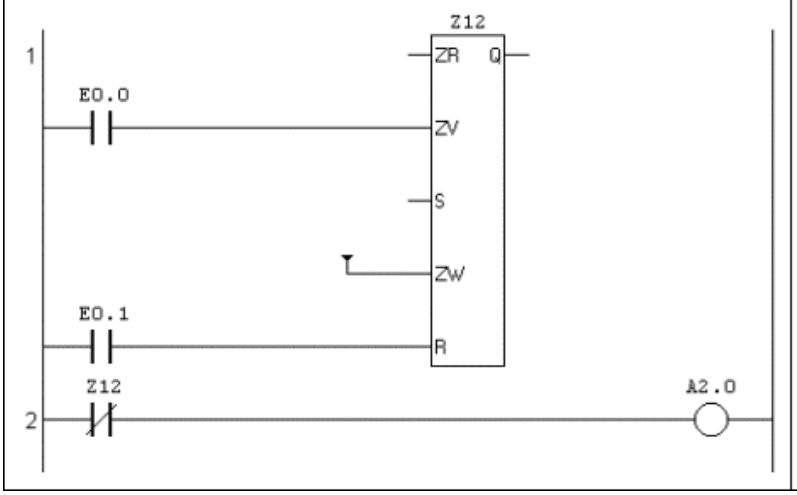
El primer grupo de instrucciones del programa AWL se encarga de incrementar en 1 el valor de conteo del contador Z4 a cada flanco ascendente de E1.2. El segundo efectúa la programación del conteo al valor cargado en ACCU1 (10) en correspondencia con el flanco ascendente de E1.3. El último grupo se encarga de transferir el estado del contador a la salida A2.7.

Ejemplo 18
Conteo hacia delante

El contador Z12 se incrementa a cada cierre de la entrada E0.0 y se pone a cero correspondiendo con el cierre de la entrada E0.1. La salida A2.0 es activa cuando el valor de conteo es cero.

La entrada de reset (R) del contador Z12 está conectada a un contacto de E0.1, mientras que la de incremento (ZV) lo está a un contacto de E0.0. En el flanco de cierre de E0.1 se produce el reset del contador, es decir, la programación del valor de conteo a 0. A cada flanco de cierre de E0.0, el conteo se incrementa en 1. La salida del contador (Q) es de potencial alto cuando el valor de conteo es distinto de 0. Bastará pues con invertirla para pilotar la salida A2.0 del PLC, tal como se ha hecho en el recorrido 2.

LADDER



AWL

```
: U   E0.0
: ZV  Z12

: U   E0.1
: R   Z12

: UN  Z12
: =   A2.0

: BE
```

Ejemplo 19
Conteo del tiempo de cierre de una entrada (en segundos)

Determinar durante cuantos segundos se mantiene cerrada la entrada E0.5 y utilizar la entrada E0.6 para poner a cero el conteo del tiempo.

En primer lugar, es preciso realizar una base de tiempo de un segundo, es decir, un tren de impulsos que tenga este período. A continuación, será preciso contar cuantos impulsos de la base de tiempos se generan durante el cierre de la entrada, es decir, los instantes en los cuales la entrada y el impulso son ciertos a la vez.

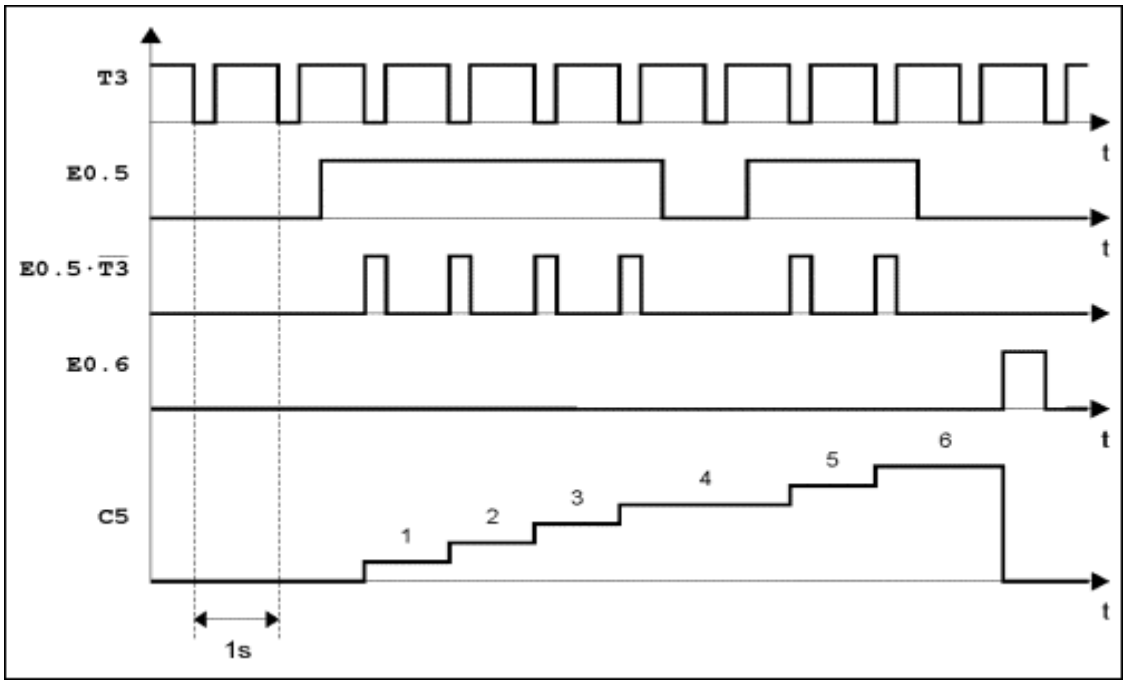
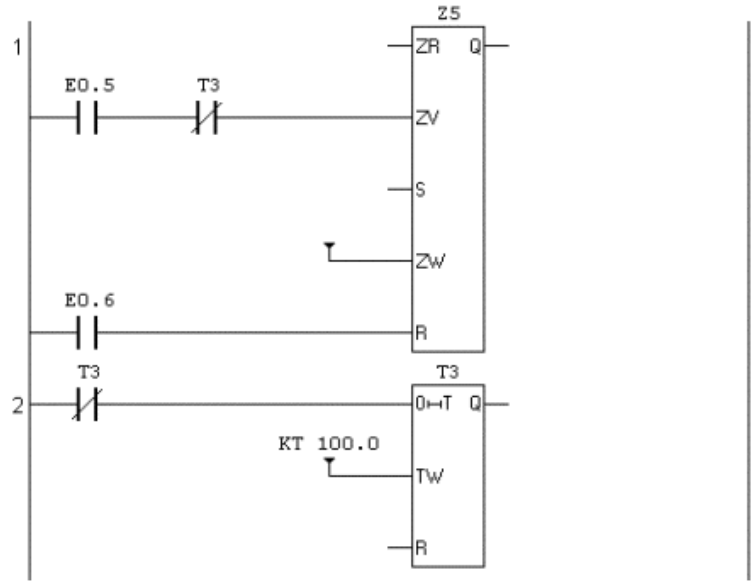


Diagrama del contador del tiempo de cierre de una entrada

El recorrido 2 del esquema de contactos implementa el tren de impulsos en 1 segundo de período, tal como hemos visto en el ejemplo 16. En el recorrido 1 se observa que la entrada ZV del contador está pilotada por la combinación lógica AND (serie de contactos en el diagrama) entre la entrada E0.5 y el tren de impulsos, es decir, precisamente los impulsos que hay que contar. Por su parte, un contacto de E0.6 pilota la entrada reset del contador para la puesta a cero del conteo, tal como se ha requerido.

LADDER	AWL
	: U E0.5 : UN T3 : ZV Z5 : UN T3 : L KT 100.0 : SA T3 : U E0.6 : R Z5 : BE

En el programa AWL, el primer grupo de instrucciones se encarga del incremento del contador en correspondencia con el flanco ascendente de la AND entre E0.5 y el tren de impulsos generado por T3 en el segundo grupo de instrucciones. El último grupo realiza el reset del conteo sobre el flanco ascendente de la entrada E0.6 (cierre del contacto respectivo).
El valor de conteo de Z5 representa el número de segundos durante los cuales la entrada se ha mantenido cerrada, con el límite propio de los contadores de este PLC, de 999.

Ejemplo 20
Conteo del tiempo de cierre de una entrada (en horas, minutos y segundos)

Determinar cuantas horas, minutos y segundos la entrada E0.5 permanece cerrada y utilizar la entrada E0.6 para reponer el conteo del tiempo.

Para programar la solución a este problema se ha hecho uso de símbolos. Su correspondencia con los operandos absolutos se ha establecido según la tabla siguiente.

Op. absoluto	Simbolo	Comentario
E0.5	ENTRADA	Entrada para conteo de tiempo
E0.6	RESET	Entrada para reset conteo
T3	CLOCK	Tren de impulsos con período de 1 seg.
Z5	SEG	Contador de los segundos
Z6	MIN	Contador de los minutos
Z7	HORAS	Contador de las horas

El programa propuesto termina con la construcción de un tren de impulsos con el temporizador CLOCK, que funcionará como base de tiempos con un periodo de 1 segundo (ver Ejemplo 16). Al inicio del mismo, las tres primeras instrucciones hacen avanzar el conteo del contador SEC cuando un impulso de CLOCK se detecta durante el cierre de ENTRADA. SEC, o bien Z5, es, por consiguiente, el contador de los segundos.
El segundo grupo de instrucciones se ocupa de la carga del valor de los segundos en ACCU2 y de la constante 60 en ACCU1. Por tanto, los dos valores se confrontan para igualarse y, en caso de test afirmativo, el contador MIN aumenta y el contador SEC se repone a cero. Así pues, el contador MIN va aumentando cada 60 segundos y constituye así el contador de los minutos.
El tercer grupo de instrucciones se ocupa de la carga del valor de los minutos en ACCU2 y de la constante 60 en ACCU1. Luego, los dos valores se confrontan para igualarse y, en caso de test afirmativo, el contador HORAS se incrementa y el contador MIN se repone a cero. Así pues, el contador HORAS se incrementa cada 60 minutos y constituye así el contador de las horas.
En resumen, el cuentatiempo de software construido nos permite contar hasta a 999 horas, 59 minutos y 59 segundos (¡precisos!). Todo ello, partiendo de la base que pueda considerarse un valor tan preciso, sobre un tiempo tan largo, teniendo presentes los inevitables errores de los relojes internos, tanto del PLC real como del PC en el que 'gira' el simulador.
Como ejercicio, modifique el programa añadiendo un contador DIAS que se incrementará en 1 cada 24 horas.
Para probar el programa, sin tener que esperar tiempos muy largos, pueden reducir la constante de tiempo en la carga de CLOCK, aumentando así la frecuencia del tren de impulsos, o bien forzar manualmente valores de conteo próximos a los de comparación.

AWL

```
: U   -ENTRADA              'cada segundo incrementa el contador
: UN  -CLOCK                'de los segundos
: ZV  -SEG

: L   -SEG                  'si se llega a los 60 segundos
: L   KF +60
: !=F                       'incrementa el contador de los
: ZV  -MIN                  'minutos y restaura el de los
: R   -SEG                  'segundos

: L   -MIN                  'si se llega a los 60 minutos
: L   KF +60
: !=F
: ZV  -HORAS                'incrementa el contador de las horas
: R   -MIN                  'y restaura el de los minutos

: U   -RESET                'pone a cero los contadores a
: R   -SEG                  'continuación del cierre de la
: R   -MIN                  'entrada RESET
: R   -HORAS

: UN  -CLOCK                'construcción tren de impulsos con
: L   KT 100.0              'periodo de 1 segundo
: SA  -CLOCK

: BE
```

Ejemplo 21
Generador de onda cuadrada

La salida A2.7 debe estar controlada por una señal de onda cuadrada con Ton =0.5s y Toff =1.5s.

En la solución propuesta se emplean dos temporizadores que se 'rebotan' la activación.
Durante la primera carga del programa ladder, el contacto NC de T2 está cerrado y por lo tanto, la salida del temporizador T1 (retardo a la desactivación) se activa. Ahora, también el contacto NA de T1 en el recorrido 2 está cerrado y la salida de T2 también se activa.
Al ciclo siguiente, el contacto NC de T1 en esta ocasión está abierto y el tiempo de T1 empieza a avanzar y su salida continua estando alta, dejando a T2 en el mismo estado.
Finalizado el tiempo de T1 (1.5 seg) la salida del mismo pasa a nivel bajo y el contacto NA en el recorrido 2 se abre, el temporizador T2 se pone en marcha y su tiempo empieza a avanzar. Mientras, su salida continua alta.
Transcurrido el tiempo T2, la salida del temporizador pasa a nivel bajo. Hemos vuelto así a la condición inicial y tal como ya se ha descrito, el ciclo se repite indefinidamente.
El desarrollo temporal de las señales T1 y T2 se muestra en las dos primeras líneas del siguiente diagrama; mientras que la tercera línea muestra el desarrollo que debería tener la salida A2.7. Observemos que esta salida es el complemento de la señal T1, salvo en una pequeña diferencia de tiempo, igual a un ciclo de ejecución que, para mayor claridad, se ha exagerado voluntariamente en el diagrama.
Concluyamos el programa implementando al tercer recorrido la función de asignación de la salida del PLC según este criterio.

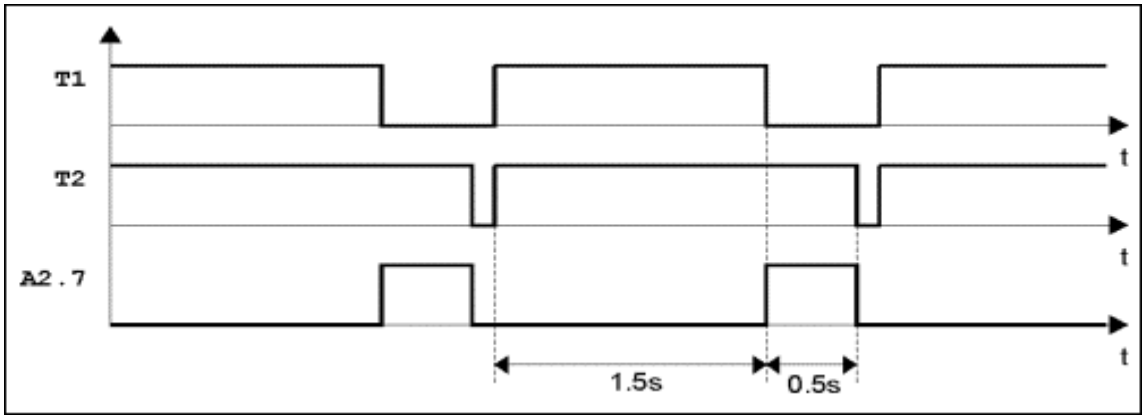
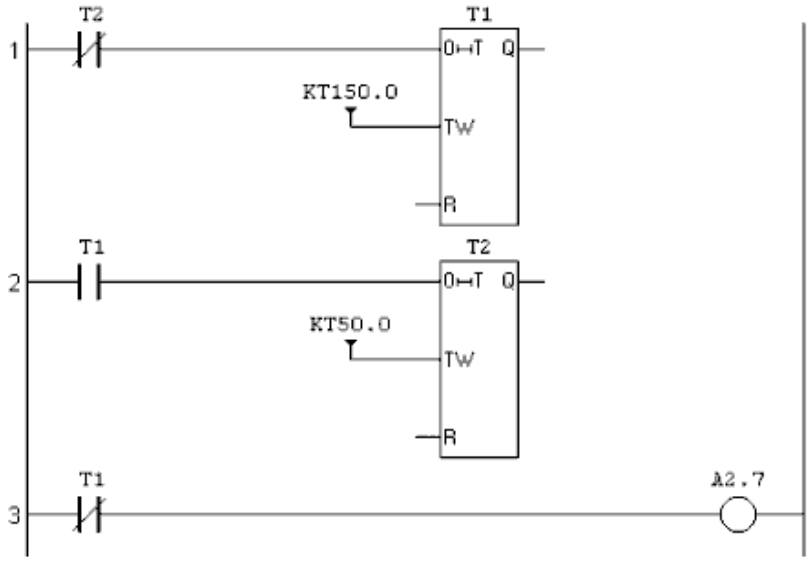


Diagrama temporal del generador de onda cuadrada

El programa AWL es la conversión pura y simple del programa ladder y consideramos que, a estas alturas, debería ser de fácil comprensión.
Programando oportunamente el valor de las dos constantes de tiempo se puede variar Ton y Toff, realizando una onda cuadrada con la frecuencia y el ciclo de trabajo que se quieran.

LADDER



AWL

```
: UN  T2                    'activación tiempo OFF
: L   KT 150.0
: SA  T1

: U   T1                    'activación tiempo ON
: L   KT 50.0
: SA  T2

: UN  T1
: =   A2.7                  'programación salida

: BE
```

Ejemplo 22
Otro generador de onda cuadrada

La salida A2.7 debe estar dirigida por una señal de onda cuadrada con Ton =0.5s y Toff =1.5s.

La solución que aquí se presenta es diferente de la propuesta en el ejemplo anterior para el mismo trazado. En efecto, aquí se ha utilizado un único temporizador de autoarranque que se programa, en el primer grupo de instrucciones, con un tiempo igual al período de la señal requerida (2 seg). Así pues, mientras transcurre, el tiempo varia entre 200 y 0 centésimas de segundo. Entre los valores 200 y 50, la salida A2.7 debe programarse al estado bajo, mientras que para valores menores de 50, esa misma deberá asumir el estado alto.
El segundo grupo de instrucciones se encarga de comparar el valor de tiempo actual con la constante 50 y, si es menor, activar la salida.
También en este caso se puede cambiar tanto el período de la señal, variando la constante de tiempo de T3, como el tiempo en el estado alto, variando la constante decimal de comparación.

AWL

```
: UN   T3                  'puesta en marcha del temporizador
: L    KT 200.0            'programación periodo onda cuadrada
: SA   T3

: L    T3                  'comparación con el tiempo
: L    KF +50              'en estado ON
: <F
: =    A2.7                'activación salida

: BE
```

Ejemplo 23
Control temporizado de luces

Un pulsador conectado a la entrada E0.0 activa, durante tres minutos, un grupo de luces conectadas a la salida A2.1. Junto con éstas se activa un piloto luminoso conectado a la salida A2.2 que, 15 segundos antes de que se apaguen las luces empieza a parpadear para avisar de la inminente finalización del tiempo. El piloto se apaga definitivamente a la vez que las luces.

Los temporizadores T2 y T3 se utilizan para generar una onda cuadrada con periodo de 1s y ciclo de trabajo del 50%. Ambos están cargados con una constante de tiempo de 50 centésimas de segundo. Ver Ejemplo 21.
Además, se utilizan otros dos temporizadores a impulso prolongado. T1, cargado con un tiempo de 3 minutos (KT180.2 , es decir, 180 segundos), dirige directamente la salida del grupo de luces. T0, cargado con un tiempo inferior en 15 segundos (KT165.2), se utilizará para discriminar el instante de inicio del parpadeo del piloto indicador.

