Lección 6
Lenguajes de Programación

Clasificación de los Lenguajes de Programación

Niveles de los Lenguajes
- Bajo Nivel
- Alto Nivel

Niveles de los Lenguajes Específicos para PLC

Lenguajes de Programación para PLC
- Listas
- Plano de Contactos
- Diagrama de Bloques Funcionales
- Organigrama de Bloques Secuenciales

Lección 7
Operaciones Lógicas

Operaciones Lógicas

Ejemplos de Aplicación

Lección 8
Lenguaje de Plano de Contactos

Lenguaje de Plano de Contactos
Reglas del Lenguaje
Elementos del Lenguaje
Elementos de Entrada
Elementos de Salida

Lección 9
Temporizadores y Contadores

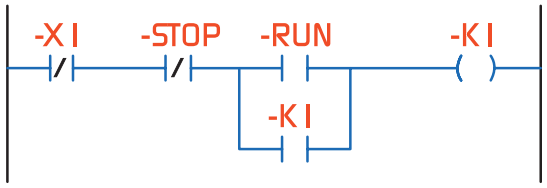
Temporizadores
Definición de Tiempo de Retardo
Contadores
Ejemplos de Aplicación

Lección 6
Lenguajes de Programación

Los lenguajes de programación son necesarios para la comunicación entre el usuario (sea programador u operario de la máquina o proceso donde se encuentre el PLC) y el PLC. La interacción que tiene el usuario con el PLC la puede realizar por medio de la utilización de un cargador de programa (loader Program) también reconocida como consola de programación o por medio de un PC (computador Personal).Tenga en cuenta que: En procesos grandes o en ambientes industriales el PLC recibe el nombre también de API (Autómata Programable Industrial) y utiliza como interfase para el usuario pantallas de plasma, pantallas de contacto (touch screen) o sistemas SCADA (sistemas para la adquisición de datos, supervisión, monitoreo y control de los procesos), cuyo contenido no serán presentados ni tenidos en cuenta en este curso.

Clasificación de los Lenguajes de Programación:

Los lenguajes de programación para PLC son de dos tipos, visuales y escritos. Los visuales admiten estructurar el programa por medio de símbolos gráficos, similares a los que se han venido utilizando para describir los sistemas de automatización, planos esquemáticos y diagramas de bloques. Los escritos son listados de sentencias que describen las funciones a ejecutar.




```
AN    -X I
AN    -STOP
(
A     -RUN
O     -K I
)
=     -K I
```

Lenguaje Visual Lenguaje Escrito

Los programadores de PLC poseen formación en múltiples disciplinas y esto determina que exista diversidad de lenguajes. Los programadores de aplicaciones familiarizados con el área industrial prefieren lenguajes visuales, por su parte quienes tienen formación en electrónica e informática optan, inicialmente por los lenguajes escritos.

Niveles de los Lenguajes

Los lenguajes de programación de sistemas basados en microprocesadores, como es el caso de los PLC, se clasifican en niveles; al microprocesador le corresponde el nivel más bajo, y al usuario el más alto.

Lenguajes de Bajo Nivel:

Lenguaje de Máquina:

Código binario encargado de la ejecución del programa directamente en el microprocesador.

Lenguaje Ensamblador:

Lenguaje sintético de sentencias que representan cada una de las instrucciones que puede ejecutar el microprocesador. Una vez diseñado un programa en lenguaje ensamblador es necesario, para cargarlo en el sistema, convertirlo o compilarlo a lenguaje de máquina. Los programadores de lenguajes de bajo nivel deben estar especializados en microprocesadores y demás circuitos que conforman el sistema.

```
BANKSEL   T1CON

CLRF      TMR1H
CLRF      TMR1L

MOVLW     HIGH      Valor1
MOVWF     CCPR2H

MOVLW     LOW       Valor1
MOVWF     CCPR2H

BSF       T1CON,TMR1ON     ;INICIA CONTEO DE PERIODO

BSF       INTCON,PEIE      ;HABILITA INTERRUPCIONES PERIFERICAS
BSF       INTCON,GIE       ;HABILITA INTERRUPCIONES GLOBALES
```

Lenguajes de Alto Nivel:

Se basan en la construcción de sentencias orientadas a la estructura lógica de lo deseado; una sentencia de lenguaje de alto nivel representa varias de bajo; cabe la posibilidad que las sentencias de un lenguaje de alto nivel no cubran todas las instrucciones del lenguaje de bajo nivel, lo que limita el control sobre la máquina. Para que un lenguaje de alto nivel sea legible por el sistema, debe traducirse a lenguaje ensamblador y posteriormente a lenguaje de máquina.

```
If  Val(TxtDesde) <= 0 Then
    MsgBox "Verifique en Número Inicial", vbOKOnly
    TxtDesde.SetFocus
    Exit Sub
End  If

If  Val(TxtDesde) <= 0 Then
    MsgBox "Verifique en Número Final", vbOKOnly
    TxtHasta.SetFocus
    Exit Sub
End  If
```

Tipos	Descripción	Nivel	Características	
			Acceso a los Recursos	Preferencias de Uso
Visuales	Utilizan los símbolos de planos esquemáticos y diagramas de bloques.	Alto	Restringido a los símbolos que proporciona el lenguaje.	Profesionales en áreas de automatización industrial, mecánica y afines.
Escritos	Utilizan sentencias similares a las de programación de computadores.	Bajo	Total a los recursos de programación.	Profesionales en áreas de electrónica e informática.

Lenguajes de Programación para PLC:

Los fabricantes de PLC han desarrollado una cantidad de lenguajes de programación en mayoría de los casos siguiendo normas internacionales, con el fin de suplir las necesidades y expectativas de los programadores. En la siguiente tabla se presentan lenguajes de uso común.

Lenguaje	Características	Ejemplos*	Tipo	Nivel
Listas	Lista de Instrucciones	IL AWL STL IL/ST	Escrito	Bajo
Plano	Diagrama Eléctrico	LADDER LD KOP	Visual	Alto
Diagrama de Bloques Funcionales	Diagrama Lógico	FBD FBS FUD		
Organigrama de Bloques Secuenciales	Diagrama Algorítmico	AS SFC PETRI GRAFCET		
Otros	Lenguajes Usados en Otras Áreas de la computación	BASIC C	Escrito	
* Los nombres fueron asignados por el fabricante				

Niveles de los Lenguajes Específicos para PLC

1. Bajo Nivel:

En el ámbito de programación de PLC no se utiliza directamente el lenguaje de máquina o del ensamblador. Se emplea el lenguaje de lista de instrucciones,

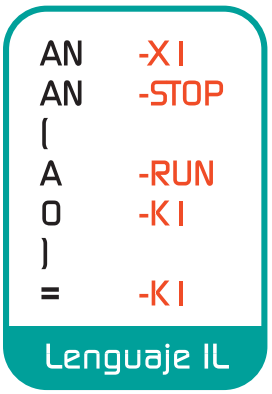
similar al lenguaje ensamblador, con una sintaxis y vocabulario acordes con la terminología usada en PLC.

2. Listas:

Lenguaje que describe lo que debe hacer el PLC instrucción por instrucción.

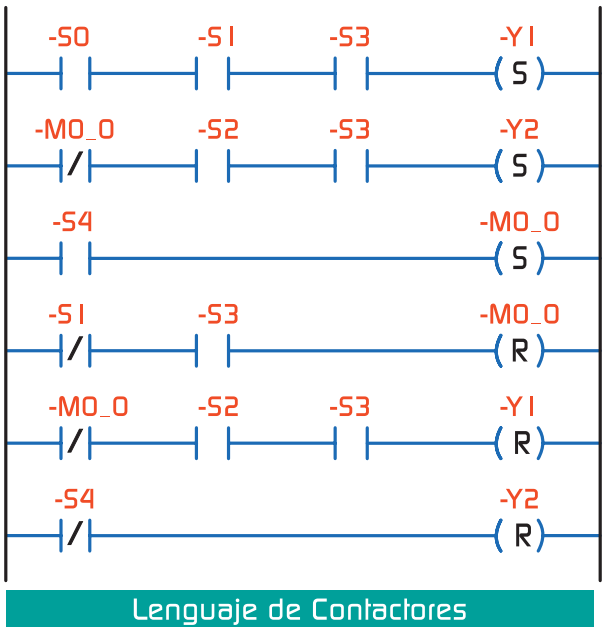
3. Alto Nivel:

Se caracterizan principalmente por ser visuales, aunque existen también lenguajes escritos de alto nivel.



