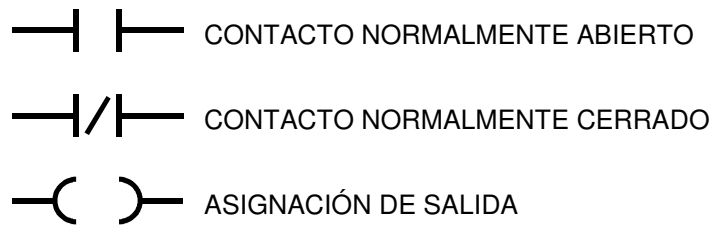


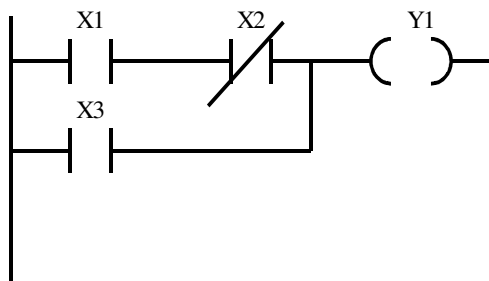
Diagrama de contactos (Ladder)

Es un lenguaje gráfico, derivado del lenguaje de relés. Mediante símbolos representa contactos, bobinas, etc. Su principal ventaja es que los símbolos básicos están normalizados según el estándar IEC y son empleados por todos los fabricantes. Los símbolos básicos son:



En estos diagramas la línea vertical a la izquierda representa un conductor con tensión, y la línea vertical a la derecha representa tierra.

Por ejemplo:



Programa:

```
STR NOT X1
AND X2
OR X3
OUT Y1
```

Con este tipo de diagramas se describe normalmente la operación eléctrica de distintos tipos de máquinas, y puede utilizarse para sintetizar un sistema

de control y, con las herramientas de software adecuadas, realizar la programación del PLC.

Se debe recordar que mientras que en el diagrama eléctrico todas las acciones ocurren simultáneamente, en el programa se realizan en forma secuencial, siguiendo el orden en el que los "escalones" fueron escritos, y que a diferencia de los relés y contactos reales (cuyo número está determinado por la implementación física de estos elementos), en el PLC se puede considerar que existen infinitos contactos auxiliares para cada entrada, salida, relé auxiliar o interno, etc.

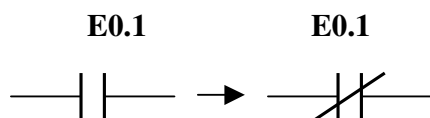
Los contactos

Los elementos a evaluar para decidir si activar o no las salidas en determinado "escalón", son variables lógicas o binarias, que pueden tomar solo dos estados: 1 ó 0, Estos estados que provienen de entradas al PLC o relés internos del mismo.

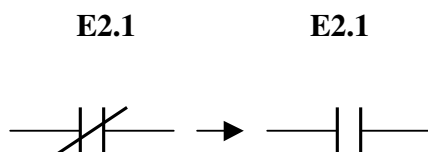
En la programación Escalera (Ladder), estas variables se representan por contactos, que justamente pueden estar en solo dos estados: abierto o cerrado.

Los contactos se representan con la letra "E" y dos números que indicaran el modulo al cual pertenecen y la bornera al la cual están asociados

Ejemplo: E0.1 Entrada del Modulo "0" borne "1"



Los contactos abiertos al activarse se cerraran

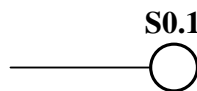


Los contactos cerrados al activarse se abrirán

Las salidas de un programa Ladder son equivalentes a las cargas (bobinas de relés, lámparas, etc.) en un circuito eléctrico.

Se las identifica con la letra "S", "A" u otra letra, dependiendo de los fabricantes, y dos números que indicaran el modulo al cual pertenecen y la bornera al la cual están asociados

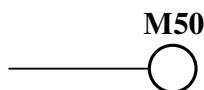
Ejemplo: S0.1 Salida del Modulo "0" borne "1"



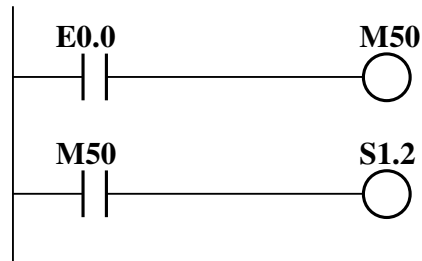
Relés Internos o Marcas

Como salidas en el programa del PLC se toma no solo a las salidas que el equipo posee físicamente hacia el exterior, sino también las que se conocen como **"Relés Internos o Marcas"**. Los relés internos son simplemente variables lógicas que se pueden usar, por ejemplo, para memorizar estados o como acumuladores de resultados que utilizaran posteriormente en el programa.

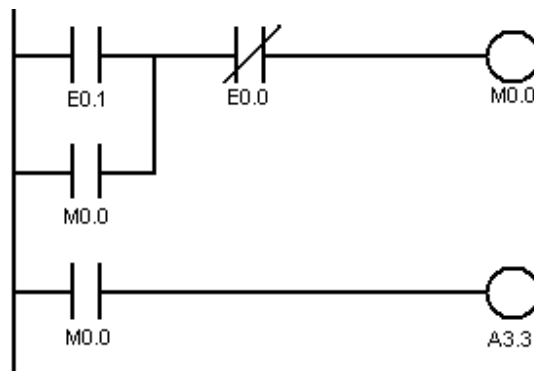
Se las identifica con la letra "M" y un número el cual servirá para asociarla a algún evento



Por ejemplo:



El estado de la salida M50 depende directamente de la entrada E0.0, pero esta salida no esta conectada a un borne del modulo de salidas, es una marca interna del programa. Mientras que el estado de la salida S1.2 es resultado de la activación del contacto M50



Las marcas remanentes son aquellas que en el caso de haber un fallo de tensión, cuando se restablece recuerdan su estado anterior, o sea, si estaban a 1 se pondrán a 1 solas (las salidas **NO** son remanentes).

Las funciones lógicas más complejas como:

- Temporizadores
- Contadores
- Registros de desplazamiento
- etc.

Se representan en formato de bloques.

Estos no están normalizados, aunque guardan una gran similitud entre sí para distintos fabricantes.
Resultan mucho más expresivos que si se utiliza para el mismo fin el lenguaje en lista de instrucciones.