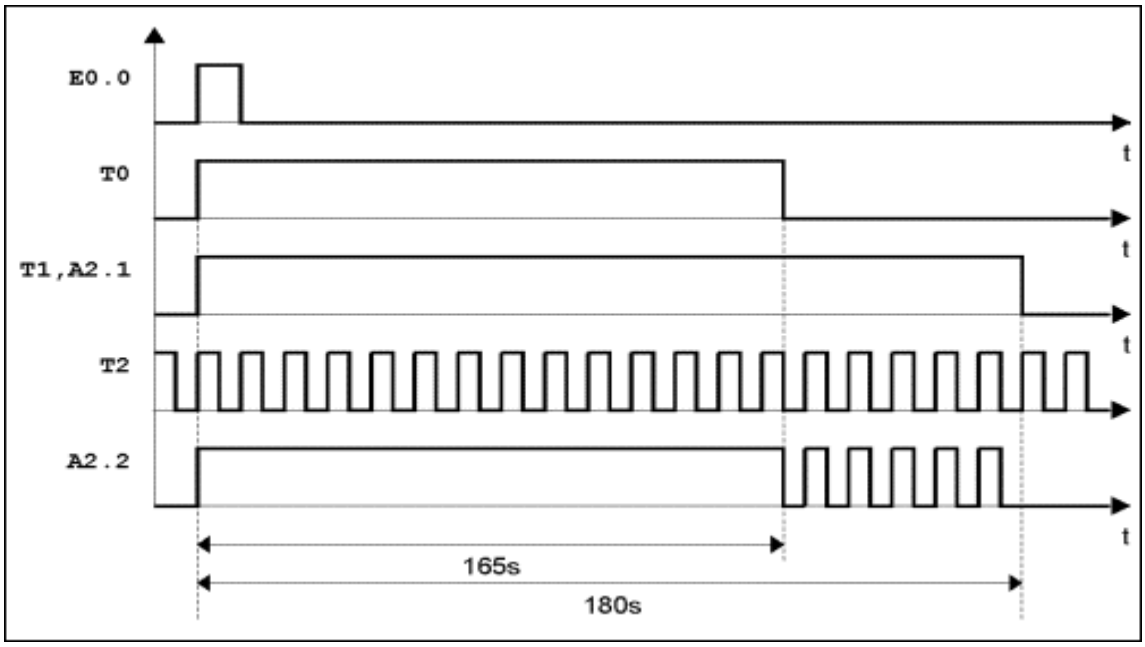
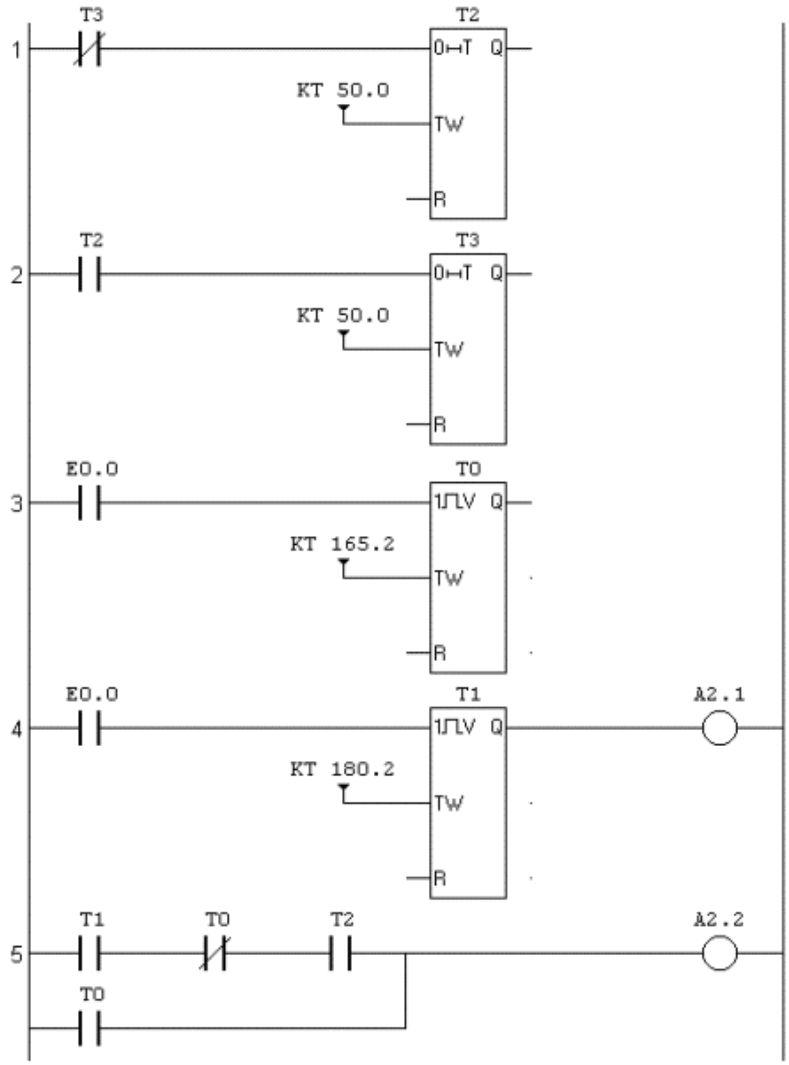


Diagrama del control temporizado de luces

Los desarrollos temporales de T0 y T1 a continuación del impulso sobre E0.0, se detallan en la segunda y tercera línea del diagrama superior. La cuarta línea indica el trazado, no a escala, del temporizador T2.
Observando la quinta línea se observa que el piloto indicador debe permanecer encendido o bien cuando está activo T0.0 o bien cuando están activos A2.1 y T2 y simultáneamente T0 está inactivo. Es decir, en términos de función booleana:
A2.2 = T0 OR (A2.1 AND NOT T0 AND T2)
Las soluciones propuestas implementan con precisión todo cuanto se ha descrito.
En la primera, realizada en ladder, se utilizan los dos primeros recorridos para la construcción de la señal de onda cuadrada. Los dos siguientes activan los temporizadores de las luces y del piloto indicador.
En el programa AWL, el primer grupo de instrucciones genera la señal de onda cuadrada. El segundo pone en marcha los temporizadores para las luces y el piloto indicador. El tercero enciende las luces y, por último, el cuarto enciende el piloto según las modalidades determinadas por la expresión indicada anteriormente.

LADDER



AWL

```
: UN   T3                    'oscilador 1Hz
: L    KT 50.0
: SA   T2
: U    T2
: L    KT 50.0
: SA   T3

: U    E0.0                  'puesta en marcha del temporizador
: L    KT 165.2              'para el inicio del parpadeo
: SV   T0                    '(luz intermitente)
: L    KT 180.2              'puesta en marcha del temporizador
: SV   T1                    'de las luces

: U    T1                    'encendido luces
: =    A2.1

: U    T1                    'encendido y
: UN   T0                    'parpadeo piloto automático
: U    T2
: O    T0
: =    A2.2

: BE
```

Ejemplo 24
Divisor de frecuencia (x4)

Realizar un divisor de frecuencia por 4: cada cuatro impulsos en la entrada se activa un impulso en la salida A2.1.

La primera parte de las soluciones propuestas realiza un tren de impulsos, tal como hemos aprendido a hacer en el Ejemplo 16, y muestra la señal en la salida A2.0. La segunda parte implementa efectivamente el divisor que no es otra cosa que un contador que va disminuyendo a cada impulso del generador (entrada ZR pilotado por A2.0) y que, cuando llega a cero, se autoprograma a valor 4 (salida Q llevada sobre la entrada S a través de M0.0).
Cada 4 impulsos de A2.0, en un sólo ciclo de ejecución, el conteo vuelve a cero. En este ciclo, la salida del contador se desactiva y también el merker M0.0 conectado a ella. Por el contrario, la salida A2.1 del PLC se activa, a causa de la negación del contacto de M0.0 que la dirige.

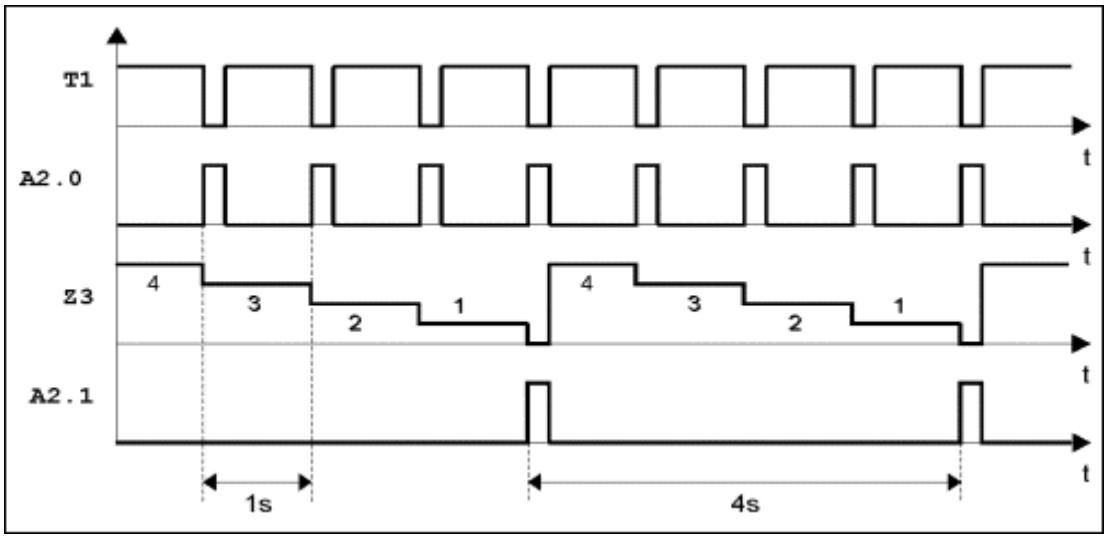
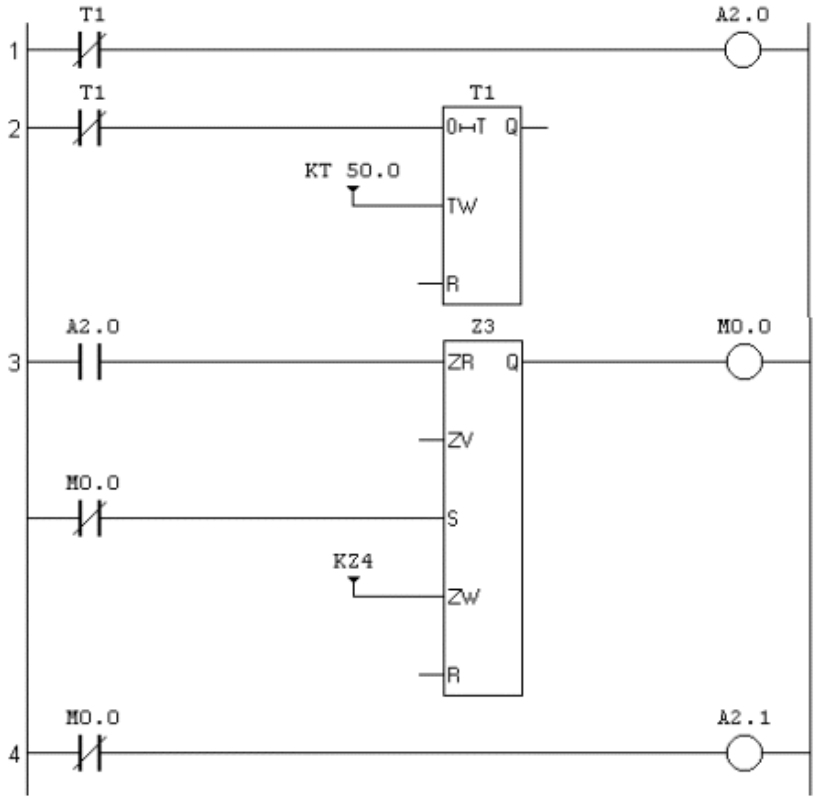


Diagrama temporal del divisor de frecuencia por 4

LADDER



AWL

```
: UN   T1                    'conteo impulsos
: ZR   Z3
: =    A2.0                  'activación salida cada segundo

: UN   Z3
: =    A2.1                  'activación salida cada 2 segundos
: L    KZ 4                  'restaurar contador a 4
: S    Z3

: UN   T1                    'construcción tren de impulsos
: L    KT 50.0               'a dividir
: SA   T1

: BE
```

Ejemplo 25
Conteo entradas cerradas (solución I)

Contar el número de las entradas cerradas del módulo 0.

Para resolver este ejemplo son necesarias operaciones que pertenecen al set integrativo. Por tanto es necesario escribir el programa en un bloque funcional y luego reclamar a éste desde el OB1 para permitir la ejecución cíclica. No será pues posible programar en ladder.
Depositaremos en MB80 el número de las entradas cerradas y utilizaremos W10 como puntero a la entrada normalmente examinada. La primera parte del programa en OB1 consiste en la inicialización a 0 de estas dos variables. La instrucción siguiente llama al bloque FB4 que contiene el núcleo del programa.
Como consecuencia de la instrucción de llamada de bloque incondicional, la elaboración del programa prosigue desde la primera instrucción de FB4.
Las dos primeras instrucciones de este bloque permiten cargar en RLC el complemento de la entrada apuntado por MW10, con dirección de canal en el byte alto y dirección de módulo en el byte bajo. Al primer paso, al estar MW10 a 0, se cargará el complemento del estado de E0.0.
La instrucción siguiente es un salto condicionado: si RLC=1, es decir, si el complemento de E0.0 es igual a 1, o sea, si la entrada está abierta, la elaboración prosigue desde la etiqueta INC. En cambio, si la entrada está cerrada, el salto no se efectúa y la elaboración continua con la instrucción siguiente.
Las cuatro instrucciones siguientes incrementan el valor de MB80, es decir, del número que representa el conteo de entradas que se hayan encontrado cerradas, cargando primero el valor en ACCU1, y luego sumando a 1 a éste y, por último, transfiriendo el resultado de nuevo a MB80. El conjunto de la función de estas instrucciones y de las anteriores es el de incrementar la variable MB80 si la entrada está cerrada y de no incrementarla si la entrada está abierta.
En uno y otro caso, la elaboración llega al grupo de instrucciones señalado con la etiqueta INC que se encarga, como primera acción, de incrementar el byte alto de MW10, es decir, MB10 y el byte que representa el número del canal de la entrada en la operación de carga dirigida que ya habíamos visto (con fines didácticos, hemos utilizado una operación diferente para obtener el incremento del byte). A continuación, se confronta el valor recién obtenido con el valor 7. Si es menor o igual, la entrada existe y debemos valorar su estado volviendo a la etiqueta TEST, tal como se ha especificado en la instrucción de salto condicionado. Por el contrario, en caso de que el valor sea 8, el canal no existe y tenemos que salir del bloque, habiendo examinado las ocho entradas posibles, de E0.0 a E0.7.
La parte de programa que va de la etiqueta TEST a la instrucción SPB =TEST se realiza pues ocho veces, antes de regresar al bloque emisor OB1 por medio de la última instrucción BE. Cada vez, MW10 contará un valor diverso: en una sucesión y en hexadecimal 0000, 0100, 0200, 0300, 0400, 0500, 0600, 0700; y cada vez, el grupo de instrucciones B MW10 y UN E0.0 cargará en RLC el complemento de una entrada distinta del módulo, desde el primero hasta el último. La parte de programa que va de L MB80 hasta T MB80 se cargará únicamente si la entrada verificada normalmente está cerrada, con el resultado de ejecutar un incremento de MB80 solo con esta condición, y, puesto que el valor inicial de este merker byte es igual a 0, al final de la ejecución del bloque, este contará efectivamente el número de las entradas cerradas.

AWL

OB1

```
    : L    KF +0                  'programaciones iniciales
    : T    MB80
    : T    MW10

    : SPA FB4 ()

    : BE
```

FB4

```
TEST: B    MW10                   'detección de entrada
    : UN   E0.0
    : SPB =INC
    : L    MB80                   'la entrada está activa:
    : L    KF +1                  'incremento del merker byte
    : +F                          'de conteo
    : T    MB80

INC : L    MB10                   'incremento del índice para
    : I   1                       'el examen de la entrada siguiente
    : T    MB10
    : L    KF +7                  'control al finalizar
    : <=F
    : SPB =TEST

    : BE
```

Si desea analizar las entradas cerradas del módulo 1 bastará con cargar 1 como valor de inicialización para MW10. También es posible valorar el total de entradas cerradas para los dos módulos. Se trata de reclamar dos veces el FB4, la primera inicializando MW10 a 0 y la segunda inicializándolo a 1. En cambio, para MB80 hará falta una sola inicialización a 0 al principio del OB1. Dejamos al lector la realización de este programa.

Ejemplo 26
Conteo entradas cerradas (solución II)

Contar el número de entradas cerradas del módulo 0.

En este ejercicio, con el propósito de utilizar una operación de desplazamiento, afrontamos una solución diferente a la programada en el ejemplo precedente.
El contador de entradas cerradas sigue siendo MB80 y su valor sigue inicializado a 0 al principio del OB1. Por su parte, MB10 representa una máscara de 8 bit, uno solo de los cuales será, por turno, a 1. Su valor inicial es 1, por tanto, (00000001)2, donde solo el bit 0 es cierto.
Las tres primeras instrucciones del bloque FB4 cargan en los acumuladores los valores de la máscara y del byte de entrada relativo al módulo 0 y ejecutan la AND bit a bit. Al primer paso, con el valor de la máscara apenas visto, la AND da resultado distinto de 0 sólo si E0.0 está cerrado. Así pues, el salto sobre cero previsto por la siguiente instrucción solo se efectúa por entrada abierta.
Si la entrada está cerrada, como en el ejemplo anterior, incrementamos el byte de conteo MB80.
En uno u otro caso, los dos recorridos de elaboración se reúnen en la etiqueta INC donde, tras haber cargado la máscara en ACCU1, se dispone el desplazamiento hacia la izquierda de una posición. El resultado del desplazamiento o, mejor aun, la parte baja de éste, se retransfiere nuevamente a MB10. Después de la instrucción L KF+256, ACCU1 contendrá el valor 256 y ACCU2 contendrá el resultado del desplazamiento. Si el bit de máscara, después de 8 desplazamientos, ha acabado en la posición 8 del registro de 16 bit del acumulador, el valor de este último será 28=256, la instrucción de salto condicionado se ignora y la elaboración del bloque termina. Si el bit ocupa posiciones inferiores, y entonces el contenido del acumulador resulta menor de 256, el salto a la etiqueta TEST se ejecuta para repetir la elaboración con el fin de examinar la entrada siguiente.

AWL

OB1

```
    : L    KF +0                'inicialización merker byte
    : T    MB80                 'de conteo
    : L    KF +1                'inicialización máscara
    : T    MB10

    : SPA FB4 ()

    : BE
```

FB4

```
TEST: L    MB10                 'control de la entrada
    : L    EB0                  'no enmascarada
    : UW
    : SPZ =INC

    : L    MB80                 'la entrada está activa:
    : L    KF +1                'incremento del merker byte
    : +F                        'de conteo
    : T    MB80

INC : L    MB10                 'shift de la máscara para el
    : SLW 1                     'examen de la entrada siguiente
    : T    MB10
    : L    KF +256              'control al finalizar
    : <F
    : SPB =TEST

    : BE
```

Ejemplo 27
Semáforo para Formula 1

Con la activación del pulsador conectado a la entrada E0.0, las cinco luces de un semáforo deben encenderse una tras otra, una a cada segundo. Al cabo de un segundo del encendido completo, las luces deberán apagarse.
Para programar la solución a este problema se ha hecho uso de símbolos. Su correspondencia con los operandos absolutos se ha establecido según la tabla siguiente.

Op. absoluto	Simbolo	Comentario
E0.0	START	Pulsador de START de la secuencia
A2.0	L1	Lámpara 1
A2.1	L2	Lámpara 2
A2.2	L3	Lámpara 3
A2.3	L4	Lámpara 4
A2.4	L5	Lámpara 5

El diagrama temporal siguiente muestra, en las líneas intermedias, el desarrollo de las salidas del PLC que controlan las luces del semáforo, en función de la entrada START indicada en la primera línea.