4. Diagrama de Contactos:

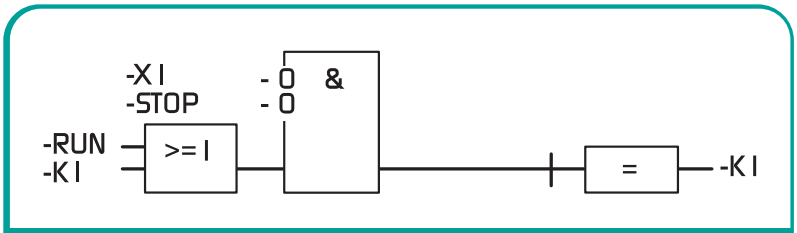
Representa el funcionamiento deseado, como en un circuito de contactores



y relés, fácil de entender y utilizar para usuarios con experiencia en lógica alambrada. En general, nos referimos a este lenguaje como LADDER (escalera), ya que la forma de construcción de su esquema se asemeja a una escalera.

5. Diagrama de Bloques Funcionales:

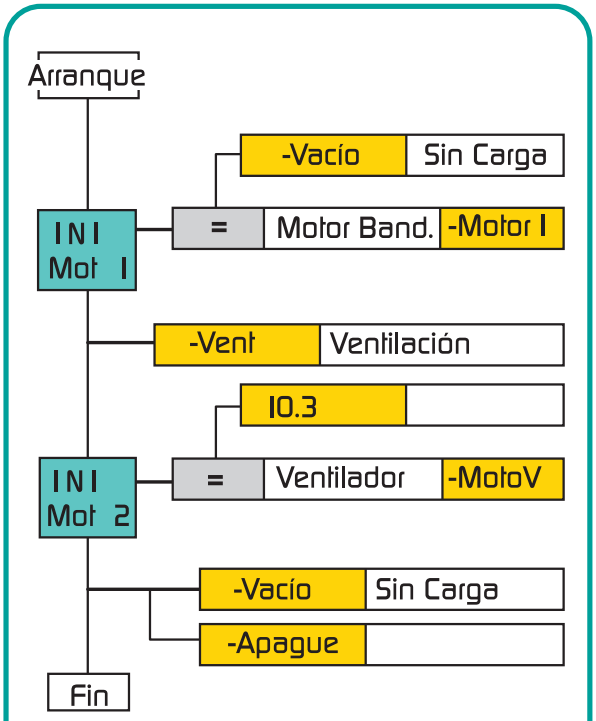
Utiliza los diagramas lógicos de la electrónica digital.



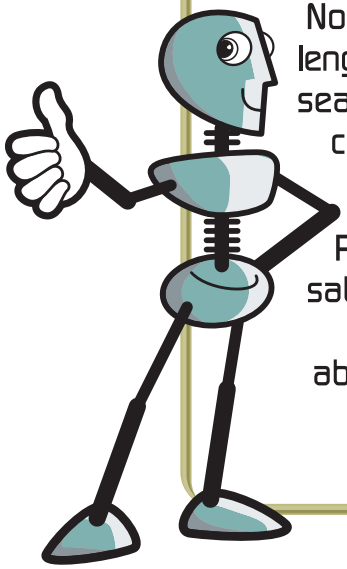
Programa FDB

6. Organigrama de Bloques Secuenciales:

Explota la concepción algorítmica que todo proceso cumple con una secuencia. Estos lenguajes son los más utilizados por programadores de PLC con mayor trayectoria.



Programa As



No podemos decir que alguno de los lenguajes abordados hasta el momento sea mejor que otro, cada uno de ellos cumple con una función propia que depende del tipo de aplicación.

Para aprender de PLC es necesario saber cuando menos un lenguaje de programación, en este curso se abordará el estudio sobre el tema de Plano de Contactos y Lista de Instrucciones.

Lección 7
Operaciones Lógicas

Las operaciones lógicas más utilizadas son: AND, OR, NOT, EXOR. A continuación se presentan las tablas de verdad que las definen.

Los programadores de PLC tienen formación en múltiples disciplinas y esto determina que exista una diversidad de lenguajes. Los programadores de aplicaciones familiarizados con el área industrial prefieren lenguajes visuales, por su parte quienes tienen formación en electrónica o informática optan inicialmente por los lenguajes escritos.

AND - Conjunción

La operación lógica AND -conjunción- entrega como resultado V si todas las entradas son V. Esta se aplica en situaciones en las que se requiere realizar una acción si y sólo sí se cumplen un determinado número de condiciones. En lenguaje de contactos se realiza disponiendo los contactos en serie.

AND		
Entradas		Salidas
A	**B**	**Y**
F	F	F
F	V	F
V	F	F
V	V	V

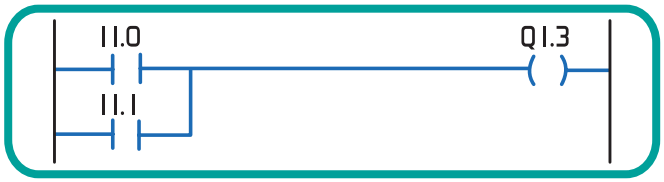
Ejemplo:
En el circuito se activa Q1.2 cuando I1.0, I1.1 e I1.2 son verdaderas. De hecho, el PLC evalúa la rama ejecutando la operación lógica Q1.2 = I1.0 AND I1.1 AND I1.2.

OR - Disyunción

La operación lógica OR -disyunción - entrega como resultado V siempre que alguna de las entradas sea V, lo que se logra poniendo los contactos en paralelo.

OR		
Entradas		Salidas
A	**B**	**Y**
F	F	F
F	V	V
V	F	V
V	V	V

Ejemplo:
En el circito se activa Q1.3 si alguna de las entradas I1.0 o I1.1 se activa. La operación lógica es Q1.3 = I1.0 OR I1.1.

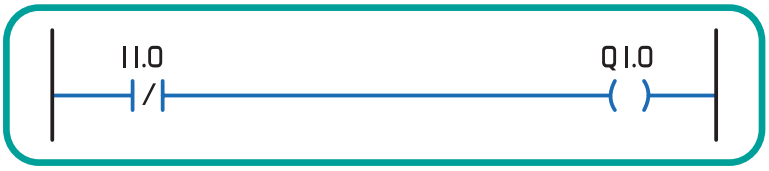


NOT - Inversión

La operación lógica NOT – inversión- entrega como resultado el estado contrario al presente en la entrada, esto se logra con el uso de Contactos Normal Cerrado.

NOT	
Entradas	Salidas
A	**Y**
F	v
V	F

Ejemplo:
Función y operación realizada es Q1.0 = NOT I1.0.



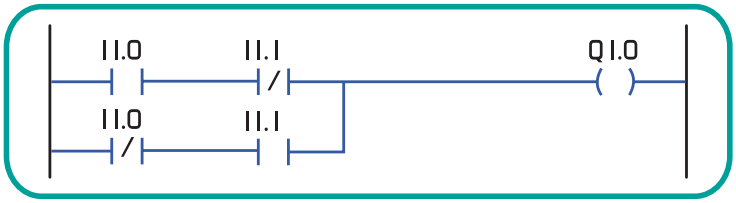
EXOR - OR - Exclusiva

La EXOR - OR -exclusiva- es V si alguna de las entradas, pero nunca ambas, es V también; se puede decir que es V si y sólo si las entradas son distintas. Analicemos detenidamente el circuito que la realiza.

EXOR		
Entradas		Salidas
A	**B**	**Y**
F	F	F
F	V	V
V	F	V
V	V	F

Ejemplo:

Para realizar la operación Q1.0 = I1.0 EXOR I1.1, se debe efectuar una combinación de operaciones AND y OR: Q1.0 = ((I1.0 AND (NOT I1.1)) OR ((NOT I1.0) AND I1.1)). En el lenguaje de contactos es frecuente aquel caso en el cual las operaciones lógicas deben resolverse a partir de contactos normal abierto y normal cerrado.

