TÉCNICO PLC

> ESPECIALIZACIÓN SUPERIOR

> CONTROLADOR LÓGICO PROGRAMABLE (PLC)

AMÉRICA EDUCA, Uruguay
Estudia, Tutores en Red y
La Academia de Paso de los Toros
presentan este curso.

Responsable técnico docente:
Ingeniero Jorge Velazco Barranque
Registro 1139 02
Contacto: +59897436180

NUESTRA RED:
Ingeniería Electrónica:
www.facebook.com/groups/electronicauruguay
Equipo de tutores:
https://www.facebook.com/groups/tutoresenred
https://www.facebook.com/desarrollo.permanente
https://www.facebook.com/recursosabiertos
https://www.facebook.com/americaeduca
https://www.facebook.com/uruguayestudia

CORREOS:
educa@terra.com
uruguayestudiaenlinea@gmail.com

Las empresas que piensan en el futuro se encuentran provistas de modernos dispositivos electrónicos en sus máquinas y procesos de control. En la actualidad, las fábricas automatizadas deben proporcionar en sus sistemas: alta confiabilidad, gran eficiencia y flexibilidad. Una de las bases principales de dichas fábricas es un dispositivo electrónico llamado Controlador Lógico Programable (PLC)

Hoy los Controladores Lógicos Programables son diseñados usando lo último en diseño de microprocesadores y circuitería electrónica, esto proporciona una mayor confiabilidad en su operación, así como también en las aplicaciones industriales donde existen peligros ambientales: alta repetibilidad, elevadas temperaturas, ruido ambiente o eléctrico, suministro de potencia eléctrica no confiable, vibraciones mecánicas, entre otros.

Nuestra meta es enseñar el funcionamiento interno y la programación de este tipo de controladores, asimismo exponer algunas de sus aplicaciones en la industria.

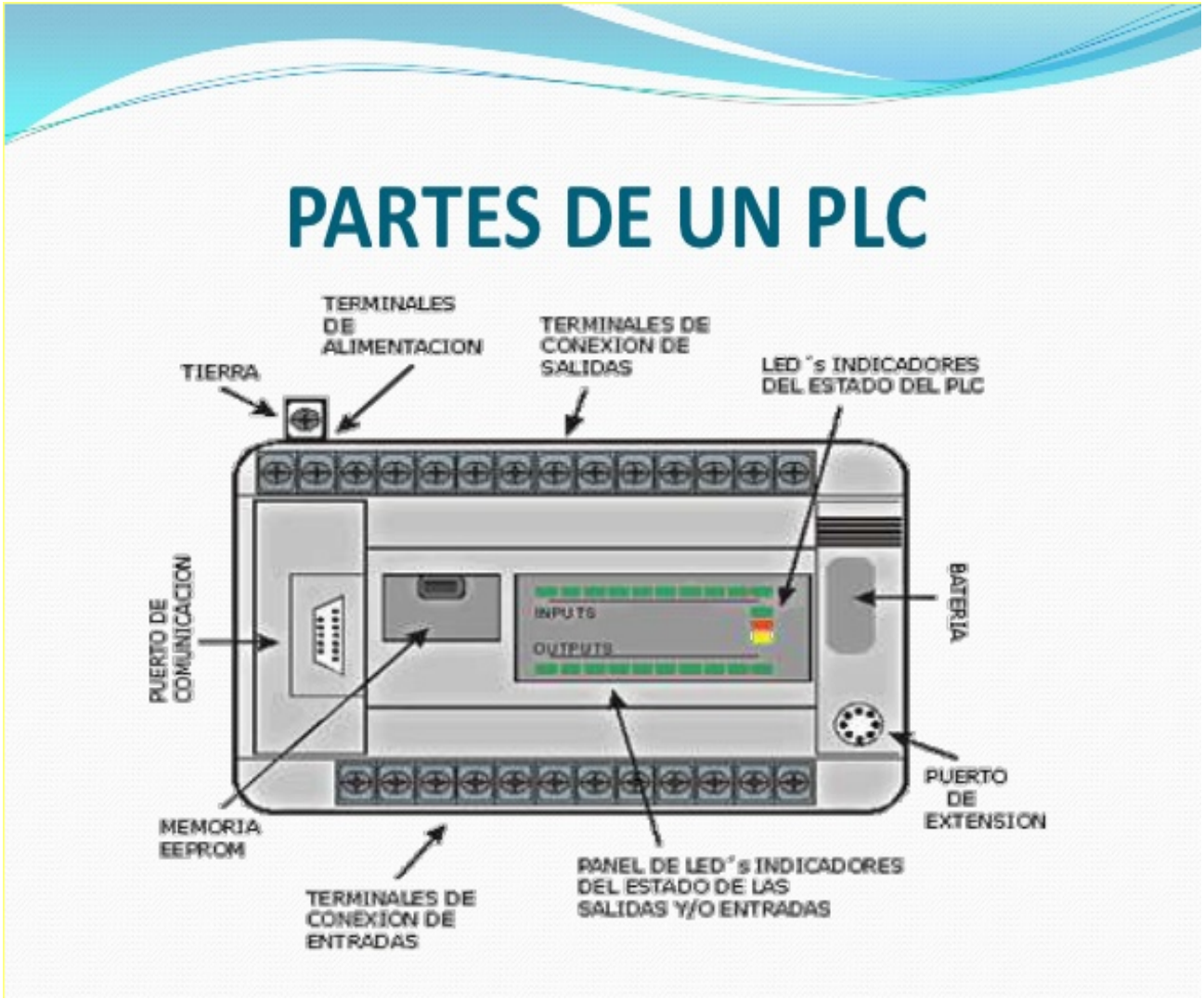
En LA ACADEMIA, a través de los cursos de capacitación, pretendemos crear un espacio deformación y entrenamiento en el área de la automatización industrial; para estudiantes, profesores, operadores, técnicos e ingenieros que decidan completar la propiaformación. El diseño del manual está elaborado con criterios eminentemente prácticos, para facilitar un estudio ágil y actualizado de cada uno de los temas.
El objetivo de éste y de todos los cursos es ofrecer un sistema de aprendizaje dinámico e interactivo de clases teórico-prácticas, en el cual el alumno avance en la especialidad, ejecutando de una forma práctica los conocimientos desarrollados en las clases teóricas. Siempre con una visión real y profesional, para aplicar estos conocimientos a las necesidades de su empresa, tanto en el campo de mantenimiento,como en el de producción.

Esperamos haber construido una herramienta que les permita apropiarse significativamente del nuevo saber.

Para contribuir al logro de los objetivos reseñados, sus comentarios al final del curso serán de inestimable utilidad.

Ingeniero Jorge Velazco Barranque

ESPECIALIZACIÓN SUPERIOR
Controlador Lógico Programable (PLC)



1 Conceptos básicos

Introducción

El control automático, como actualmente lo conocemos, tiene su primer antecedente en el Regulador de Watt, el famoso sistema que controlaba la velocidad de una turbina de vapor en el año 1774. A partir de aquel regulador, se desarrollaron innumerables aplicaciones prácticas.
Las industrias de procesos contiguos tuvieron sus primeras necesidades al requerir mantener las variables de proceso en un determinado rango, a fin de lograr los objetivos de diseño.
Las primeras industrias realizaban el control de las variables de forma manual, a través de operadores que visualizaban el estado del proceso mediante indicadores ubicados en las cañerías y/o recipientes y equipos.
El operador conocía el valor deseado de la variable a controlar, y en función del error tomaba acciones correctivas sobre un elemento final de control a fin de minimizarlo.
Por supuesto, el control manual era descentralizado. Cuando las plantas de producción crecieron y se tornaron más complejas, se requirió cada vez mayor cantidad de mano de obra.
El primer intento de reemplazar al hombre en las tareas de control se realizó a través de elementos mecánicos. Mecanismos como las válvulas de control de nivel a flotante permitieron al hombre dedicarse a estas tareas.
Sin embargo, el hecho de que el elemento mecánico de control estuviera ubicado directamente sobre el proceso, mantenía la obligación de ir al campo para conocer el verdadero estado de las variables, así como dejaba expuesto al medio ambiente a elementos de regulación delicados.
A medida que las plantas crecían, fue surgiendo la necesidad de tener más información en forma ordenada y accesible. De esta forma, comenzaron aparecer los primeros tableros de control, muchas veces ubicados cerca de los equipos de proceso, y con frecuencia transportando la variable a medir hasta el indicador instalado en el panel.

1 . 1 Conceptos básicos. Definiciones

¿Qué es un PLC?

Según lo define la Asociación Nacional de Fabricantes Eléctricos de los Estados Unidos un PLC – Programable Logic Controller (Controlador Lógico Programable) es un dispositivo digital electrónico con una memoria programable para el almacenamiento de instrucciones, permitiendo la implementación de funciones específicas como ser: lógicas, secuenciales, temporizadas, de conteo y aritméticas; con el objeto de controlar máquinas y procesos.
También se puede definir como un equipo electrónico, el cual realiza la ejecución de un programa de forma cíclica. La ejecución del programa puede ser interrumpida momentáneamente para realizar otras tareas consideradas más prioritarias, pero el aspecto más importante es la garantía de ejecución completa del programa principal.
Estos controladores son utilizados en ambientes industriales donde la decisión y la acción deben ser tomadas en forma muy rápida, para responder en tiempo real.
Los PLC son utilizados donde se requieran tanto controles lógicos como secuenciales o ambos a la vez.

1.2 Campos de aplicación

El PLC por sus especiales características de diseño tiene un campo de aplicación muy extenso. La constante evolución del hardware y software amplía constantemente este campo, para poder satisfacer las necesidades que se detectan en el espectro de sus posibilidades reales.
Su utilización se da fundamentalmente en aquellas instalaciones en donde es necesario un proceso de maniobra, control y señalización. Por tanto, su aplicación abarca desde procesos de fabricación industriales de cualquier tipo a transformaciones industriales, o control de instalaciones, entre otras.
Sus reducidas dimensiones, la extremada facilidad de su montaje, la posibilidad de almacenar los programas para su posterior y rápida utilización, la modificación o alteración de los mismos, hace que su eficacia se aprecie principalmente en procesos en que se producen necesidades tales como:

- Espacio reducido
- Procesos de producción periódicamente cambiantes
- Procesos secuenciales
- Maquinaria de procesos variables
- Instalaciones de procesos complejos y amplios
- Chequeo de programación centralizada de las partes del proceso

Ejemplos de aplicaciones generales:

- Maniobra de máquinas
- Maquinaria industrial de plástico
- Máquinas transfer
- Maquinaria de embalajes
- Maniobra de instalaciones: instalación de aire acondicionado, calefacción
- Instalaciones de seguridad
- Señalización y control

1.3 Ventajas e inconvenientes

Sabemos que no todos los autómatas ofrecen las mismas ventajas sobre la lógica cableada, ello es debido, principalmente, a la variedad de modelos existentes en el mercado y las innovaciones técnicas que surgen constantemente. Tales consideraciones obligan a referirse a las ventajas que proporciona un autómata de tipo medio.

Ventajas

- Menor tiempo empleado en la elaboración de proyectos, debido a que no es necesario dibujar previamente el esquema de contactos, es preciso simplificar las ecuaciones lógicas, ya que por lo general la capacidad de almacenamiento del módulo de memoria es lo suficientemente grande.
- La lista de materiales queda sensiblemente reducida, y al elaborar el presupuesto correspondiente eliminaremos parte del problema que supone el contar con diferentes proveedores, distintos plazos de entrega.
- Posibilidad de introducir modificaciones sin cambiar el cableado ni añadir aparatos.
- Mínimo espacio del tablero donde se instala el **autómata programable**.
- Menor costo de mano de obra de la instalación.
- Economía de mantenimiento. Además de aumentar la fiabilidad del sistema, al eliminar contactos móviles, los mismos autómatas pueden indicar y detectar averías.

- Posibilidad de gobernar varias máquinas con un mismo autómata.
- Menor tiempo para la puesta en funcionamiento del proceso al quedar reducido el tiempo de cableado.
- Si por alguna razón la máquina queda fuera de servicio, el autómata sigue siendo útil para otra máquina o sistema de producción.

Inconvenientes

- Como inconvenientes podríamos hablar, en primer lugar, de que hace falta un programador, lo que obliga a adiestrar a uno de los técnicos en tal sentido. Esta capacitación puede ser tomada en distintos cursos, inclusive en universidades.
- El costo inicial.

1 . 4 Reseña Histórica

Los PLC fueron introducidos a fines de los años 60. La razón de su aparición fue la necesidad de eliminar los complicados y costosos sistemas de control de máquinas basados en relés. Bedfor Associates propuso algo llamado Controlador Modular Digital (MODICON) a la General Motors. Al mismo tiempo, otras compañías propusieron esquemas basados en computadoras, uno de los cuales fue PKP-8. El MODICOM 084 llegó a ser el primer PLC en producción a escala comercial.
Cuando hay cambios en los requerimientos de producción, éstos involucran al sistema de control. Estas modificaciones llegan a ser muy caras si los cambios requeridos son frecuentes. Debido a que los relés son aparatos mecánicos, éstos tienen una vida limitada que obliga a apegarse a estrictos programas de mantenimiento. El encontrar las fallas en uno de estos sistemas, es una tarea complicada cuando involucra una cantidad importante de relés.
Estos nuevos controladores debían ser fáciles de programar por los ingenieros de mantenimiento o de planta. También debían ser capaces de funcionar en los agresivos ambientes industriales. La forma de lograr esto fue usar técnicas de programación con las que los programadores estaban familiarizados y reemplazar los relés mecánicos con elementos electrónicos de estado sólido.
A mediados de los años 70 los PLC comenzaron a tener habilidades de comunicación. El primer sistema de comunicación fue el MODBUS de MODICON. Ahora los controladores se podían comunicar entre sí para coordinar el accionar de un conjunto de máquinas. También se les agregaron capacidades de transmitir y recibir voltajes variables que le permitían recibir señales analógicas. Desdichadamente, la carencia de estandarización en estos sistemas, unido a los protocolos y redes físicas, originó la decadencia de su aplicación.
Durante los años 80 se apreció un intento por estandarizar las comunicaciones con el protocolo de automatización de manufactura de la General Motors (MAP) Al mismo tiempo, se tendió a la miniaturización de los equipos y la utilización de lenguajes simbólicos de programación en computadoras personales o programadoras portátiles. Hoy en día los PLC más pequeños son de tamaño de un sólo relé.
En los 90 se ha visto una reducción gradual en la introducción de protocolos nuevos, y la modernización de las capas físicas de algunos de los protocolos más populares que sobrevivieron a los años 80. El último modelo ha tratado de reunir los lenguajes de los PLC bajo un estándar internacional único.
Ahora se cuenta con controladores programables con función de diagramas de bloques, lista de instrucciones, lenguajes de programación C o texto estructurado, todo al mismo tiempo. También se ha visto que se están introduciendo computadoras personales para reemplazar en algunas aplicaciones específicas a los PLC. Es el caso de la General Motors, que ha llevado sus sistemas a control basado en computadoras.