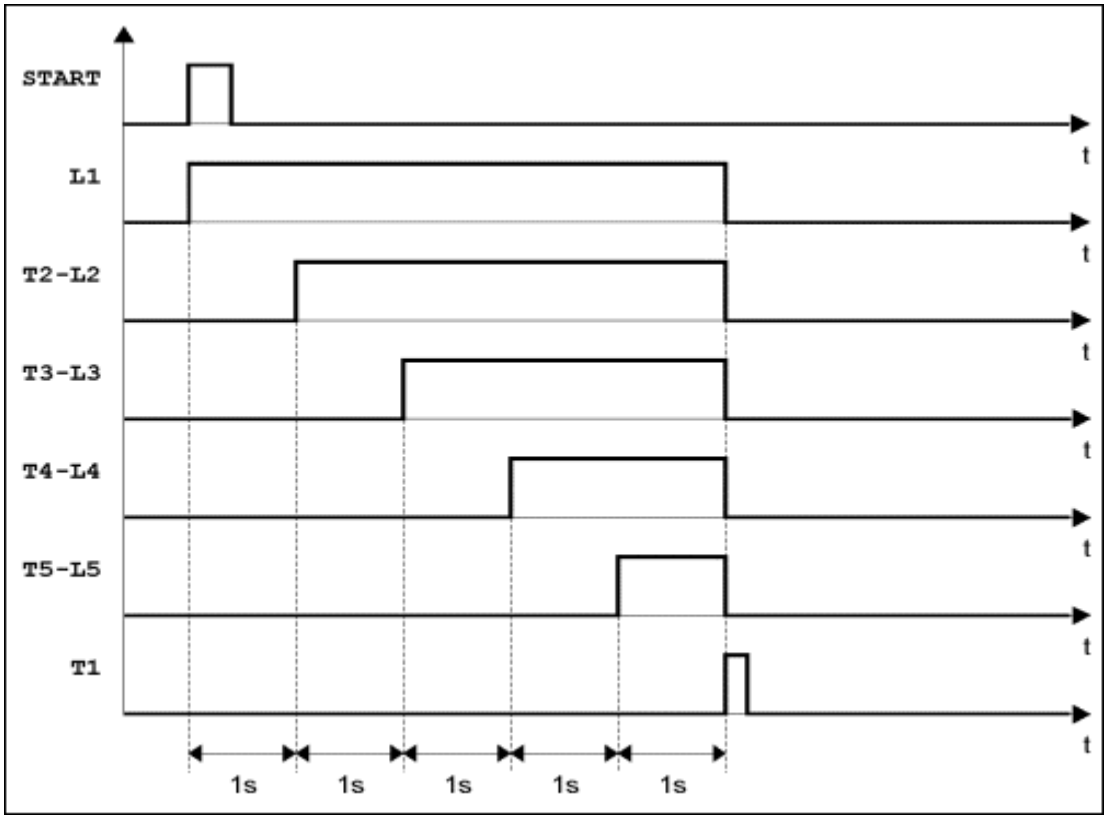
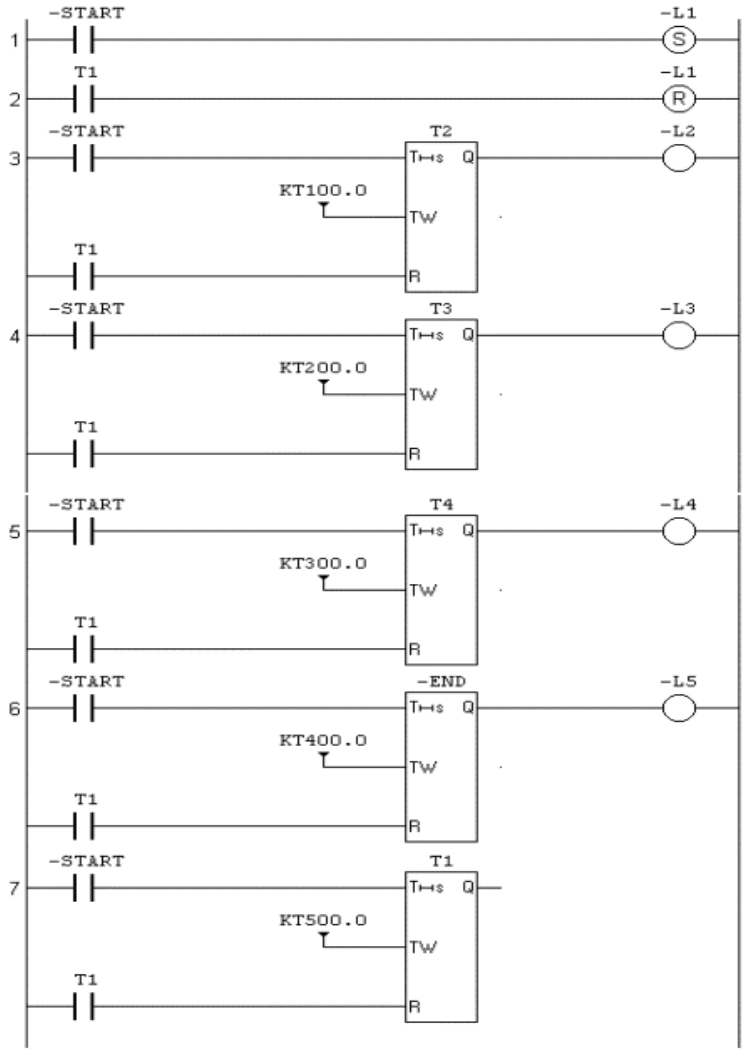


Diagrama temporal para un semáforo de Fórmula 1

El cierre de esta entrada, además de activar la salida que controla la primera luz, que se enciende inmediatamente, activa cinco temporizadores del tipo retardo a la activación con memoria, con tiempos de 1 a 5 segundos. La salida de cada uno de ellos, T1 excluido, una vez transcurrido el tiempo programado, se llevará al estado alto, activando la correspondiente luz y obteniendo con facilidad la secuencia de encendido. Por su parte, el temporizador T1, se encarga del apagado de todas las luces desactivando L1 y todos los demás temporizadores, incluido él mismo.
Ponga en Run el PLC, trasforme el interruptor 0.1 en un pulsador, accione el pulsador y …¡que gane el mejor!

LADDER

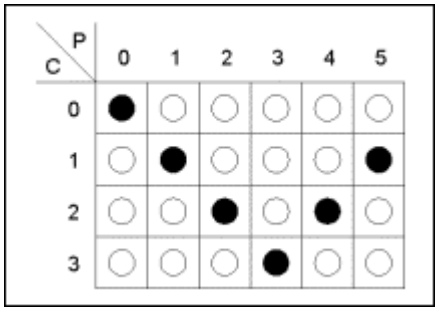


AWL

```
: U   -START                'programación temporizadores por secuencia
: L   KT 100.0
: SS  T2
: L   KT 200.0
: SS  T3
: L   KT 300.0
: SS  T4
: L   KT 400.0
: SS  T5
: L   KT 500.0
: SS  T1

: U   -START                'programación salidas
: S   -L1
: U   T2
: =   -L2
: U   T3
: =   -L3
: U   T4
: =   -L4
: U   T5
: =   -L5

: U   T1                    'reset temporizadores
: R   -L1
: R   T1
: R   T2
: R   T3
: R   T4
: R   T5

: BE
```

Ejemplo 28
Luces secuenciales en 4 canales

Construir un secuenciador de 4 canales que prevea el siguiente esquema de encendido.



Esquema de encendido para un secuenciador de 4 canales

El esquema establece la secuencia de encendido de las luces conectadas a cuatro canales. Los círculos negros indican la activación del canal durante su paso específico. Así, durante el paso 0 estará activo el canal 0, durante el paso 1 el canal 1, y así sucesivamente. Si las luces se disponen en línea, el efecto será el de un desplazamiento de la fuente luminosa desde la primera hasta la última posición y luego al contrario.
Al paso 5 le sucede un paso 6 idéntico al 0 y luego otro idéntico al paso 1, es decir, el diagrama se va recorriendo cíclicamente. Imaginen que lo recortan y lo enroscan formando un cilindro y hacen coincidir los límites opuestos del paso 0 y del 5, sería algo similar al tambor de un carillón: cuando se acaba la musiquilla, vuelve a empezar de nuevo.
Continuando con nuestro símil sonoro, para que un carillón funcione necesitamos un cilindro con unas levas dispuestas de un modo adecuado sobre su superficie lateral y un mecanismo que lo haga girar.
Empezaremos precisamente por construir este último. El mecanismo de avance de nuestro secuenciador será un tren de impulsos con período de 0.2 segundos, es decir, una base de tiempo con un período elegido arbitrariamente. Las instrucciones

```
UN -TIMER
L  KT20.0
SA -TIMER
```

constituyen un tren de impulsos, tal como hemos aprendido a hacer en el Ejemplo 16. El cilindro, por su parte, estará formado por un contador que, partiendo de 0, va incrementándose a cada impulso.

```
UN -TIMER
ZV -COUNTER
```

Cuando llegue a 6, es decir, después del último paso, deberá reprogramarse al valor 0 de partida, siendo, además, idéntico el paso 6 al paso 0 .

```
L -COUNTER
L KF+6
!=F
R -COUNTER
```

Antes de disponer las levas sobre el cilindro identificamos los tramos del cilindro que corresponden a cada paso especifico. Las instrucciones

```
L -COUNTER
L KF+0
!=F
= -PASO0
```

programan a 1 el merker PASO0 cuando el contador vale 0, identificando entre los posibles valores del contador el correspondiente a dicho paso. Los siguientes grupos de instrucciones programan los merker correspondientes para cada uno de los demás pasos. Así, al final, cada 0.2s será activo un merker diverso, a continuación de PASO0 a PASO5 y, luego, volviendo a empezar desde PASOO0.
Ahora que hemos identificado las posiciones podemos insertar las levas. Empezamos por el canal 0 y observamos de nuevo la parrilla de encendido. El canal 0 está activo sólo durante el paso 0, o sea:

```
O -PASO0
= -CH0
```

El canal 1 debe estar activo tanto durante el paso 1 como durante el paso 5:
O -PASO1
O -PASO5
= CH1
Proseguimos así para los otros dos canales, hasta terminar el carillón o, abandonando ya el símil didáctico, el secuenciador.
En este ejemplo se puede aumentar o disminuir la duración de los pasos simplemente cambiando la constante con la que se carga el temporizador, produciendo el efecto de variar la velocidad del desplazamiento aparente de la fuente luminosa. Se puede modificar el número de pasos, cambiando la constante del valor de conteo para la reposición del contador y añadiendo otros valores para confrontar los nuevos pasos. También es posible cambiar la secuencia de encendido de las luces, modificando las condiciones en los grupos de OR que constituyen la última parte del programa.

AWL

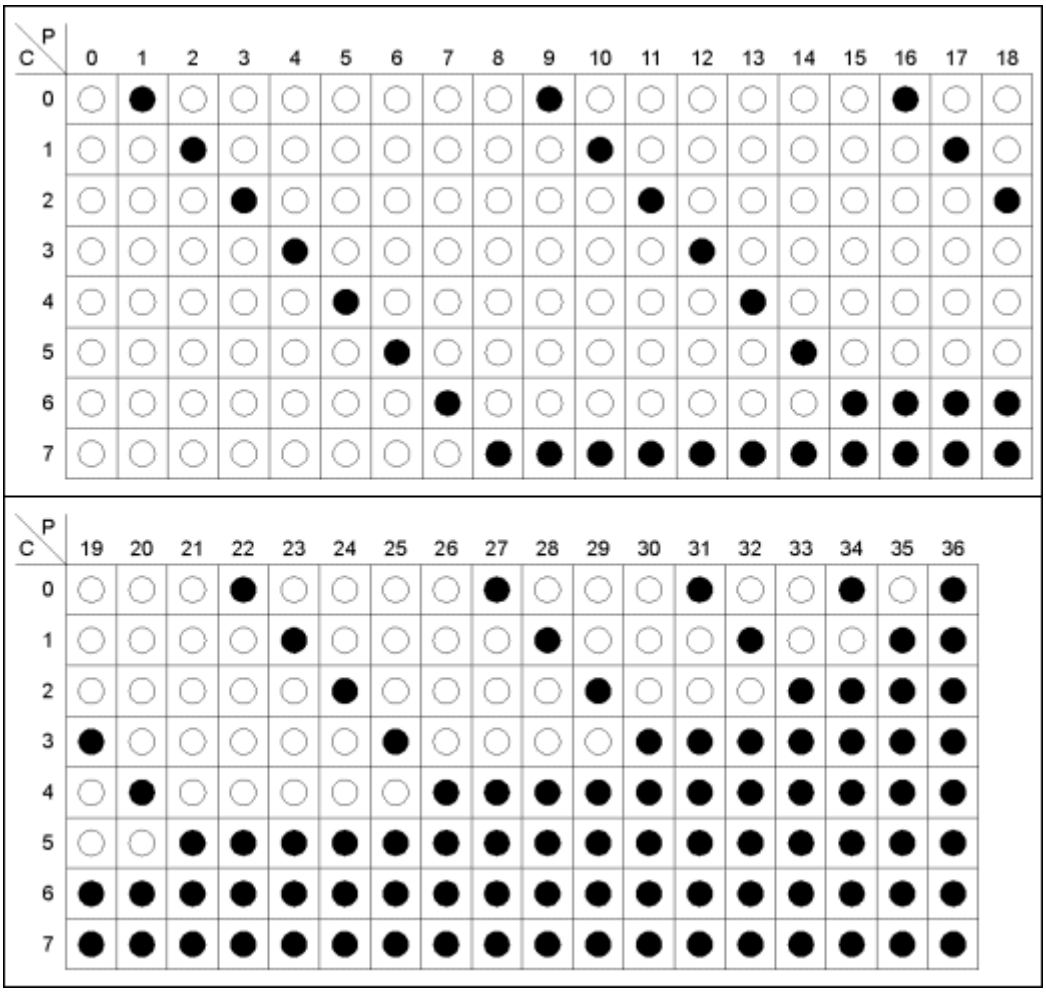
```
: UN   -TIMER                'avance paso
: ZV   -COUNTER

: UN   -TIMER                'construcción base tiempos
: L    KT 10.0
: SA   -TIMER

: L    -COUNTER              'reinicialización ciclo
: L    KF +6
: !=F
: R    -COUNTER

: L    -COUNTER              'test paso 0
: L    KF +0
: !=F
: =    -PASO0

: L    -COUNTER              'test paso 1
: L    KF +1
: !=F
: =    -PASO1

: L    -COUNTER              'test paso 2
: L    KF +2
: !=F
: =    -PASO2

: L    -COUNTER              'test paso 3
: L    KF +3
: !=F
: =    -PASO3

: L    -COUNTER              'test paso 4
: L    KF +4
: !=F
: =    -PASO4

: L    -COUNTER              'test paso 5
: L    KF +5
: !=F
: =    -PASO5

: O    -PASO0              'programación canal 0
: =    -CH0

: O    -PASO1              'programación canal 1
: O    -PASO5
: =    -CH1

: O    -PASO2              'programación canal 2
: O    -PASO4
: =    -CH2

: O    -PASO3              'programación canal 3
: =    -CH3

: BE
```

Seguramente, la que hemos presentado no es la única solución al problema y, por lo general, a medida que éste se hace más complejo, las posibles soluciones aumentan. Posiblemente ni siquiera es la mejor en términos de versatilidad, simplicidad, de mantenimiento o de elegancia de programación, si bien en el

próximo ejemplo propondremos una solución que responde mejor a estos requisitos, pero es la aproximación más sencilla, que utiliza en definitiva las instrucciones más comunes, que hemos logrado reproducir, y esto es un requisito fundamental para la tarea que nos hemos propuesto: acompañarles en sus primeros pasos en el mundo de la programación de los PLC.

Ejemplo 29
Luces secuenciales en 'barra'

Construir un secuenciador de 8 canales que prevea el siguiente esquema de encendido.



Esquema de encendido para luces secuenciales en barra

El efecto óptico, si las luces están dispuestas en línea vertical al igual que los LED del módulo del PLC, es el de un llenado progresivo de la barra debido a fuentes luminosas que aparecen por arriba y que poco a poco van descendiendo hasta ocupar la última posición libre, es decir, apagada. Una vez iluminada toda la barra, el ciclo vuelve a empezar, apagando las luces y reiniciando su llenado.
La secuencia es bastante más complicada que la anterior como para obligarnos a buscar una solución distinta, más funcional y, tal vez, más versátil. La solución prevista utiliza un bloque de datos para memorizar las combinaciones de encendido de la secuencia. Cada data word contiene, en el byte bajo, el código de encendido de las luces. La última word contiene el dato (FFFF)16 que actúa de finalizador de secuencia con las modalidades que en seguida veremos.
El corazón del programa está constituido por el bloque FB10 que se llama cada 2 décimas de segundo y que se encarga de leer los datos y de enviarlos al módulo de salida. Veamos como.
MW100 constituye el puntero en la data word actual. Su valor inicial es 0. El grupo de instrucciones en la etiqueta READ, siguiente a la apertura del DB20, carga primero el valor (FFFF)16 en el acumulador, y luego el valor de la data word corriente, utilizando una instrucción dirigida, y confronta los dos valores: si son distintos, la elaboración continua transfiriendo el contenido de ACCU1, el dato de la secuencia, al módulo 2 de salida. A continuación, después de haberse incrementado con el valor del puntero, se retorna al bloque reclamante de manera que, cuando sea nuevamente reclamado FB4, la data word cargada será la siguiente.

En cambio, si el valor leído es igual a (FFFF)16, la elaboración continua primero en la etiqueta INIT, donde se realiza la puesta a cero del índice para la vuelta al principio de la secuencia, y luego en la etiqueta READ para la nueva lectura de la primera combinación de la secuencia, DW0.

AWL

OB1 (Principal)

```
     : UN  TO                    'llamada a tiempo del FB10
     : SPB FB10 ()

     : UN  TO                    'base tiempos de 0.2 segundos
     : L   KT 20.0
     : SA  TO

     : BE
```

OB21 (Inicialización sobre STOP-RUN)

```
     : L   KF +0                 'reinicialización índice
     : T   MW100

     : BE
```

OB22 (Inicialización sobre OFF-ON)

```
     : L   KF +0                 'reinicialización índice
     : T   MW100

     : BE
```

FB10 (Encendido luces)

```
     : A   DB20                  'apertura bloque de datos

READ: L   KH FFFF                'carga del código fin de ciclo
     : B   MW100                 'lectura del dato en uso
     : L   DW0                   'y control si es fin de ciclo
     : !=F
     : SPB =INIT
     : T   AB2                   'encendido luces

     : L   MW100                 'incremento del índice
     : L   KF +1
     : +F
     : T   MW100

     : BEA

INIT: L   KF +0                  'reinicialización índice
     : T   MW100
     : SPA =READ

     : BE
```

DB20 (Datos para la secuencia)

```
 0 KM 00000000 00000000          datos del ciclo
 1 KM 00000000 00000001
 2 KM 00000000 00000010
 3 KM 00000000 00000100
 4 KM 00000000 00001000
 5 KM 00000000 00010000
 6 KM 00000000 00100000
 7 KM 00000000 01000000
 8 KM 00000000 10000000
 9 KM 00000000 10000001
10 KM 00000000 10000010
11 KM 00000000 10000100
12 KM 00000000 10001000
13 KM 00000000 10010000
14 KM 00000000 10100000
15 KM 00000000 11000000
16 KM 00000000 11000001
17 KM 00000000 11000010
18 KM 00000000 11000100
19 KM 00000000 11001000
20 KM 00000000 11010000
21 KM 00000000 11100000
22 KM 00000000 11100001
23 KM 00000000 11100010
24 KM 00000000 11100100
25 KM 00000000 11101000
26 KM 00000000 11110000
27 KM 00000000 11110001
28 KM 00000000 11110010
29 KM 00000000 11110100
30 KM 00000000 11111000
31 KM 00000000 11111001
32 KM 00000000 11111010
33 KM 00000000 11111100
34 KM 00000000 11111101
35 KM 00000000 11111110
36 KM 00000000 11111111
37 KH FFFF                      finalizador de ciclo
```

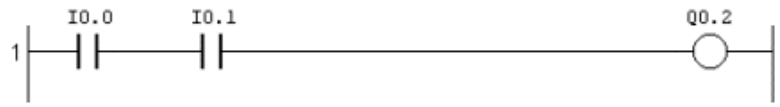
Ejemplos de programación para PLC PS3

(27 Problemas Resueltos)

INDICE

Ejemplo 1
Combinación AND

La salida Q0.2 debe activarse tan sólo si los dos interruptores conectados a las entradas I0.0 y I0.1 están cerrados.
La solución ladder se obtiene pilotando la bobina QO.2 a través de la serie de dos contactos con operandos I0.0 y I0.1. De hecho, la combinación lógica AND, traducida al lenguaje ladder, equivale a la serie de dos contactos: en la disposición en serie 'se lee' el cierre del circuito tan sólo cuando los dos están cerrados. Por lo tanto, ésta es la única condición que activa la bobina.

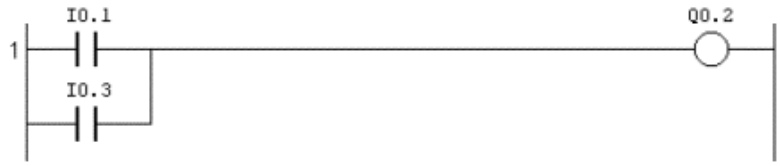
LADDER	AWL
	000: L I0.0 001: A I0.1 002: = Q0.2

La solución AWL se obtiene cargando primero el estado de I0.0 en el registro general de bit (L I0.0), a continuación, se efectúa una AND entre este último y la entrada I0.1 (A I0.1) y el resultado se deposita de nuevo en el registro general. La última instrucción (= Q0.2) se encarga de transferir el contenido del registro general, que en ese momento representa la combinación lógica I0.0 AND I0.1, a la salida Q0.2

Ejemplo 2
Combinación OR

Realizar Q0.2 = I0.1 OR I0.3

La salida Q0.2 debe activarse si por lo menos uno de los interruptores conectados a las entradas I0.0 o I0.1 está cerrado.
La solución ladder se obtiene pilotando la bobina Q0.2 por medio del paralelo de dos contactos, con operandos I0.0 y I0.1. De hecho, la combinación lógica OR, traducida al lenguaje ladder, equivale al paralelo de dos contactos: en la disposición en paralelo 'se lee' el cierre del circuito cuando al menos uno de los contactos está cerrado. Por lo tanto, esta es la condición que conduce a la activación de la bobina.

LADDER	AWL
	000: L I0.1 001: O I0.3 002: = Q0.2

La solución AWL se obtiene cargando primero el estado de I0.0 en el registro general de bit (L I0.0), a continuación se efectúa una OR entre este último y la entrada I0.1 (O I0.1) y el resultado se deposita de nuevo en el registro general. La última instrucción (= Q0.2) se encarga de transferir el contenido del registro general, que en este momento representa la combinación lógica I0.0 OR I0.1, a la salida Q0.2.

Ejemplo 3
Combinación AND de OR

Realizar Q0.0 = (I0.0 OR I0.1) AND (I0.2 OR I0.3)

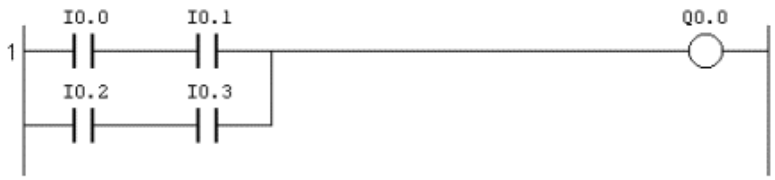
Después de haber realizado los ejercicios anteriores, debería encontrar la solución ladder por pura intuición: La serie (AND) de dos paralelos (OR) de contactos pilota la bobina Q0.0, en correspondencia con los cuales se han conectado oportunamente los operandos.