Sobre estos bloques se define:

La base de los tiempos y el tiempo final en el caso de temporizadores

El módulo de conteo y condiciones de paro y reset en el caso de contadores.

Existen también bloques funcionales complejos que permiten la manipulación de datos y las operaciones con variables digitales de varios bits.

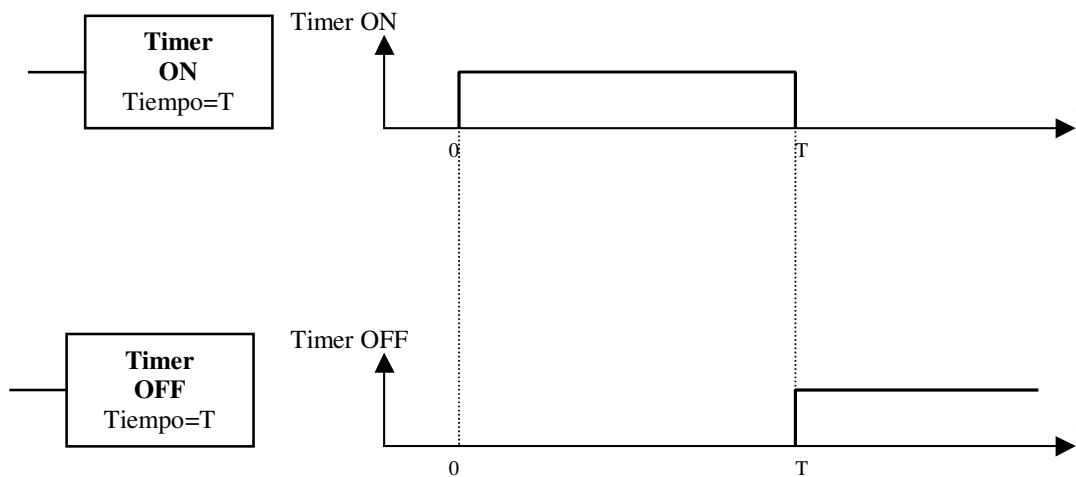
La presencia de estos bloques de ejecución dependiente de una o más condiciones binarias, multiplica la potencia de programación sin dejar de mantener las ventajas de la representación gráfica del programa. Así, pueden programarse situaciones de automatización compleja que involucren variables digitales, registros, transferencias, comparaciones, señales analógicas, etc.

Por supuesto, no todos los Autómatas, aun del mismo fabricante, pueden manejar todas las posibilidades de programación con contactos: solo las gamas más altas acceden a la totalidad de extensiones del lenguaje.

Los temporizadores

Como lo indica su nombre, cada vez que alcanzan cierto valor de tiempo activan un contacto interno. Dicho valor de tiempo, denominado PRESET o meta, debe ser declarado por el usuario. Luego de haberse indicado el tiempo de meta, se le debe indicar con cuales condiciones debe empezar a temporizar, o sea a contar el tiempo. Para ello, los temporizadores tienen una entrada denominada START o inicio, a la cual deben llegar los contactos o entradas que sirven como condición de arranque. Dichas condiciones, igual que cualquier otro renglón de Ladder, pueden contener varios contactos en serie, en paralelo, normalmente abiertos o normalmente cerrados.

Una de las tantas formas de representación sería:



Las operaciones de tiempo permiten programar los temporizadores internos del autómeta. Existen diversos tipos de temporizadores y para utilizarlos se deben ajustar una serie de parámetros:

Arranque del temporizador: conjunto de contactos que activan el temporizador, conectados como se desee.

Carga del tiempo: la forma habitual es mediante una constante de tiempo, pero pueden haber otros ajustes, p.e. leyendo las entradas, un valor de una base de datos, etc.

Esta carga del valor se debe realizar con la instrucción **L** que lo almacena en una zona de memoria llamada acumulador (AKKU1) para luego transferirlo al temporizador.

formato **L KT xxx.yy KT** constante de tiempo.

xxx tiempo (máx. 999).

y base de tiempos.

0 = 0.01 seg. (centésimas).

1 = 0.1 seg. (décimas).

2 = 1 seg.

3 = 10 seg. (segundos x 10)

ejemplo: **KT 243.1** 24,3 segundos

KT 250.2 250 segundos

T0...MAX: número de temporizador. El número MAX depende del fabricante

Paro del temporizador: es opcional y pone a cero el valor contado en el temporizador.

A continuación definimos diferentes tipos de temporizadores.

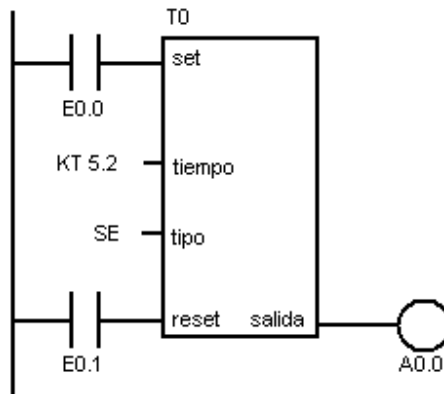
Tipos de temporizador: **SE** - Con retardo a la conexión

SS - Con retardo a la conexión activado por impulso en **set**

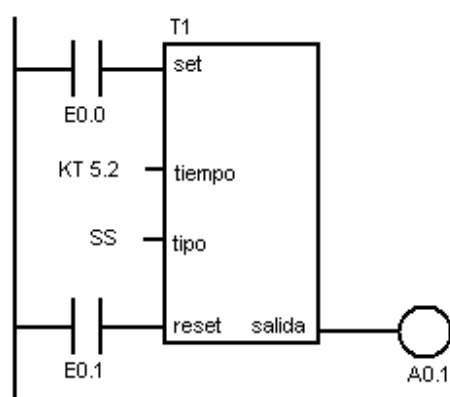
SI - mientras mantenemos conectada la señal set, la salida estará activa durante **KT**.

SV - mantiene la salida activa durante **KT**

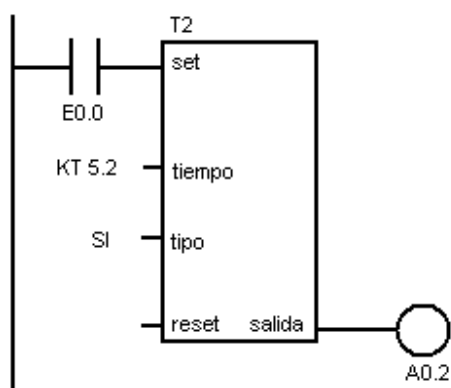
Temporizador SE: retardo a la conexión manteniendo la entrada **set** a 1. La entrada **reset** desconecta el temporizador.