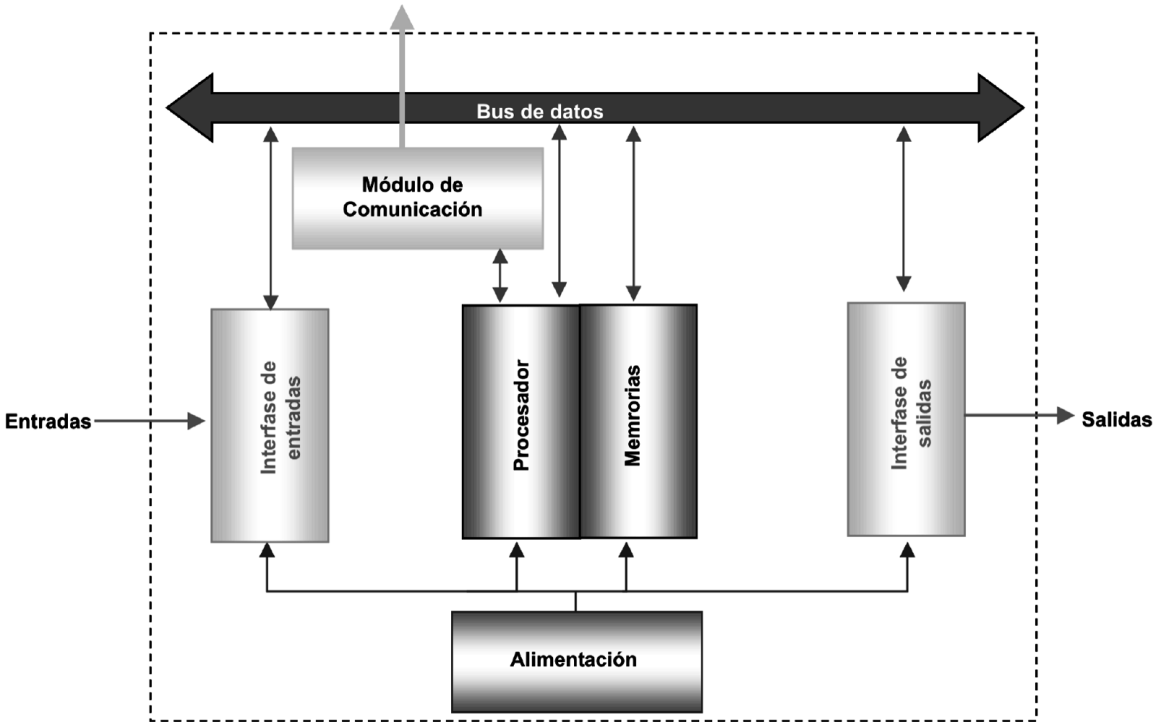
2 Estructura de un PLC

Introducción

La estructura básica de un PLC está compuesta por:

- La CPU.
- Las interfases de entradas.
- Las interfases de salidas.

Esta estructura se puede observar en la figura siguiente:



2.1 Definición y descripción de los componentes de la estructura básica de un PLC

2.1.1

Procesador: es el "cerebro" del PLC, el responsable de la ejecución del programa desarrollado por el usuario.

Tareas Principales:

- Ejecutar el programa realizado por el usuario.
- Administración de la comunicación entre el dispositivo de programación y la memoria, y entre el microprocesador y los bornes de entrada/ salida.
- Ejecutar los programas de autodiagnósticos.

Para poder realizar todas estas tareas, el procesador necesita un programa escrito por el fabricante, llamado **sistema operativo**. Este programa no es accesible por el usuario y se encuentra grabado en una memoria que no pierde la información ante la ausencia de alimentación, es decir, en una memoria no volátil.

2.1.2 Memoria

Los PLC tienen que ser capaces de almacenar y retirar información, para ello cuentan con memorias. Las memorias son miles de cientos de localizaciones donde la información puede ser almacenada. Estas localizaciones están muy bien organizadas. En las memorias el PLC debe ser capaz de almacenar:

Datos del Proceso:

- Señales de entradas y salidas.
- Variables internas, de bit y de palabra.
- Datos alfanuméricos y constantes.

Datos de Control

- Instrucciones de usuario, programa.
- Configuración del autómata.

Tanto el sistema operativo como el programa de aplicación, las tablas o registros de entradas/ salidas y los registros de variables o bits internos están asociados a distintos tipos de memoria.
La capacidad de almacenamiento de una memoria suele cuantificarse en bits, bytes (grupo de 8 bits), o words (grupo de 16 bits)

Un bit es una posición de memoria que puede tomar valor “0” ó “1”:

Un byte son 8 posiciones de memoria agrupadas:

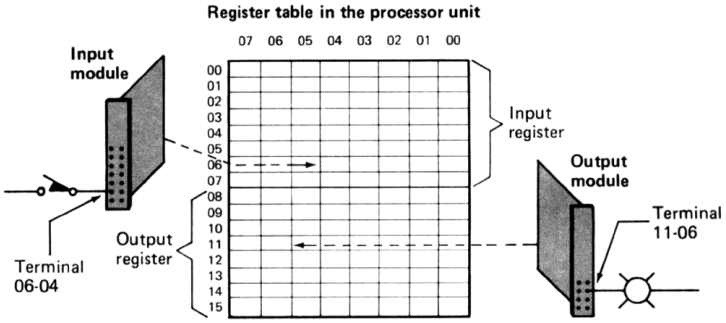
Una palabra o word son 16 posiciones de memoria agrupadas:

El sistema operativo viene grabado por el fabricante. Como debe permanecer inalterado y el usuario no debe tener acceso a él, se guarda en una memoria como las ROM (Read Only Memory), que son memorias cuyo contenido no se puede alterar inclusive con ausencia de alimentación.

Tipos de memoria

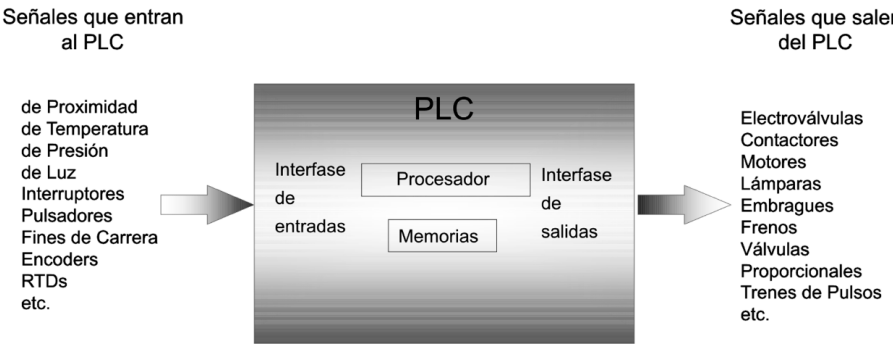
- **La memoria de datos**:
  También llamada tabla de registros, se utiliza tanto para grabar datos necesarios a los fines de la ejecución del programa, como para almacenar datos durante su ejecución y/o retenerlos luego de haber terminado la aplicación. Este tipo de memorias contiene la información sobre el estado presente de los dispositivos de entrada y salida. Si un cambio ocurre en los dispositivos de entrada o salida, ese cambio será registrado inmediatamente en esta memoria.
  En resumen, esta memoria es capaz de guardar información originada en el microprocesador incluyendo: tiempos, unidades de conteo y relés internos.

En la figura que sigue se puede ver como los terminales de entrada o de salida están relacionados con una localización específica en el registro de entradas/ salidas.



Los bornes de conexión de los PLC tienen la misma identificación que la dirección de los registros. Por ejemplo, los bornes de la entrada 001 están relacionados con el lugar de la memoria de datos que se encuentra en la palabra 00, bit 01.
Como puede verse, esta codificación asigna a una única entrada o salida, una terminal y consecuentemente un dispositivo de entrada o salida.

- **Memoria del usuario**:
  Es la memoria utilizada para guardar el programa.
  El programa construido por el usuario debe permanecer estable durante el funcionamiento del equipo, además debe ser fácil de leer, escribir o borrar. Por eso es que se usa para su almacenamiento memorias tipo RAM, o EEPROM. A estas memorias se la llama memoria del usuario o memoria de programa. En el caso de usar memorias tipo RAM será necesario también el uso de pilas, ya que este tipo de memoria se borra con la ausencia de alimentación. En el caso de usar memorias EEPROM la información no se pierde al quitar la alimentación.



RECUERDE que...
La velocidad con que se pueden escribir y leer el estado de las entradas y salidas juega un papel importante en la velocidad de operación del PLC, por tal motivo para guardar esta información se utilizan memorias tipo RAM (Random Access Memory) que son muy rápidas.

2.1.3 Entradas y salidas

Dispositivos de entrada

Los dispositivos de entrada y salida son aquellos equipos que intercambian (o envían) señales con el PLC.
Cada dispositivo de entrada es utilizado para conocer una condición particular de su entorno, como temperatura, presión, posición, entre otras.

Entre estos dispositivos podemos encontrar:

- Sensores inductivos magnéticos, ópticos, pulsadores, termocuplas, termoresistencias, encoders, etc.

Dispositivos de salida

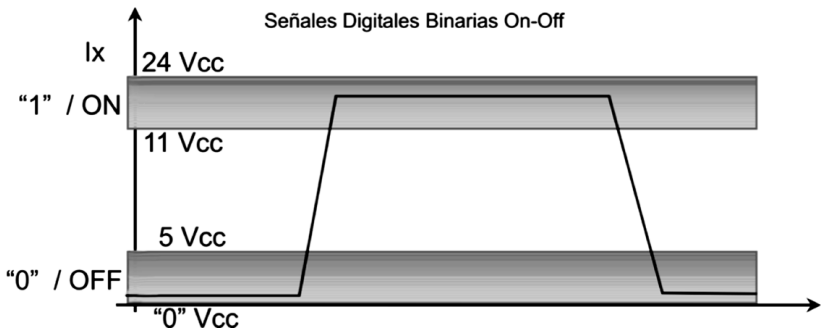
Los dispositivos de salida son aquellos que responden a las señales que reciben del PLC, cambiando o modificando su entorno.

Entre los dispositivos típicos de salida podemos hallar:

- Contactores de motor
- Electroválvulas
- Indicadores luminosos o simples relés

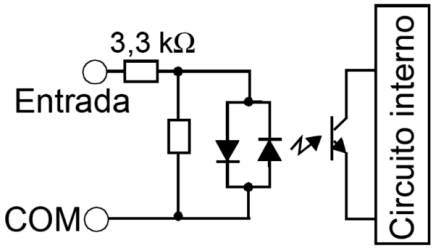
Generalmente los dispositivos de entrada, los de salida y el microprocesador trabajan en diferentes niveles de tensión y corriente. En este caso las señales que entran y salen del PLC deben ser acondicionadas a las tensiones y corrientes que maneja el microprocesador, para que éste las pueda reconocer. Ésta es la tarea de las interfases o módulos de entrada o salida.
Las entradas se pueden clasificar en:

Entradas Digitales: también llamadas binarias u "on-off", son las que pueden tomar sólo dos estados: encendido o apagado, estado lógico 1 ó 0.
Los módulos de entradas digitales trabajan con señales de tensión. Cuando por un borne de entrada llega tensión, se interpreta como "1" y cuando llega cero tensión se interpreta como "0". Existen módulos o interfases de entradas de corriente continua para tensiones de 5, 12, 24 ó 48 Vcc y otros para tensión de110 ó 220 Vca.

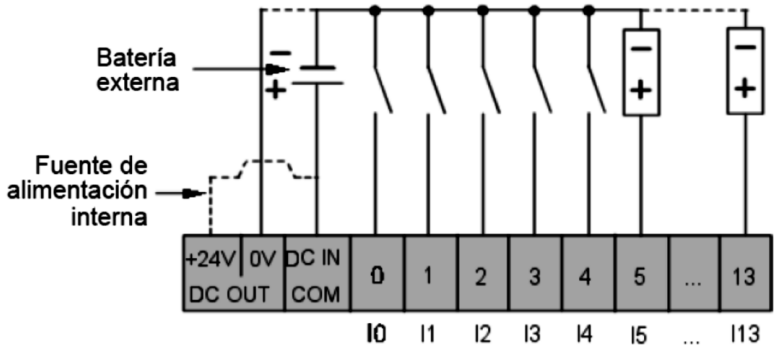


Los PLC modernos tienen módulos de entrada que permiten conectar dispositivos con salida PNP o NPN en forma indistinta. La diferencia entre dispositivos con salida PNP o NPN es como la carga (en este caso la carga es la entrada del PLC) está conectada con respecto al neutro o al positivo.

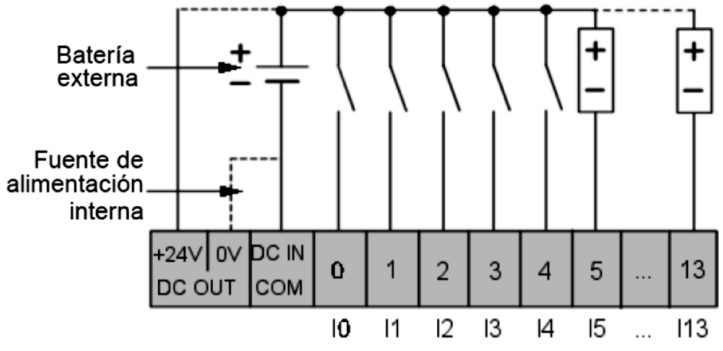
Entrada de común positivo o negativo estándar



Esquema de cableado de entradas de común negativo de CC de un PLC TWIDO



Esquema de cableado de entradas de común positivo de CC de un PLC TWIDO



RECUERDE que...

Las señales digitales en contraste con las señales analógicas no varían en forma continua, sino que cambian en pasos o en incrementos discretos en su rango. La mayoría de las señales digitales utilizan códigos binarios o de dos estados.

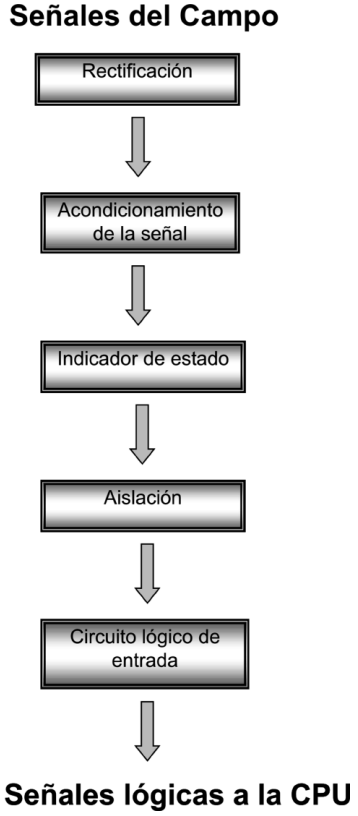
Las entradas discretas, tanto las de la corriente continua como las de la corriente alterna, están compuestas por una estructura típica que se puede separar en varios bloques:

- Rectificador: en el caso de una entrada de corriente alterna, convierte la señal en continua. En el caso de una señal de corriente continua, impide daños por inversión de polaridad.

- Acondicionador de señal: elimina los ruidos eléctricos, detecta los niveles de señal para los cuales conmuta el estado lógico, y lleva la tensión al nivel manejado por la CPU.

- Indicador de estado: en la mayoría de los PLC existe un indicador luminoso por cada entrada. Este indicador (casi siempre un LED) se encenderá con la presencia de tensión en la entrada y se apagará en caso contrario.

- Aislación: en la mayoría de los PLC las entradas se encuentran aisladas para que, en caso de sobretensiones externas, el daño causado no afecte más que a esa entrada, sin perjudicar el resto del PLC.



- Circuito lógico de entrada: es el encargado de informar a la CPU el estado de la entrada cuando éste lo interrogue.