LADDER	AWL
	000: L I0.0 001: O I0.1 002: L I0.2 003: O I0.3 004: A 005: = Q0.0

En cambio, la solución AWL requiere alguna aclaración, ya que se ha introducido una nueva operación que utiliza el registro de stack. En primer lugar, observamos que después de la ejecución de la instrucción 001, el registro general de bit contiene el resultado de la combinación lógica OR entre I0.0 y I0.1 (ver Ejemplo 2). La siguiente operación de carga, y puesto que la secuencia aun no está acabada, introduce primero el contenido del registro general en el registro de stack, y luego carga el estado del operando especificado en el registro general. Una vez se ha ejecutado la instrucción 003, el registro de stack contiene aún, en primera posición, el valor calculado con anterioridad I0.0 OR I0.1 y el registro general contiene el valor de I0.2 OR I0.3, tal y como se ha obtenido ejecutando las instrucciones 002 y 003. La instrucción 004, una AND sin operando, ordena a la CPU del PLC que ejecute una AND entre el registro de stack y el registro general, es decir, en este caso, entre los resultados de las dos combinaciones OR, y que deposite de nuevo el resultado en el registro general. Tan sólo queda copiar el contenido de este último en la salida Q0.0 con la última instrucción.

Ejemplo 4
Combinación OR de AND

Realizar Q0.0 = (I0.0 AND I0.1) OR (I0.2 AND I0.3). Donde los paréntesis, si bien no son necesarios por cuanto la operación AND tiene preferencia sobre la OR, se han añadido para mayor claridad.

LADDER	AWL
	000: L I0.0 001: A I0.1 002: L I0.2 003: A I0.3 004: O 005: = Q0.0

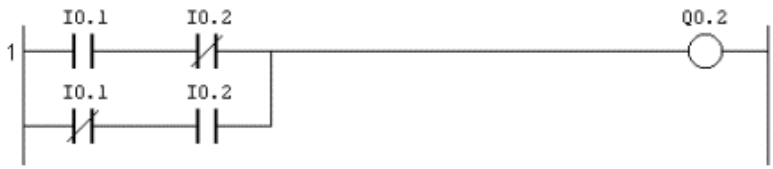
En la solución AWL observamos en primer lugar, que después de la ejecución de la instrucción 001, el registro general de bit contiene el resultado de la combinación lógica AND entre I0.0 y I0.1 (ver Ejemplo 1). La siguiente operación de carga, dado que aun no ha concluido la secuencia, introduce primero el contenido del registro general en el registro de stack, y luego carga el estado del operando especificado en el registro general. Una vez se ha ejecutado la instrucción 003, el registro de stack contiene aun, en primera posición, el valor calculado con anterioridad I0.0 AND I0.1 y el registro general contiene el valor de I0.2 AND I0.3, tal y como se ha obtenido siguiendo las instrucciones 002 y 003. La instrucción 004, una OR sin operando, ordena a la CPU del PLC que ejecute una OR entre el registro de stack y el registro general, es decir, en este caso, entre los resultados de las dos combinaciones AND, y que deposite de nuevo el resultado en el registro general. Tan sólo queda copiar el contenido de este último en la salida Q0.0 con la última instrucción.

Ejemplo 5
Combinación XOR

Realizar Q0.0 = I0.1 XOR I0.2

La operación lógica XOR aplicada a dos variables booleanas da un resultado cierto cuando una y sólo una de las dos variables es cierta.
La primera serie de contactos del programa ladder está cerrada únicamente cuando I0.1 está cerrado y I0.2 está abierto. La segunda serie, por el contrario, está cerrada tan sólo cuando I0.1 está abierto y I0.2 está cerrado. Realizando el paralelo de las dos se obtiene la función deseada. Es decir, la bobina se activa tan sólo cuando una entrada está cerrada y la otra está abierta.

LADDER



AWL

```
000:    L    I0.1
001:    AN   I0.2
002:    LN   I0.1
003:    A    I0.2
004:    O
005:    =    Q0.2
```

```
000:    L    I0.1
001:    XO   I0.2
002:    =    Q0.2
```

La primera solución AWL no es más que la traducción del programa ladder y su funcionamiento es muy similar al ejemplo anterior. La segunda solución utiliza, por el contrario, la operación XO que realiza directamente la XOR entre los operandos.

Ejemplo 6
Autorretención

Un pulsador conectado a la entrada I0 debe activar la salida Q0.15 y un segundo pulsador conectado a la entrada I0.1 debe desactivarla.

En el programa ladder propuesto se realiza un circuito con autorretención. Accionando el pulsador conectado a I0.0 la bobina Q0.15 se activa y entonces, el contacto con el mismo operando en la segunda línea se cierra (imaginen que el contacto y la bobina son parte de un mismo relé Q0.15) y continua manteniendo activada la bobina incluso después de la apertura de I0.0. El cierre del pulsador en la entrada I0.1 provoca la apertura del contacto, normalmente cerrado en el esquema, desactivando la bobina y cortando la autorretención.

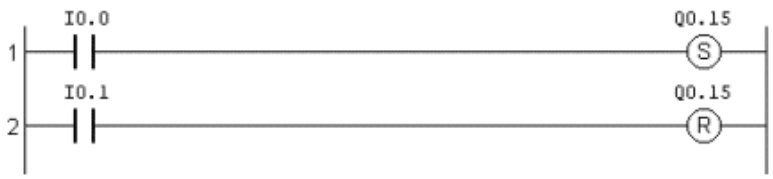
LADDER	AWL
	000: L IO.0 001: O Q0.15 002: AN IO.1 003: = Q0.15

El programa AWL propone la conversión de lo anteriormente descrito. El valor de la salida Q0.15 en la última instrucción se calcula cargando el estado de I0.0, valorando luego la OR con Q0.5 y, por último, poniendo en AND el resultado con el complemento de I0.1.

Ejemplo 7
Set y reset

Un pulsador conectado a la entrada I0.0 debe activar la salida Q0.15 y un segundo pulsador, conectado a la entrada I0.1, debe desactivarla

El ejercicio es idéntico al precedente pero, en esta ocasión, en la solución se utilizan bobinas de set y reset de la misma salida Q0.15 pilotadas separadamente por contactos de I0.0 y I0.1.

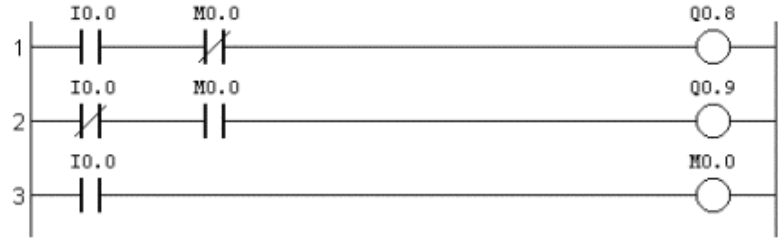
LADDER	AWL
	000: L I0.0 001: S Q0.15 002: L I0.1 003: R Q0.15

Si hacemos funcionar el programa, observamos que al presionar dos pulsadores a la vez se desactiva la salida. En efecto, en ambos lenguajes, cuando las condiciones de test son válidas a la vez, el operando Q0.15 se activa primero, en el recorrido 1 o con las dos primeras instrucciones, y luego se desactiva, en el recorrido 2 o con la tercera y cuarta instrucción. Pero recordemos que Q0.15 no representa efectivamente la salida física del PLC, sino el correspondiente bit en el interior de la memoria de la imagen del proceso. Dicho bit de memoria es llevado efectivamente a 1 y luego a 0 pero, tan sólo al final de la carga del programa utilizado, el valor elaborado por él se transfiere al canal físico de salida correspondiente, que se mantiene constantemente en el valor bajo cuando ambas entradas están cerradas.
Con esta escritura de programas hemos hecho prevalecer el reset respecto al set. Si desea obtener lo contrario, le bastará con invertir la posición de los recorridos en el esquema de contactos o cambiar la primera secuencia por la segunda en el programa AWL.

Ejemplo 8
Activación por flancos

Activar las salidas Q0.8 y Q0.9 respectivamente con los flancos ascendente y descendente de la entrada I0.0.

Observamos que el último recorrido del esquema ladder y las dos últimas instrucciones del programa AWL imponen, al final de la ejecución del programa, la igualdad del merker bit (marca o flag) M0.0 al estado de la entrada I0.0. Pero, en correspondencia con los flancos y para los recorridos o las instrucciones precedentes, se da el hecho que el estado de las dos variables es opuesto y que tan sólo al final de la carga del programa utilizado se convierten en iguales. Todo ello queda representado en las dos primeras líneas del diagrama con un retraso temporal entre M0.0 respecto a I0.0 que equivale a un ciclo de ejecución.

LADDER	AWL
	000: L IO.0 001: AN M0.0 002: = Q0.8 003: LN IO.0 004: A M0.0 005: = Q0.9 006: L IO.0 007: = M0.0

La parte inicial de los dos programas activa la bobina Q0.8, para un ciclo de ejecución, cuando I0.0 esta a 1 y M0.0 está a 0, es decir, en correspondencia con el flanco ascendente de I0.0, tal como aparece indicado en la tercera línea del diagrama.
En cambio, la bobina Q0.9 se activará en el segundo recorrido o a la segunda secuencia, siempre para un ciclo de ejecución, cuando I0.0 está a 0 y M0.0 está a 1, es decir, en correspondencia con el flanco descendente de I0.0, tal como aparece indicado en la cuarta línea del diagrama.

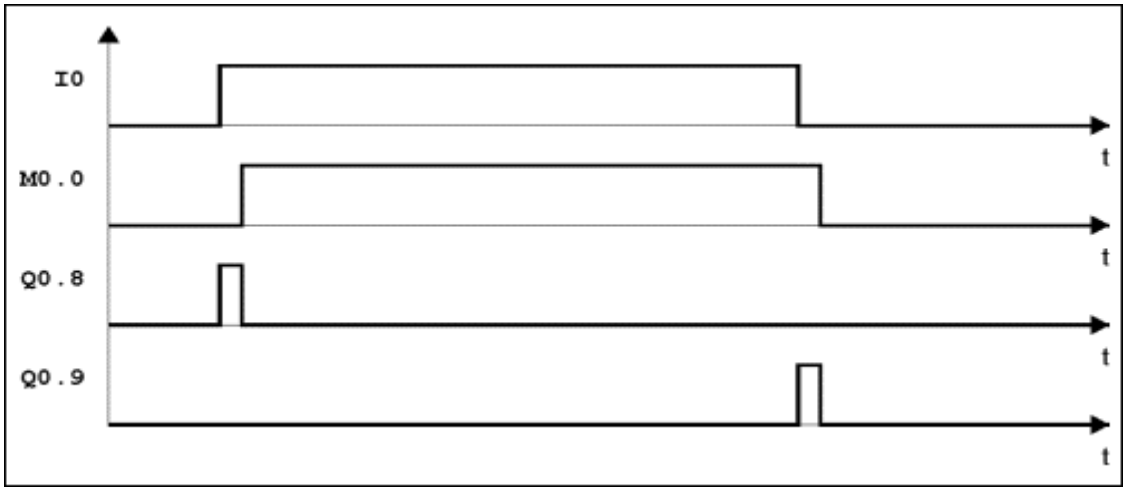
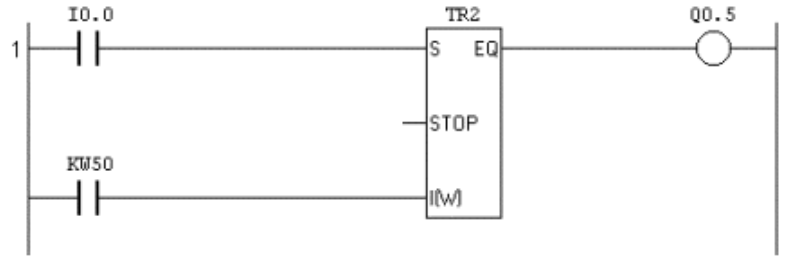


Diagrama temporal de un generador de flancos

Ejemplo 9
Temporizador con retardo a la activación

La salida Q0.0 se activa 5 segundos después de la activación de la entrada I0.2. Cuando la entrada vuelve a estar baja, la salida se desactiva.

Los programas que siguen, resuelven el problema en Ladder y AWL.

LADDER	AWL
	000: TR2 *S: I0.0 *STP: *IW: KW50 *EQ: Q0.5

La solución es sumamente sencilla ya que existe un módulo de sistema que realiza precisamente la funcionalidad requerida por el trazado: el temporizador con retardo a la activación, único tipo de temporizador que este PLC posee.
En ladder se trata de conectar las entradas y la salida del módulo de sistema a contactos y bobinas. En correspondencia con estos deberá indicar los operandos apropiados. Así, la entrada S se ha conectado a un contacto NA de I0.0 mientras que la salida EQ se ha conectado a una bobina de Q0.5 y la entrada I(W) se ha conectado a un contacto con operando KW50 (50 décimas de segundo).
Las operaciones que hay que realizar para la escritura del programa AWL son parecidas. Después de haber insertado la sigla del temporizador, hay que teclear los operandos de los que el módulo deberá tomar los datos o a los que el módulo deberá enviar los controles.
Tanto en el programa ladder como en el AWL, la entrada STOP no se utiliza. En el primer caso es suficiente con no conectar nada a la entrada. En el segundo, basta con dejar vacía la línea correspondiente. En ambos casos, la entrada del módulo será automáticamente situada a valor lógico bajo.

Ejemplo 10
Temporizador con retardo a la desactivación

La salida Q0.5 debe activarse a la vez que el cierre de la entrada I0.0 y desactivarse 5 segundos después de su apertura.

Para resolver este problema, dado que este PLC no está dotado de temporizadores de este tipo, es preciso utilizar el único temporizador con que cuenta, retardo a la desactivación, y escribir un programa que modifique su funcionamiento. Observen la figura que aclara el método utilizado para resolver este problema.

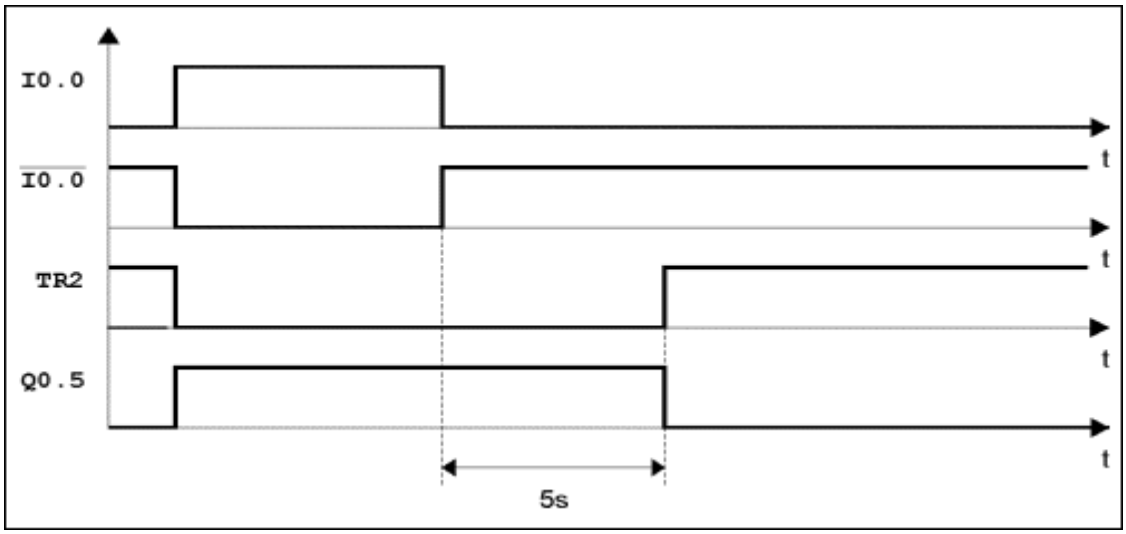
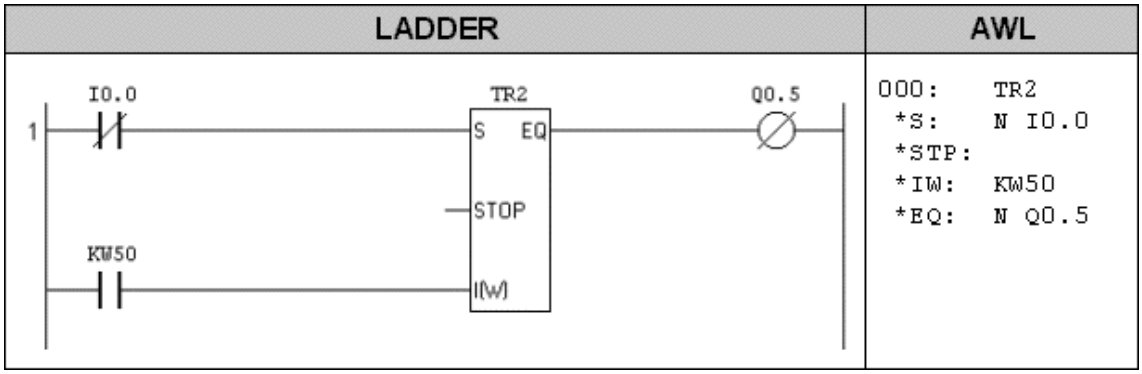


Diagrama temporal del temporizador con retardo a la desactivación

La primera y la última línea del diagrama representan el desarrollo temporal de la entrada y de la salida de un temporizador con retardo a la desactivación: la salida se activa en correspondencia con el flanco ascendente de la entrada y se desactiva, después de un tiempo prefijado, con el flanco descendente de la misma señal.
La segunda línea del diagrama representa la señal que se obtiene complementando la entrada. En cambio, la tercera línea representa la salida de un temporizador con retardo a la activación que tiene como entrada la señal de la línea superior. Observemos como esta señal representa el complemento de la que buscamos. Es decir, como conclusión, un temporizador con retardo a la desactivación se obtiene de uno con retardo a la activación, simplemente negando las señales de entrada y de salida.



En ladder la negación de una entrada se obtiene utilizando un contacto normalmente cerrado (test sobre el estado negado de la señal) y la negación de una salida se obtiene utilizando una bobina inversa. En lenguaje AWL basta con hacer que el indicador de complemento 'N' preceda a los operandos.

Ejemplo 11
Temporizador a impulso

La salida Q0.0 se activa al cierre de la entrada I0.0 y se desactiva 5 segundos más tarde. Si la entrada se reabre durante ese período, la salida queda desactivada inmediatamente.

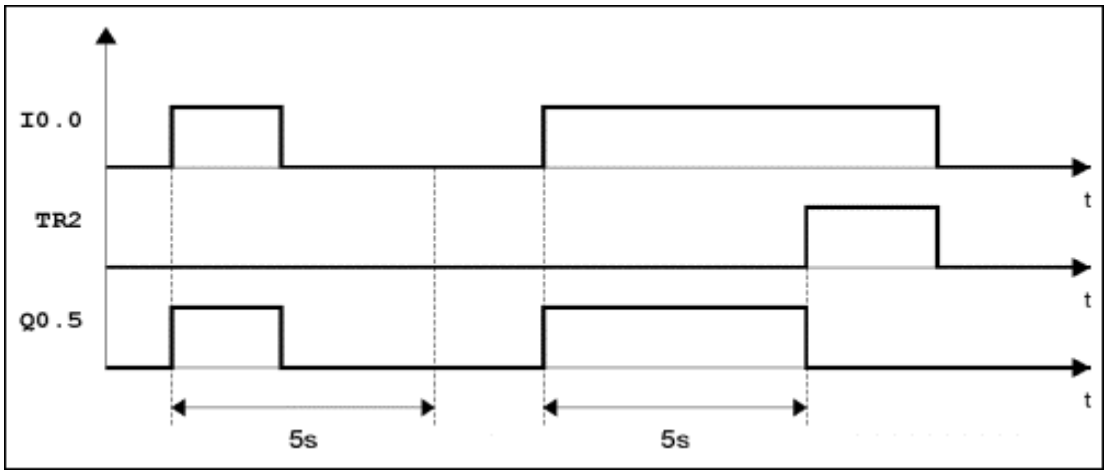
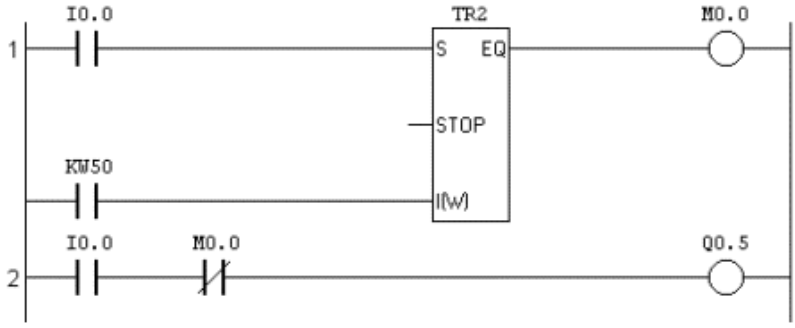


Diagrama del temporizador a impulso

El diagrama temporal de aquí encima indica en la primera línea la entrada del temporizador a impulso y en la última su salida.
También en este caso debemos utilizar el único temporizador que este PLC posee, el retardo a la activación, para conseguir un temporizador del tipo deseado. Con ese propósito se ha construido la línea intermedia del diagrama que representa la salida de un temporizador con retardo a la activación, a cuya entrada se ha conectado la señal de la primera línea.
Observamos pues que la salida Q0.5 debe ser cierta (estado lógico 1) cuando la entrada I0.0 está en el estado alto y, a la vez, la salida del temporizador se encuentra en el estado bajo. En términos de expresión booleana: Q0.5 = I0.0 AND NOT TR2.