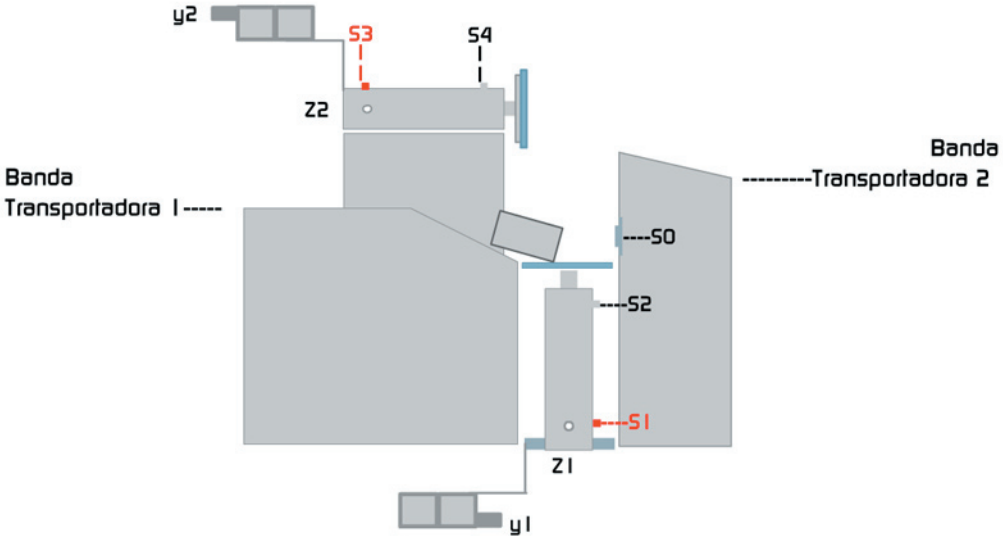


Ejemplos de Aplicación

Elevador de Piezas

En la etapa de un proceso de fabricación, se desplazan piezas de una banda transportadora a otra banda, la segunda esta ubicada a una altura mayor. Para esto se dispone de un elevador basado en un cilindro neumático (Z1). Cuando el elevador llega a su máxima altura, otro cilindro (Z2) empuja la pieza hasta la segunda banda transportadora.

Descripción de la Secuencia - Elevador de Piezas



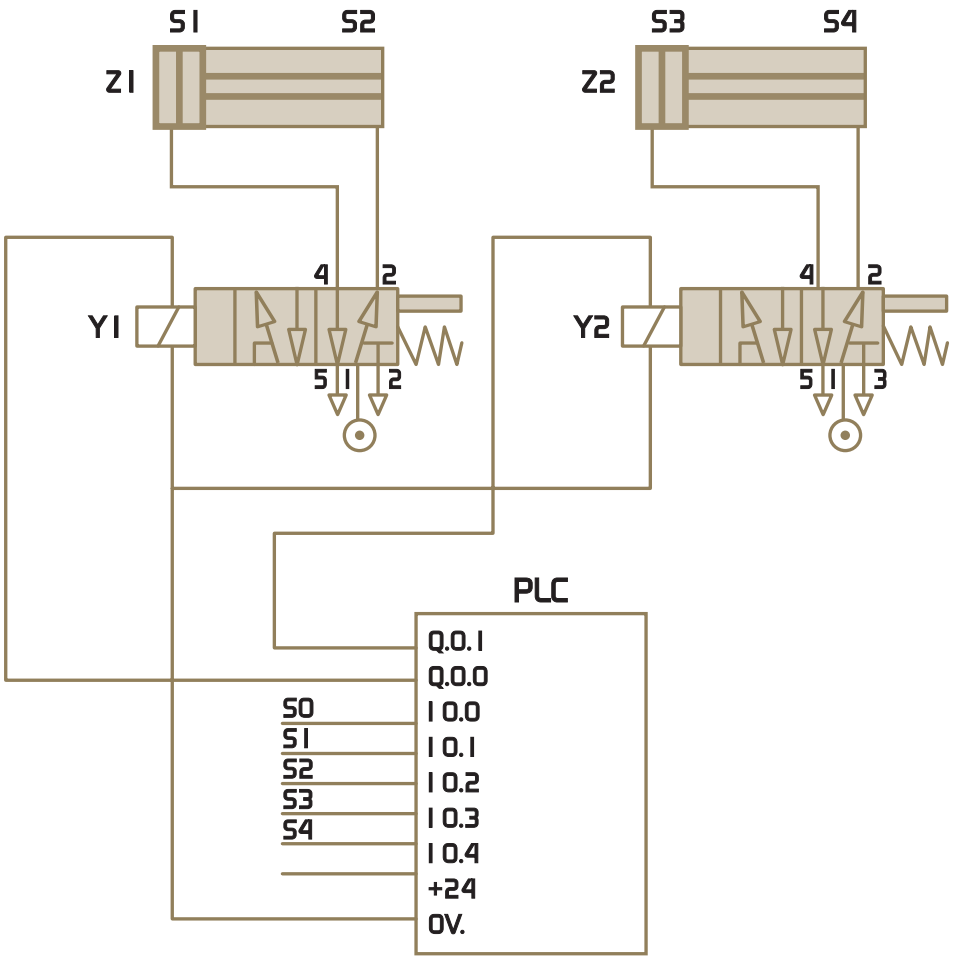
1. El Sistema se encuentra en estado inicial, ambos cilindros Z1 y Z2 están retraídos y no hay piezas presentes en la plataforma del elevador.
2. La pieza situada en la plataforma del elevador B1 es detectada por S0, éste activa el movimiento de Z1 por medio del actuador Y1.
3. S2, determina que Z1 llega a su fin de carrera y Z2 da inicio al movimiento, accionado por el actuador Y2.
4. S4, detecta la posición de fin de carrera y Z2 empieza a retraerse concluido el trabajo de empujar la pieza a la segunda banda transportadora.
5. S3, determina que el cilindro Z2 llega a su posición de inicio de carrera, con lo cual Z1 empieza a retraerse.

6. Cuando S1 detecta que Z1 esta en posición de inicio de carrera, el sistema esta a punto para iniciar un nuevo ciclo.
7. El ciclo se repite, una vez una nueva pieza llega a la plataforma del elevador.

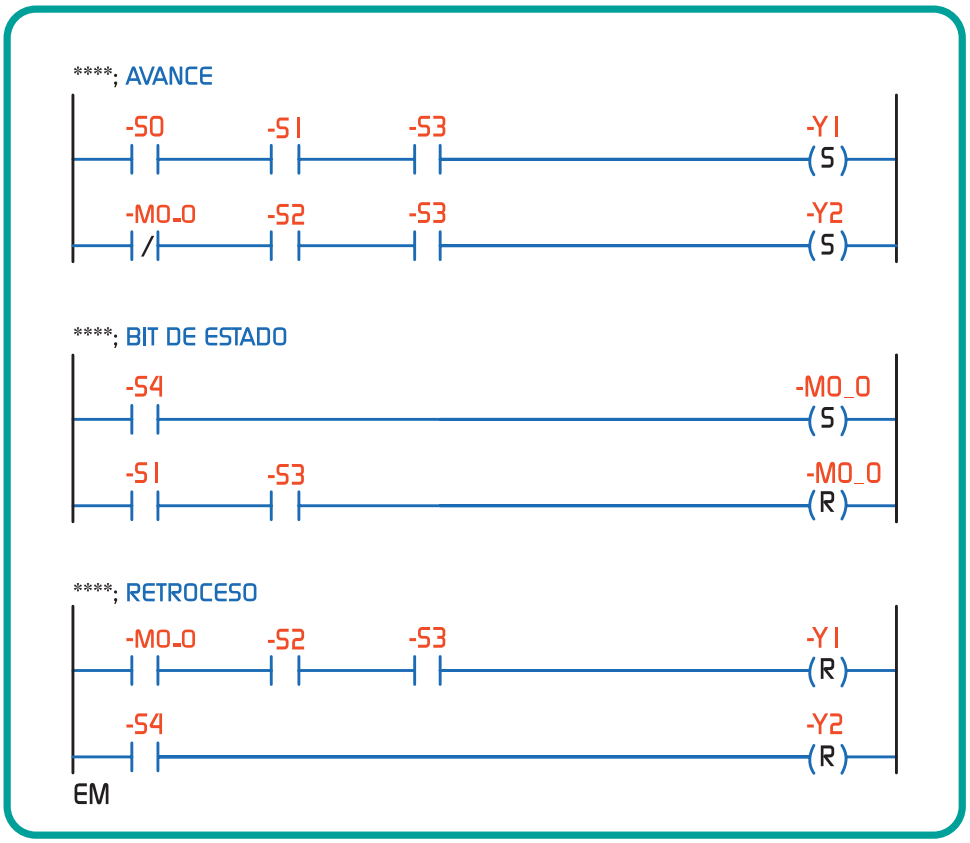
Esquema de conexiones:

Símbolo	Circuito	Descripción
S0	I 0.0	Detector de proximidad. Determina que hay una pieza lista para ser elevada
S1	I 0.1	Detector fin de carrera. Determina que el cilindro Z1 se halla en su posición inicial.
S2	I 0.2	Detector fin de carrera. Determina que el cilindro Z1 se halla en su posición final.
S3	I 0.3	Detector fin de carrera. Determina que el cilindro Z2 se halla en su posición inicial.
S4	I 0.4	Detector fin de carrera. Determina que el cilindro Z2 se halla en su posición final.
Y1	Q 0.0	Electro válvula 3/2 vías. Activa al cilindro Z1.
Y2	Q 0.1	Electro válvula 3/2 vías. Activa al cilindro Z2.

Con la descripción de la secuencia se establecen las preposiciones lógicas para determinar el diagrama de contactos.



Una vez determinadas las operaciones lógicas de la secuencia es fácil hacer el programa en lenguaje de contactos.




```
I 0.0    S0      ; DETECTOR DE PROXIMIDAD DE LA PIEZA
I 0.1    S1      ; Z1 RETRAIDO
I 0.3    S3      ; Z2 RETRAIDO
I 0.4    S4      ; Z2 AVANZADO
M0.0     M0_0    ; 0= PASOS a Al o  1= PASOS d Al f
Q 0.0    Y1      ; Z1  1= AVANCE   0= RETROCESO
Q 0.1    Y2      ; Z2  1= AVANCE   0= RETROCESO
```

El programa ha sido realizado en WINSPS versión 3.22. Se introdujo una tabla de asignación de símbolos que permite que el esquema sea más legible.

Además, se separó el programa en tres redes o circuitos: avance, bit de estado y retroceso, lo cual no sólo permite analizar con facilidad el esquema, sino que también da orden a la ejecución del programa.

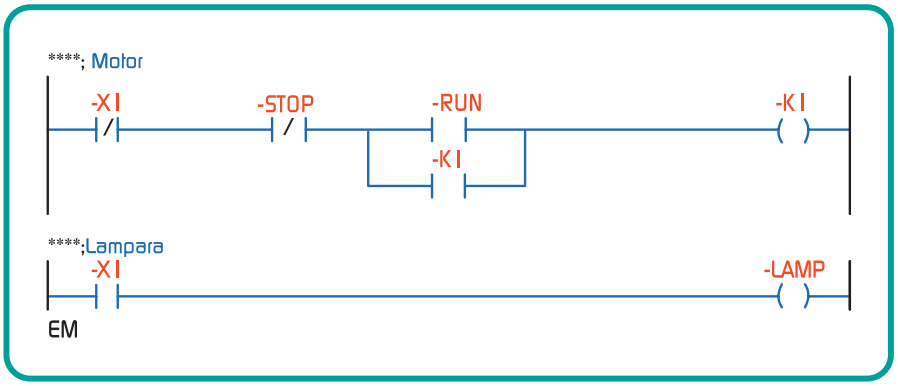
La instrucción de fin de módulo (EM) al final del programa es de uso obligatorio en la mayoría de los ambientes de programación para PLC.

Arranque y Parada

Una aplicación clásica en automatización industrial es la de proveer a un motor eléctrico el control de arranque y parada.
Dado:

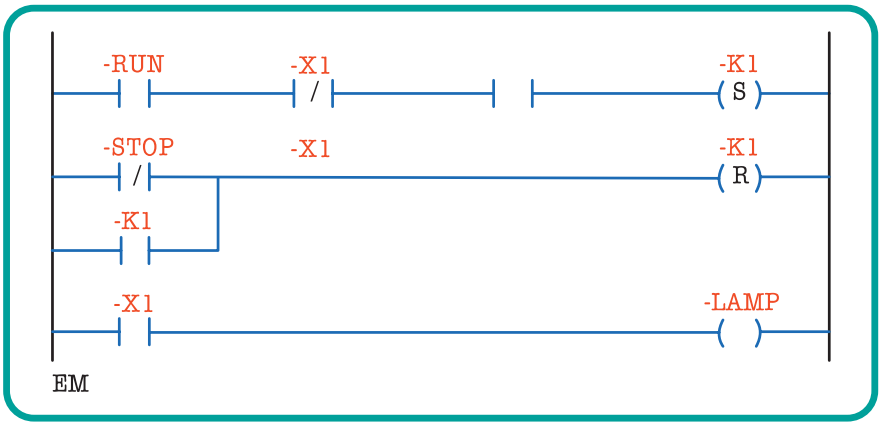
Símbolo	Asignación	Descripción
X1	I 0.0	Relé Térmico
RUN	I 0.1	Pulsador de marcha
STOP	I 0.2	Pulsador de parada
K1	Q 0.0	Contactor del Motor M1
LAMP	Q 0.1	Lámpara de emergencia

El Programa

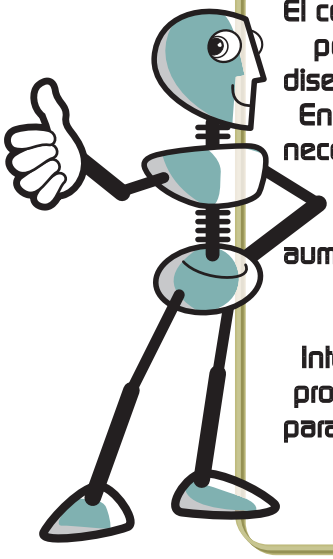


Al pulsar RUN se activan la bobina K1 y el contacto normal abierto K1, lo cual provoca una retención de K1 haciendo que el motor quede encendido permanentemente a pesar de que se suelte RUN, para apagarlo se oprime STOP momentáneamente. El motor también se apaga a causa del relé X1 y en este caso enciende la lámpara LAMP.

Veamos a continuación otra posible solución.



Reflexiones sobre lo visto



El conocimiento de las operaciones lógicas nos permite un mejor desempeño en el análisis y diseño de programas en Lenguaje de Contactos. En este nivel del curso ya tienes las nociones necesarias para realizar programas en Lenguaje de Contactos para una gran cantidad de aplicaciones. En la siguiente lección aumentaremos este potencial aprendiendo sobre el uso de los temporizadores y contadores.

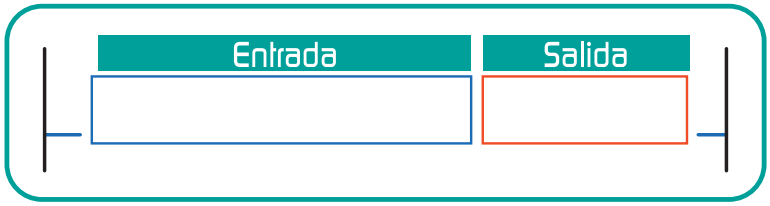
Intenta una búsqueda en Internet para obtener programas que te permitan escribir aplicaciones para PLC. Te recomiendo el WINSPS versión 3.22 en español el cual puedes descargar en la siguiente dirección:

http:// 193.108.217.183/ATProducts/plcwebsite/englisch/default.htm

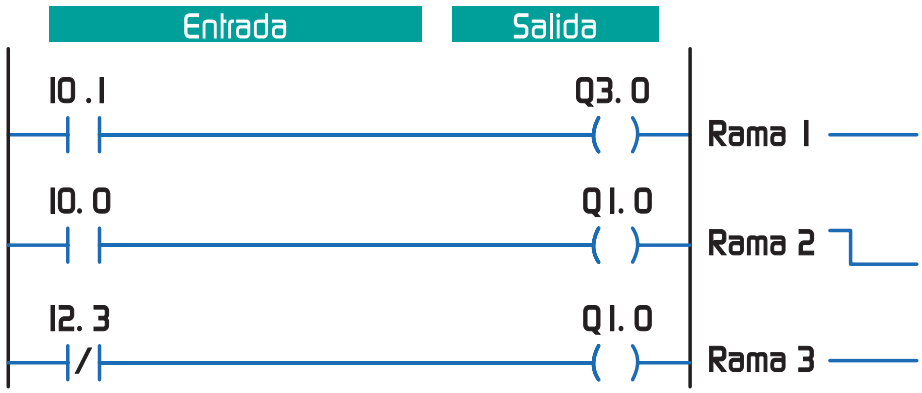
Reglas del Lenguaje de Plano de Contactos

El esquema se realiza entre dos líneas o barras de alimentación dispuestas verticalmente a ambos lados del diagrama, entre ellas se dibujan los elementos del lenguaje.