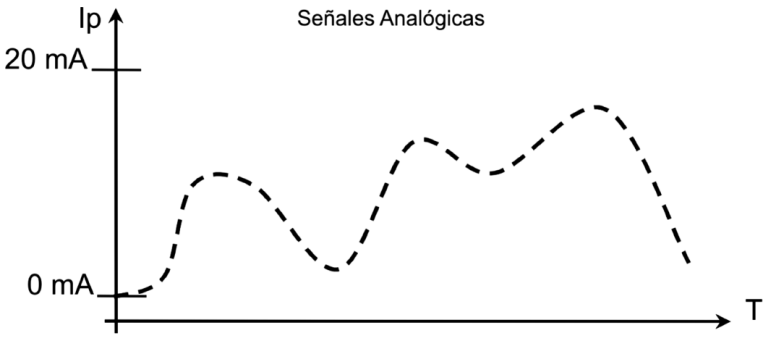
Cuando la señal llega hasta los bornes del PLC tiene que atravesar todos estos bloques. Recorrer este camino le lleva un tiempo que es llamado: **tiempo de respuesta de la entrada**.
Un aspecto a analizar es el mínimo tiempo de permanencia o ausencia de una señal requerido para que el PLC la interprete como 0 ó 1. Si una variable de proceso pasa al estado lógico 1, y retorna al estado 0 en un tiempo inferior al tiempo de respuesta de la entrada, es posible que le PLC no llegue a leerla.

Ejemplo
Si una tarjeta tuviera un tiempo de respuesta de 10 mseg, no sería capaz de identificar con certeza una señal que presentó un pulso de menos mseg. Para aquellos casos en que se produzca esta situación, se requiere tarjetas con capacidad de retención, en las que el estado lógico es sostenido por un período mayor que la duración del pulso de señal.

Entradas Analógicas: estos módulos o interfases admiten como señal de entrada valores de tensión o corriente intermedios dentro de un rango, que puede ser de 4-20 mA, 0-5 VDC o 0-10 VDC, convirtiéndola en un número. Este número es guardado en una posición de la memoria del PLC.
Los módulos de entradas analógicas son los encargados de traducir una señal de tensión o corriente proveniente de un sensor de temperatura, velocidad, aceleración, presión, posición, o cualquier otra magnitud física que se quiera medir en un número para que el PLC la pueda interpretar. En particular es el conversor analógico digital (A/D) el encargado de realizar esta tarea.
Una entrada analógica con un conversor A/D de 8 bits podrá dividir el rango de la señal de entrada en 256 valores (28)

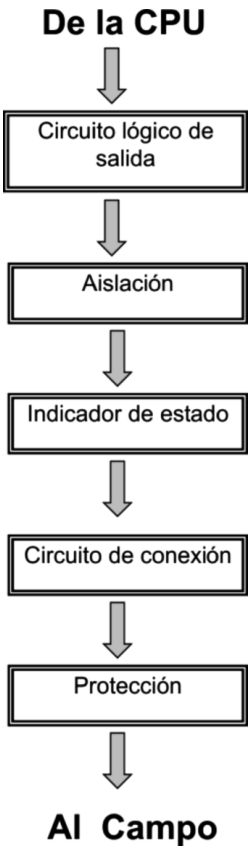
Ejemplo
Si la señal de entrada es de una corriente entre 4 y 20 mA la resolución será de (20-4)/256 = 0.0625 mA. Recordemos que se define como resolución al mínimo cambio que un conversor puede discriminar en su entrada. Si el conversor A/D fuera de 12 bits se podrá dividir el rango de la señal de entrada en 4096 valores (212), con lo que se logra una resolución para una señal de 4-20 mA de (20-4)/4096= 0,0039 mA.

En la medida que el conversor A/D tenga mayor número de bits será capaz de ver o reconocer variaciones más pequeñas de la magnitud física que estamos observando.



Los módulos de salida digital permiten al autómata programable actuar sobre elementos que admitan órdenes de tipo prendido - apagado, todo o nada u “on - off”.
El valor binario de las salidas digitales se convierte en la apertura o cierre de un relé interno del autómata, en el caso de módulos de salidas a relé.
Existe una gran cantidad de módulos de salida discreta, todos ellos con la misma estructura que se presenta a continuación.

RECUERDE que...
Una señal es analógica cuando las magnitudes de la misma se representan mediante variables continuas, análogas (relación de semejanza entre cosas distintas) a las magnitudes que dan lugar a la generación de esta señal.



- Circuitos lógicos de salida: es el receptor de la información enviada por la CPU.
- Aislación: cumple la misma función que en las interfases de entrada.
- Indicador de estado: también tiene la misma función que en la entrada.
- Circuitos de conexión: esta compuesto por el elemento de salida al campo que maneja la carga conectada por el usuario. Existen tres tipos de circuitos de conexión que se describirán más adelante.
- Protección: son internas al PLC y pueden ser fusibles en serie con los contactos de salida, alguna protección electrónica por sobrecarga, o algún circuito RC. Recordar que en caso de que más de una salida use un solo borne de referencia, es éste el que lleva asociada la protección. Por lo cual si esta protección actúa dejarán de funcionar todas las salidas asociadas a ese borne común.

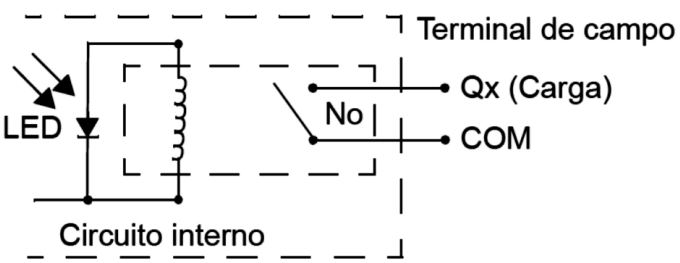
Tiempo de respuesta de la salida: al igual que en las entradas, se denomina tiempo de respuesta de la salida al tiempo que tarda una señal para pasar por todos los bloques. Existen cuatro posibilidades para el circuito de conexión de una salida:

1. Salida a relé:

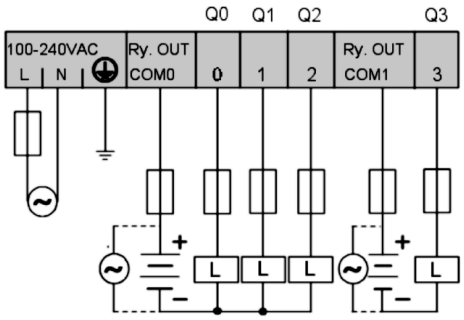
Es una de las más usuales. Con ellos es posible conectar tanto cargas de corriente alterna como continua. Suelen soportar hasta 2A de corriente. Una buena práctica en la instalación es verificar que la corriente máxima que consume la carga esté dentro de las especificaciones de la salida del PLC.

Los tiempos de conmutación de estos tipos de salidas llegan a los 10 mseg. tanto para la conexión como para la desconexión. Algunas cargas son muy problemáticas, por ejemplo las cargas inductivas, que tienen la tendencia a devolver corriente al circuito cuando son conectadas. Siendo la corriente estimada en unas 30 veces a la corriente de consumo nominal. Esto genera picos de voltaje que pueden dañar la salida a la que esta conectada la carga. Para minimizar estos riesgos se utilizan comúnmente diodos, varistores u otros circuitos de protección.

Contacto de salidas de relé



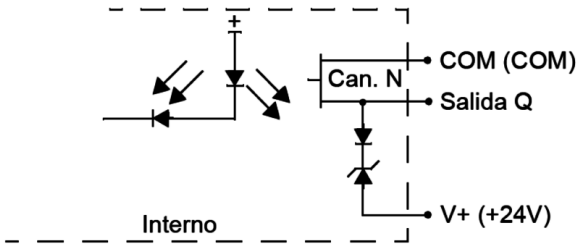
Modelo de cableado de salidas de relé y de alimentación de CA de un TWIDO



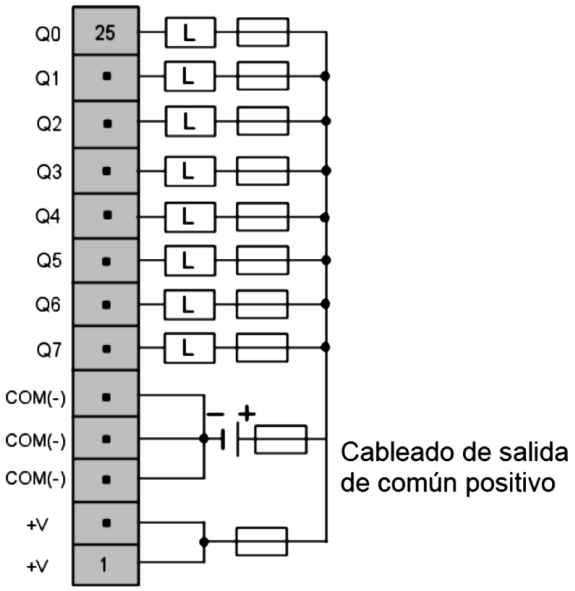
Los relés son internos al PLC. El circuito típico es el que se muestra en la figura de arriba. Cuando el programa active una salida, el PLC aplicará internamente tensión a la bobina del relé. Esta tensión hará que se cierren los contactos de dicho relé. En ese momento una corriente externa pasará a través de esos contactos y así se alimentará la carga. Cuando el programa desactiva una salida, el PLC desactiva la bobina abriendo así los contactos.

2. Salidas a transistor:
Sólo son capaces de operar con corriente continua, de baja potencia (hasta 0,5 A) Pero tienen tiempos de conmutación que rondan el milisegundo y una vida útil mucho mayor que la de los relés. En este tipo de salida el transistor es el encargado de conectar la carga externa cuando el programa lo indique.

Contacto de salidas de común positivo de transistor



Esquema de cableado



3. Salidas por triac:
Manejan corrientes alternas. Al igual que los transistores, por ser semiconductores tienen una vida útil mucho mayor que la del relé, que es un elemento electromecánico.

4. Salidas analógicas:
Los módulos de salida analógica permiten que el valor de una variable numérica interna del autómata se convierta en tensión o corriente.

Internamente en el PLC se realiza una conversión digital analógica (D/A), puesto que el autómata sólo trabaja con señales digitales. Esta conversión se realiza con una precisión o resolución determinada (número de bits) y en un intervalo determinado de tiempo (período muestreo)
Esta tensión o intensidad puede servir de referencia de mando para actuadores que admitan mando analógico, como pueden ser las válvulas proporcionales, los variadores de velocidad, las etapas de los tiristores de los hornos, los reguladores de temperatura, etc. Permitiendo al autómata realizar funciones de regulación y control de procesos continuos.

2.1.4 Alimentación

La fuente de alimentación proporciona las tensiones necesarias para el funcionamiento de los distintos circuitos del sistema.
La alimentación a la CPU frecuentemente es de 24 Vcc, o de 110/220 Vca. En cualquier caso es la propia CPU la que alimenta las interfaces conectadas a través del bus interno.
La alimentación a los circuitos E/S puede realizarse, en alterna a 48/110/220 Vca o en continua a 12/24/48 Vcc.

2.1.5 Equipos o Unidades de programación

El autómata debe disponer de alguna forma de programación, la cual se suele realizar empleando algunos de los siguientes elementos:

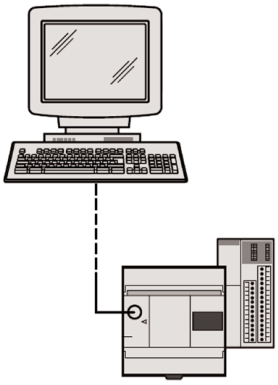
• Unidad de programación
Suele ser en forma de calculadora. Es la forma básica de programar el autómata, y se suele reservar para pequeñas modificaciones del programa o la lectura de datos en el lugar de colocación del autómata.

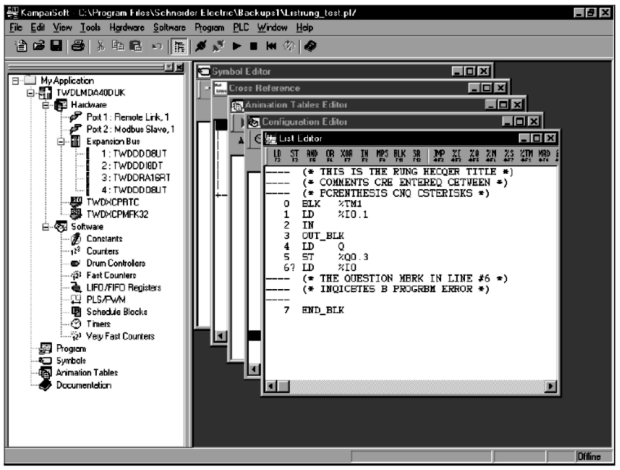
• Consola de programación

Es un terminal a modo de ordenador que proporciona una forma más favorable de realizar el programa de usuario y observar parámetros internos del autómata. Obsoleto actualmente.

• PC

Es la forma más cómoda empleada en la actualidad. Permite programar desde un ordenador personal estándar, con todo lo que ello supone: herramientas más potentes, posibilidad de almacenamiento en soporte magnético, impresión, transferencia de datos, monitorización mediante software SCADA, entre otros.





Para cada caso el fabricante proporciona lo necesario, el equipo o el software y/o los cables adecuados. Cada equipo, dependiendo del modelo y del fabricante, puede poseer una conexión a uno o varios de los elementos anteriores.

3 Clasificación

Introducción

El parámetro indicador que habitualmente define un PLC es la clasificación por cantidad de entradas y salidas (E/S), a pesar de su arbitrariedad.
Los fabricantes ofrecen características tales como: la capacidad de memoria, operaciones aritméticas, en directa relación a la cantidad de entradas y salidas que el controlador puede manejar.
Así, por ejemplo, suele haber una directa relación entre la clasificación de PLC como integrales, y los clasificados como micro PLC por la cantidad de E/S.

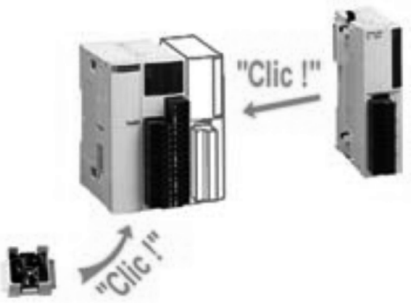
3.1 Cantidad de Entradas y Salidas

Una de las clasificaciones más comunes de los PLC hace referencia en forma directa a la cantidad de entradas y salidas (E/S o I/O) de un PLC y nos dice que un PLC es considerado micro PLC cuando tienen menos de 64 E/S, pequeños cuando tienen menos de 256 E/S, medianos cuando tienen menos de 1024 E/S y grandes cuando tienen más de 1024 E/S.

3.2 Estructura

Otras de las clasificaciones que se suelen hacer con respecto a los PLC son por su construcción. Estos pueden ser compactos o modulares.
Un PLC es compacto cuando todas sus partes se encuentran en la misma caja, compartimiento o chasis.

Los PLC compactos suelen ser los más baratos y pequeños, pero tienen la desventaja de sólo poder ampliarse con muy pocos módulos.
Un PLC es modular cuando se pude componer o armar en un bastidor o base de montaje, sobre el cual se instalan la CPU, los módulos de entradas/salidas y los módulos de comunicaciones si fueran necesarios, entre otros.



La principal ventaja de un PLC modular es que el usuario puede componer su equipo como sea necesario, y luego puede ampliarlo si su aplicación lo requiere. También suelen poseer instrucciones más complejas, un lenguaje de programación más potente y posibilidades de comunicaciones.
La desventaja es que suele ser un poco más caro y voluminoso que el integral.
Algunos módulos de E/S tienen forma de tarjetas con una bornera en el frente y un conector macho en su parte posterior. A estos módulos muchas veces se los denomina tarjetas de entradas y/o salidas. Estos módulos o tarjetas existen con distintos números de entradas y/o salidas. Podemos encontrar entre 4, 8, o 16, puntos de entradas y/o salidas en la misma tarjeta. Algunas empresas tienen módulos de alta densidad con 32 o más puntos de E/S.
Algunos PLC modulares tienen en sus tarjetas o módulos las borneras desmontables. Esto es particularmente útil en caso de tener que reemplazar algunos de los módulos. Pues no será necesario recablear las entradas o salidas.