LADDER



AWL

```
000:    TR2
 *S:    I0.0
 *STP:
 *IW:   KW50
 *EQ:   M0.0
001:    L    I0.0
002:    AN   M0.0
003:    =    Q0.5
```

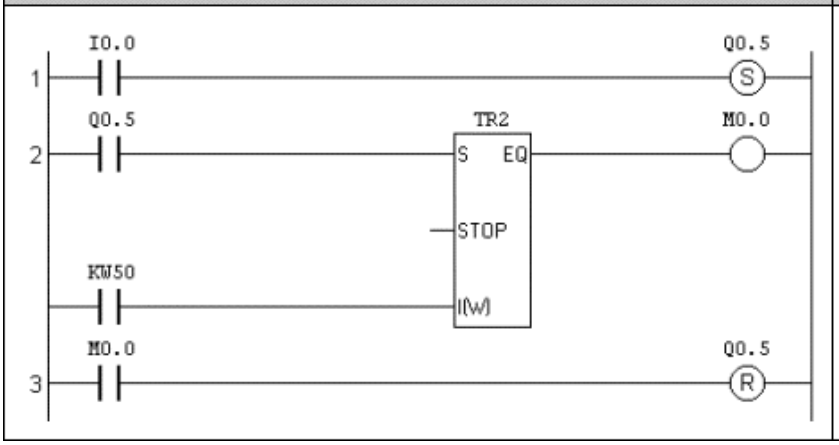
La solución en ladder prevé entonces la escritura de un primer recorrido para la activación de un temporizador con retardo a la activación con entradas (S) I0.0 y (IW) KW50 (5 segundos), la salida está apoyada en un merker bit (M0.0). En el segundo recorrido, la bobina de la salida Q0.5 está pilotada por la serie de la entrada y por el complemento de M0.0 que corresponde a la salida del temporizador.
La solución AWL, aunque presente una escritura diversa, es análoga a la ladder.

Ejemplo 12
Temporizador a impulso prolongado

La salida Q0.5 se activa el cierre de la entrada I0.0 y se desactiva 5 segundos después, independientemente de si la entrada se reabre o no durante ese período.

El funcionamiento del temporizador a impulso prolongado se deduce de la comparación entre la primera y la última línea del diagrama temporal. En la primera línea está representada la señal de entrada y en la última, la correspondiente salida.
También en este caso debemos utilizar el único temporizador que este PLC posee, el retardo a la activación, para obtener un temporizador del tipo deseado. La tarea a realizar es llevar al estado alto (activar) la salida Q0.5 cuando la entrada está a nivel alto y llevarla al estado bajo (desactivarla) cuando haya transcurrido el tiempo.

LADDER



AWL

```
000:    L    I0.0
001:    S    Q0.5
002:    TR2
 *S:    Q0.5
 *STP:
 *IW:   KW50
 *EQ:   M0.0
003:    L    M0.0
004:    R    Q0.5
```

El primer recorrido del diagrama ladder se encarga de la primera tarea: activar la salida cuando la entrada está alta. El temporizador del recorrido siguiente se inicia por medio de un contacto de la propia salida y, pasados 5 segundos, se encarga de activar el merker M0.0. En el último recorrido, el mismo merker lleva de nuevo a 0 la salida Q0.5. El apoyo de la salida de TR2 en M0.0 tan sólo es necesario porque no puede utilizarse la bobina de reset de Q0.5 directamente a la salida del temporizador. Recuerde que, a las salidas de los módulos de sistema, únicamente pueden conectarse bobinas directas o inversas.
La figura que aparece aquí debajo representa el diagrama temporal de los principales operandos presentes en este ejemplo. Además de la entrada y la salida del PLC, representadas respectivamente en la primera y la última línea, en la línea intermedia está representada la salida del temporizador TR2 (que corresponde perfectamente al merker M0.0). Observe como esta señal se mantiene activa un tiempo muy breve, sólo durante un ciclo de ejecución del programa, y que en el diagrama se ha prolongado para hacerla más leíble. En efecto, justo en el momento en que está activa, la salida del temporizador se encarga de la desactivación de Q0.0 pero, ya que este representa su entrada, en la siguiente carga del programa se obtiene incluso la reposición del estado del temporizador.

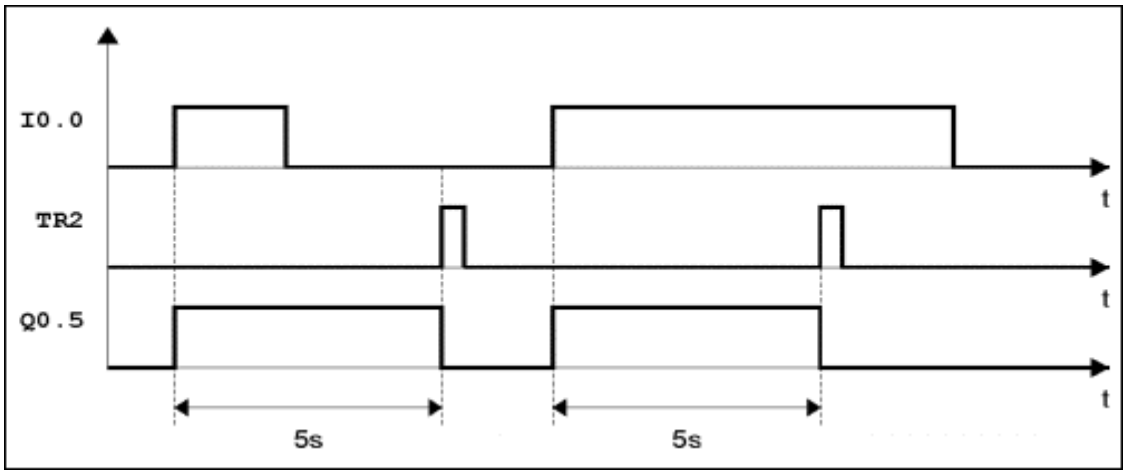


Diagrama temporal del temporizador a impulso prolongado

Observemos por último como, en realidad, en el primer recorrido se continua programando la salida durante todo el tiempo en que la entrada es alta pero el siguiente reset al recorrido 3 sobreescribe la imagen de las salidas, impidiendo que el estado alto en esta condición se propague a la periferia durante la transferencia de la imagen.
Una vez más, el programa AWL, es la fiel traducción del programa ladder.

Ejemplo 13
Temporizador con retardo a la activación con memoria

La salida Q0.5 se activa 5 segundos después del cierre de la entrada I0.0 (aunque si mientras tanto esta última se vuelve a abrir) y se desactiva en correspondencia con el cierre de la entrada I0.1.

El funcionamiento del temporizador con retardo a la activación con memoria se deduce confrontando las dos primeras líneas del diagrama temporal de la Figura 24 con la última. Las primeras representan las señales de entrada y la última se refiere a la correspondiente salida.

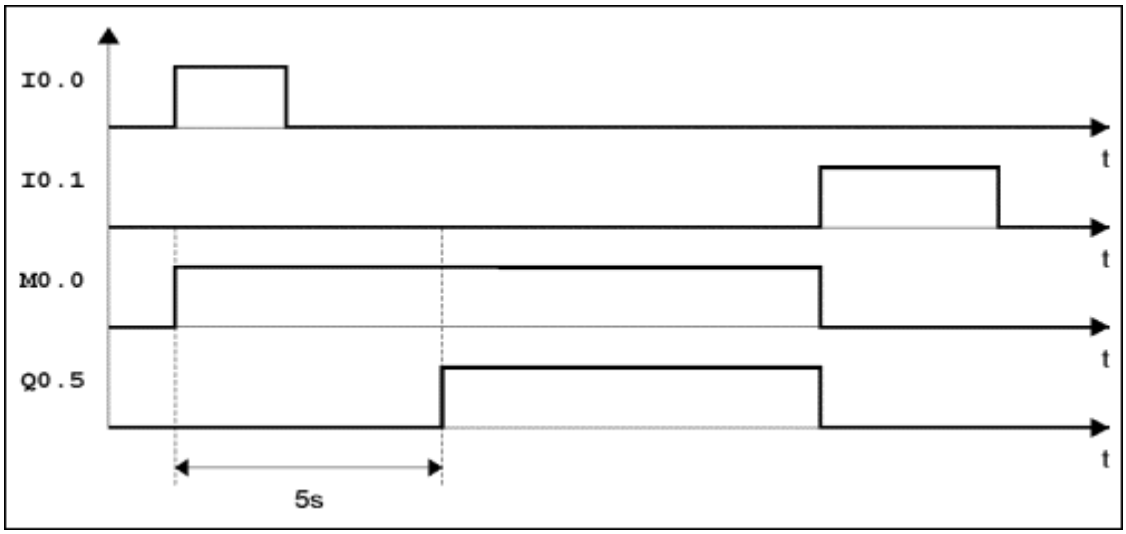
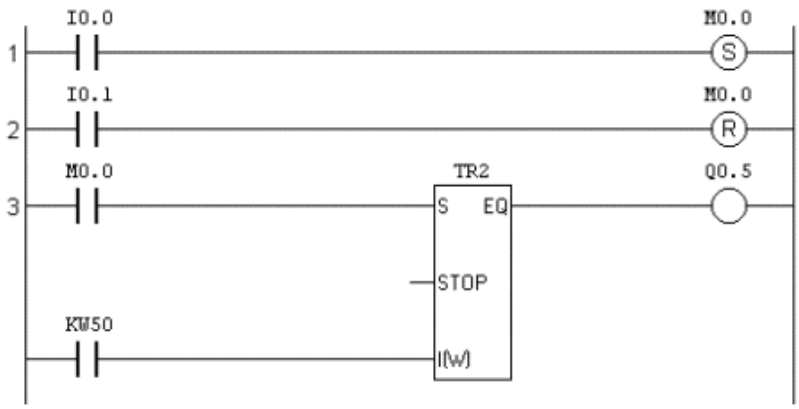


Diagrama temporal de un temporizador con retardo a la activación con memoria

También en este caso debemos utilizar el único temporizador que este PLC posee, el retardo a la activación, para conseguir un temporizador del tipo deseado. Con tal propósito utilizamos un merker que se activa desde I0.0 y se desactiva desde I0.1 y cuyo trazado se indica en la tercera línea del diagrama. Observamos entonces que la salida Q0.5, respecto a este último, no es más que un retardo a la activación: ¡Hemos encontrado la solución!

LADDER



AWL

```
000:    L    I0.0
001:    S    M0.0
002:    L    I0.1
003:    R    M0.0
004:    TR2
 *S:    M0.0
 *STP:
 *IW:   KW50
 *EQ:   Q0.5
```

En el esquema de contactos, los dos primeros recorridos tienen la finalidad de programar el estado del merker M0.0. Desde éste, en el tercer recorrido, se obtiene la salida, por medio de un temporizador con retardo a la activación.
El programa AWL es la traducción del programa ladder.

Ejemplo 14
Temporizador con retardo a la activación y a la desactivación

La salida Q0.7 se activa 2 segundos después del cierre de la entrada I0.1 y se desactiva 7 segundos después de su reapertura.

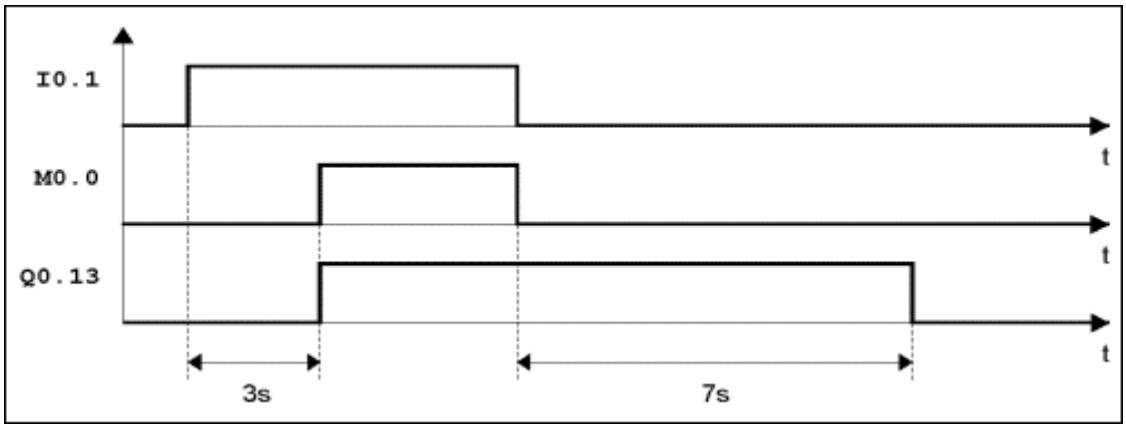
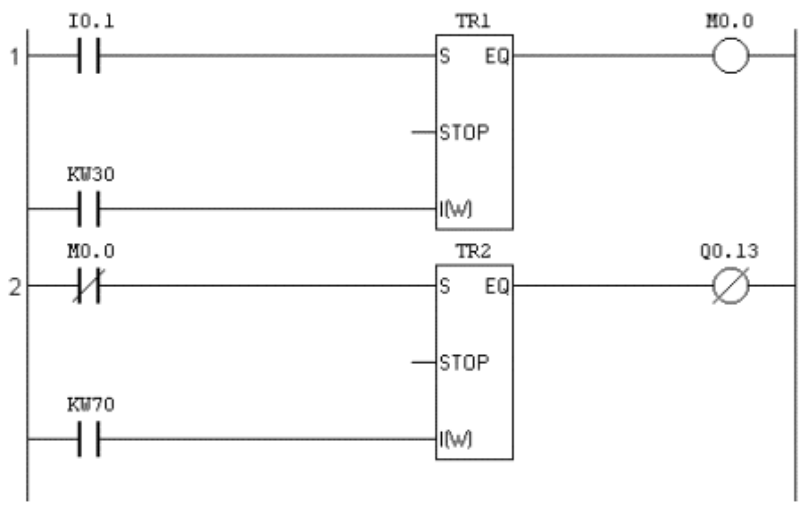


Diagrama de un temporizador con retardo a la activación y a la desactivación

La figura precedente ilustra en la primera y la última línea del diagrama, el proceso de las dos señales del trazado. La línea intermedia indica el proceso de un merker que es la salida de un temporizador con retardo a la activación cuya entrada es precisamente I0.1. Observe que el proceso de la salida Q0.13, respecto al desarrollo del merker, representa un retardo a la desactivación. El temporizador con retardo a la activación está presente en el PLC mientras que, en uno de los ejemplos precedentes, hemos aprendido a construir uno con retardo a la desactivación. ¡Hagan juego, señores! Se trata de escribir un programa que contenga dos temporizadores: el primero, un retardo a la activación, tiene por entrada I0.1 y como salida M0.0. El segundo, un retardo a la desactivación, que tiene como entrada M0.0 y como salida Q0.13.
Los programas ladder y AWL indicados implementan esta solución.

LADDER



AWL

```
000:    TR1
 *S:    I0.1
 *STP:
 *IW:   KW30
 *EQ:   M0.0
001:    TR2
 *S:    N M0.0
 *STP:
 *IW:   KW70
 *EQ:   N Q0.13
```

Ejemplo 15
Impulso retardado

La salida Q0.7 se activa 2 segundos después de la apertura de la entrada I0.13 durante 1 segundo.

El diagrama de la figura siguiente ilustra, en la primera línea, el proceso de la entrada, y en la última, el de la salida del temporizador solicitado. Por su parte, la segunda línea indica el desarrollo del complemento de I0.13. Esta señal se utilizará para poner en marcha dos temporizadores con retardo a la activación TR10 y TR11, en 2 y 3 segundos respectivamente, y cuyas señales de salida quedan reseñadas en las siguientes líneas del diagrama.

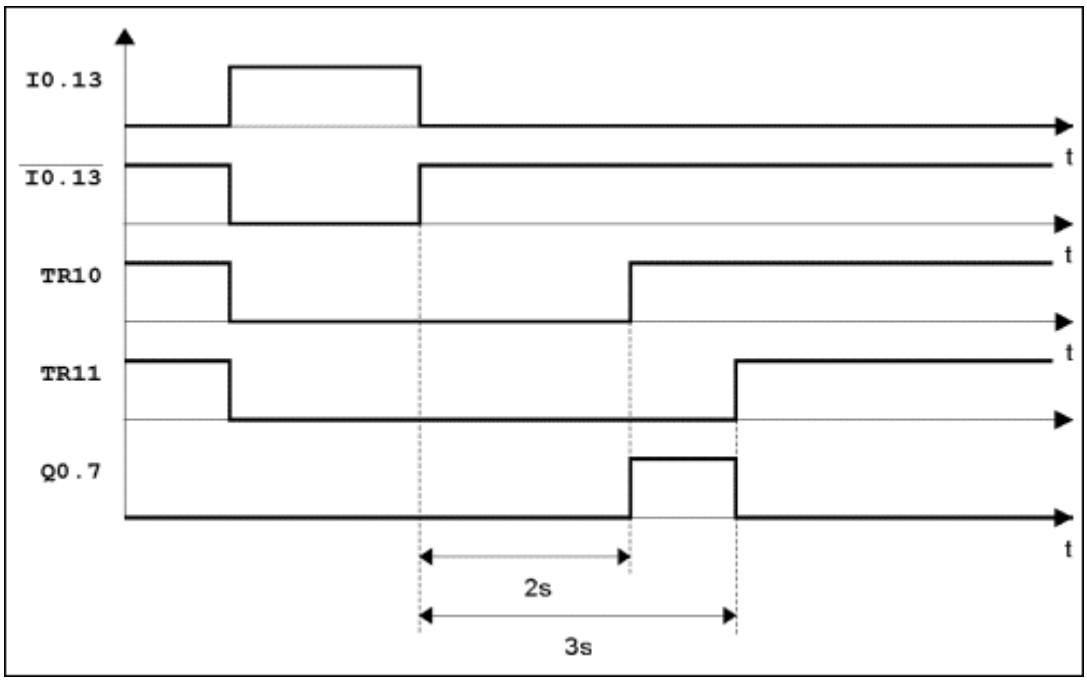
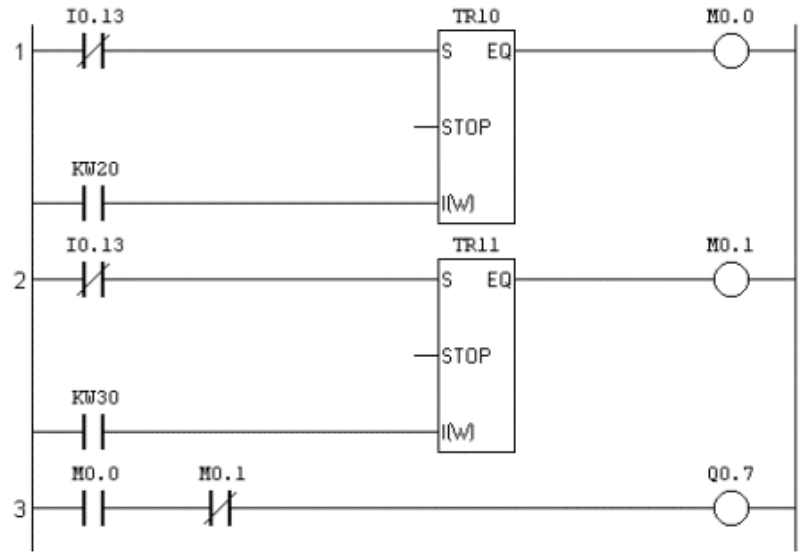


Diagrama temporal del impulso retardado

Se observa entonces que la salida Q0.7 debe ser cierta cuando se dan simultáneamente las siguientes condiciones: salida de TR10 cierta y salida de TR11 falsa , es decir, en términos de expresión booleana:

Q0.7 = TR10 AND NOT TR11.