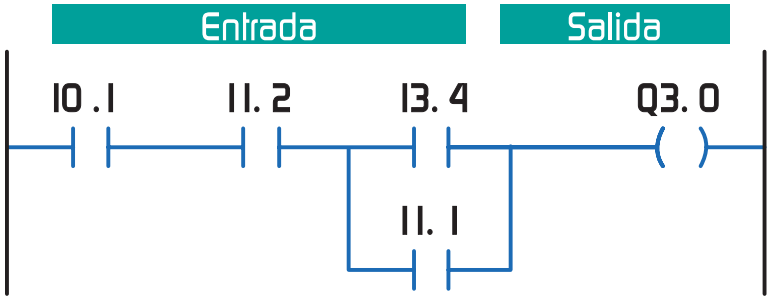
A la derecha del esquema se ubican los elementos de salida y a la izquierda los de entrada



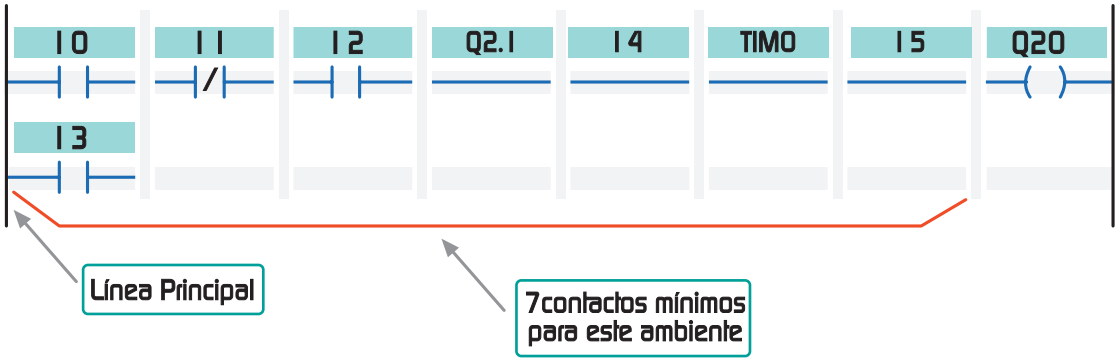
El diagrama puede tener varias ramas o escalones.



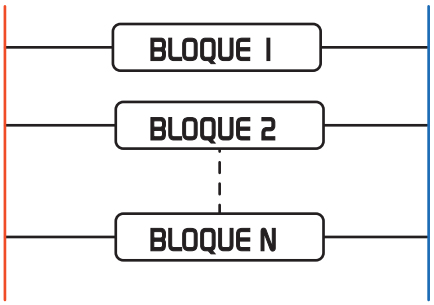
Cada rama permite ubicar varios elementos de entrada pero sólo uno de salida.



La programación en cada bloque de contactos se realiza en el orden de izquierda a derecha.

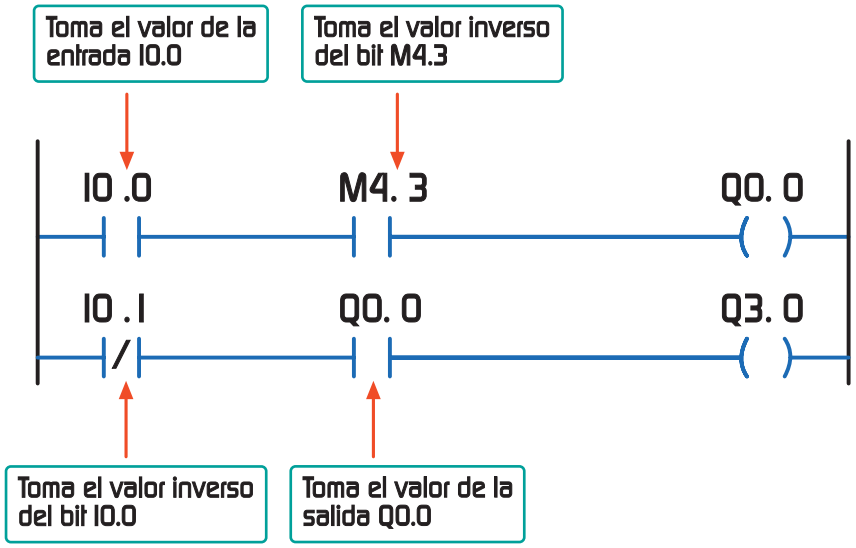


El sentido de programación de los bloques de contactos de un programa de ejecuta en el sentido de arriba abajo.

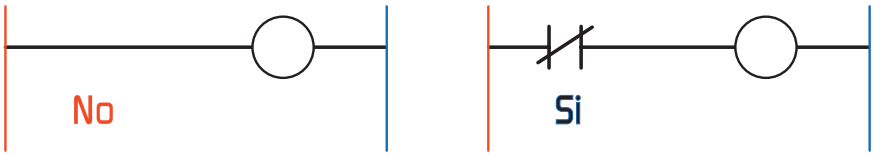


Reglas del Lenguaje

El número de contactos que se pueden colocar en un bloque, desde el comienzo de la linea principal hasta la salida, es ilimitado. **Limitación práctica:** Anchura del papel cuando queramos sacar el programa por impresora o anchura en el amhiente de programación.



No se puede conectar una salida directamente a la línea principal, en estos casos se intercala un contacto cerrado de una marca o bit o relé interno cualquiera.



Con relación a los contactos, tenga presente lo siguiente:

Contactos de entrada	El número de contactos abiertos o cerrados que se pueden utilizar en un programa, por cada una de las entradas, es ilimitado, es decir que, se puede repetir el mismo número de contacto cuantas veces se quiera.
Contactos de salida	El número de salidas o bobinas de salida o relés de salida OUT es fijo, por lo que no se puede repetir un mismo número de salida. Sin embargo, el número de contactos asociados a cada una de ellas es ilimitado.

Elementos del Lenguaje

Se clasifican en elementos de entrada y salida. Su estado es evaluado por el PLC para determinar un valor lógico, que recibe distintas denominaciones dependiendo del contexto de trabajo.

A continuación, se presenta una tabla donde se relacionan las denominaciones de los contextos con las usadas en este curso (activo e inactivo).

Valores Lógicos		
Contexto	**Activo**	**Inactivo**
Informática	True	False
	Verdadero	Falso
Algebra de Boole	V	F
	1	0
Electrónica Digital	High	Low
	H	L

Elementos de Entrada:

Los contactos, únicos elementos que se colocan a las entradas, son de tipo normal abierto ⊣ ⊢ y normal cerrado ⊣/⊢.