4 Funcionamiento de un PLC

Introducción

En la mayoría de los PLC (Autómata Programable o Controladores Lógicos Programables) el funcionamiento es de tipo cíclico y secuencial, es decir, que las operaciones tienen lugar una tras otra, y se van repitiendo continuamente mientras el autómata está bajo tensión.

4 . 1 Tiempo de Barrido o “Scan Time”

Tiempo de Barrido o Scan Time: es el tiempo que demanda al PLC completar un ciclo. A cada ciclo de tareas se lo denomina Barrido o Scan.
Una típica secuencia se detalla a continuación:

- Autodiagnóstico: el autodiagnóstico se realiza cuando el PLC es conectado a tensión y es una verificación de todos sus circuitos. Si existiera algún problema el PLC emitiría alguna señal luminosa indicando el tipo de error que ha detectado.

- Lectura del registro de entradas y creación de una imagen de las entradas en la memoria: el PLC revisa cada entrada para determinar si está encendida o apagada (entrada binaria o de dos estados) Revisa las entradas desde la primera a la última, graba estos estados en la memoria creando la imagen de las entradas para ser utilizada en el paso siguiente.

- Lectura y ejecución del programa: acudiendo a la imagen de las entradas y salidas en memoria, la CPU ejecuta el programa realizado por el usuario. La ejecución del programa se realiza instrucción por instrucción y en el orden en que se determinó. Como ya se ha revisado el estado de las entradas, el programa puede tomar decisiones basado en los valores que fueron guardados. Las decisiones que toma el programa, en última instancia, corresponden a los valores que van a tomar cada una de las salidas, estos valores son almacenados en registros para ser utilizados en la etapa final.

- Atención de las comunicaciones.

- Actualización del registro de salidas: renovación de todas las salidas, en forma simultánea, en función de la imagen de las mismas, obtenidas al final de la ejecución del programa.

Los fabricantes en general dan el tiempo de barrido para ejecutar 1K (1024) de instrucciones de lógica booleana. Sin embargo, al no estar normalizados el tipo de instrucciones a utilizar en el ensayo, el dato no alcanza para comparar distintos PLC. Puede darse el caso de que un PLC ejecute un cierto tipo instrucciones más rápido que otro o viceversa. Para determinar en forma certera el tiempo de barrido se requiere la determinación del tiempo que le insume al procesador le ejecución de cada una de las instrucciones utilizadas, así como el tiempo consumido por las demás funciones que ejecuta la CPU.

4.1.1 Ciclo de funcionamiento

Existen dos posibilidades en cuanto al ciclo de ejecución, que el autómata esté en RUN o en STOP.
En cada uno de estos casos el autómata se comporta de la siguiente manera:

- **Autómata en RUN**: el procesador ejecuta el tratamiento interno, la confirmación de entradas, el tratamiento del programa y la actualización de las salidas.

- **Autómata en STOP**: en este caso no se ejecuta el tratamiento del programa.

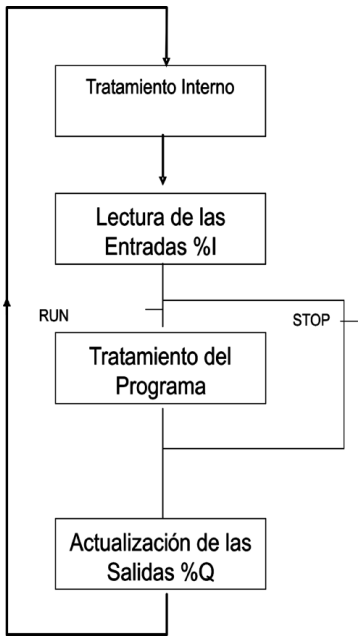
En la mayoría de los PLC existe un indicador luminoso en la parte frontal con la leyenda de RUN, que nos muestra cuando el microprocesador está ejecutando el programa. Cuando este indicador se encuentra en apagado el controlador no está ejecutando el programa o bien se encuentra en modo Stop.
Otro indicador luminoso, con la leyenda de ERROR, nos muestra cuando se ha encontrado una falla en la etapa de autodiagnóstico. En la mayoría de los casos cuando se detecta un error se detiene automáticamente la ejecución del programa.

4.2 Modos de funcionamiento del controlador Twido

El ciclo de ejecución o tareas del autómata se puede realizar de dos maneras:

- Ejecución Normal (exploración cíclica), configurada por defecto.
- Ejecución Periódica.

Ejecución Normal (cíclica)
Por defecto, el ciclo de tareas del autómata se ejecuta en forma cíclica de la siguiente manera:



Terminado el ciclo de ejecución actual, el autómata comienza inmediatamente con uno nuevo.

Desbordamiento del tiempo de ejecución

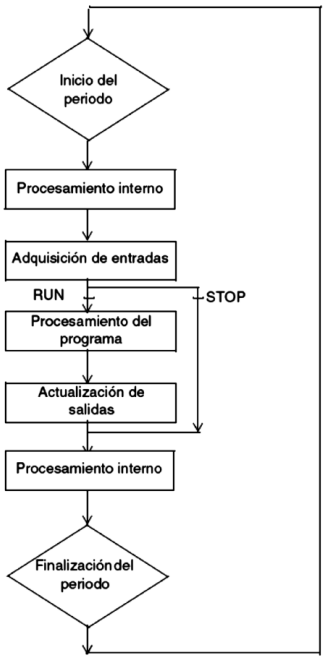
El temporizador watchdog del controlador supervisa el tiempo de ciclo del programa del usuario. Éste no debe exceder los 150 ms, ya que de lo contrario se producirá un fallo que provoque la detención inmediata del controlador en modo de parada o stop. Las salidas en este modo se fuerzan a su estado de retorno predeterminado.

Casos posibles de funcionamiento:

- Tiempo de ciclo < watch dog: el funcionamiento es normal, una vez finalizado el ciclo, se inicia el siguiente.
- Tiempo de ciclo > watch dog: el autómata pasa a STOP, los indicadores RUN y ERR parpadean y el bit del sistema %S11 pasa a 1.

Ejecución periódica

En este caso, la lectura de las entradas, el tratamiento del programa y la actualización de las salidas se realizan de forma periódica, según un tiempo definido por el usuario durante la configuración (2 a 150ms), tal como se indica en la figura siguiente:



En el inicio del ciclo del autómata, un temporizador de programa se ajusta al valor definido en configuración. El ciclo del autómata debe finalizar antes de que expire este temporizador. Al final del ciclo del temporizador, se inicia el siguiente. Si el tiempo del ciclo supera al tiempo programado, el bit del sistema (%S19) pasará a 1. La comprobación y reinicio a 0 correrán a cargo del programa del usuario.

Desbordamiento del tiempo de ejecución

La duración del tiempo de ejecución del programa usuario es controlada por el autómata (watch dog) y no debe superar los 150ms. En caso contrario, aparecerá un fallo que provocará la parada inmediata del autómata (indicadores RUN y ERR intermitentes)

Casos posibles de funcionamiento:

- Tiempo de ciclo < período: funcionamiento normal, el ciclo siguiente se inicia una vez alcanzado el final del período programado.
- Período < tiempo de ciclo < watch dog: el sistema pone el bit de sistema %S19 en estado 1 y el ajuste al estado 0 depende del programa usuario. El autómata-permanece en RUN.
- Tiempo de ciclo > watch dog: el autómata pasa a STOP, los indicadores RUN y ERR parpadean y el bit de sistema %S11 pasa a 1.

4.3 Comprobación del tiempo de ciclo

El ciclo de tarea master se controla mediante un temporizador watchdog, llamado Tmax (duración máxima del ciclo de tarea master)
Permite mostrar errores de aplicación (bucles infinitos, etc.) y garantiza una duración máxima para actualizar las salidas.

WatchDog del software (operación periódica o cíclica)

En una operación periódica o cíclica, la activación del watchdog provoca un error del software. La aplicación pasa a estado de pausa y establece el bit %S11 a 1. La nueva ejecución de la tarea necesita una conexión a Twido Soft con el fin de analizar la causa del error, la modificación de la aplicación para corregir el error y la nueva ejecución de las solicitudes de inicio y ejecución.

Comprobación de la operación periódica

En una operación periódica, se utiliza una comprobación adicional para detectar el período que se está excediendo:

- **%S19** indica que se ha superado el período. Se establece a 1 por el sistema cuando el tiempo de ciclo es mayor que el período de la tarea o por el usuario.
- **%SW0** contiene el valor del periodo (0-150 ms), es decir, comienza a partir de un inicio en frío mediante el valor establecido en la configuración. El usuario puede modificarlo.

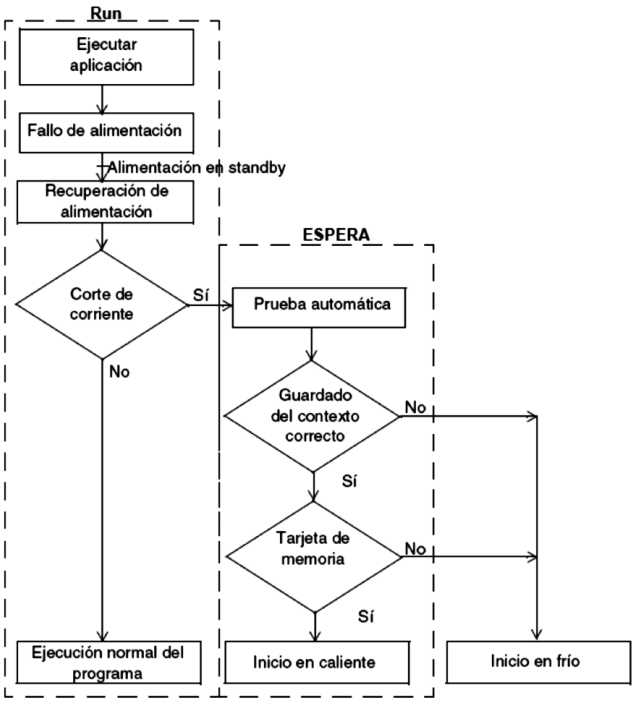
Uso del tiempo de ejecución de la tarea master

Las siguientes palabras del sistema se utilizan para ofrecer información sobre el tiempo de ciclo de exploración del controlador:

- **%SW11** se inicia con el tiempo de vigilancia máximo de watchdog (10 a 500 ms)
- **%SW30** contiene el tiempo de ejecución para el último ciclo de exploración del controlador.
- **%SW31** contiene el tiempo de ejecución para el ciclo de exploración del controlador más largo.
- **%SW32** contiene el tiempo de ejecución para el ciclo de exploración del controlador más corto.

RECUERDE que...
El estado de pausa se produce cuando la aplicación se detiene inmediatamente, debido a un error del software de la aplicación, como un desborde de ciclo.
Los datos conservan los valores actuales que permiten un análisis de la causa del error. Todas las tareas se detienen en la instrucción actual.
Está disponible la comunicación con el controlador.

La ilustración que aparece a continuación muestra los distintos tipos de reinicio de alimentación detectados por el sistema. Si la duración del corte de corriente es inferior al tiempo de filtrado de suministro de alimentación (unos 10 ms para el suministro de corriente alterna o 1 ms para el suministro de corriente continua), el programa no lo advierte y sigue funcionando con normalidad.



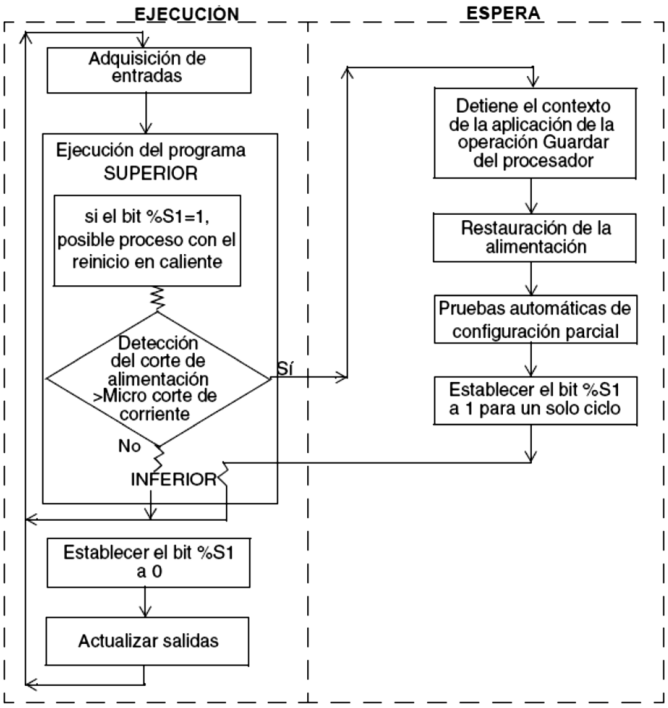
RECUERDE que...
El contexto se guarda en una memoria RAM mantenida con batería. Durante el arranque, el sistema comprueba el estado de la batería y el contexto guardado, y decide si puede producirse un arranque en caliente.

Uso de un reinicio en caliente. Causa de un reinicio en caliente

Un inicio en caliente puede producirse:

- Cuando se restaura la alimentación sin pérdida de contexto de las aplicaciones.
- Cuando el programa establece el bit **%S1** a estado 1.
- Desde la visualización del operador, cuando el controlador está en modo de detención.

El dibujo que aparece a continuación describe una operación de reinicio en caliente, en modo de ejecución.



Reinicio de la ejecución del programa

En la tabla siguiente se describen las fases de reinicio para ejecutar un programa después de un reinicio en caliente.

Fase	Descripción
1	La ejecución del programa se reanuda a partir del mismo elemento donde estaba antes el corte de alimentación, sin actualizar las salidas. **Nota**: sólo se reinicia el mismo elemento del código de usuario. El código del sistema (por ejemplo, la actualización de salidas) no se reinicia.
2	Al final del ciclo de reinicio, el sistema: • Elimina la reserva de la aplicación si se reservó (y hace que la aplicación se detenga en caso de depuración) • Reinicia los mensajes.
3	El sistema realiza un ciclo de reinicio en el que: • Ejecuta de nuevo la tarea con los bits **%S1** (indicador de reinicio en frío) y **%S13** (primer ciclo en ejecución) ajustados a 1. • Restablece los bits **%S1** y **%S13** a 0 al final de este primer ciclo de tarea.

Procesamiento de un inicio en caliente

En caso de un inicio en caliente, si es necesario un proceso de aplicación determinado, el bit **%S1** debe comprobarse al comienzo del ciclo de tarea y debe llamarse al programa correspondiente.

Salidas después de un fallo de alimentación

Tan pronto como se detecta un fallo de alimentación, las salidas se ponen a un estado de recaída (predeterminado) de 0. Cuando se recupera la alimentación, las salidas permanecen con el último estado hasta que la tarea las actualice de nuevo.