En el programa ladder, los dos primeros recorridos están dedicados a los temporizadores: ambos tienen por entrada el complemento de I0.13 (contacto NC = test sobre el estado 0); las salidas están apoyadas en dos merker. En el tercer recorrido se implementa la expresión lógica recién obtenida: los dos contactos puestos en serie desarrollan la función de la operación AND. En particular, para el segundo se ha usado el tipo NC con el fin de ejecutar el complemento de su operando.
De manera análoga, en la solución AWL, se requieren y se conectan al programa utilizado los dos temporizadores. Entonces, la secuencia final elabora el valor de la salida.

LADDER	AWL
	000: TR10 *S: N I0.13 *STP: *IW: KW20 *EQ: M0.0 001: TR11 *S: N I0.13 *STP: *IW: KW30 *EQ: M0.1 002: L M0.0 003: AN M0.1 004: = Q0.7

Ejemplo 16
Tren de impulsos

La salida Q0.4 debe activarse un instante a cada segundo.

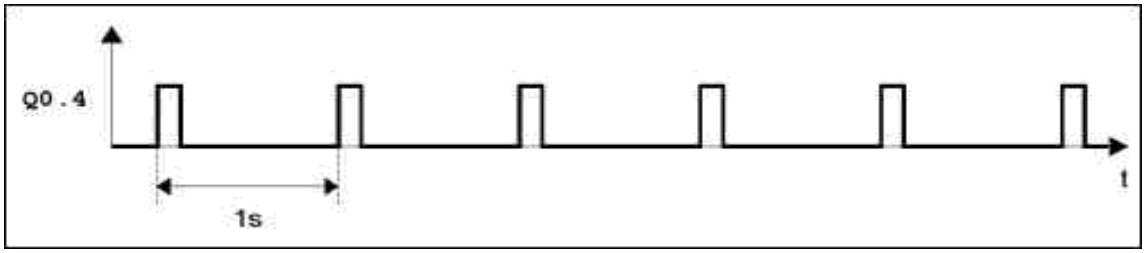
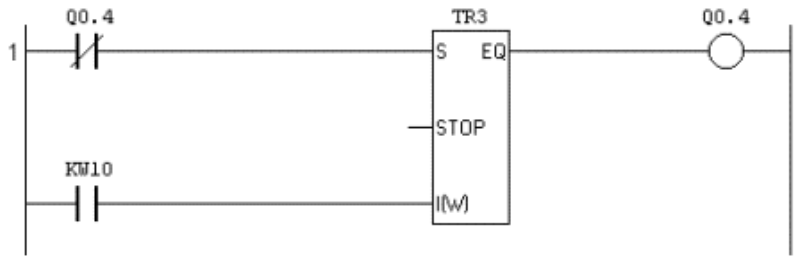


Diagrama temporal del tren de impulsos

El ejercicio se resuelve utilizando un temporizador que se autoarranca cíclicamente.
En el primer ciclo de ejecución, el complemento de Q0.4 está alto. La temporización empieza y, mientras el tiempo va transcurriendo, la salida de TR3 se mantiene baja. Cuando el tiempo llega a cero, la salida de TR3 se hace alta así como el estado de Q0.4 conectado a ella. En el ciclo siguiente, la entrada del temporizador será baja y también lo será su salida.
De ese modo, hemos vuelto a las condiciones iniciales: el funcionamiento descrito se repetirá indefinidamente, generando el tren de impulsos requerido por el trazado.

LADDER	AWL

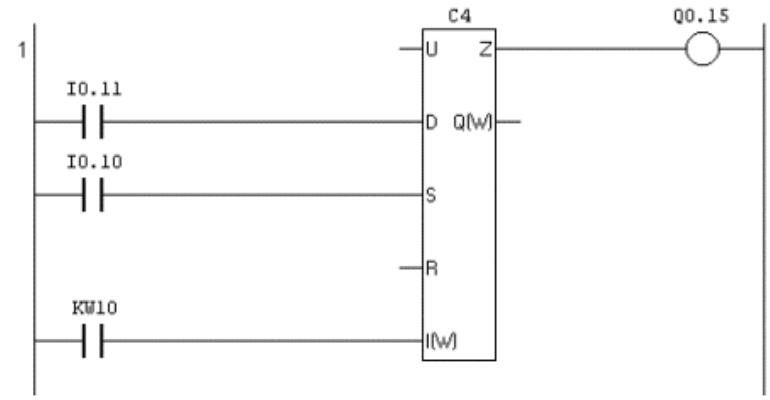
	`000: TR3`
	`*S: N Q0.4`
	`*STP:`
	`*IW: KW10`
	`*EQ: Q0.4`

Ejemplo 17
Conteo hacia atrás

El contador C4 se programa a valor 10 en correspondencia con la activación de la entrada I0.10 y va disminuyendo a cada cierre de la entrada I0.11. La salida Q0.15 se activa en correspondencia con el valor 0 del conteo.

El programa ladder es muy sencillo. La constante KW10, asignada a la entrada IW, facilitará el valor de preselección 10. La entrada de set (S) del contador C4 se conecta a un contacto de I0.10, mientras que el decremento (D) se conecta a un contacto de I0.11. En el flanco de cierre de I0.10 se produce la programación del valor de conteo a 10 y a cada flanco de cierre de I0.11 el conteo va disminuyendo en 1. La salida del contador (Q) está alta cuando el valor de conteo es igual a 0. Por tanto, será suficiente conectar a ésta una bobina de la salida Q0.15 del PLC.

LADDER



AWL

```
000:   C4
 *U:
 *D:   I0.11
 *S:   I0.10
 *R:
 *IW:  KW10
 *QW:  Q0.15
```

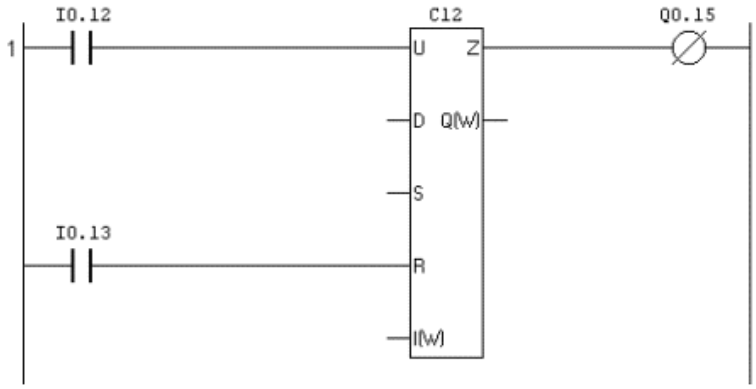
La instrucción 000 del programa AWL constituye una llamada al módulo de sistema C4 que está conectado de una manera análoga a como se hizo en el programa ladder.

Ejemplo 18
Conteo hacia adelante

El contador C12 se incrementa a cada cierre de la entrada I0.12 y es llevado hasta cero en correspondencia con el estado alto de la entrada I0.13. La salida Q0.15 se desactiva cuando el valor de conteo es 0.

La entrada de reset (R) del contador C12 está conectada a un contacto de I0.13, mientras que la de incremento (U) está conectado a un contacto de I0.12. En el flanco de cierre de I0.13 se produce el reset del contador, es decir, la programación del valor de conteo a 0 y a cada flanco de cierre de I0.12, el conteo se incrementa en 1. La salida del contador (Q) está alta cuando el valor de conteo es igual a 0. Por lo tanto, bastará con conectarle una bobina inversa de la salida Q0.15 del PLC para lograr el funcionamiento deseado.

LADDER



AWL

```
000:    C12
 *U:    I0.12
 *D:
 *S:
 *R:    I0.13
 *IW:
 *QW:   N Q0.15
```

La instrucción 000 del programa AWL constituye una llamada al módulo de sistema C12 que está conectado de una manera análoga a como se hizo en el programa ladder.

Ejemplo 19
Conteo del tiempo de cierre de una entrada (en segundos)

Determinar durante cuantos segundos se mantiene cerrada la entrada I0.5 y utilizar la entrada I0.6 para poner a cero el conteo del tiempo.

En primer lugar, es preciso realizar una base de tiempos de un segundo, es decir, un tren de impulsos que tenga este período. Luego será necesario contar cuantos impulsos de la base de tiempos se generan durante el cierre de la entrada, o sea, contar los instantes en los cuales la entrada y el impulso son ciertos a la vez. (ver figura siguiente)

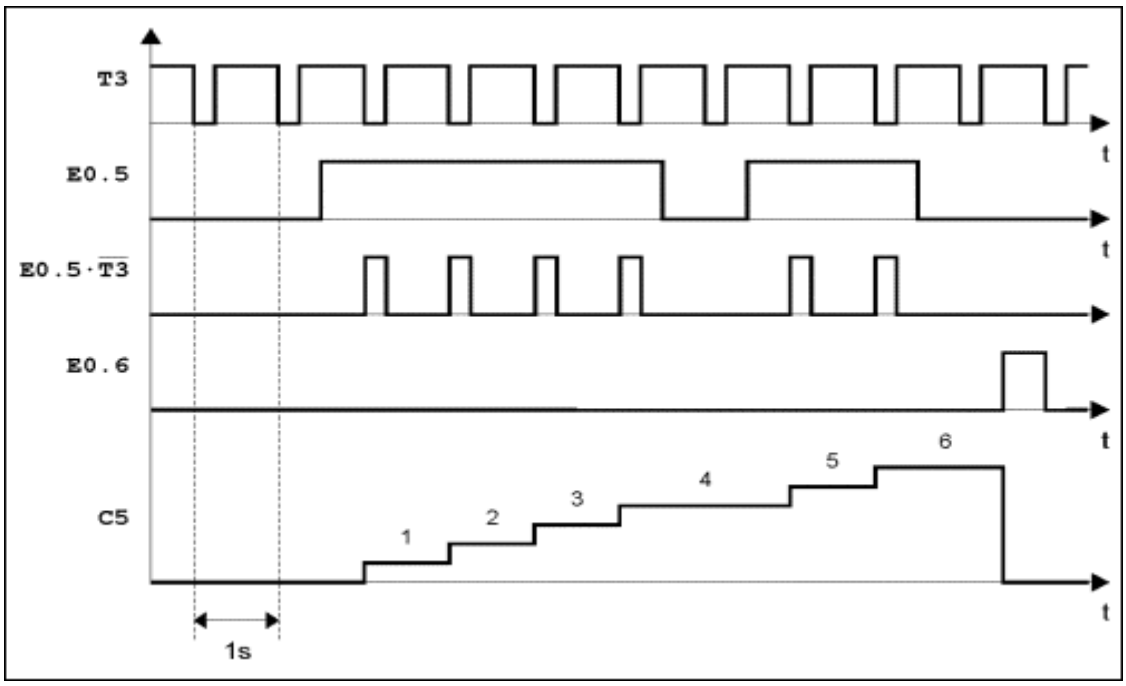
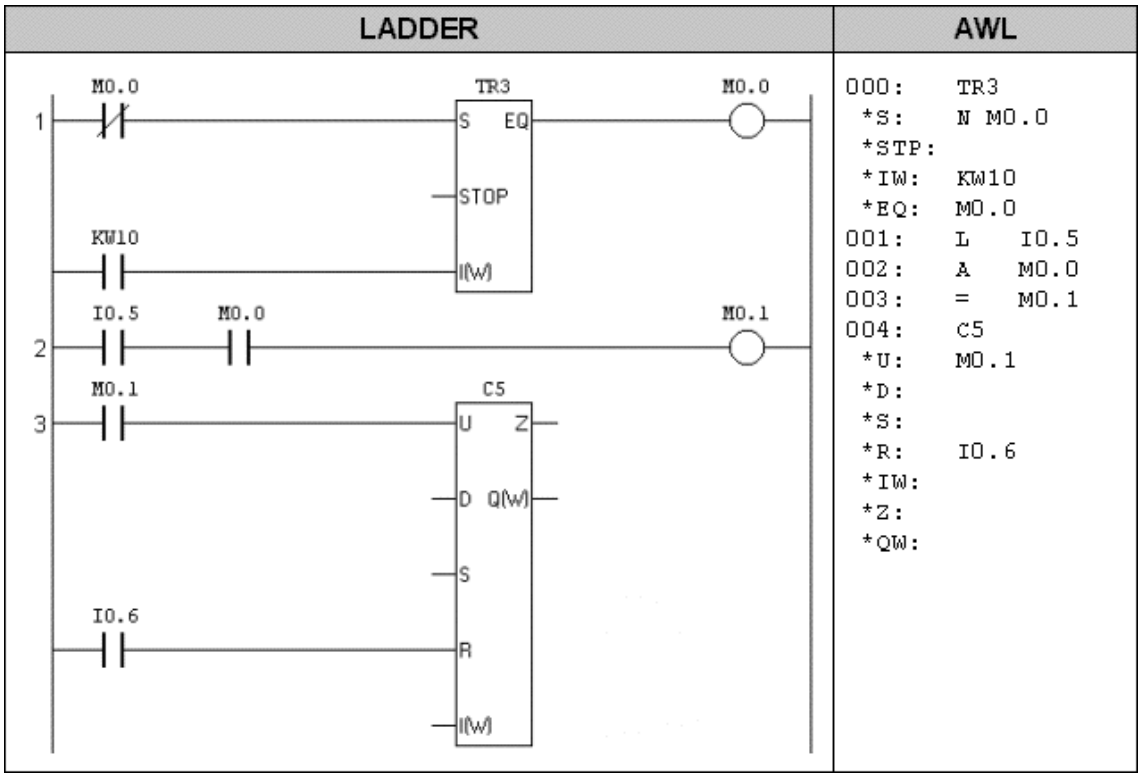


Diagrama temporal del contador del tiempo de cierre de una entrada



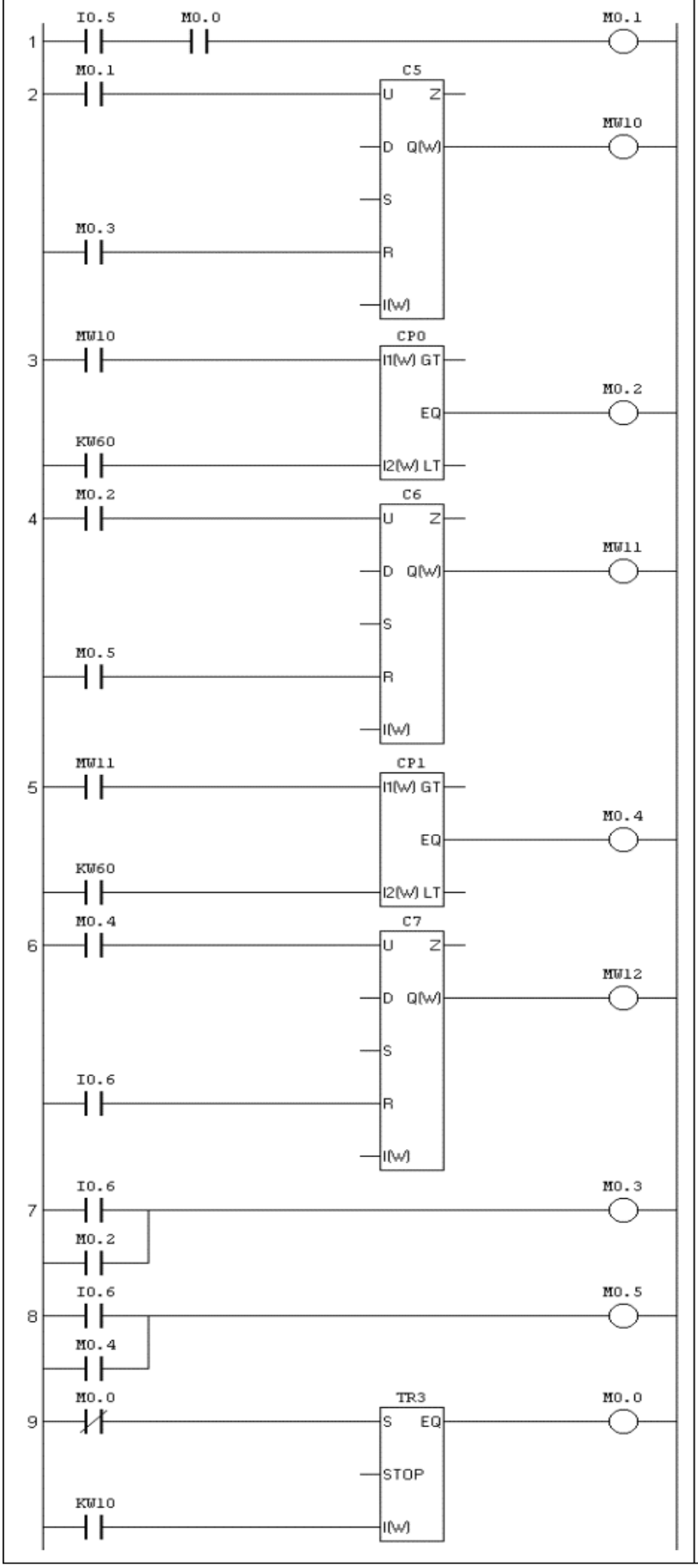
El recorrido 1 del programa ladder genera el tren de impulsos de 1 segundo de período, tal como hemos visto en el Ejemplo 16, pero esta vez utilizamos un merker (M0.0) como entrada/salida del temporizador, ya que no se requiere llevar hasta el exterior del PLC la señal del generador. Así pues, el merker M0.1 representa la combinación lógica AND (serie de contactos en el diagrama) entre la entrada y el tren de impulsos, es decir, justo los impulsos a contar. Este merker se emplea en el recorrido siguiente para pilotar la entrada de conteo hacia adelante del contador C5. Por su parte, la entrada I0.6 pilota la entrada reset del contador para obtener la reposición del conteo, tal como se ha requerido.
El programa AWL realiza las mismas funciones de un modo análogo
El valor del contador C5 representa el número de segundos que la entrada ha estado cerrada, con el limite de 65535 propio de los contadores de este PLC.

Ejemplo 20
Conteo del tiempo de cierre de una entrada (en horas, minutos y segundos)

Determinar cuantas horas, minutos y segundos, la entrada I0.5 se mantiene cerrada y utilizar la entrada I0.6 para poner a cero la cuenta del tiempo.

Los programas propuestos como solución acaban construyendo un tren de impulsos con merker M0.0, que servirá de base de tiempos con un período de 1 segundo. Pero al inicio de los mismos, se carga el valor del merker M0.1, que se programará a 1 en presencia de un impulso de la base de tiempos, cuando el contacto en la entrada I0.5 esté cerrado. Entonces, M0.1 se activa cada segundo cuando la entrada está cerrada: el conteo de los impulsos de este merker nos permitirá valorar el tiempo tal como se requiere en el trazado.
En efecto, la entrada de conteo hacia adelante (U) del contador C5 está pilotada precisamente por M0.1; por lo tanto, C5 constituye el contador de los segundos. El valor actual del conteo, presente en su salida QW, se deposita en la merker word MW10.
A continuación de la llamada de C5, encontramos la del otro módulo de sistema CP0 (comparador) que se emplea para comparar MW10, es decir, el número de segundos contados con la constante numérica 60 (KW60). Cuando las dos cantidades resultan iguales, es decir, cuando se han contado 60 segundos, el merker M0.2 conectado a la salida EQ se pone a 1.
Este último merker pilota la entrada U del contador C6 que de ese modo se verá incrementado cada 60 segundos. Por lo tanto, C6 constituirá el contador de los minutos. El número de minutos contados se deposita en MW11.
El comparador CP1 desempeña una función análoga a CP0: lleva a 1 el merker M0.4 cuando los minutos contados alcanzan el valor 60. El contador C7, que tiene a M0.4 conectado a la entrada U, contará por tanto las horas de cierre del contacto y depositará la cuenta en MW12.
Falta por comprender como los contadores de segundos y minutos se reponen al impulso de entrada que hace 60. Observamos que la entrada de reset (R) de C5 está conectada al merker M0.3 y que éste vale 1 si I0.6 es 1 o si lo es M0.2 y recordamos que este último es la salida del comparador de los segundos. Por tanto, el contador de segundos se pondrá a cero, bien correspondiendo con el cierre de la entrada I0.6, tal como se requiere en el trazado, o bien cuando el conteo de segundos llegue a 60.
Del mismo modo, observando la elaboración del merker M0.5 en el recorrido 8, el contador de los minutos marca de nuevo cero cuando I0.6 se ha cerrado o bien cuando el conteo de los minutos ha alcanzado el valor 60.
Por el contrario, el contador de las horas tan solo se repone una vez se ha cerrado la entrada I0.6.
En resumen, el cuentatiempo de software que hemos construido nos permite contar hasta 65535 horas, 59 minutos y 59 segundos (¡precisos!). Todo ello, partiendo de la base de que pueda considerarse un valor tan preciso en el contexto de un tiempo tan largo si tenemos presentes los inevitables errores de los relojes internos, tanto del PLC real como del PC en el que funciona el simulado.
Como ejercicio, puede intentar la modificación del programa añadiendo un contador de días, que se incremente en 1 cada 24 horas.
Para probar el programa sin tener que esperar tiempos muy largos, puede disminuir la constante de tiempo en la carga de TR3, aumentando así la frecuencia del tren de impulsos, o bien forzar manualmente valores de conteo próximos a los de comparación.