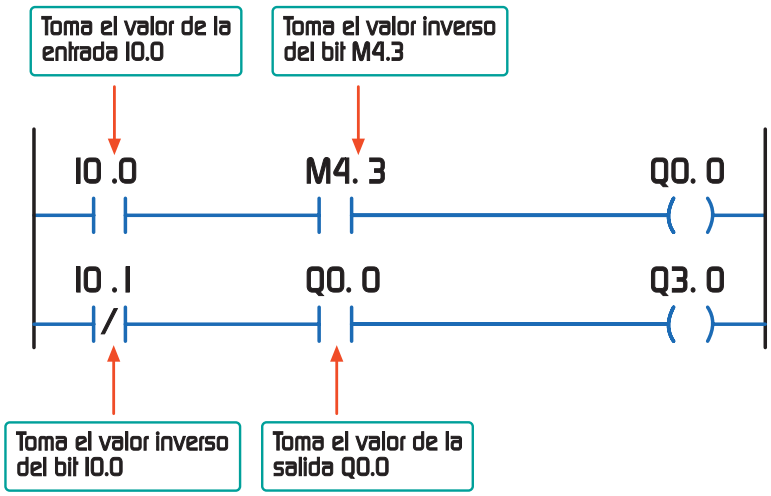
Encima del contacto se escribe la variable a la cual hace referencia. El valor lógico del contacto depende directamente del valor lógico de su variable. Para los contactos normal abierto, si la variable es V el contacto también será V y, si la variable es F el contacto será F.

Los contactos normal cerrado toman el valor inverso de su variable, si la variable es V el contacto será evaluado como F y viceversa.

Contacto	Variable	
Normal Abierto	V	V
	F	F
Normal Cerrado	F	V
	V	F

Las variables a las cuales pueden referirse los contactos son:

Variable	Contacto	Ejemplo
Entradas Digitales	Entrada Digital	**I1.3**
Salidas Digitales	Valor Salida Digital	**Q0.0**
Bits en Memoria	Bit localizado en la memoria con posibilidad de ser definido por el usuario (también se conocen como relés internos, bits de estado, control de temporizadores y contadores)	



Elementos de Salida:

A los elementos de salida, al igual que para los de entrada, se les escribe encima la variable a la cual están referidos. El valor lógico del elemento de salida es determinado por el PLC a partir de los elementos de entrada.
El elemento de salida principal se denomina Asignación o Bobina.
Las Bobinas son de tres tipos: Asignación Simple, Puesta a uno (SET) ⊣(S)⊢ y Puesta a Cero (RESET) ⊣(R)⊢.

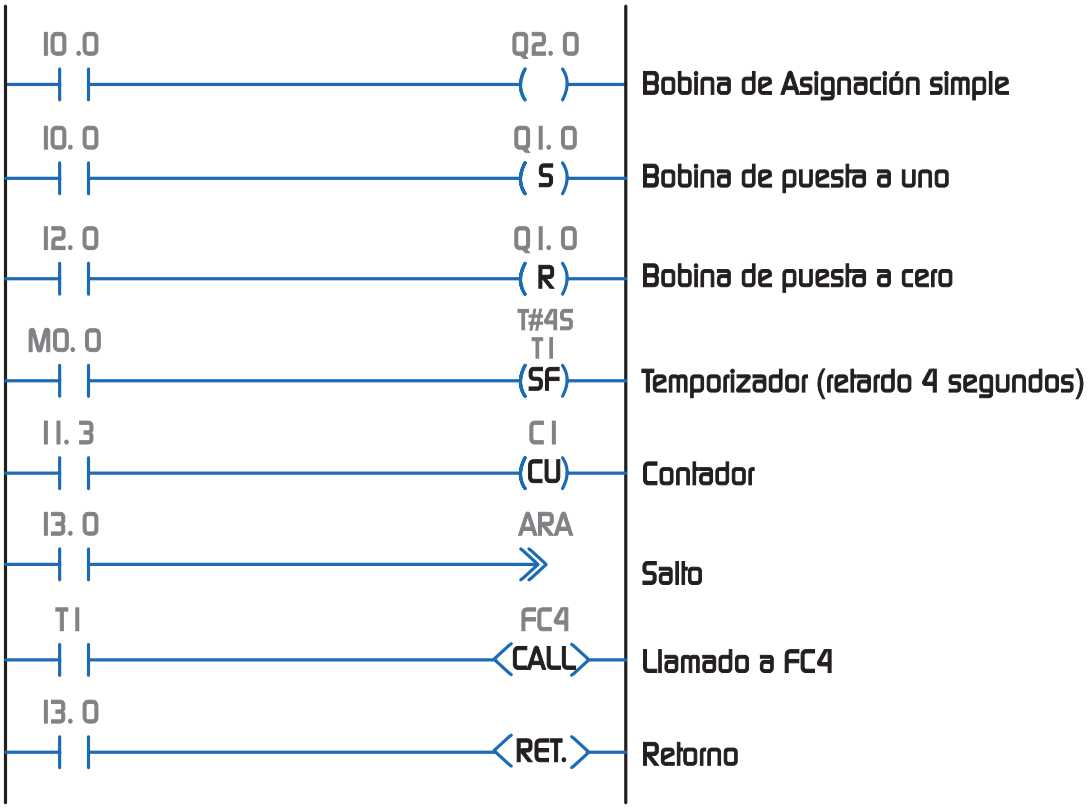
Bobina de Asignación Simple: Su valor lógico es igual al resultado de la combinación de los contactos en la rama. Si el resultado de la evaluación de los contactos es V entonces la bobina será V; si el resultado es F, la bobina toma el valor F.

Bobina de Puesta a Uno (SET): Cuando llega el valor V a esta bobina, su variable asociada se pone y mantiene indefinidamente en estado V sin importar que a la bobina llegue posteriormente un valor F. Una vez retenida la variable en el valor V, para pasarla a F será necesario el uso de una bobina de puesta a 0 (cero).

Bobina de Puesta a Cero (RESET): Cuando llega un valor V a esta bobina, su variable asociada se pone y mantiene indefinidamente en estado F sin importar que a la bobina llegue posteriormente un valor F. La única manera de cambiar el estado de la variable es usando una Bobina de Puesta a Uno.

Otros tipos de elementos de salida son:

Temporizadores, Contadores, Saltos, Llamadas y Retornos, algunos de los cuales se explicarán en las lecciones siguientes.

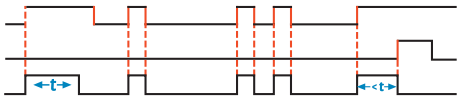
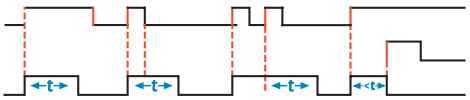
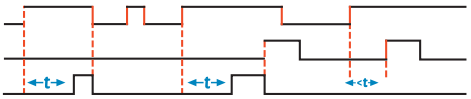
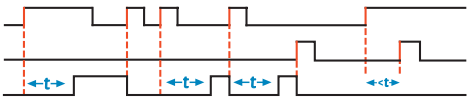
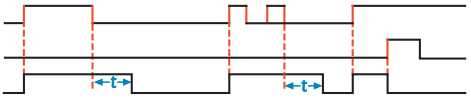


El estudio del lenguaje LADDER resulta sencillo si nos remitimos a nuestros conocimientos previos de automatización con relés y contactores, ya que las reglas y elementos del lenguaje son muy similares.

Lección 9
Temporizadores y Contadores

Temporizadores

En las tablas, a continuación, se listan los diversos tipos de temporizadores disponibles en lenguaje de plano de contactos especificando su simbología y diagrama de tiempos.

Función	Símbolo		Diagrama de Tiempos
Temporizador de Impulso			
Disparo	T#30S TI (SI)	Disparo	
Reposición	TI (RT)	Reposición	
Detención	TI (TH)	Detención	
Temporizador de Impulso Prolongado -Monoestable-			
Disparo	T#30S TI (SV)	Disparo	
Reposición	TI (RT)	Reposición	
Detención	TI (TH)	Detención	
Temporizador de Impulso de Retardo de Conexión			
Disparo	T#30S TI (SE)	Disparo	
Reposición	TI (RT)	Reposición	
Detención	TI (TH)	Detención	
Temporizador de Retorno de Conexión Memorizado			
Disparo	T#30S TI (SS)	Disparo	
Reposición	TI (RT)	Reposición	
Detención	TI (TH)	Detención	
Temporizador de Retardo de Conexión			
Disparo	T#30S TI (SA)	Disparo	
Reposición	TI (RT)	Reposición	
Detención	TI (TH)	Detención	

Definición del Tiempo de Retardo:

El Tiempo de Retardo (**T#xx**) se establece:
En la parte superior del símbolo de disparo del temporizador, en segundos o en milisegundos.
Mediante el formato T#multiplicador.escala, como producto entre la base de tiempo estipulada por la escala y multiplicador.