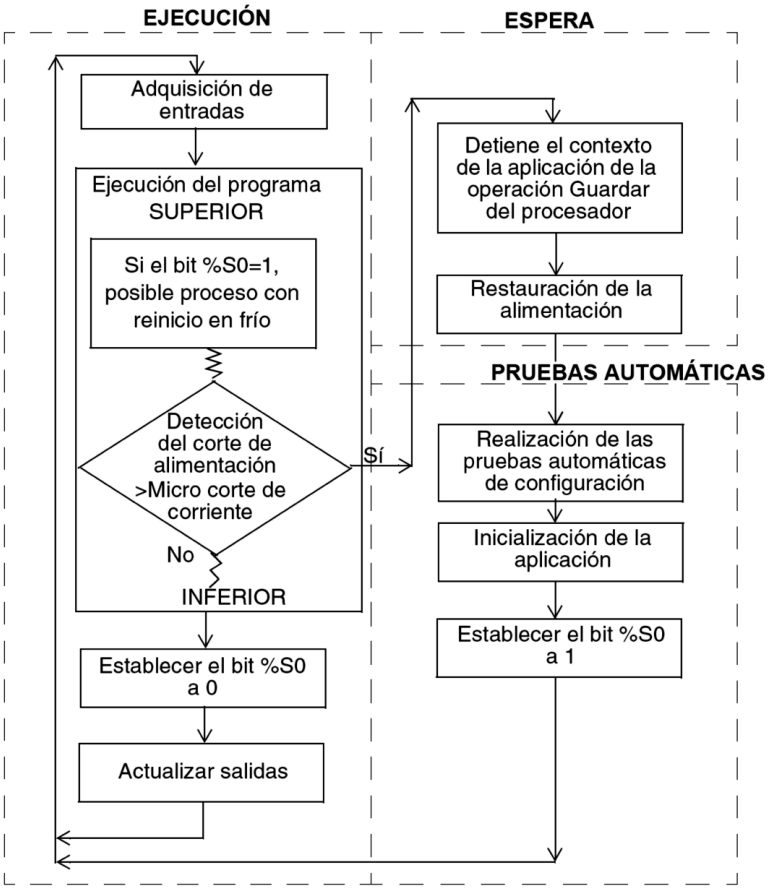
Comportamiento ante un inicio en frío. Causas de un inicio en frío

Un inicio en frío puede producirse:

- Al cargar una aplicación nueva en la RAM.
- Cuando se restaura la alimentación con pérdida de contexto de las aplicaciones.
- Cuando el programa ajusta el bit **%S0** a estado 1.
- Desde el monitor de operación, cuando el controlador está en modo de detención.

RECUERDE que...
Los controladores compactos siempre arrancan en frío. Los controladores modulares siempre se reinician en caliente.

El dibujo de abajo describe una operación de reinicio en frío en modo de ejecución.



Operación
En la tabla siguiente se describen las fases de reinicio para ejecutar un programa después de un reinicio en frío.

Fase	Descripción
1	Durante el arranque, el controlador está en modo de ejecución. Durante un reinicio en frío tras una detención debida a un ERROR, el sistema fuerza se reinicia en frío. La ejecución del programa se reinicia al comienzo del ciclo.
2	El sistema: • Restablece las palabras y los bits internos y las imágenes de E/S a 0. • Inicio de las palabras y los bits de sistema. • Inicio de los bloques de función de los datos de configuración.
3	Durante este primer ciclo de reinicio, el sistema: • Ejecuta de nuevo la tarea con los bits **%S0** (indicador de reinicio en frío) y **%S13** (primer ciclo en ejecución) ajustados a 1. • Restablece los bits **%S0** y **%S13** a 0 al final de este primer ciclo de tarea.

Procesamiento de un inicio en frío
En caso de inicio en frío, si se requiere un proceso de aplicación particular, se debe verificar el bit %S0 (que permanece a 1) durante el primer ciclo de la tarea.

Salidas después de un fallo de alimentación
Tan pronto como se detecta un fallo de alimentación, las salidas se ponen a un estado de recaída (predeterminado) de 0. Cuando se recupera la alimentación, las salidas permanecen a 0 hasta que la tarea las actualice de nuevo.

4.4 Iniciación del controlador

Los controladores se pueden iniciar mediante TwidoSoft ajustando los bits de sistema **%S0** (reinicio en frío) y **%S1** (reinicio en caliente)

- Comienzo de inicio en frío: para realizar un comienzo de inicio en frío, el bit de sistema **%S0** se debe ajustar a 1.
- Comienzo de inicio en caliente mediante **%S0** y **%S1**: para realizar un comienzo de inicio en caliente, los bits de sistema **%S1** y **%S0** se deben ajustar a 1.

El siguiente ejemplo explica cómo programar un reinicio en caliente mediante los bits de sistema:

LD %S1 Si %S1 = 1 (reinicio en caliente), ajuste %S0 a 1 para el inicio del controlador. ST %S0 el sistema restablece a 0 estos dos bits al final del ciclo siguiente.

- Inicio en caliente mediante el comando INIT

También se puede solicitar el inicio en caliente mediante un comando INIT. El comando INIT envía al controlador al estado IDLE, el reinicio de los datos de aplicación y el estado de la tarea al estado STOPPED.

RECUERDE que...
No debe ajustar %S0 a 1 durante más de un ciclo del controlador.

4.5 Evaluación

1. ¿En qué tipo de instalaciones se utiliza los PLC? ¿Por qué?
2. Comenta algunas ventajas e inconvenientes de los autómatas.
3. ¿Por qué diferentes unidades o partes está compuesto el autómata?
4. ¿Qué función desempeña la CPU dentro del autómata?
5. ¿Qué función desempeñan las fuentes de alimentación dentro del autómata programable?
6. ¿Qué tipos de interfaces existen?
7. ¿Qué tipos de entradas existen? Explicar.
8. ¿Qué es lo que almacena las memorias internas? ¿De que tipo es?
9. ¿Qué es lo que almacena la memoria de programa? ¿De que tipo es?
10. ¿De que procesos está compuesto el ciclo de funcionamiento?
11. ¿Que es el tiempo de ejecución? ¿De que depende dicho tiempo?

5 Hardware Twido

Introducción

El trabajo con los autómatas necesita un software para su funcionamiento. El software a utilizar se elegirá en función del hardware instalado y del nivel de las aplicaciones a realizar.

5.1 Presentación del producto

El autómata Twido surge del desarrollo conjunto entre Modicon y Telemecanique, marcas de Schneider Electric y especialistas en autómatas programables industriales (PLC). Dedicado a la automatización de instalaciones industriales simples y de máquinas pequeñas, Twido se encuentra disponible en dos versiones: compacto y modular. Comparten opcionales, extensiones de E/S y el software de programación, otorgándole máxima flexibilidad y simplicidad de uso. Tiene dimensiones reducidas y con una puesta en marcha muy sencilla, dispone de dos formas de programación:

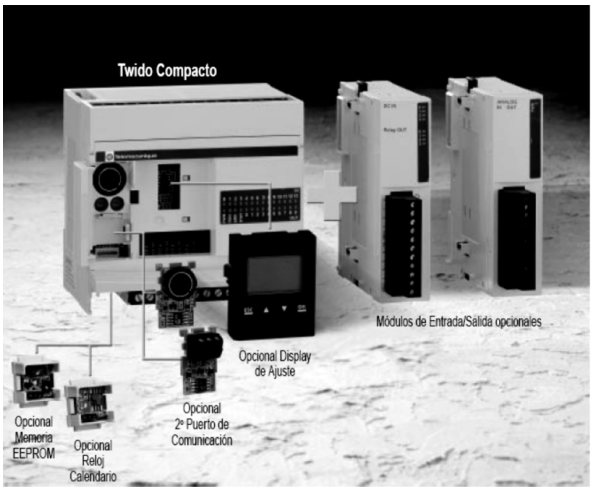
a. Lenguaje lista de instrucciones «list»

b. Lenguaje a contactos «ladder»

Twido permite, además, la creación de páginas GRAFCET, facilitando la programación de procesos secuenciales.
La programación se efectúa con la ayuda de una PC, con el software TwidoSoft.

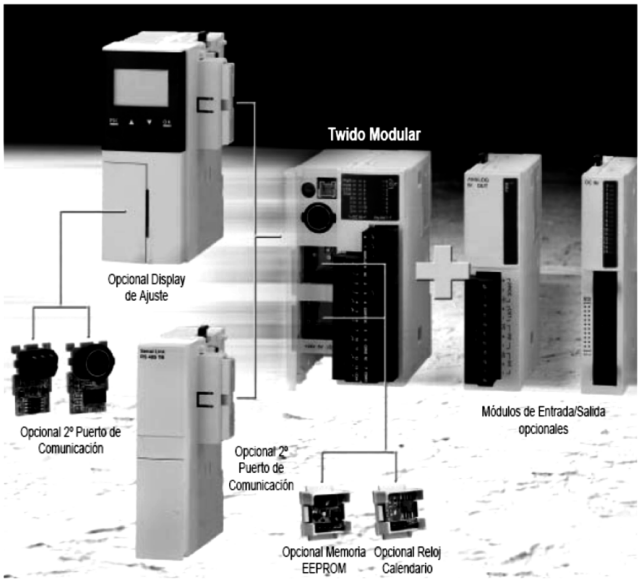
5.1.1 Twido Compacto

Para optimizar tiempos costos en la instalación, el Twido compacto está disponible en tres tallas: 10, 16 y 24 E/S, este último con la posibilidad de ser ampliado, incorporándole módulos de entradas o salidas digitales o analógicas. La alimentación del modelo compacto es en corriente alterna (100 – 240 Vca), posee entradas de 24 Vcc, y salidas a relé.



5.1.2 **Twido Modular**

Para soluciones hechas a medida, maximizando la eficiencia de las máquinas, el Twido modular está disponible en dos tallas: 20 y 40 E/S. La alimentación del modelo modular es en 24 Vcc, y posee entradas de 24 Vcc y salidas transistores, a relés o mixtas (transistores + relé) Además, cada Twido modular trae de base una entrada analógica de 0 a 10 Vcc.



5.2 **Principales características**

Mayor flexibilidad para componer un autómata programable acorde a su necesidad:

- Con sus 6 modelos de CPU compactas y modulares, Twido le ofrece múltiples posibilidades para resolver su automatismo.
- Gracias a una gran variedad de módulos, usted puede encontrar exactamente lo que necesita en aplicaciones estándar de 10 a 100 E/S.
- Ya sea si necesita un reloj calendario o un 2° puerto serie, etc. Twido le ofrece un amplio abanico de opciones. Evalúe su necesidad y utilice lo estrictamente necesario.

Mayor comunicación:

- Posibilidad de un 2° puerto serie opcional para los Twido compactos y modulares (en estos últimos a través de los módulos de comunicación)
- Cada CPU Twido compacto o modular puede extenderse con otras con:
  - E/S descentralizadas, en este caso en las bases no pueden adicionarse módulos de extensión de E/S.
  - Twidos conectados como CPU´s, en este caso en las bases pueden adicionarse módulos de extensión de E/S.
  - Cada Twido tiene su propio programa de aplicación y tiene reservadas cuatro palabras de entradas (%INW) y cuatro de salidas (%QNW) para intercambiar datos entre los Twidos.

- Hasta 7 Twidos pueden conectarse a un Twido compacto o modular. La distancia máxima del Bus RS485 es 200 m. Pueden utilizarse tanto los puertos integrados como los opcionales.
- Twido comunicado en Modbus. Puede integrarse fácilmente a los equipos existentes en campo como ser: otros autómatas programables, variadores de velocidad, monitores de circuito, arrancadores suaves, etc.

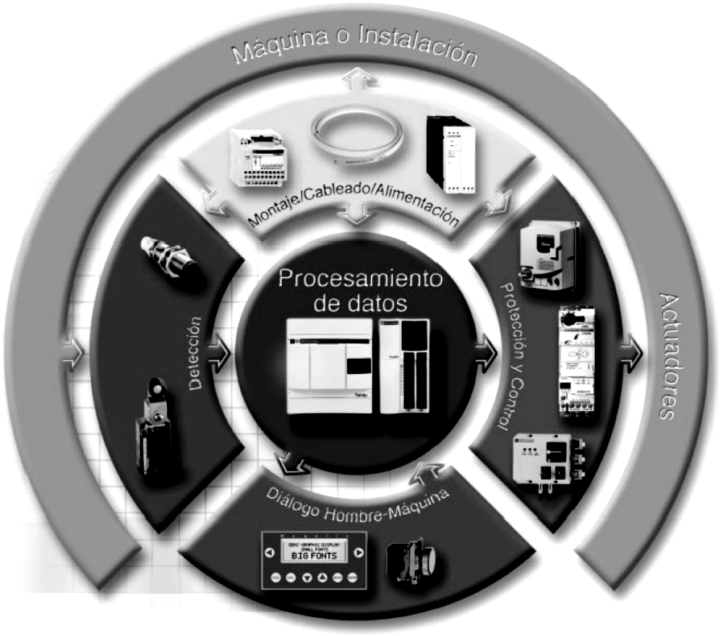
Más posibilidades de ajuste de parámetros:
- El visualizador de 4 botones puede ser utilizado para realizar los ajustes básicos directamente sobre el controlador.

Más simplicidad para ganar tiempo y disponibilidad:
- Fácil de cablear
  - Twido le propone una gran variedad de conexiones:
  - Soluciones con borneras a tornillo (extraíbles o fijas)
  - Soluciones pre-cableadas para una conexión rápida y confiable (conectores HE10, Twido Fast)
  - Soluciones de E/S remotas u otras CPU´s remotas (hasta 50 m)
  - Nuevas borneras a resorte, asociando un cableado rápido y una conexión segura.
- Fácil de ensamblar
  - Con un simple "click", podrá agregar las extensiones y/o los opcionales que necesite.
- Fácil de instalar
  - Su pequeño tamaño facilita la integración en los tableros.
- Fácil de aprender

Mayor capacidad:
- Con el opcional reloj calendario.
- Con memoria suplementaria de 32 y 64 k, permitiendo una rápida puesta en marcha a distancia de su aplicación.
- Con las siguientes funciones integradas:
  - Contadores rápidos (5 y 20 kHz)
  - Posicionamiento con funciones PLS (generador de pulsos) y PWM (modulación de ancho de pulso) en los Twido Modulares (2 salidas configurables)
  - 1E analógica integrada, en tensión (0-10VCC) en todas las CPU´s Twido Modular.
  - Potenciómetros analógicos.

Mayor compatibilidad para garantizar funcionamiento sin costos extra

Twido. Sinergia total con los productos Schneider Electric.

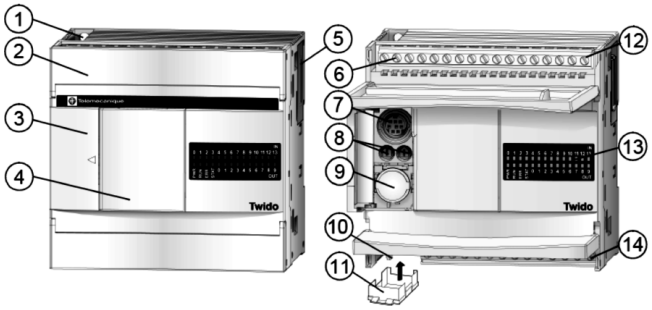


El más pequeño y poderoso entre sus pares

Imagine un autómata programable de 40 E/S y numerosas funciones integradas, todo en un tamaño no mayor a una tarjeta personal. Twido, supera todo lo imaginado.