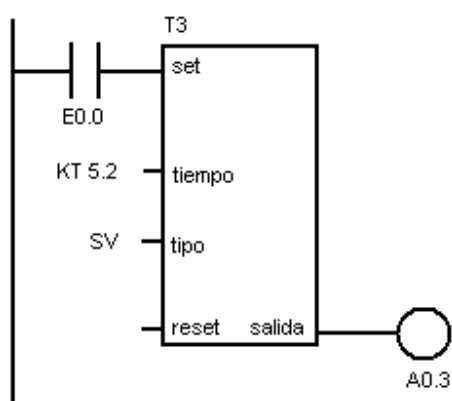
Temporizador SS: retardo a la conexión activado por impulso en **set**. Sólo se desconectará la salida por la entrada **reset**.



Temporizador SI: mientras mantenemos conectada la señal **set**, la salida estará activa durante **KT**.



Temporizador SV: mantiene la salida activa durante **KT** independientemente del tiempo de la señal **set** esté activa.

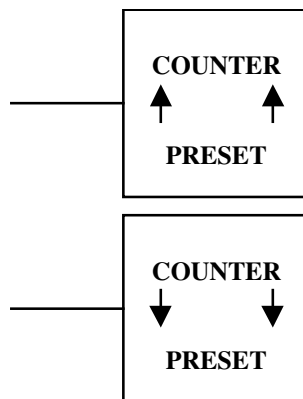


Los contadores

Definidos como posiciones de memoria que almacenan un valor numérico, mismo que se incrementa o decrementa según la configuración dada a dicho contador.

Como los temporizadores, un contador debe tener un valor prefijado como meta o PRESET, el cual es un número que el usuario programa para que dicho contador sea activo o inactivo según el valor alcanzado.

Por ejemplo, si el contador tiene un *preset* de 15 y el valor del conteo va en 14, se dice que el contador se encuentra inactivo, sin que por ello se quiera decir que no esté contando. Pero al siguiente pulso, cuando el valor llegue a 15, se dice que el contador es activo porque ha llegado al valor de *preset*.



Dependiendo del software, puede ocurrir que el contador empiece en su valor de *preset* y cuente hacia abajo hasta llegar a cero, momento en el cual entraría a ser activo.

Nos permitirán contar y/o descontar impulsos que enviemos al contacto que lo activa (p.e. número de botes, sacos, piezas, etc.) entre 0 y 999.

Los parámetros son:

Z0... MAX – número de contador

ZV – incrementa el valor del contador (no supera el valor 999).

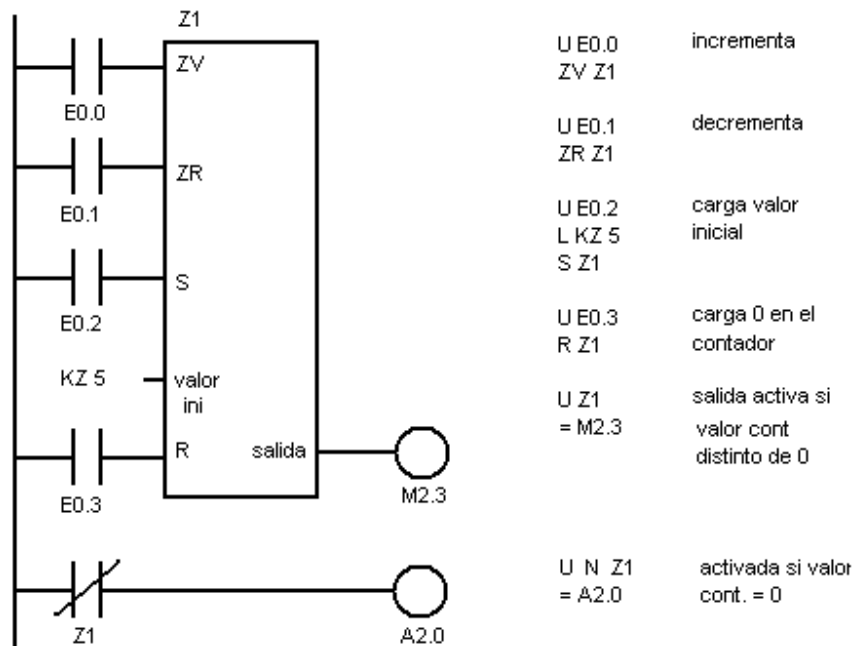
ZR – decrementa el valor del contador (no decremента por debajo de 0).

S - carga el valor inicial en el contador.

KZ xxx – valor inicial.

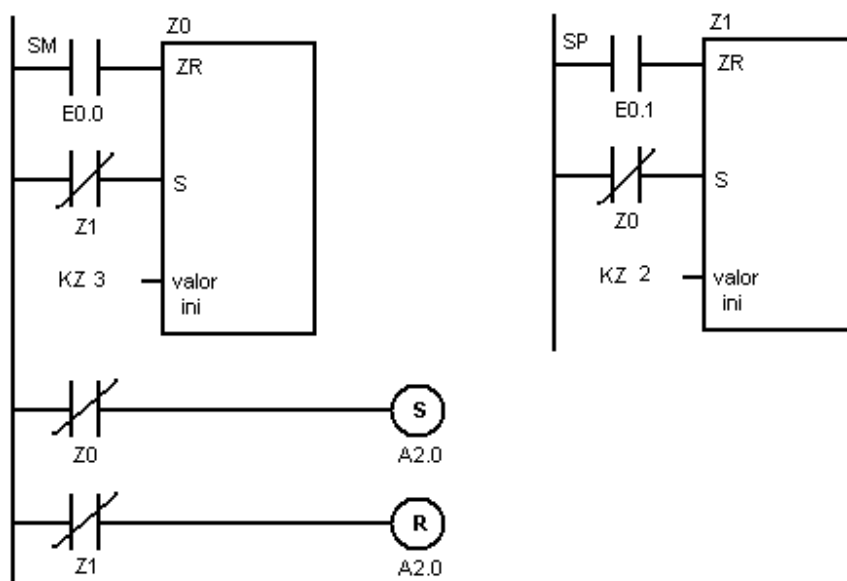
R - resetea el valor del contador.