Así que

Retardo = Base de tiempo * multiplicador

Observe en la tabla los posibles valores de base de tiempo.

Valores Base de Tiempo		
Valor de Escala	**Base de Tiempo**	**Ejemplo**
0	0.01 S	T#20.0 Retardo= 0.2 S
1	0.1 S	T#15.1 Retardo= 1.5 S
2	1 S	T#30.1 Retardo= 30 S
3	10 S	T#60.3 Retardo= 600 S

La salida del temporizador es cualquier contacto al cual se le haya asignado como variable de referencia el nombre del temporizador.

Contadores

Las opciones de programación de los contadores son:

Asignación: 

Con éste elemento se define el nombre del contador a ser utilizado y el valor inicial de la cuenta.

Cuenta Ascendente: 

Un flanco de subida en la entrada del elemento hace que el valor de la cuenta se incremente en uno. El flanco de subida se define como el cambio de una señal de F a V.

Cuenta Descendente: 

Con un flanco de subida se hace que el valor de la cuenta descienda en uno.

Reposición: 

Obliga a que el contador se reinicie con su valor inicial.

La salida de un contador es un contacto cuya variable de referencia sea el nombre del contador, la variable es F mientras el valor de la cuenta sea 0 y es V si la cuenta es diferente de 0.

Ejemplos de Aplicación:

Control de apertura y cierre de puerta con luz de pasillo temporizada.

Supongamos que para proveer el control automático de una puerta disponemos de los siguientes componentes:

Motor M1 eléctrico para abrir y cerrar la puerta

Contactores:

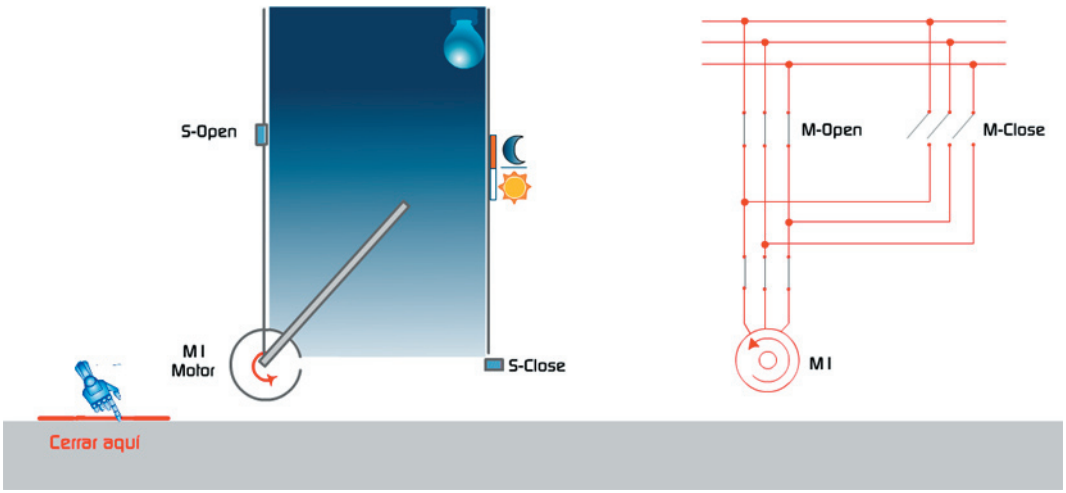
1. M ON, encendido y apagado el motor.
2. M_open, direccionamiento del giro al motor necesario para abrir la puerta.
3. M_close, direccionamiento de giro de cerrar la puerta.
4. Interruptor ABRIR que accionado ordena la apertura de la puerta y sin accionar ordena que se cierre.

Sensores:

1. S_open, fin de carrera puerta totalmente abierta
2. S_close, fin de carrera puerta cerrada.
3. S_Día, tipo Día/Noche que se activa cuando hay suficiente luz solar.
3. Bombilla LUZ, para iluminar el pasillo.
4. Interruptor ILUM, para el encendido manual de la luz del pasillo.
5. Cuando se activa ABRIR la puerta debe abrirse y al desactivarse la puerta debe cerrarse.

Ejemplos de Aplicación:

Motor M1 eléctrico para abrir y cerrar la puerta



Si es de noche, la Bombilla LUZ debe encenderse durante 30 segundos, adicionales al comando de cierre de la puerta; ella también puede encenderse manualmente en cualquier momento.

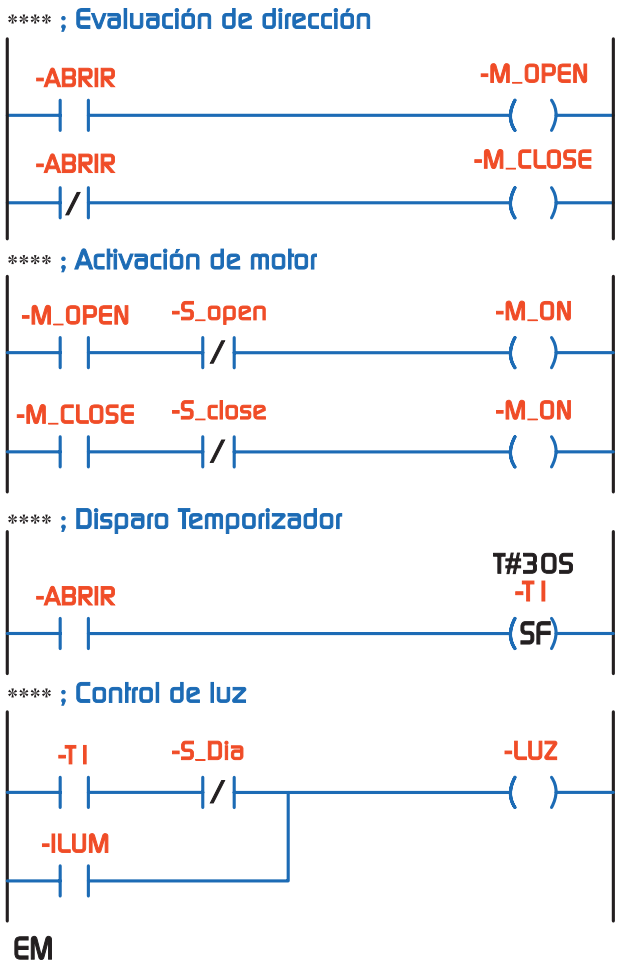
Iniciemos con la asignación de circuitos:

Símbolo	Variable	Descripción
M_ON	Q0.0	V= motor encendido, F= motor apagado
M_open	Q0.1	Sentido de giro para abrir la puerta
M_close	Q0.2	Sentido de giro para cerrar la puerta
ABRIR	I0.0	Interruptor, V= orden de abrir F= orden de cerrar
S_close	I0.1	Sensor fin de carrera de puerta cerrada
S_open	I0.2	Sensor fin de carrera puerta abierta
S_Dia	I0.3	Sensor luz solar
LUZ	Q0.3	Bombilla del pasillo
ILUM	I0.4	Interruptor de luz del pasillo
	T1	Temporizador de Retardo de desconexión

El estado de ABRIR dará los valores para M_open y M_close.

El contactor del motor M_ON debe estar activo hasta alcanzar el fin de carrera S_close si M_close está activo, ó hasta alcanzar S_open si M_open esta activo.

La LUZ debe encender si ABRIR está activo y S_Día es falso (noche) ó si ILUM es activo. Además ABRIR debe disparar un temporizador T1 de retardo a la desconexión por 30 segundos para mantener LUZ encendido.



Ejemplos de Aplicación:

Estampadora

En una línea de producción en serie se estampan 1500 piezas, el proceso inicia al presionar el pulsador y termina cuando se enciende una luz y suena una sirena para dar aviso que se han fabricado las 1500 piezas. Cuando se posiciona una pieza en el área de estampado se acciona la prensa la cual se mantiene presionando la pieza durante 10 segundos, al cabo de los cuales se retira la prensa. Se esperan tres segundos adicionales para que la pieza repose y entonces se retira. Desde un proveedor se desplazará, por gravedad, otra pieza al área de estampado.