5.3 Descripción

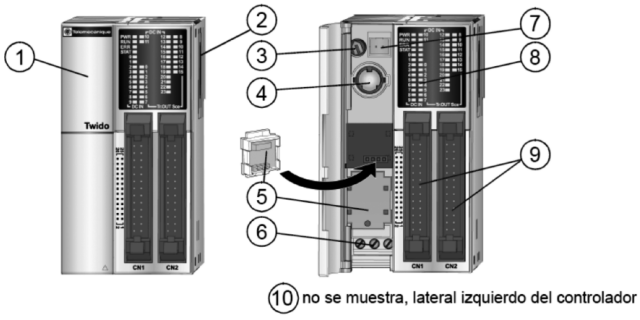
Twido compacto



Referencias

1. Orificio de montaje.
2. Cubierta de terminales.
3. Tapa con bisagra.
4. Cubierta extraíble del conector de visualización del operador.
5. Conector de ampliación - sólo en el controlador TWDLCAA24DRF.
6. Terminales de alimentación de sensores.
7. Puerto serie 1.
8. Potenciómetros analógicos - TWDLCAA10DRF y TWDLCAA16DRF tienen uno.
9. Conector de puerto serie 2 - TWDLCAA10DRF no tiene ninguno.
10. Terminales de fuentes de alimentación de 100 a 240 V CA.
11. Conector de cartuchos - ubicado en la parte inferior del controlador.
12. Terminales de entrada.
13. LED.
14. Terminales de salida.

Twido Modular



Referencias

1. Tapa con bisagra.
2. Conector de ampliación.
3. Potenciómetro analógico.
4. Puerto serie 1.
5. Cubiertas de los cartuchos.
6. Terminales de fuente de alimentación de 24 V CC.
7. Conector de entrada de tensión analógica.
8. LED.
9. Terminales de E/S.
10. Conector de comunicaciones.

5.4 Referencias de productos

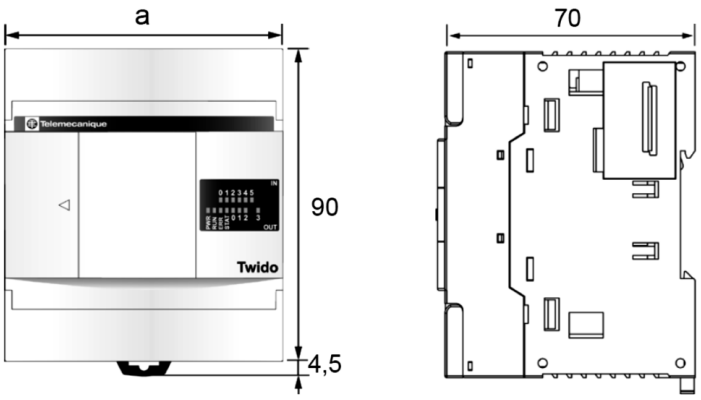
Descripción de las referencias y sus características a partir del código.

TWDL A
Tipo
CA: modelo compacto, alimentación en 100/240 Vca.
MD: modelo modular, alimentación en 24 Vcc.

Cantidad de Entradas / Salidas
10: 6 entradas + 4 salidas.
16: 9 entradas + 7 salidas.
20: 12 entradas + 8 salidas.
24: 14 entradas + 10 salidas.
40: 24 entradas + 16 salidas.

Características de Entradas / Salidas
Dxx: entradas 24 Vcc NPN/PNP
DFR: salidas a Relé.
DUK: salidas a transistor NPN
DTK: salidas a transistor PNP
DRT: salidas a relé + salidas a transistor PNP

5.5 Dimensiones

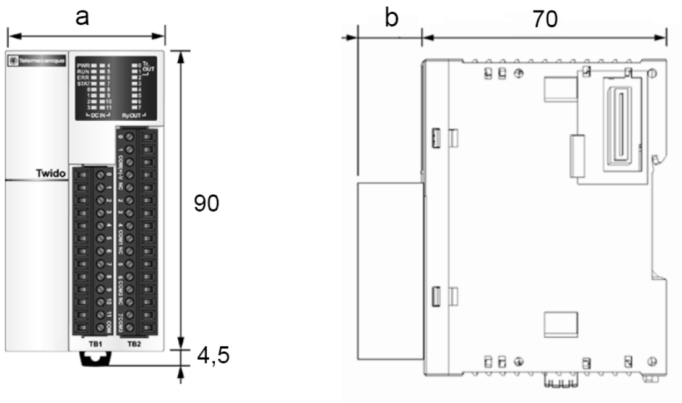


5.5.1 Modelos compactos

Referencias "a"

TWDLCAA 10DRF 80
TWDLCAA 16DRF 80
TWDLCAA 24DRF 95
Nota: dimensiones en milímetros

5.5.2 Modelos Modulares



Referencias "a – b"

TWDLMDA 20DTK/DUK 35,4 0 *
TWDLMDA 20DRT 47,5 14,6
TWDLMDA 40DTK/DUK 47,5 0 *
Nota: dimensiones en milímetros
* Sin el conector

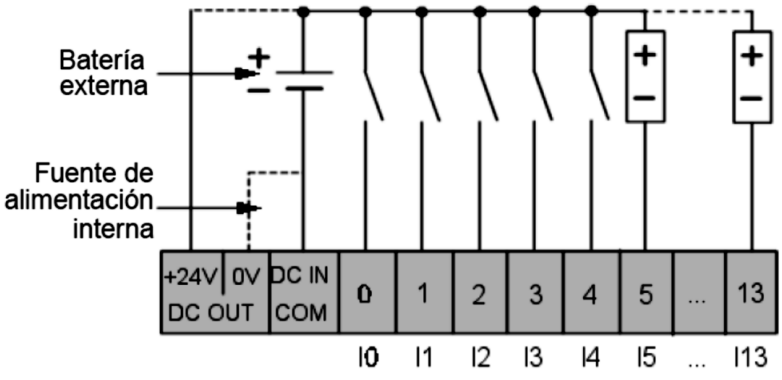
5.6 Conexionado

En esta sección se muestra un resumen del conexionado de las entradas y salidas digitales del autómata Twido, para mayor información sobre conexionado de los distintos módulos (E/S digitales o analógicas, módulos de comunicación, etc.) recurra a la guía de referencia de Hardware TWD USE 10AS.

5.6.1 Conexionado de las entradas digitales

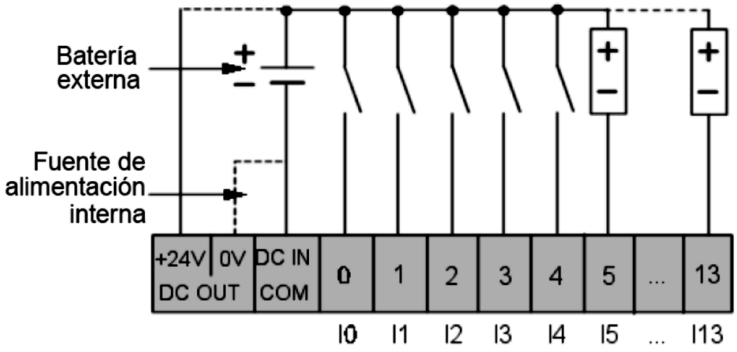
A continuación se describe la forma de conexión de las entradas del TWIDO.

5.6.1.1 Entradas con lógica positiva



*Conexión de detectores **PNP***

5.6.1.2 **Entradas con lógica negativa**

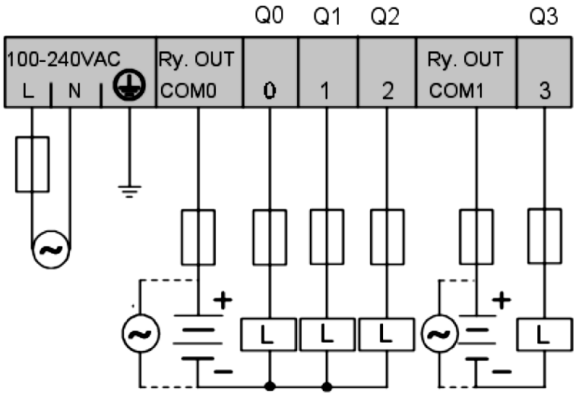


*Conexión de detectores **NPN***

5.6.2 **Conexionado de las salidas digitales**

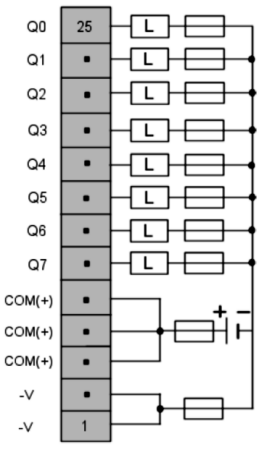
A continuación se describe la forma de conexión de las salidas del TWIDO.

5.6.2.1 **Salidas con relé**

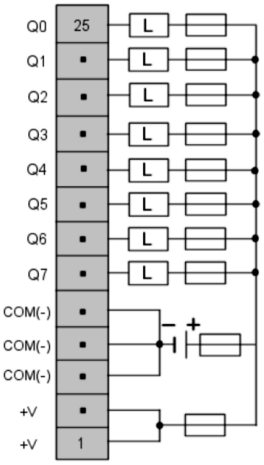


5.6.2.2 **Salidas a transistor**

Con lógica negativa



Con lógica Positiva



5.7 Estructura de la memoria de usuario de un PLC TWIDO

La memoria del controlador accesible a través de una aplicación de usuario está dividida en dos partes diferentes:

- **Memoria de bits**
  La memoria de bits se almacena en la memoria RAM interna que está integrada en el controlador. Contiene el mapa de 1280 objetos de bit.

- **Función de la memoria de palabras**
  La memoria de palabras (16 bits) admite:

DATOS	Datos de sistema y datos de aplicación dinámicos.
PROGRAMA	Descriptores y código ejecutable para tareas.
CONSTANTES	Palabras constantes, valores iniciales y configuración de entrada / salida.

5.7.1 Tipos de memoria

A continuación se señalan los diferentes tipos de memoria para los controladores Twido:

- RAM interna (integrada)
  Esta es la memoria RAM integrada del controlador. Los 10 primeros KB de la memoria RAM interna constituyen la RAM rápida. Los 32 KB siguientes constituyen la RAM estándar. La RAM interna contiene el programa, constantes y datos.

- EEPROM interna
  EEPROM integrada de 32 KB, proporciona una copia de seguridad interna en el controlador de una aplicación. Protege la aplicación contra los daños provocados por fallos de batería o cortes de corriente superiores a 30 días. Contiene el programa y constantes.

- Cartucho de copia de seguridad de memoria externa, cartucho de EEPROM externa opcional para realizar copias de seguridad de una aplicación o para dar cabida a una aplicación más grande. Se puede utilizar para actualizar la aplicación en la RAM del controlador. Contiene el programa y constantes, pero ningún dato.

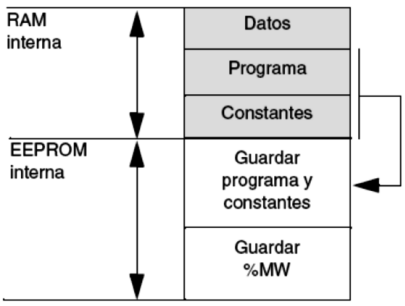
5.7.1.1 Estructura sin cartucho de memoria externa

En el diagrama que aparece a continuación se describe la estructura de memoria sin cartucho de memoria externa.

La EEPROM interna está integrada en el controlador y proporciona 32 KB de memoria para lo siguiente:

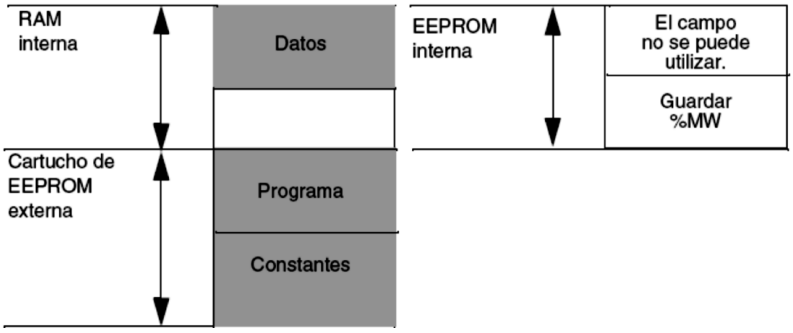
- El programa de aplicación (32 KB)

∑ 512 palabras internas (%MWi)



5.7.1.2 **Estructura con cartucho de memoria externa**

El cartucho de memoria externa opcional proporciona una copia de seguridad de los programas y constantes, al mismo tiempo que ofrece memoria ampliada para aplicaciones de mayor tamaño.
En el siguiente diagrama se describe la estructura de memoria con cartucho de memoria externa:



La EEPROM interna de 32 KB puede almacenar 512 palabras internas (%MWi)

5.7.2 **Almacenamiento de la memoria**

La memoria RAM interna del controlador se puede almacenar mediante:

- Batería interna (hasta 30 días)
- EEPROM interna (32 KB como máximo)
- Cartucho de memoria externa opcional (64 KB como máximo)

La transferencia de la aplicación desde la memoria EEPROM interna hasta la memoria RAM se realiza automáticamente cuando la aplicación se pierde en la RAM (si no se ha guardado o si no hay batería)
También se puede realizar una transferencia manual mediante TwidoSoft.

6 Lenguajes de Programación

Introducción

Cuando se habla de los lenguajes de programación se hace referencia a diferentes formas de poder escribir el programa usuario.
Los softwares actuales permiten traducir el programa usuario de un lenguaje a otro, pudiendo así escribir el programa en el lenguaje que más convenga.
La creciente complejidad en la programación de los autómatas programables requiere más que nunca de la estandarización de la misma. Bajo la dirección del IEC el estándar IEC 1131-3 (IEC 65) para la programación de PLC ha sido definida. Alcanzó el estado de estándar internacional en agosto de 1992. Con la idea de hacer el modelo adecuado para un gran abanico de aplicaciones, cinco lenguajes han sido definidos en total:

- Gráfico secuencial de funciones (Grafcet)
- Lista de instrucciones.
- Texto estructurado.
- Diagrama de flujo.
- Diagrama de contactos o Lógica de Escalera o Ladder Logic.

No obstante, los lenguajes de programación más empleados en la actualidad son: el listado de instrucciones y el esquema de contactos o Ladder Logic.

6.1 Diagrama de Contactos o Lógica de Escalera

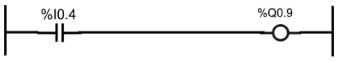
Tradicionalmente los diagramas de lógica de escalera están compuestos por dos líneas verticales que representan las líneas de alimentación, mientras que los renglones contienen los cableados, los arreglos de contactos y las bobinas de relés.
En los PLC, los diagramas de Lógica de Escalera o Ladder Logic son una manera fácil de dibujar los programas.
Una ventaja importante es que los símbolos básicos están normalizados según NEMA y son empleados por todos los fabricantes.
En la tabla que sigue se puede ver una comparación entre lo que significa los dibujos para la antigua lógica de escalera y para la moderna programación de un PLC.

Comparación de los diagramas Ladder

Simbología	Conexión Física de Relé	Programación por PC
Líneas Verticales	Bus Principal	Comienzo y Fin del renglón
Renglones o Peldaños	Ramas del Circuito	Conjunto de Instrucciones
	Contactos	Dirección de dispositivos de entradas y salida
	Bobinas de Relés	Dirección en registros de salida
Implementación	Conexión de cables siguiendo el esquema.	Entrada de símbolos con el dispositivo de programación

Cada contacto y cada bobina de relé representan una localización en el registro de entradas o salidas. Debe quedar claro que los dibujos sólo “representan” relés que no existen físicamente. El símbolo de una bobina de relé representa un bit del registro de las salidas, que podrá estar encendido (puesto en “1”) o apagado (puesto en “0”) durante la ejecución del programa.
Cada renglón o peldaño del diagrama de lógica de escalera del PLC corresponde a un conjunto de instrucciones para el PLC, ese conjunto de instrucciones le dirá al PLC que hacer en respuesta al estado de las entradas (contactos)

La figura que se muestra a continuación grafica esta situación.