La salida del contador estará a “1” siempre que el valor del contador sea diferente de “0”.

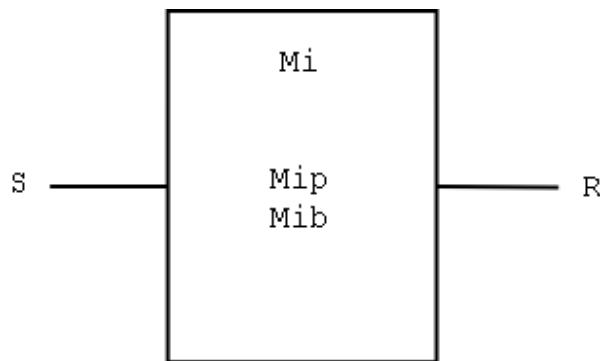


Ejemplos:

Conectar una salida al accionar el pulsador de marcha 3 veces y pararla al pulsar el de paro 2 veces.



Monoestables



constante de tiempo

Mip tiempo

Mib base de tiempos.

El monoestable es un elemento capaz de mantener activada una salida durante el tiempo con el que se haya programado, desactivándola automáticamente una vez concluido dicho tiempo. Una de sus principales ventajas es su sencillez ya que sólo posee una entrada y una salida como podemos observar en la figura.

- **Entrada STAR (S):** Cuando se activa o se le proporciona un impulso comienza la cuenta que tiene programada.
- **Salida RUNNING (R):** Se mantiene activada mientras dura la cuenta y se desactiva al finalizarla. Al igual que con el temporizador, para programar la cuenta hay que introducir los valores de Mip y Mib.

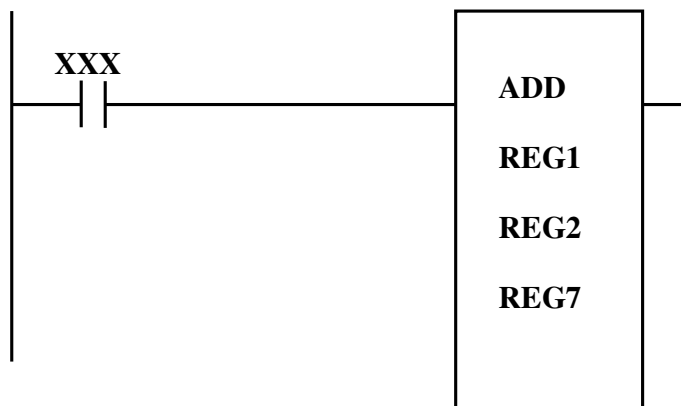
Las operaciones aritméticas

Puede haber operaciones matemáticas como sumas, restas, comparaciones, multiplicaciones, divisiones, desplazamientos de bits, etc. Todas ellas utilizan valores contenidos en registros de memoria referenciados a contadores, entradas, salidas, temporizadores y demás. Las funciones matemáticas son usadas especialmente para la manipulación de variables analógicas.

Las operaciones aritméticas con números enteros son representadas por cajas (Boxes) en las que se indica la operación a efectuar y los operandos. El funcionamiento sigue las reglas generales del diagrama de contactos, cuando se cierra el contacto XXX se realiza la operación.

Ejemplo:

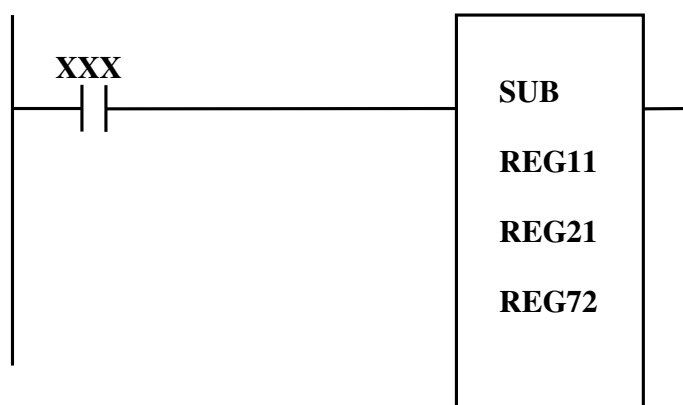
Suma: $REG7 = REG1 + REG2$



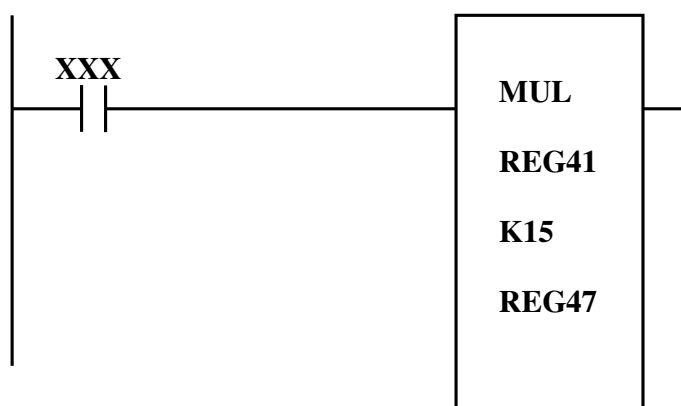
En este ejemplo se suman los contenidos de las memorias de datos REG1 Y REG2 y se almacena el resultado en REG7, cuando la condicion XXX se vuelve verdadera.

Los siguientes ejemplos ilustran las operaciones más comunes disponibles en la mayoría de los PLC.