Proceso del Estampado

1. Posicionamiento de la pieza (en bruto) en el área de estampado.
2. Accionamiento de la prensa.
3. Presión de 10 seg.
4. Apertura de la prensa.
5. Reposo de la pieza 3 seg.
6. Retiro de la pieza fabricada

El sistema consta de los siguientes componentes:

1. Z1, cilindro de simple efecto, encargado de realizar el estampado.
2. Y1, electro válvula comandado.
3. S_HEAT, sensor de fin de carrera, el cual detecta cuando Z1 está completamente extendido.
4. Z2, cilindro de simple efecto, encargado de expulsar la pieza.
5. Y2, electro válvula comando.
6. S_INI, 2 sensores de fin de carrera, que indica cuando Z2 está retraído.
7. S_EXP, sensor que indica cuando Z2 está extendido.
8. S0, sensor de proximidad que indica cuando la pieza se halla en el área de estampado.
9. LUZ, luz indicadora de fin de producción.
10. SIREN, sirena indicadora de fin de producción.
11. START, pulsador inicio de producción

Iniciemos con la asignación de circuitos:

Símbolo	Variable	Descripción
START	I0.0	Pulsador, accionado por el operario para iniciar la producción
S_HEAT	I0.1	Fin de carrera, indica que Z1 está en posición de estampado.
S_INI	I0.2	Fin de carrera, indica que Z2 está en su posición inicial.
S_EXP	I0.3	Fin de carrera, indica que Z2 acaba de expulsar la pieza.
S0	I0.4	Detector de proximidad, indica que hay una pieza en el área de estampado.
Y1	Q0.0	Cilindro de simple efecto, conforma la prensa de estampado.
Y2	Q0.1	Cilindro de simple efecto, para retirar la pieza procesada.
LUZ	Q0.2	Bombilla, indica al operario que la producción se completó.
SIREN	Q0.3	Sirena, indica al operario que la producción se completó.
T_ ESTAMP	T1	Temporizador de retardo de conexión, utilizado para contar los 10 segundos de prensado de la pieza.
T_COLD	T2	Temporizador de retardo de conexión memorizado, para contar el tiempo de reposo de la pieza.
CUENTA	C1	Contador, para contar los 1500 estampados.
MO	M0.0	Marcador interno, 1= en producción, 0= fin de producción

MO, bit en memoria, señala que el proceso de producción esta en marcha, se activa al presionar el botón de START y se desactiva cuando el contador desciende hasta 0 (cero).

Ejemplos de Aplicación:

Estampadora

El contador CUENTA es inicializado por START. MO debe encargarse de que la CUENTA sólo se inicie una vez durante la producción ya que el botón START podría presionarse varias veces, accidentalmente, durante el proceso.

(SET M0) = (NOT MO) AND (START).
(C1, #1500) = (NOT MO) AND (START).

Z1 avanza cuando se está en producción, caso en el cual una hay pieza localizada en el área de estampado, y por ende el cilindro expulsor Z2 esta retraído. (SET Y1) = M0 AND INI AND S0.

Z1, se retraerá cuando T_STAMP completa su tiempo de retardo.
(RESET Y1) = T_STAMP.

El disparo a T_STAMP se efectúa cuando Z2 alcanza la posición HEAT.
(T_STAMP, T#10Seg)= HEAT.

Se aprovecha HEAT para disparar T_COLD, por 3 segundos más.
(T_COLD, T#13Seg)= HEAT.

Al cabo del retardo de T_COLD se inicia el proceso de expulsión.
(SET Y2)= T_COLD.

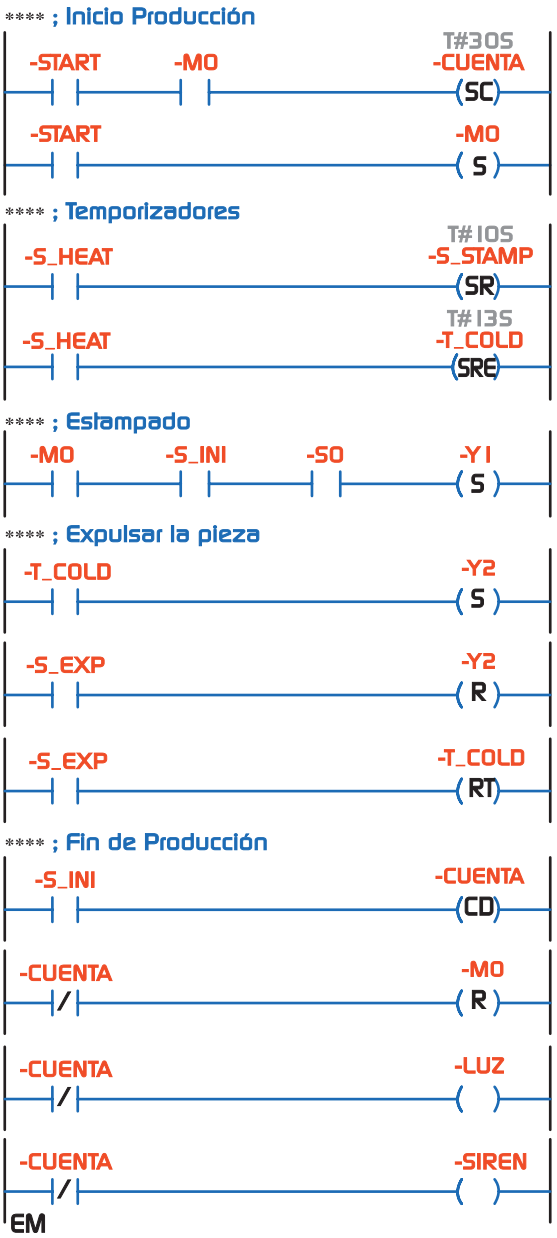
Z2 se retrae con la señal S_EXP, siendo necesario efectuar la reposición del temporizador T_COLD
(RESET Y2)= S_EXP.
(RESET T_COLD)= S_EXP.

Al alcanzar Z2 a INI se aprovecha para el contador CUENTA descienda
(conteo descedente CUENTA)= INI.

En este punto el sistema está listo para un nuevo ciclo de estampado. Al cumplirse los 1500 ciclos el valor de la cuenta habrá llegado a 0 (cero), con lo cual se debe desactivar MO y encender la luz y la sirena.

(RESET MO)= (NOT CUENTA).
LUZ= (NOT CUENTA).
SIREN= (NOT CUENTA).

El Programa




```
; asignación se símbolos para estampadora
; ENTRADAS

I 0.0   START     ; PULSADOR PARA ARRANCAR LA PRODUCCION
I 0.1   S_HEAT    ; FIN DE CARR.  Y1, LA PIEZA SE ESTA PRENSANDO
I 0.2   S_INI     ; EL EXPULSADOR, Y2, ESTA RETRAIDO
I 0.3   S_EXP     ; EL EXPULSADOR, Y2, HA RETIRADO LA PIEZA
I 0.4   SO        ; LA PIEZA ESTA EN POSICION PARA SER ESTAMPADA

; INTERNAS
T1      T_STAMP   ; TEMPORIZADOR RETARDO DE CONEX.  10 SEG
T2      T_COLD    ; TEMPORIZADOR RETARDO DE CONEX. MEMORI ENFRIAR PIEZA
C1      CUENTA    ; CONTADOR DE PIEZAS ESTAMPADAS
M0.0    MO        ; MARCADOR INTERNO.  1= EN PRODUCCION  0= FIN DE PRODUCCION

; SALIDAS
Q0.0    Y1        ; CILINDRO DE ESTAMPADO
Q0.1    Y2        ; CILINDRO DE EXPULSION
Q0.2    LUZ       ; LUZ INDICADORA DE FIN DE PRODUCCION
Q0.3    SIREN     ; SIRENA INDICADORA DE FIN DE PRODUCCION
```

Reflexiones sobre lo visto:

El potencial de lenguaje de contactos crece fuertemente al incluir en su estructura temporizadores y contadores. No obstante, estamos al límite de sus capacidades por lo que encontraremos aplicaciones en las cuales la programación en lenguaje de contactos es más compleja de lo sugerido por la misma solución.