Algunos símbolos usados

Como se sabe, existen dos símbolos para la programación de PLC: uno para representar contactos normalmente abiertos y otro para representar contactos normalmente cerrados.
Estos contactos pueden representar entradas, salidas o variables internas, es decir, un bit del registro de entradas, o bits del registro de salida, o de los bits internos o auxiliares, también llamados relés internos o auxiliares.
Una forma conveniente de ver estos contactos en un programa es pensarlos como una instrucción que examina si esa entrada está encendida o si está apagada, o dicho de otra forma examina si el bit que representa esa entrada está encendido o apagado.
Un contacto normalmente abierto representa una interrogación por si un bit está encendido y un contacto normalmente cerrado representa una interrogación por si un bit está apagado. Una condición de verdadero o falso es otorgada al contacto si el PLC encuentra la requerida condición de bit.
Si el PLC encuentra una condición de verdadero para todos los contactos del renglón, el bit de salida es encendido o apagado según lo indique el símbolo de esa salida.

En la tabla de abajo se muestra un resumen de lo dicho anteriormente.

Tipo de contacto	**Símbolo**	**Instrucción dada al PLC**	El PLC establece continuidad si el bit es buscado.
NO / NA		Examina si está encendido	1 (ON)
NO / NA		Examina si está apagado	0 (OFF)

Un contacto de entrada (salida, variable interna) NA (normalmente abierto) hace que el PLC revise en el registro de las entradas (salidas o variables internas) si esa entrada en particular se encuentra activa o encendida. Si el PLC encuentra la entrada activa permitirá la continuidad a través de ese contacto, en particular en el programa lógico realizado por el usuario.
De forma similar un contacto de entrada (salida o variable interna) NC hace que el PLC revise en el registro de las entradas (salidas o variables internas) si esa en particular se encuentra inactiva o desactivada. Esto es, que el PLC revisa a esa localización en el registro de las entradas para ver si está desactivada. De ser cierto, el PLC permite la continuidad a través de ese contacto en el programa del usuario.

Cuando el símbolo del contacto NA representa una localización en el registro de las salidas, nos provee un reporte del estado del dispositivo de salida. Un contacto de salida NA hace que el PLC revise esa dirección de salida en particular. El contacto de salida tendrá continuidad si la salida está encendida, pero mostrará discontinuidad si la salida está apagada.

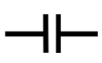
Un contacto de salida NC hace que el PLC revise esa dirección de salida en particular. El contacto de salida tendrá continuidad si la salida está apagada, pero mostrará discontinuidad si la salida está encendida.
El símbolo más usado para representar las salidas es el de la bobina de un relé. Estos símbolos no son bobinas reales, sino que son dibujos utilizados para graficar la localización de una salida en el registro de las salidas.
Para las líneas de funciones más complejas como temporizadores, registros de desplazamiento, etc., se emplea el formato de bloques. Éstos no están estandarizados, aunque guardan una gran similitud entre sí para distintos fabricantes y resultan mucho más expresivos que si se utiliza para el mismo fin el lenguaje en lista de instrucciones o mnemónico.

6.2 Listado de instrucciones (mnemónico)

Utiliza instrucciones derivadas de las operaciones del álgebra de Boole, combinadas con otras que permiten representar funciones como temporizadores, contadores, movimientos de datos en la memoria y cálculos (suma, resta, multiplicación, división, raíz cuadrada, cálculo de porcentaje, cambios en el sistema de numeración, etc.)
Cada instrucción está formada por un mnemónico o código, (abreviatura que representa una función), y uno o varios argumentos (variables que indican la dirección de memoria sobre la que se va a trabajar)
Como puede imaginarse existe una equivalencia o correspondencia entre la lógica de escalera y el listado de instrucciones. En muchos PLC esta equivalencia se puede ver en forma inmediata sólo con activar un icono de la pantalla de programación.

6.3 Diagramas de funciones

El diagrama de funciones (function block diagram o FBD) es un lenguaje gráfico que permite programar elementos que aparecen como bloques para ser cableados entre si de forma análoga al esquema de un circuito. El uso de FBD es adecuado para muchas aplicaciones que involucren el flujo de información o datos entre componentes de control.



6.4 Texto estructurado

El texto estructurado (structured text o ST) es un lenguaje de alto nivel estructurado por bloques que posee una sintaxis parecida al PASCAL. El ST puede ser empleado para realizar rápidamente sentencias complejas que manejen variables con un amplio rango de diferentes tipos de datos, incluyendo valores analógicos y digitales. También se especifica tipos de datos para el manejo de horas, fechas y temporizaciones, algo importante en procesos industriales. El lenguaje posee soporte para bucles como REPEAR UNTIL, ejecuciones condicionales empleando sentencias IF-THEN-ELSE y funciones como SQRT() y SIN()

6.5 Grafcet

El gráfico secuencial de funciones (SFC o Grafcet) es un lenguaje gráfico que proporciona una representación en forma de diagrama de las secuencias del programa. Soporta selecciones alternativas de secuencia y secuencias paralelas. Los elementos básicos son pasos y transiciones. Los pasos consisten en piezas de programas que son inhibidas hasta que una condición especificada por las transiciones es conocida. Como consecuencia de que las aplicaciones industriales funcionan en forma de pasos, el SFC es la forma lógica de especificar y programar el más alto nivel de un programa para PLC.

7 Instrucciones Tipo

Introducción

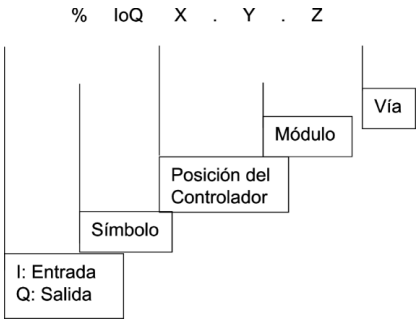
George Boole (1815- 1864) nació el 2 de Noviembre de 1815 en Lincoln, Lincolnshire (Inglaterra) En el 1854 publicó Las leyes del pensamiento sobre las cuales son basadas las teorías matemáticas de Lógica y Probabilidad.
Boole aproximó la lógica en una nueva dirección reduciéndola a un álgebra simple, incorporando lógica en las matemáticas. Agudizó la analogía entre los símbolos algebraicos y aquellos que representan formas lógicas. Su álgebra consiste en un método para resolver problemas de lógica que recurre solamente a los valores binarios 1 y 0 y a tres operadores: AND (y), OR (o) y NOT (no). Comenzaba el álgebra de la lógica llamada Álgebra Booleana, la cual ahora encuentra aplicación en la construcción de computadores, circuitos eléctricos, etc.

7.1 Tratamiento Booleano

Definición de los principales objetos de bits

Bits de entradas/salidas: estos bits son las "imágenes lógicas" de los estados eléctricos de las entradas/salidas. Están almacenados en la memoria de datos y se actualizan en cada explotación del programa.

El direccionamiento de estos bits es el siguiente:



Referencias:
Símbolo: IEC61131
Tipo de objeto: %I: para las Entradas y %Q: para las Salidas.

X. Posición del controlador: 0 Controlador master, 1 a 7 controlador remoto.
Y. Módulo: 0 unidad de E/S local, 1 a 7 módulos de ampliación.
Z. Vía, número de la entrada o salida.

Bits internos: los bits internos (%Mi) memorizan los estados intermedios durante la ejecución del programa.

Bits de sistema: los bits de sistema (%Si) controlan el buen funcionamiento del autómata, así como el desarrollo del programa de aplicación.

Existen otros bits que pueden usarse en el tratamiento booleano, como son los bits de los bloques de función y los bits extraídos de palabras.

Síntesis de los bits utilizados por el Twido

Tipo	Dirección	Cantidad máxima	Escritura
Bits entrada	%IX.Y.Z	Depende Twido	No
Salida	%QX.Y.Z	Depende Twido	Sí
Bits internos	%Mi	128 ó 256 según modelo	Sí
Bits Sistema	%Si	128	Según i

7.2 Introducción a los diagramas Ladder Logic

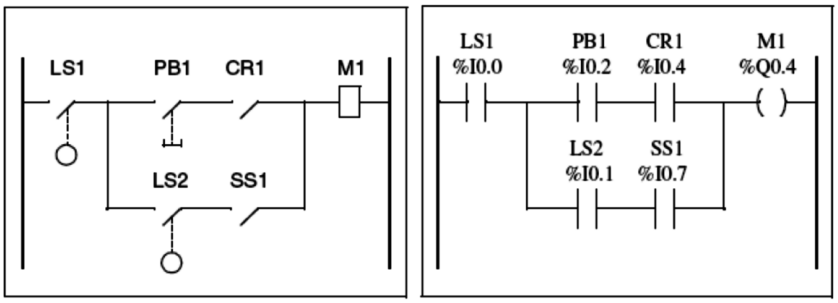
Los diagramas Ladder Logic son similares a los diagramas de lógica de relé. Las principales diferencias entre los dos son las Funciones de la Programación de Ladder Logic que no aparecen en los diagramas de lógica de relé.

Características:

- Todas las entradas están representadas por símbolos de contactos
- Todas las salidas están representadas por símbolos de bobinas
- Las operaciones numéricas están incluidas en el conjunto de instrucciones de Ladder Logic gráfico

7.2.1 Equivalentes Ladder Logic a los circuitos de relé

La siguiente ilustración muestra un diagrama simplificado del cableado de un circuito de lógica de relé y el diagrama Ladder Logic equivalente.



Circuito de lógica de relé — Diagrama Ladder Logic

Si observamos la ilustración anterior, podemos percatarnos de que todas las entradas asociadas al dispositivo de conmutación, en el diagrama de lógica de relé, aparecen como contactos en el diagrama Ladder Logic.
La bobina de salida M1 del diagrama de lógica de relé se representa con un símbolo de bobina de salida en el diagrama Ladder Logic.
Los números de dirección que aparecen sobre cada uno de los símbolos de contactos o bobinas en el diagrama Ladder Logic hacen referencia a la posición que ocupan las conexiones de entrada/salida con el controlador.

7.2.2 Escalones Ladder Logic

Un programa escrito en lenguaje Ladder Logic está compuesto por escalones, que son conjuntos de instrucciones gráficas dibujadas entre dos barras verticales de potencia. El controlador ejecuta los escalones secuencialmente.
El conjunto de instrucciones gráficas representa las siguientes funciones:

- Entradas/salidas del controlador (sensores, relés, luces de pilotos, etc.)
- Funciones del controlador (temporizadores, contadores etc.)
- Operaciones lógicas y matemáticas (adición, división, AND, y XOR entre otras)
- Operadores de comparación y otras operaciones numéricas (A<B, A=B, desplazamiento, rotación, etc.)
- Variables internas del controlador (bits, palabras, etc.)

Estas instrucciones gráficas se organizan con conexiones horizontales y verticales que eventualmente llevan a una o varias salidas o acciones. Una red no puede admitir más de un grupo de instrucciones vinculadas.

7.2.3 Bloque de diagramas Ladder Logic

Los diagramas Ladder Logic están compuestos por bloques que representan el flujo de programas y las funciones, por ejemplo:

- Contactos
- Bobinas
- Instrucciones de flujo de programas
- Bloques de función
- Bloques de comparación
- Bloques de operación

Las instrucciones de los diagramas Ladder Logic se componen de elementos gráficos. Esta sección enumera y describe los elementos gráficos utilizados en las instrucciones Ladder de Twido.

Contactos

Los elementos gráficos de los contactos se programan en el área de prueba y ocupan una celda (el alto de una fila por el ancho de una columna)

Nombre	Elemento gráfico	Instrucción	Función
Contacto normal abierto		LD	Establece contacto cuando el objeto de bit de control está en estado 1.
Contacto normal cerrado		LDN	Establece contacto cuando el objeto de bit de control está en estado 0.
Contacto para detectar un flanco ascendente		LDR	Flanco ascendente: detecta el cambio de 0 a 1 del objeto de bit de control.
Contacto para detectar un flanco descendente		LDF	Flanco descendente: detecta el cambio de 1 a 0 del objeto de bit de control.

Elementos de conexión

Los elementos gráficos de conexión se utilizan para conectar los elementos gráficos de acción y de prueba.

Nombre	Elemento gráfico	Funciones
Conector horizontal		Conecta en serie los elementos gráficos de prueba y acción entre dos barras potenciales.
Conector inferior		Conecta los elementos de prueba y acción de forma paralela (conexión vertical).

Bobinas

Los elementos gráficos de bobina se programan en el área de acción y ocupan una celda (el alto de una fila por el ancho de una columna)

Nombre	Elemento gráfico	Instrucción	Funciones
Bobina directa		ST	El objeto de bit asociado toma el valor del resultado del área de prueba.
Bobina negada		STN	El objeto de bit asociado toma el valor negado del resultado del área de prueba.
Establecer bobina		S	El objeto de bit asociado se establece en 1 cuando el resultado del área de prueba es 1.
Restablecer bobina		R	El objeto de bit asociado se establece en 0 cuando el resultado del área de prueba es 1.
Llamada de salto o subrutina	->>%Li ->>%SRi	JMP SR	Se conecta a una instrucción etiquetada ubicada delante o detrás.
Bobina de condición de transición			Proporcionado en lenguaje Grafcet, se utiliza cuando la programación de las condiciones de transición asociadas a las transiciones provoca una inversión de corriente en el siguiente paso.
Retorno desde una subrutina	<RET>	RET	Situado al final de las subrutinas para regresar al programa principal.
Detener programa	<END>	END	Define el final del programa.

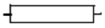
Bloques de función

Los elementos gráficos de los bloques de función se programan en la misma área de prueba y requieren cuatro filas y dos columnas de celdas (excepto para contadores muy rápidos que requieren cinco filas y dos columnas)

Nombre	Elemento gráfico	Funciones
Temporizadores, contadores, registros, etc.		Cada bloque de función utiliza entradas y salidas que permiten conexiones a otros elementos gráficos.. Nota: Las salidas de los bloques de función no pueden conectarse entre sí (conexiones verticales).

Bloques de operación y comparación

Los bloques de comparación se programan en el área de prueba, mientras que los de operación lo hacen en el área de acción.

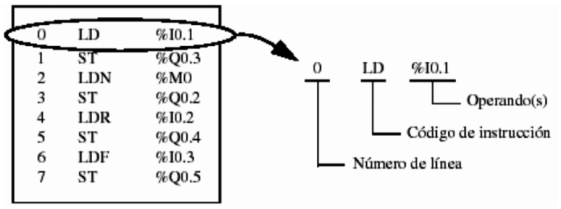
Nombre	Elemento gráfico	Funciones
Bloque de comparación		Compara dos operandos y la salida cambia a 1 cuando se comprueba el resultado. Tamaño: una fila por dos columnas
Bloque de operación		Realiza operaciones aritméticas y lógicas. Tamaño: una fila por cuatro columnas

7.3 Programas de Listado de Instrucciones

Un programa escrito en lenguaje de lista está formado por una serie de instrucciones que el controlador ejecuta de forma secuencial. Cada instrucción de lista está representada por una línea de programa y tiene tres componentes:

- Número de línea
- Código de instrucción
- Operando(s)

A continuación se muestra un ejemplo de un programa de lista.



Número de línea: los números de líneas se generan automáticamente al introducir una instrucción. Las líneas vacías y las líneas de comentario no tienen números de línea.

Código de instrucción: el código de instrucción es un símbolo para un operador, que identifica la operación que se va a realizar utilizando los operantes. Los operadores típicos especifican operaciones numéricas y boolearias.