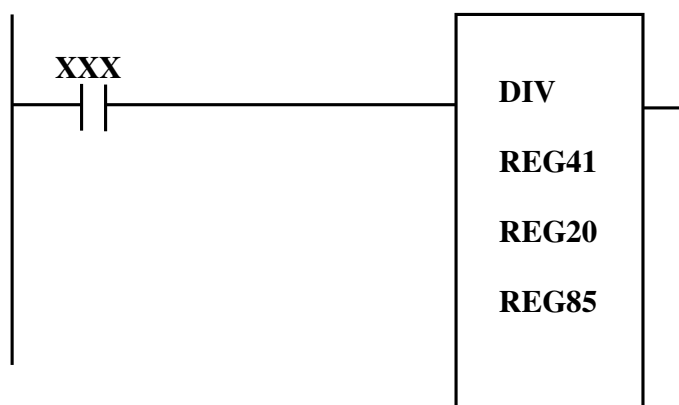
Resta: $\text{REG72} = \text{REG11} + \text{REG21}$



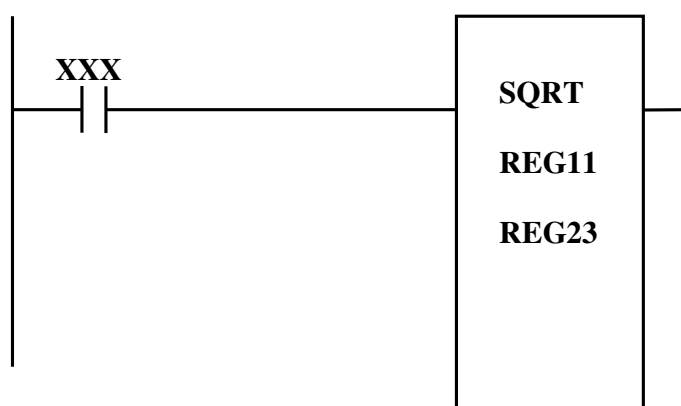
Multiplicación: $\text{REG47} = \text{REG41} * 15$



Division: $\text{REG85} = \text{REG41} / \text{REG20}$



Raíz Cuadrada: $\text{REG 23} = \text{SQRT} (\text{REG11})$

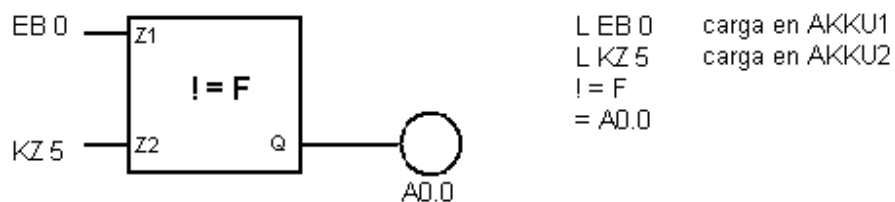


OPERACIONES DE COMPARACIÓN

Un comparador es una instrucción que nos permitirá relacionar dos datos del mismo formato (BYTE o WORD) entre sí.

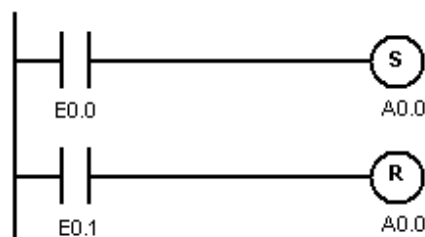
Las comparaciones pueden ser:

- ! = F** igualdad
- > < F** desigualdad
- > F** mayor
- < F** menor
- > = F** mayor o igual
- < = F** menor o igual



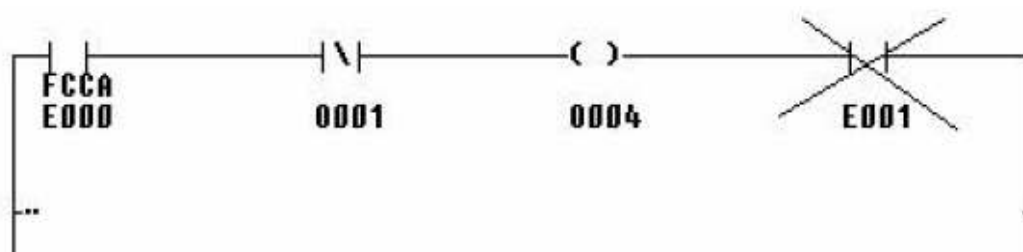
INSTRUCCIONES SET Y RESET

La instrucción SET activa la bobina correspondiente cada vez que enviamos un IMPULSO, y sólo se desactivará al enviar otro a la instrucción RESET. Podemos activar tanto salidas como marcas internas.

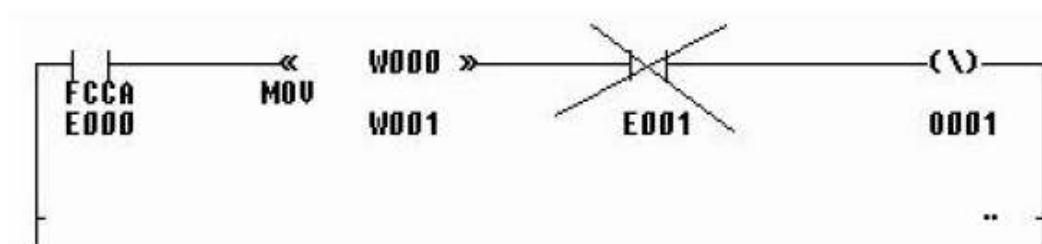


Lenguaje Ladder - Procesamiento y Limitaciones

Las bobinas pueden ir precedidas de contactos, pero no pueden estar seguidas por ninguno.



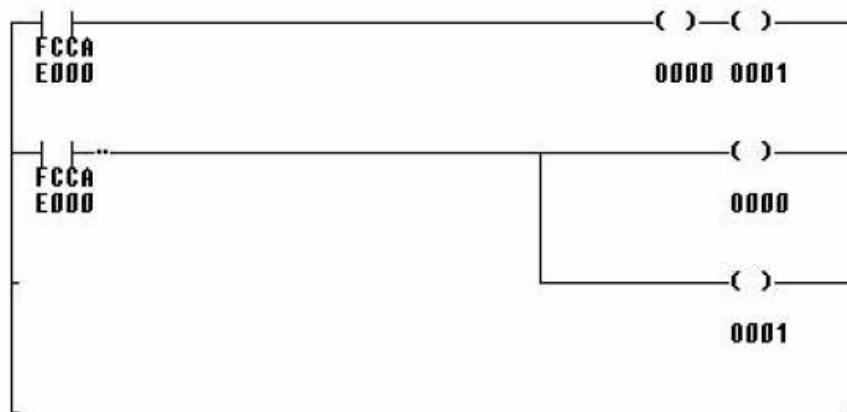
Lo mismo se aplica a los bloques Función, por ejemplo el bloque función transferencia, ya que se comporta como bobina.



Sin embargo hay una conexión que es posible en nuestro Ladder pero imposible en un tablero.