LADDER



AWL

```
000:    L    I0.5
001:    A    M0.0
002:    =    M0.1
003:    C5
 *U:    M0.1
 *D:
 *S:
 *R:    M0.3
 *IW:
 *Z:
 *QW:   MW10
004:    CP0
 *I1W:  MW10
 *I2W:  KW60
 *GT:
 *EQ:   M0.2
 *LT:
005:    C6
 *U:    M0.2
 *D:
 *S:
 *R:    M0.5
 *IW:
 *Z:
 *QW:   MW11
006:    CP1
 *I1W:  MW11
 *I2W:  KW60
 *GT:
 *EQ:   M0.4
 *LT:
007:    C7
 *U:    M0.4
 *D:
 *S:
 *R:    I0.6
 *IW:
 *Z:
 *QW:   MW12
008:    L    I0.6
009:    O    M0.2
010:    =    M0.3
011:    L    I0.6
012:    O    M0.4
013:    =    M0.5
014:    TR3
 *S:    N M0.0
 *STP:
 *IW:   KW10
 *EQ:   M0.0
```

Ejemplo 21
Generador de onda cuadrada

La salida Q0.7 debe controlarse mediante una señal de onda cuadrada con Ton = 0.5s y Toff = 1.5s.

En la solución propuesta se utilizan dos temporizadores que se "rebotan" la activación. El merker M0.0, inicialmente inactivo, pone en marcha el temporizador TR1 por un tiempo de 1.5 seg.. Al acabarse el tiempo, la salida Q0.7 se activa por medio de TR1. El contacto de Q0.7 pone en marcha entonces el temporizador TR2 con un tiempo de 0.5 seg. durante los cuales Q0.7 se mantiene activo. Finalizado este tiempo, el merker M0.0 se programa a 1 y al siguiente ciclo de ejecución, la entrada S de TR1 se lleva a nivel 0 , al igual que su salida. También la entrada S de TR2 se lleva a nivel lógico bajo, y su salida y M0.0 con él.
Hasta ahora, la salida Q0.7 ha permanecido inactiva durante 1.5 seg. y activa 5 seg.: hemos analizado un período completo de la señal. Pero, en este punto, el sistema se ha conducido de nuevo hasta sus mismas condiciones iniciales, tal como muestra el diagrama de la siguiente figura. De forma cíclica, todo el conjunto se reinicia de nuevo, realizando el generador deseado.
Programando oportunamente el valor de las dos constantes de word se puede variar Ton y Toff, realizando una onda cuadrada con frecuencia y ciclo de trabajo distintos.

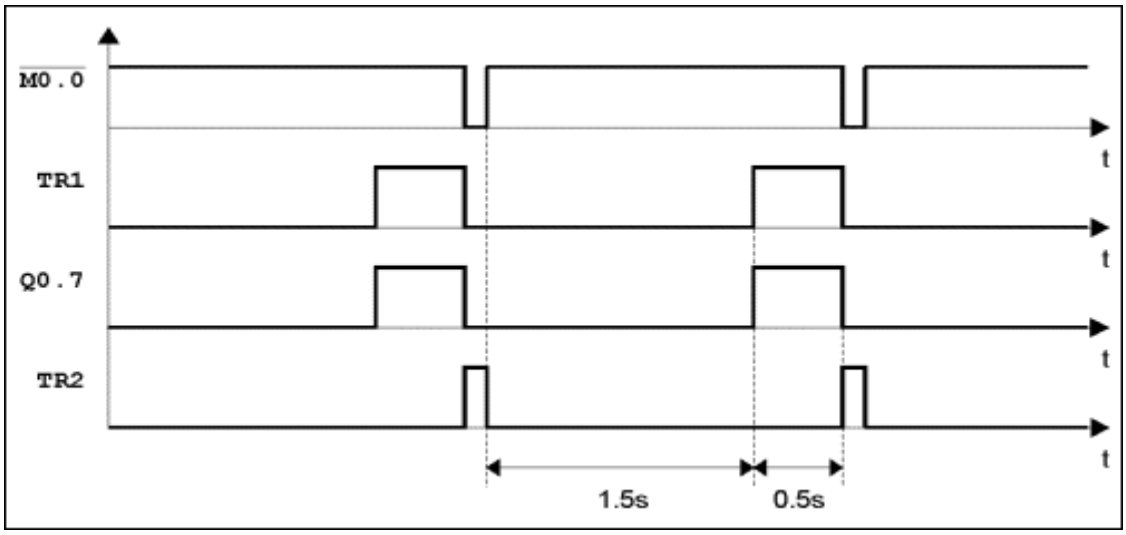
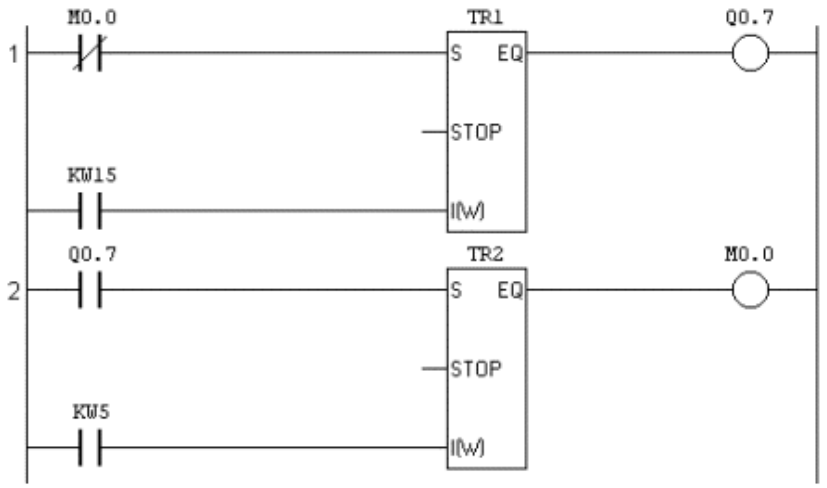


Diagrama temporal de un generador de onda cuadrada

LADDER



AWL

```
000:    TR1
 *S:    N M0.0
 *STP:
 *IW:   KW15
 *EQ:   Q0.7
001:    TR2
 *S:    Q0.7
 *STP:
 *IW:   KW5
 *EQ:   M0.0
```

Ejemplo 22
Otro generador de onda cuadrada

La salida Q0.7 debe estar controlada por una señal de onda cuadrada con Ton = 0.5s y Toff = 1.5s.

La solución que ahora se presenta es diferente de la propuesta en el ejemplo anterior, para el mismo trazado. El temporizador TR1 se usa para generar un tren de impulsos con un período de dos segundos. En la primera línea de la siguiente figura se detalla el desarrollo de su salida y en el segundo, el complemento del merker M0.0 conectado a ella.
Esta señal se aplica a la entrada de un temporizador con retardo a la activación, a cuya salida está conectada una bobina de Q0.7. Esta señal, cuyo desarrollo se ilustra en la última línea del diagrama, representa la solución al problema.
También en este caso, programando adecuadamente las dos constantes de tiempo, se puede variar la frecuencia y el ciclo de trabajo de la onda cuadrada.

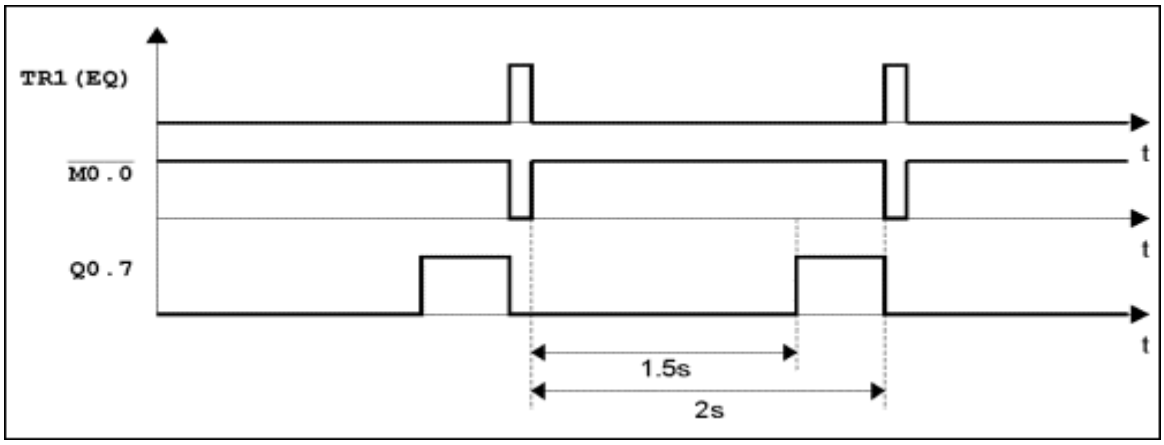
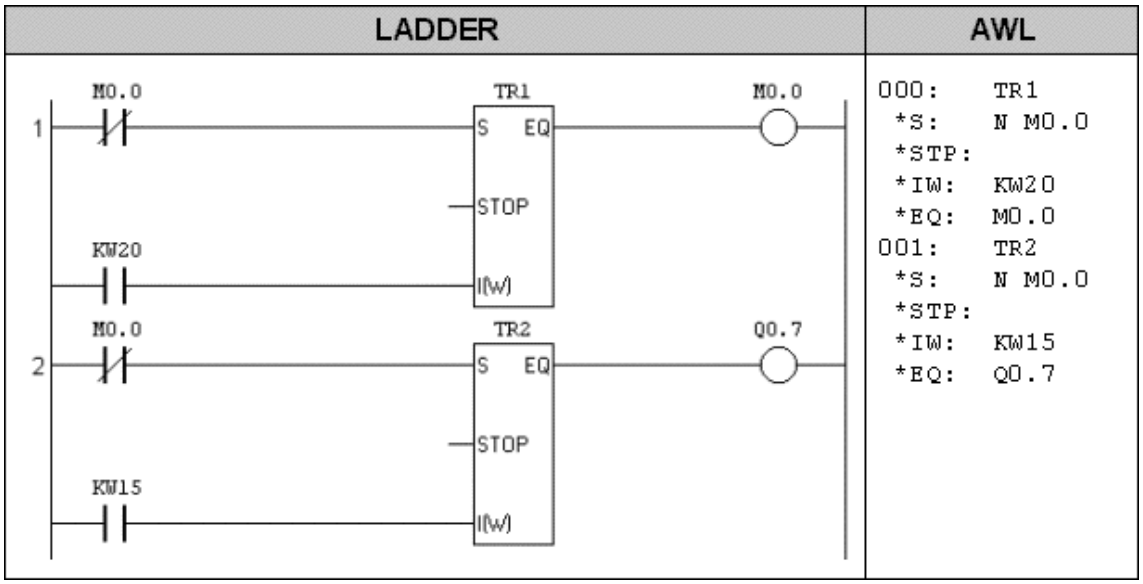


Diagrama temporal del generador de onda cuadrada.



Ejemplo 23
Control temporizado de luces

Un pulsador conectado a la entrada I0.0 activa durante tres minutos un grupo de luces conectadas a la salida Q0.1. Junto con éstas se activa un piloto luminoso conectada a la salida Q0.2 que, 15 segundos antes de que se apaguen las luces empieza a parpadear para avisar de la inminente finalización del tiempo. El piloto se apaga a la vez que las luces.

Se utilizan dos temporizadores conectados para funcionar como retardo a la desactivación. El primero, cargado con un tiempo de 3 minutos (KW1800 , es decir, 1800 décimas de segundo), controla directamente la salida del grupo de luces. El segundo, cargado con un tiempo inferior en 15 segundos (KW1650), activa un merker de apoyo.

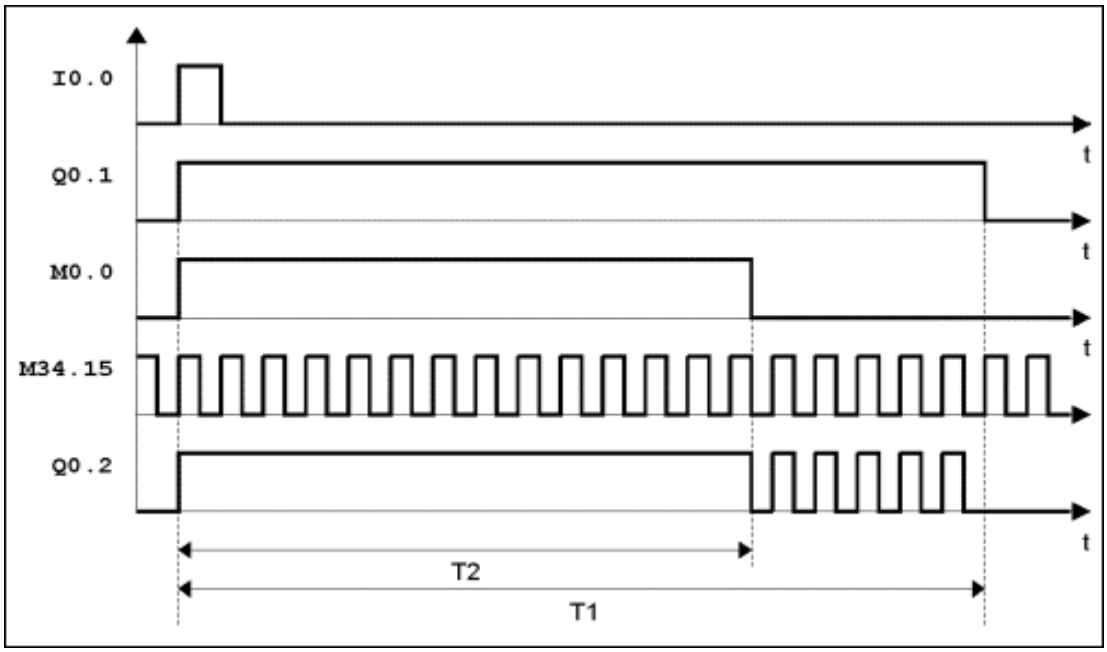
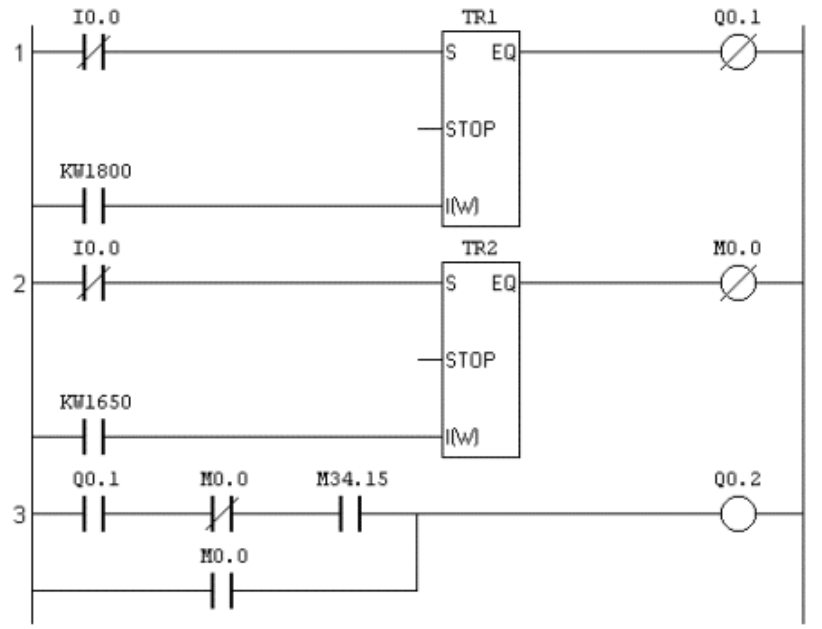


Diagrama temporal del control de luces temporizado

Los desarrollos temporales de Q0.1 y M0.0, a continuación del impulso sobre I0.0, se detallan en la segunda y tercera línea del diagrama. La cuarta línea indica el trazado, no en escala del merker M34.15, el cual actúa como multivibrador astable con un período de 2 segundos. Este merker volverá a ser útil para el parpadeo del piloto luminoso.
Observando la quinta línea del diagrama se observa que el piloto indicador debe permanecer o bien cuando está activo M0.0 o bien cuando están activos Q0.1 y M34.15 y simultáneamente M0.0 está inactivo. Es decir, en términos de función booleana:

Q0.2 = M0.0 OR (Q0.1 AND NOT M0.0 AND M34.14)

Las soluciones propuestas implementan con precisión todo cuanto se ha descrito.
En la primera, realizada en ladder, se utilizan los dos primeros recorridos para dirigir los temporizadores con retardo a la desactivación. El ultimo recorrido está dedicado a la construcción de la lógica para el funcionamiento del piloto indicador.
En el programa AWL, las dos primeras instrucciones relacionan las llamadas a los módulos de sistema de temporizadores y las siguientes implementan la lógica de funcionamiento del indicador luminoso.

LADDER	AWL
	000: TR1 *S: N I0.0 *STP: *IW: KW1800 *EQ: N Q0.1 001: TR2 *S: N I0.0 *STP: *IW: KW1650 *EQ: N M0.0 002: L Q0.1 003: AN M0.0 004: A M34.15 005: O M0.0 006: = Q0.2

Ejemplo 24
Divisor de frecuencia (x4)

Realizar un divisor de frecuencia por 4: cada cuatro impulsos en la entrada se activa un impulso en la salida A2.1.

La primera parte de las soluciones propuestas realiza un tren de impulsos, tal como hemos aprendido a hacer en el Ejemplo 16, y muestra la señal en la salida A2.0. La segunda parte implementa efectivamente el divisor que no es otra cosa que un contador que va disminuyendo a cada impulso del generador (entrada ZR pilotado por A2.0) y que, cuando llega a cero, se autoprograma a valor 4 (salida Q llevada sobre la entrada S a través de M0.0).
Cada 4 impulsos de A2.0, en un sólo ciclo de ejecución, el conteo vuelve a cero. En este ciclo, la salida del contador se desactiva y también el merker M0.0 conectado a ella. Por el contrario, la salida A2.1 del PLC se activa, a causa de la negación del contacto de M0.0 que la dirige.

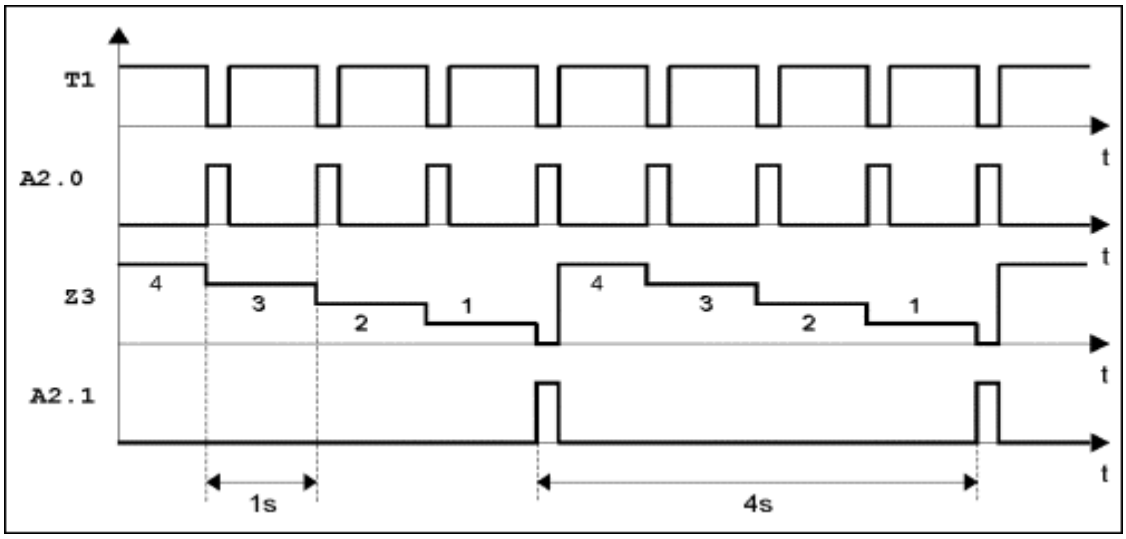
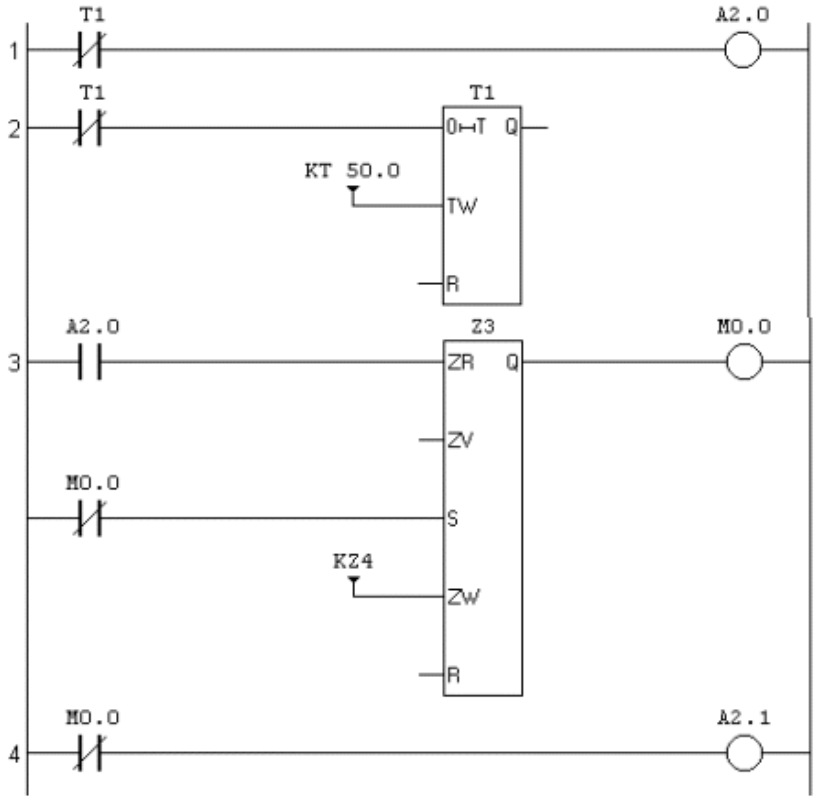


Diagrama temporal del divisor de frecuencia por 4

LADDER



AWL

```
: UN  T1                    'conteo impulsos
: ZR  Z3
: =   A2.0                  'activación salida cada segundo

: UN  Z3
: =   A2.1                  'activación salida cada 2 segundos
: L   KZ 4                  'restaurar contador a 4
: S   Z3

: UN  T1                    'construcción tren de impulsos
: L   KT 50.0               'a dividir
: SA  T1

: BE
```

Ejemplo 25
Semáforo para Formula 1

Con la activación del pulsador conectado a la entrada E0.0, las cinco luces de un semáforo deben encenderse una tras otra, una a cada segundo. Al cabo de un segundo del encendido completo, las luces deberán apagarse.
Para programar la solución a este problema se ha hecho uso de símbolos. Su correspondencia con los operandos absolutos se ha establecido según la tabla siguiente.