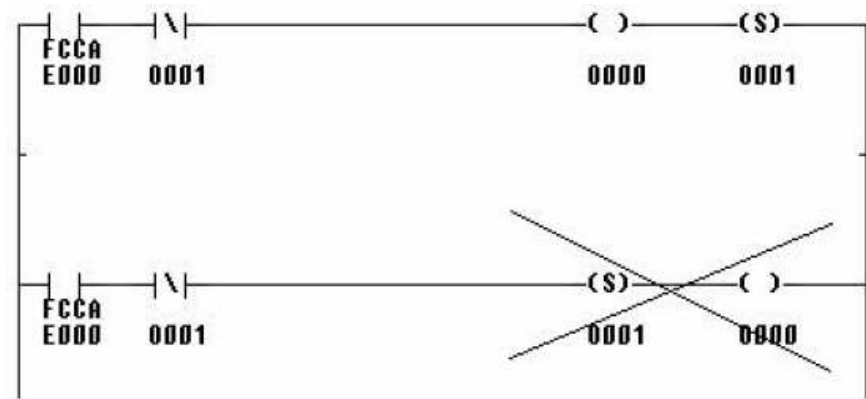
Las bobinas pueden ser conectadas en Serie comportándose en forma similar que si estuvieran en paralelo.

Si en el circuito de activación de las bobinas existen varios contactos en serie, conviene usar la conexión paralelo de las bobinas, ya que el programa se ejecuta en menor tiempo.



La diferencia ocurre cuando se utilizan contactos auxiliares, ya que debe prestarse atención al orden en que se ubican las bobinas.

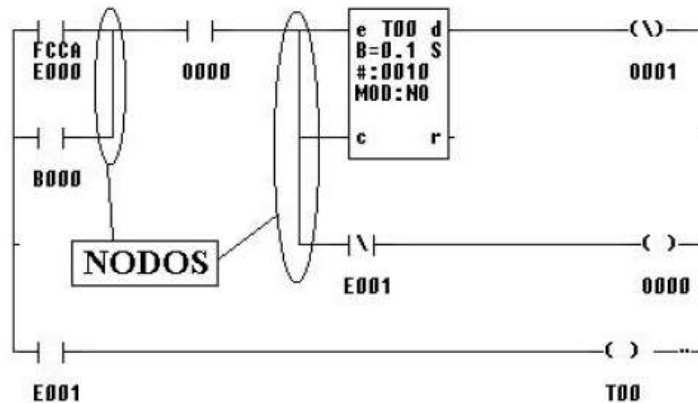
Por ejemplo:



Si adoptáramos la segunda alternativa de conexión sucedería que una vez actuada la salida O001, ya nunca se activaría la salida O000, dado que el contacto invertido de la salida O001 quedaría definitivamente abierto.

La cantidad de uniones "llamados NODOS" están limitados, no puede superar cierto numero.

Cuáles son nodos ?



Los bloques Timer, Contadores, etc. sólo pueden aparecer una vez en el programa.

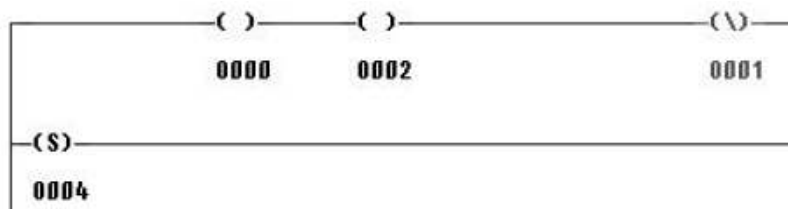
Sin embargo pueden utilizarse contactos y bobinas referidos a éstos en cualquier parte.

No pueden editarse dos bloques Timer, etc. con el mismo número.

Si las bobinas son conectadas directamente a la barra de la izquierda, entonces se las considera permanentemente activadas. Por supuesto, esto siempre que esa parte del programa esté siendo ejecutada.

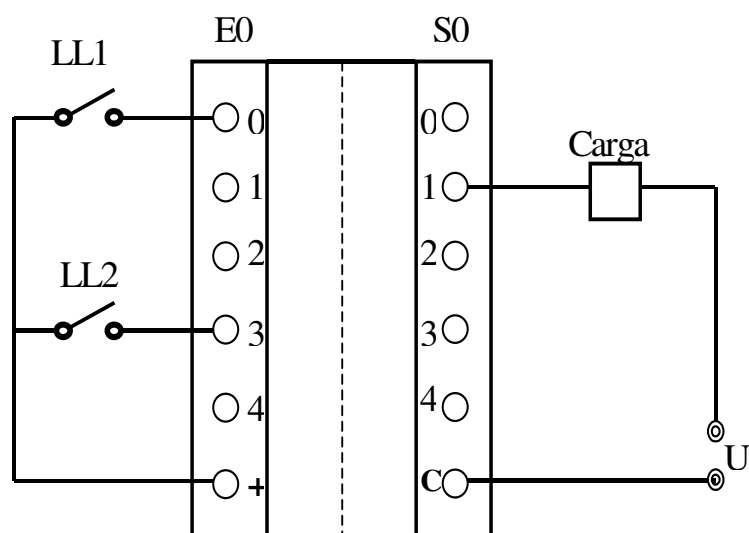
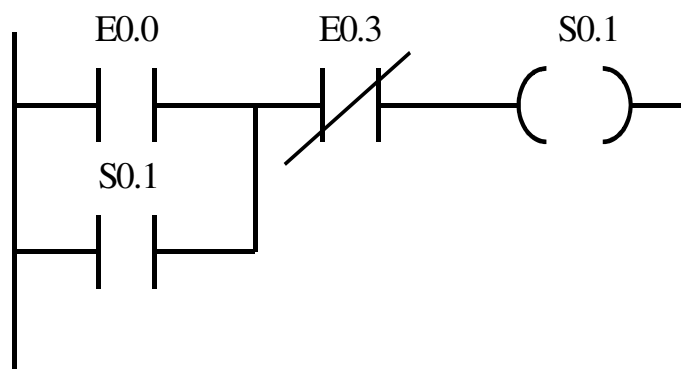
En algunos PLC esto está Prohibido, debe colocarse un contacto entre la entrada y la bobina

Ejemplo:

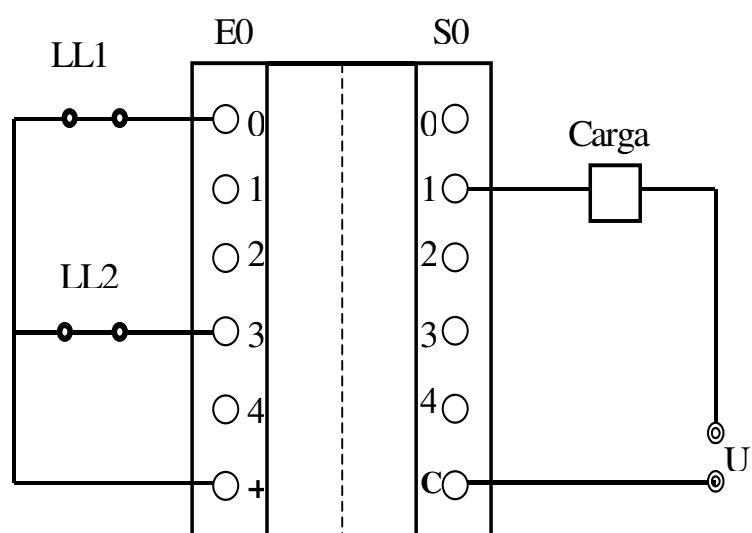
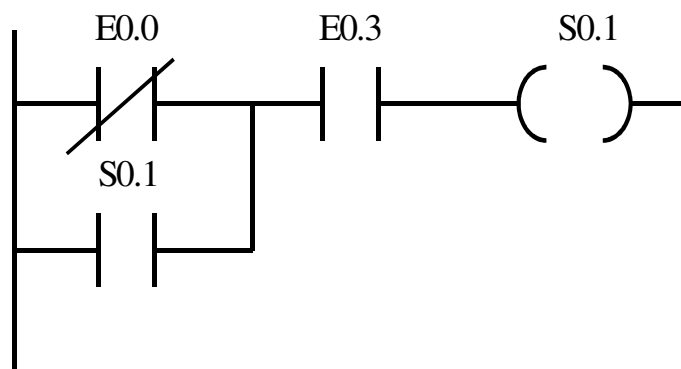


Algunos ejemplos de Contactos externos y el diagrama escalera

Si queremos hacer un enclavamiento de alguna maquina usando dos pulsadores **NA**, una forma seria:

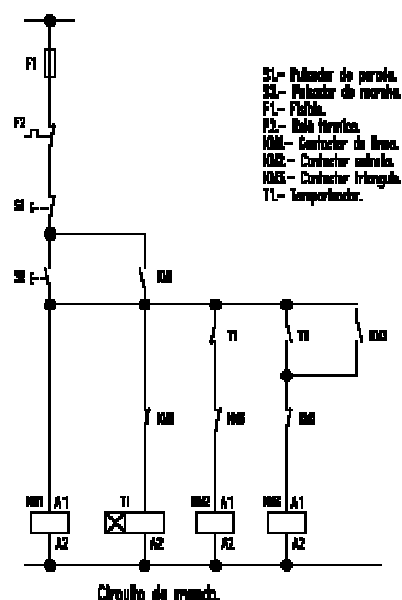
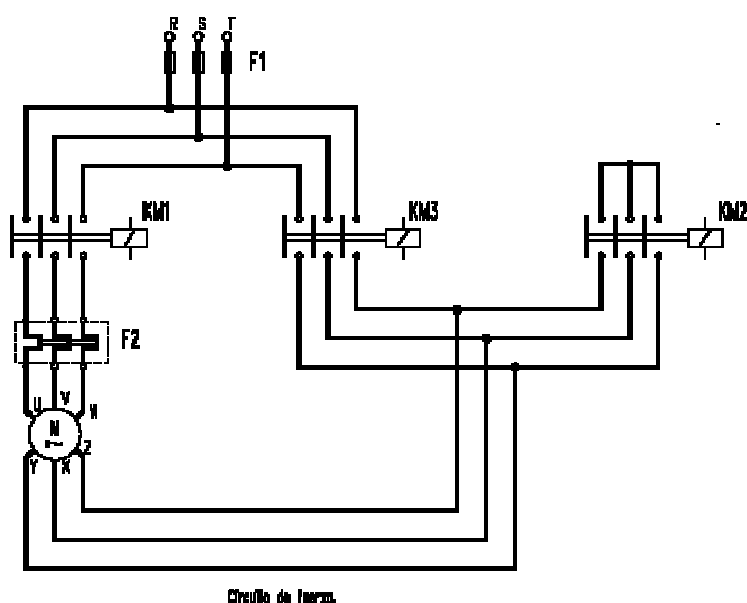


Si lo hacemos usando contactos externos **NC** será:

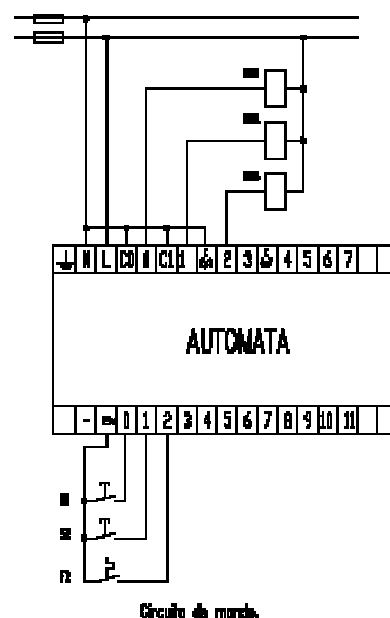
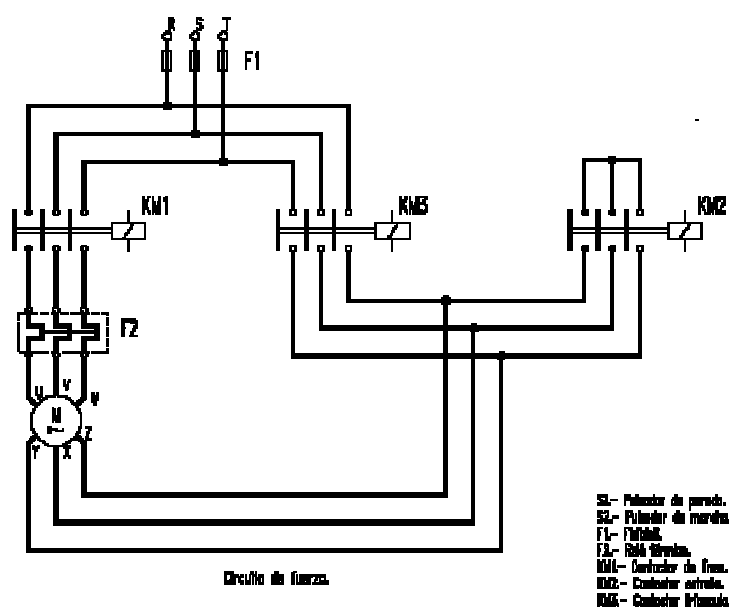


Veamos un típico circuito de automatismos. Un arrancador Estrella/Triángulo con temporizador

La figura 1 muestra como es la técnica cableada. Por una parte tenemos el circuito de fuerza, que alimenta el motor, y por otra el circuito auxiliar o de mando, que realiza la maniobra de arranque de dicho motor.



La figura 2 muestra como se realiza el mismo montaje de forma programada. El circuito de fuerza es exactamente el mismo que en la técnica cableada. Sin embargo, el de mando será sustituido por un autómata programable, al cual se unen eléctricamente los pulsadores y las bobinas de los contactores. La maniobra de arranque la realizara el programa que previamente se ha transferido al autómata.

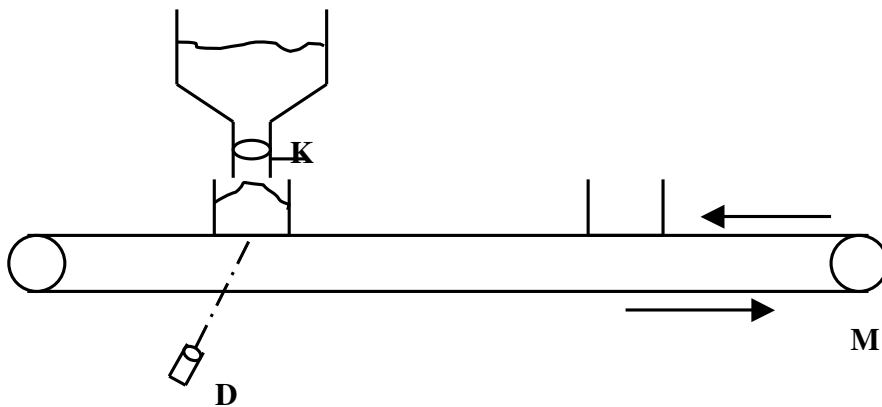


Ejemplo de Programación

Sobre una cinta transportadora impulsada por un motor M, se transportan cajas las cuales deberán detenerse bajo una tolva al ser detectadas por un sensor D. Una vez detenida la caja bajo la tolva, se abrirá una esclusa (Mediante el contactor K1) durante 10 seg., tiempo en el cual la caja se llena. Pasado este tiempo, la esclusa deberá cerrarse y la cinta comenzará a moverse quitando la caja de esa posición. Este proceso se deberá repetir cuando pase otra caja bajo la tolva.

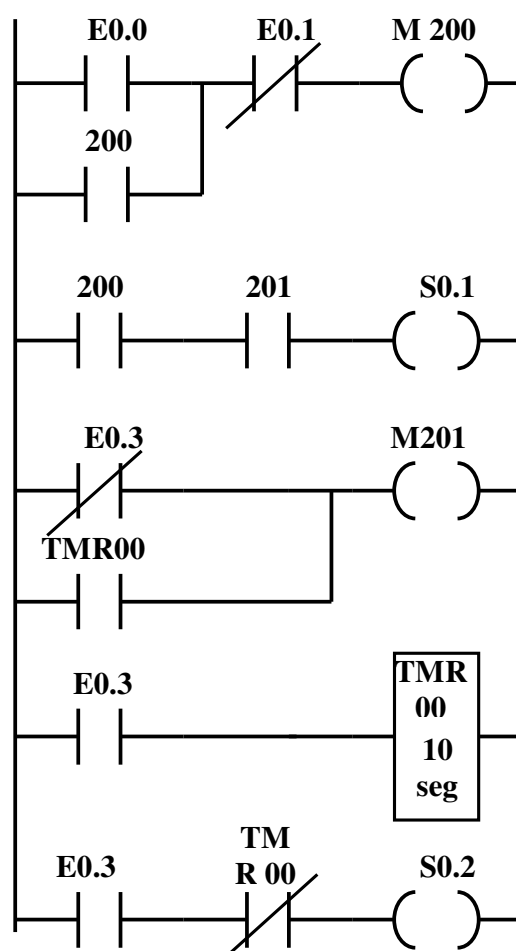
Se pide realizar el programa en diagrama escalera, cuadro de asignaciones y esquema de conexiones.

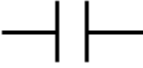
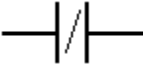



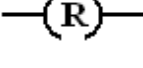

Nota: La esclusa se abre cuando es activado el contactor K1 y se cierra al desactivarse este. La cinta está funcionando siempre, salvo cuando una caja es detectada.



Cuadro de Asignaciones

E0.0	Start
E0.1	Stop
E0.3	Detector de la Caja
S0.1	Motor de la Cinta
S0.2	Contactor de la tolva (K1)
M200	Marca Interna
M201	Marca Interna
TMR00	Temporizador OFF de 10 Seg.



Elementos básicos en LADDER		
Símbolo	Nombre	Descripción
	Contacto NA	Se activa cuando hay un uno lógico en el elemento que representa, esto es, una entrada (para captar información del proceso a controlar), una variable interna o un bit de sistema.
	Contacto NC	Su función es similar al contacto NA anterior, pero en este caso se activa cuando hay un cero lógico, cosa que deberá de tenerse muy en cuenta a la hora de su utilización.
	Bobina NA	Se activa cuando la combinación que hay a su entrada (izquierda) da un uno lógico. Su activación equivale a decir que tiene un uno lógico. Suele representar elementos de salida, aunque a veces puede hacer el papel de variable interna.
	Bobina NC	Se activa cuando la combinación que hay a su entrada (izquierda) da un cero lógico. Su activación equivale a decir que tiene un cero lógico. Su comportamiento es complementario al de la bobina NA.
	Bobina SET	Una vez activa (puesta a 1) no se puede desactivar (puesta a 0) si no es por su correspondiente bobina en RESET. Sirve para memorizar bits y usada junto con la bobina RESET dan una enorme potencia en la programación.
	Bobina SET	Permite desactivar una bobina SET previamente activada.
	Bobina JUMP	Permite saltarse instrucciones del programa e ir directamente a la etiqueta que se desee. Sirve para realizar subprogramas.