Op. absoluto	Simbolo	Comentario
E0.0	START	Pulsador de START de la secuencia
A2.0	L1	Lámpara 1
A2.1	L2	Lámpara 2
A2.2	L3	Lámpara 3
A2.3	L4	Lámpara 4
A2.4	L5	Lámpara 5

El diagrama temporal siguiente muestra, en las líneas intermedias, el desarrollo de las salidas del PLC que controlan las luces del semáforo, en función de la entrada START indicada en la primera línea.

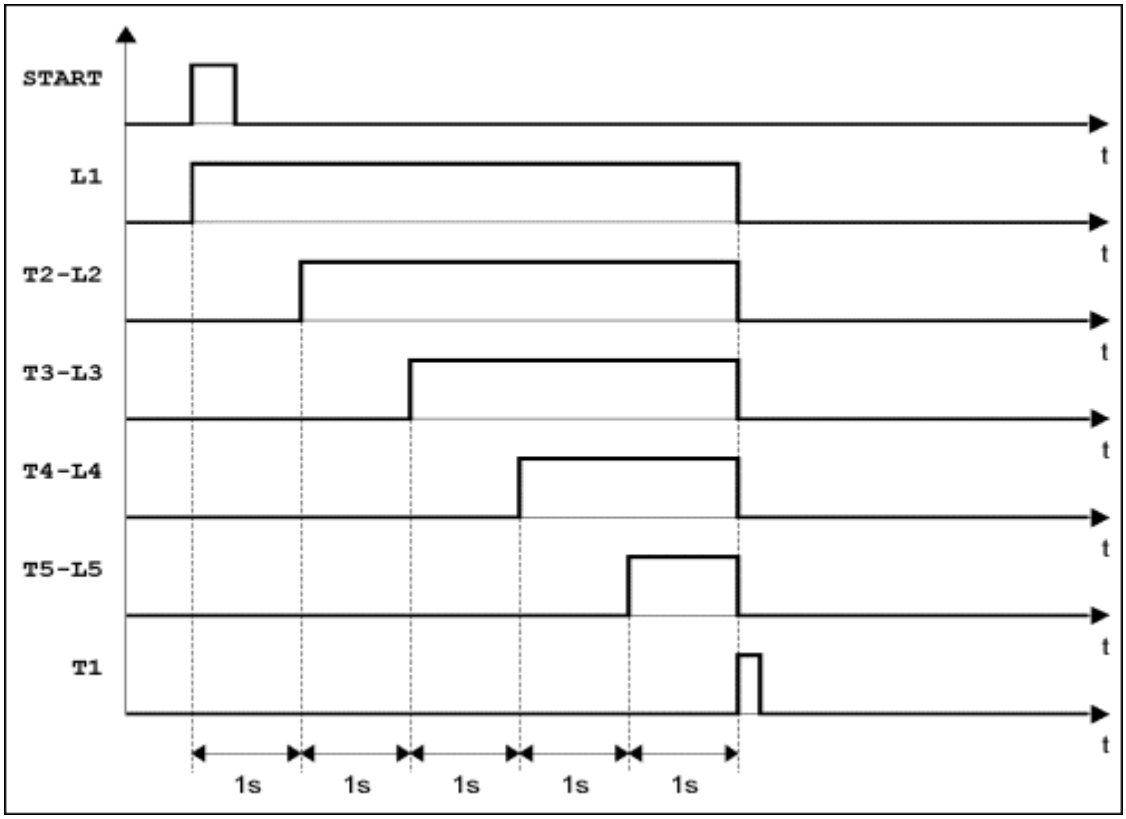
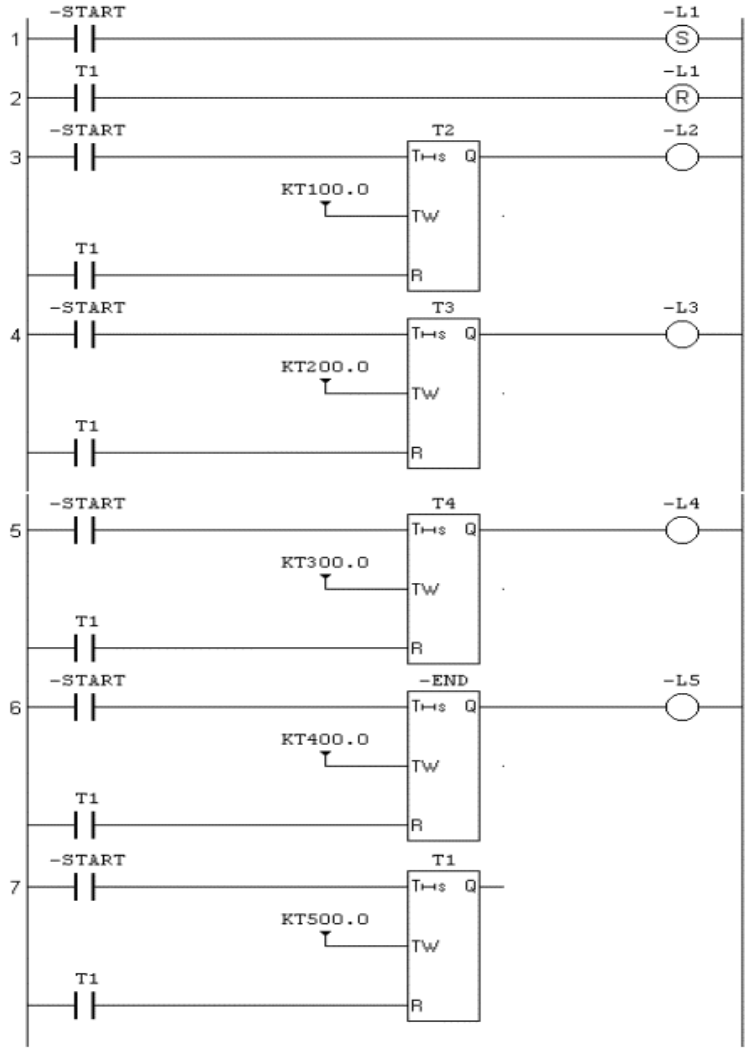


Diagrama temporal para un semáforo de Fórmula 1

El cierre de esta entrada, además de activar la salida que controla la primera luz, que se enciende inmediatamente, activa cinco temporizadores del tipo retardo a la activación con memoria, con tiempos de 1 a 5 segundos. La salida de cada uno de ellos, T1 excluido, una vez transcurrido el tiempo programado, se llevará al estado alto, activando la correspondiente luz y obteniendo con facilidad la secuencia de encendido. Por su parte, el temporizador T1, se encarga del apagado de todas las luces desactivando L1 y todos los demás temporizadores, incluido él mismo.

Ponga en Run el PLC, trasforme el interruptor 0.1 en un pulsador, accione el pulsador y …¡que gane el mejor!

LADDER

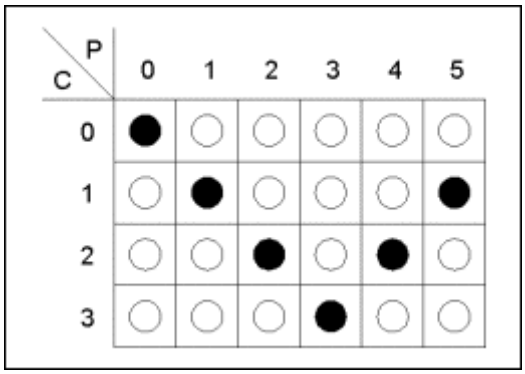


AWL

```
: U    -START                'programación temporizadores por secuencia
: L    KT 100.0
: SS   T2
: L    KT 200.0
: SS   T3
: L    KT 300.0
: SS   T4
: L    KT 400.0
: SS   T5
: L    KT 500.0
: SS   T1

: U    -START                'programación salidas
: S    -L1
: U    T2
: =    -L2
: U    T3
: =    -L3
: U    T4
: =    -L4
: U    T5
: =    -L5

: U    T1                    'reset temporizadores
: R    -L1
: R    T1
: R    T2
: R    T3
: R    T4
: R    T5

: BE
```

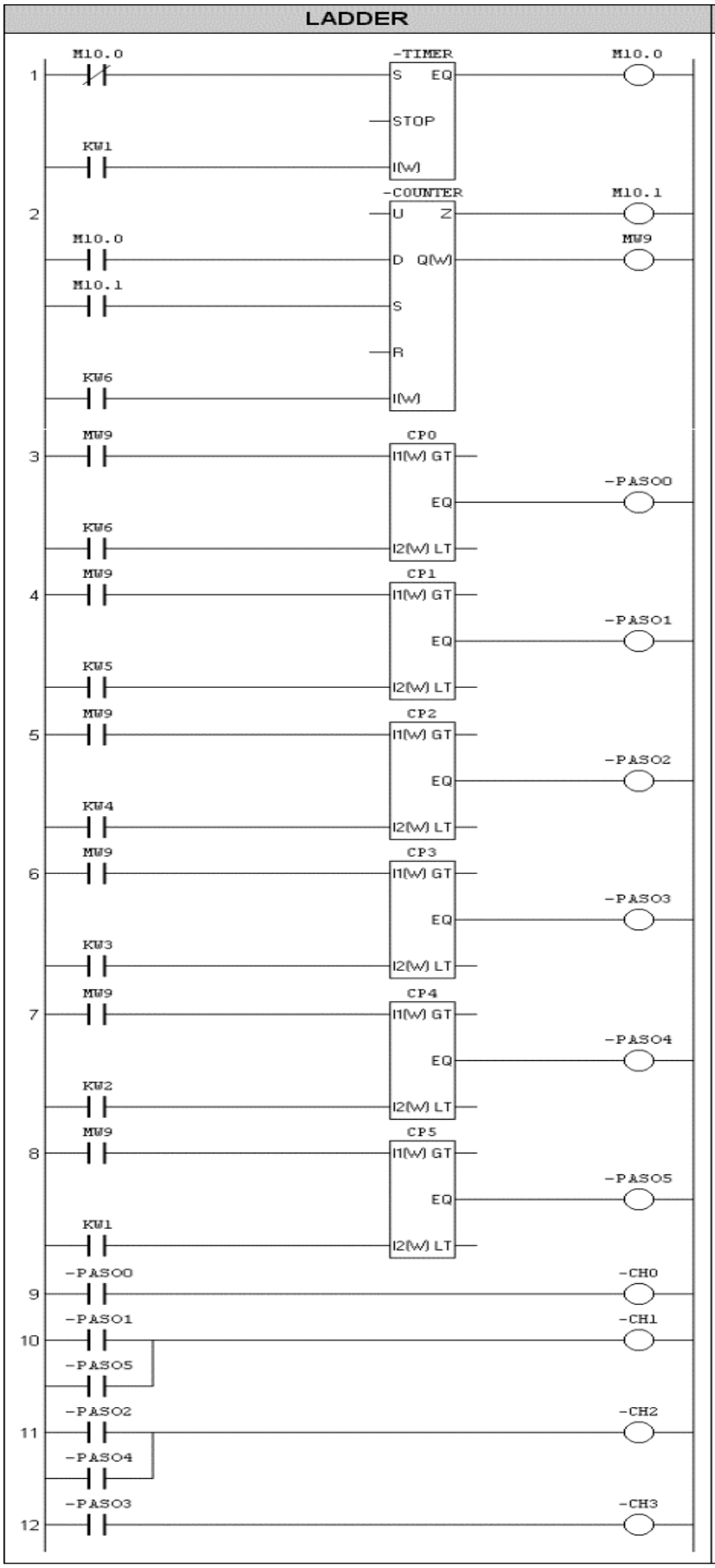
Ejemplo 26
Luces secuenciales de 4 canales

Construir un secuenciador de 4 canales que prevea el siguiente esquema de encendido.



Esquema de encendido para un secuenciador de 4 canales

El esquema establece la secuencia de encendido de las luces conectadas a cuatro canales. Los círculos negros indican la activación del canal durante su paso específico. Así, durante el paso 0 estará activo el canal 0, durante el paso 1 el canal 1, y así sucesivamente. Si las luces se colocan en línea, el efecto será el de un desplazamiento de la fuente luminosa desde la primera hasta la última posición y luego al contrario. Al paso 6 le sucede un paso 7 idéntico al 0 y luego otro idéntico al paso 1, es decir, el diagrama se va recorriendo cíclicamente. Imaginen que lo recortan y lo enroscan formando un cilindro y hacen coincidir los límites opuestos del paso 0 y del 5, sería algo similar al tambor de un carillón: cuando acaba de tocar su musiquilla, empieza de nuevo.
Continuando con nuestro símil sonoro, para que un carillón funcione necesitamos un cilindro con unas levas dispuestas de un modo adecuado sobre su superficie lateral y un mecanismo que lo haga girar. Empezaremos por construir este último. El mecanismo de avance de nuestro secuenciador será un tren de impulsos con período de 0.2 segundos, es decir, una base de tiempo con un período elegido arbitrariamente. EI temporizador TIMER se ocupa de esta función de la manera en que hemos aprendido a hacerlo en los ejercicios anteriores.
El cilindro estará formado por un contador que, partiendo del valor 6, va decreciendo a cada impulso. La salida de TIMER se envía, por medio del merker M10.0, a la entrada D del contador COUNTER. El valor de conteo actual se deposita en la merker word MW9. La salida del contador se reconduce a la entrada de set por medio del merker M10.1 y a continuación, éste se autoprograma a 6 en cuanto llega a 0.
Antes de colocar las levas en el cilindro, identificamos los tramos del cilindro que corresponden a cada paso específico. El comparador CP0 programa a 1 el merker PASO0 cuando el contador vale 6, identificando entre los posibles valores del contador el que corresponde a dicho paso. Los siguientes comparadores programan el merker correspondiente a cada uno de los demás pasos. Así, al final, a cada 0.2 seg. estará activo un merker distinto, yendo a 1 a continuación de PASO0 a PASO5 y, luego, empezando de nuevo, desde PASO0.
Ahora que ya hemos identificado las posiciones podemos insertar las levas. Empezamos desde el canal 0; observamos de nuevo la parrilla de la figura, el canal 0 está activado sólo durante el paso 0, es decir CH0=PASO0.
El canal 1 debe estar activo tanto durante el paso 1 como durante el paso 5, es decir, CH1=PASO1+PASO5
Proseguimos así para los otros dos canales hasta terminar el carillón o, dejando ya a un lado el símil didáctico, el secuenciador.
En este ejemplo se puede aumentar o disminuir la duración de los pasos simplemente cambiando la constante con la que se carga el temporizador, produciendo el efecto de variar la velocidad del desplazamiento aparente de la fuente luminosa. Se puede modificar el número de pasos, cambiando la constante con la cual se carga el contador y añadiendo otros elementos para discriminar los pasos añadidos. También es posible cambiar la secuencia de encendido de las luces, modificando las condiciones en los grupos de OR que constituyen la última parte del programa.


LADDER
AWL
M10.0
-TIMER
S EQ
STOP
KW1
I(W)
M10.0
-COUNTER
U Z
M10.1
M10.0
D Q(W)
MW9
M10.1
S
R
KW6
I(W)
MW9
CP0
I1(W) GT
EQ
-PASO0
KW6
I2(W) LT
MW9
CP1
I1(W) GT
EQ
-PASO1
KW5
I2(W) LT
MW9
CP2
I1(W) GT
EQ
-PASO2
KW4
I2(W) LT
MW9
CP3
I1(W) GT
EQ
-PASO3
KW3
I2(W) LT
MW9
CP4
I1(W) GT
EQ
-PASO4
KW2
I2(W) LT
MW9
CP5
I1(W) GT
EQ
-PASO5
KW1
I2(W) LT
-PASO0
-CH0
-PASO1
-CH1
-PASO5
-PASO2
-CH2
-PASO4
-PASO3
-CH3
000: -TIMER
*S: N M10.0
*STP:
*IW: KW1
*EQ: M10.0
001: -COUNTER
*U:
*D: M10.0
*S: M10.1
*R:
*IW: KW6
*Z: M10.1
*QW: MW9
002: CP0
*I1W: MW9
*I2W: KW6
*GT:
*EQ: -PASO0
*LT:
003: CP1
*I1W: MW9
*I2W: KW5
*GT:
*EQ: -PASO1
*LT:
004: CP2
*I1W: MW9
*I2W: KW4
*GT:
*EQ: -PASO2
*LT:
005: CP3
*I1W: MW9
*I2W: KW3
*GT:
*EQ: -PASO3
*LT:
006: CP4
*I1W: MW9
*I2W: KW2
*GT:
*EQ: -PASO4
*LT:
007: CP5
*I1W: MW9
*I2W: KW1
*GT:
*EQ: -PASO5
*LT:
008: L -PASO0
009: = -CH0
010: L -PASO1
011: O -PASO5
012: = -CH1
013: L -PASO2
014: O -PASO4
015: = -CH2
016: L -PASO3
017: = -CH3

Ejemplo 27
Conteo de entradas cerradas

Contar el número de las entradas cerradas entre las primeras 8

El conteo de las entradas cerradas está contenido en MB20.0 que se inicializa a 0 al comienzo del programa. MB10.0 representa en cambio una máscara de 8 bit, los cuales estarán, solo de uno en uno, a 1. El valor inicial es 1, es decir, (00000001)2 donde tan sólo el bit 0 es cierto.
Las instrucciones 004 y 005 ejecutan la AND bit a bit de la máscara y del byte de entrada menos significativo del PLC. Al primer paso, con el valor de la máscara apenas visto, la AND da un resultado distinto de 0 sólo si E0.0 está cerrado. Es decir, la derivación sobre cero prevista por la siguiente instrucción solo se efectúa por entrada abierta.
Si la entrada está cerrada, incrementamos el byte de conteo MB20.0, cargamos su valor en el registro general (L MB20.0), sumando 1 (ADD KB1) y transferimos el resultado de nuevo a MB20.0 (= MB20.0).
En uno u otro caso, los dos recorridos de elaboración se reúnen en la etiqueta 010 donde, tras haber cargado la máscara en el registro general de byte (L MB10.0), se dispone el desplazamiento hacia la izquierda de una posición, que equivale a una multiplicación por 2 (MUL KB2). El resultado del desplazamiento se retransfiere nuevamente a MB10.0. La máscara vale ahora 2, es decir, (00000010)2, donde sólo el bit 1 es cierto y, siendo distinta de 0, la operación de desplazamiento sobre 0 (BNZ 4) se ejecuta y la elaboración continua desde la instrucción 004, examinando la entrada siguiente.
Después de 8 desplazamientos, el bit que poco a poco se ha ido trasladando en la máscara, sale por la izquierda y MB10.0 se convierte en 0. En esta condición, la última operación de derivación se ignora y el ciclo termina.

LADDER	AWL
	000: L KB0 001: = MB20.0 002: L KB1 003: = MB10.0 004: L MB10.0 005: A IB0.0 006: BZ 10 007: L MB20.0 008: ADD KB1 009: = MB20.0 010: L MB10.0 011: MUL KB2 012: = MB10.0 013: BNZ 4