Ejemplo

En el programa de ejemplo anterior, LD es la abreviatura del código de instrucción para una instrucción LOAD. La instrucción LOAD coloca (carga) el valor del operando %I0.1 en un registro interno llamado el acumulador.
Hay dos tipos de instrucciones básicas:

- *Instrucciones de prueba. Estas instrucciones configuran o comprueban las condiciones necesarias para realizar una acción. Por ejemplo, LOAD (LD) y AND.*
- *Instrucciones de acción. Estas instrucciones realizan acciones como resultado de las condiciones configuradas. Por ejemplo, instrucciones de asignación como STORE (ST) y RESET (R)*

Operando: un operando es un número, dirección o símbolo que representa un valor que puede manipular un programa en una instrucción.

Ejemplo
En el programa del ejemplo expuesto anteriormente, el operando %I0.1 es una dirección que tiene asignado el valor de una entrada del controlador. Una instrucción puede tener de cero a tres operandos dependiendo del tipo de código de instrucción. Los operandos pueden representar los siguientes elementos:

- *Entradas y salidas del controlador, como sensores, botones y relés.*
- *Funciones de sistema predefinidas, como temporizadores y contadores.*
- *Operaciones aritméticas, numéricas y de comparación.*
- *Variables internas del controlador, como bits y palabras.*

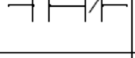
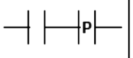
7.3.1 **Instrucciones del lenguaje de lista**

Un lenguaje de lista se compone de los siguientes tipos de instrucciones:

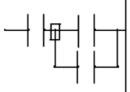
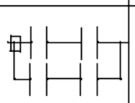
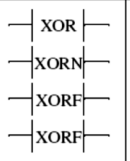
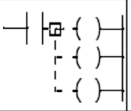
1. Instrucciones de prueba
2. Instrucciones de acción
3. Instrucciones sobre bloques de función

Esta sección identifica y describe las instrucciones Twido para la programación de listas.

1. Instrucciones de prueba

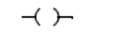
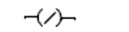
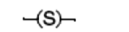
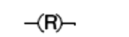
Nombre	Elemento gráfico equivalente	Función
LD		El resultado booleario es el mismo que el estado del operando.
LDN		El resultado booleario es el mismo que el estado inverso del operando.
LDR		El resultado booleario cambia a 1 durante la detección del operando (flanco ascendente) que cambia de 0 a 1.
LDF		El resultado booleario cambia a 1 durante la detección del operando (flanco descendente) que cambia de 1 a 0.
AND		El resultado booleario es igual a la instrucción lógica AND entre el resultado booleario de la instrucción anterior y el estado del operando.
ANDN		El resultado booleario es igual a la instrucción lógica AND entre el resultado booleario de la instrucción anterior y el estado inverso del operando.
ANDR		El resultado booleario es igual a la instrucción lógica AND entre el resultado booleario de la instrucción anterior y el flanco ascendente del operando (1 = flanco ascendente).
ANDF		El resultado booleario es igual a la instrucción lógica AND entre el resultado booleario de la instrucción anterior y la detección del flanco descendente del operando (1 = flanco descendente).
OR		El resultado booleario es igual a la instrucción lógica OR entre el resultado booleario de la instrucción anterior y el estado del operando.

La siguiente tabla describe las instrucciones de prueba en lenguaje de listas.

Nombre	Elemento gráfico equivalente	Función
AND(		Instrucción lógica AND (8 niveles de paréntesis)
OR(		Instrucción lógica OR (8 niveles de paréntesis)
XOR, XORN, XORR, XORF		OR exclusivo
MPS MRD MPP		Conmutación a las bobinas.
N	-	Negación (NOT)

2. Instrucciones de acción

La siguiente tabla describe las instrucciones de acción en lenguaje de listas.

Nombre	Elemento gráfico equivalente	Funciones
ST		El operando asociado toma el valor del resultado del área de prueba.
STN		El operando asociado toma el valor inverso del resultado del área de prueba.
S		El operando asociado se establece en 1 cuando el resultado del área de prueba es 1.
R		El operando asociado se establece en 0 cuando el resultado del área de prueba es 1.
JMP		Se conecta de forma incondicional a una secuencia etiquetada ubicada delante o detrás.
SRn		Conexión al comienzo de una subrutina.
RET		Retorno desde una subrutina.
END		Fin del programa.
ENDC		Fin del programa condicionado en un resultado booleario de 1.
ENDCN		Fin del programa condicionado en un resultado booleario de 0.

3. Instrucciones sobre bloques de función

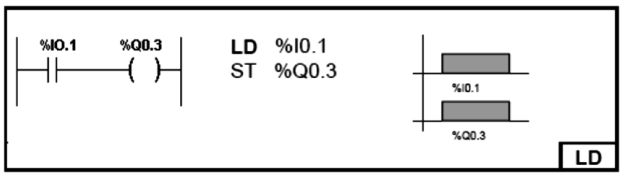
La siguiente tabla describe los bloques de función en lenguaje de listas.

Nombre	Elemento gráfico equivalente	Funciones
Temporizadores, contadores, registros, etc.		Para cada bloque de función existen instrucciones para controlar el bloque. Para cablear las entradas y salidas de bloques directamente se utiliza una forma estructurada. **Nota:** Las salidas de los bloques de función no pueden conectarse entre sí (conexiones verticales).

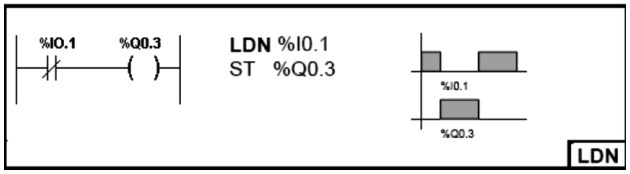
Instrucciones de carga (LD, LDN, LDR, LDF)

Las instrucciones de carga LD, LDN, LDR y LDF corresponden respectivamente a los contactos abierto, cerrado, flanco ascendente y flanco descendente (LDR y LDF sólo se utilizan con entradas del controlador)

- Contactos normales abiertos

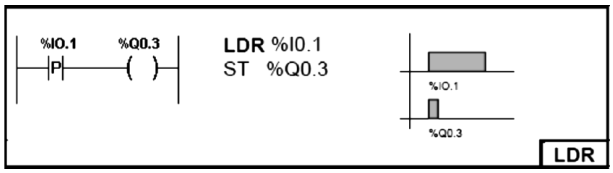


- Contactos normales cerrados



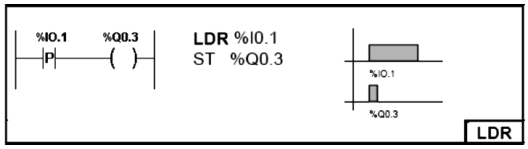
- Contacto flanco ascendente

El tiempo que permanece activa la salida equivale a un ciclo del autómata.



- Contacto flanco descendente

El tiempo que permanece activa la salida equivale a un ciclo del autómata.



Ejemplos

Los siguientes diagramas son ejemplos de instrucciones de carga.
Operandos permitidos.

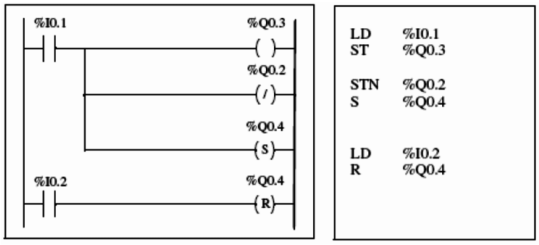
La siguiente tabla enumera los tipos de instrucciones de carga con operandos equivalentes y permitidos de Ladder Logic.

Instrucción de lista	Equivalente Ladder Logic	Operandos permitidos
LD	⊣ ⊢	0/1,%I,%Q,%M,%S,%X,%BLK.x,%•:Xk,[
LDN	⊣/⊢	%I,%Q,%M,%S,%X,%BLK.x,%•:Xk,[
LDR	⊣P⊢	%I
LDF	⊣N⊢	%I

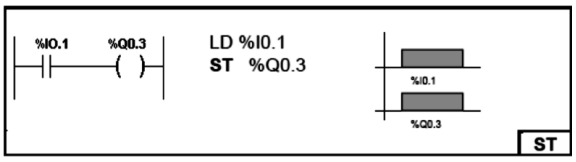
Instrucciones de almacenamiento (ST, STN, R, S)

Las instrucciones de almacenamiento ST, STN, S y R corresponden respectivamente a las bobinas directas e inversas, establecida y restablecida.

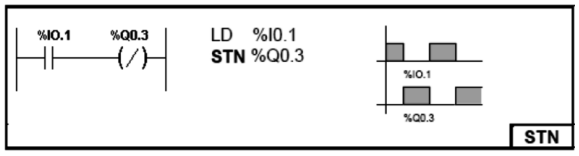
Los siguientes diagramas son ejemplos de instrucciones de almacenamiento:



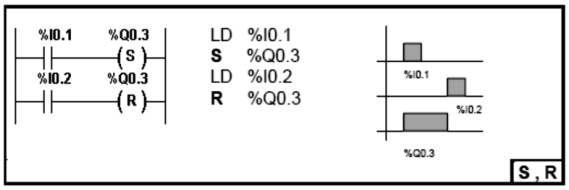
Bobina Directa



Bobina Inversa



Bobina de Set y Reset



Operandos permitidos

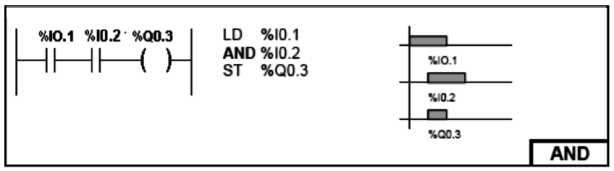
La siguiente tabla enumera los tipos de instrucciones de almacenamiento con operandos equivalentes y permitidos de Ladder Logic.

Instrucción de lista	Equivalente Ladder Logic	Operandos permitidos
ST	( )	%Q,%M,%S,%BLK.x,%•:Xk
STN	(/)	%Q,%M,%S,%BLK.x,%•:Xk
S	(S)	%Q,%M,%S,%X,%BLK.x,%•:Xk
R	(R)	%Q,%M,%S,%X,%BLK.x,%•:Xk

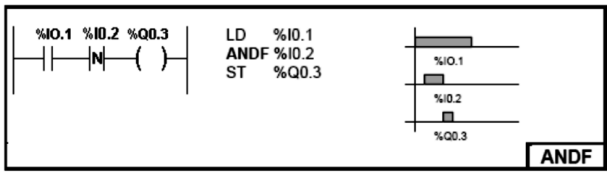
Instrucciones AND lógicas (AND, ANDN, ANDR, ANDF)

Las instrucciones AND realizan una operación lógica AND entre el operando (o su inverso, o su flanco ascendente, o descendente) y el resultado booleario de la instrucción precedente.

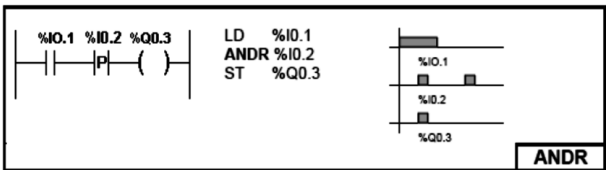
- Producto Lógico



- Producto Lógico Negado

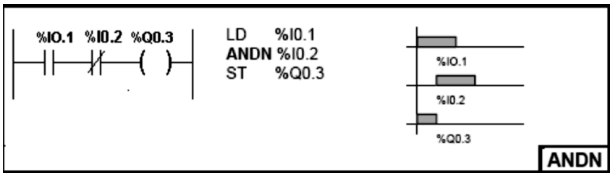


- Producto Lógico Flanco Ascendente



El tiempo que permanece activa la salida equivale a un ciclo de programa.

- Producto Lógico Flanco Descendente



El tiempo que permanece activa la salida equivale a un ciclo de programa.

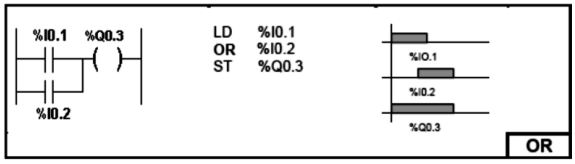
La siguiente tabla enumera los tipos de instrucciones AND con operandos equivalentes y permitidos de Ladder Logic.

Instrucción de lista	Equivalente Ladder Logic	Operandos permitidos
AND	⊣⊢	0/1,%I,%Q,%M,%S,%X,%BLK.x,%e:Xk, [
ANDN	⊣/⊢	%I,%Q,%M,%S,%X,%BLK.x,%e:Xk, [
ANDR	⊣P⊢	%I
ANDF	⊣N⊢	%I

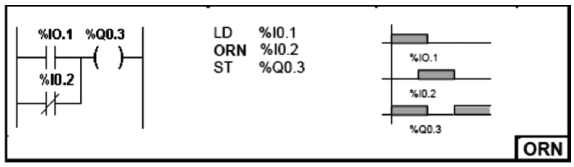
Instrucciones OR lógicas (OR, ORN, ORR, ORF)

Las instrucciones OR realizan una operación lógica OR entre el operando (o su inverso; o su flanco ascendente o descendente) y el resultado booleario de la instrucción precedente.

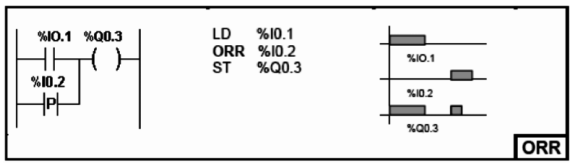
- Suma Lógica



- Suma Lógica Negada

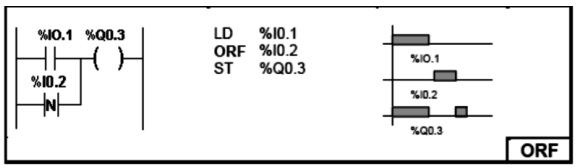


- Suma Lógica con Flanco Ascendente



El tiempo que permanece activa la salida equivale a un ciclo del autómata.

- Suma lógica con Flanco Descendente



El tiempo que permanece activa la salida equivale a un ciclo del autómata.

- Operandos permitidos

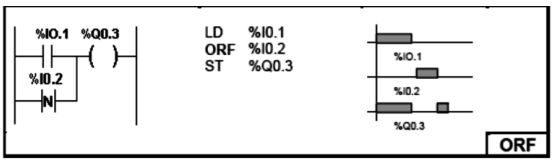
La siguiente tabla enumera los tipos de instrucciones OR con operandos equivalentes y permitidos de Ladder Logic.

Instrucción de lista	Equivalente Ladder Logic	Operandos permitidos
OR		0/1,%I,%Q,%M,%S,%X,%BLK.x,%*:Xk
ORN		%I,%Q,%M,%S,%X,%BLK.x,%*:Xk
ORR		%I
ORF		%I

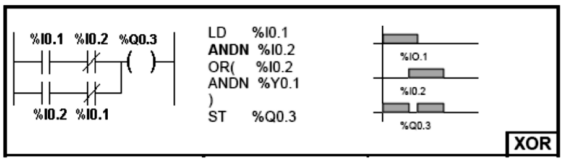
Instrucciones de OR exclusivo (XOR, XORN, XORR, XORF)

Estas instrucciones realizan un O exclusivo entre el operando (o su inverso, o flanco ascendente o descendente) y el resultado booleano de la instrucción anterior. Esta operación es conocida también como **comparador de desigualdad**, puesto que el resultado es 1 (ON), cuando los operandos involucrados son distintos.

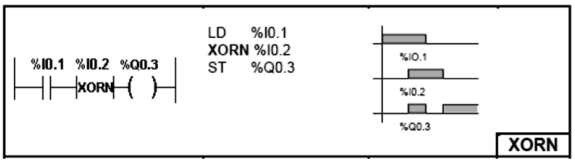
- Suma Lógica Exclusiva



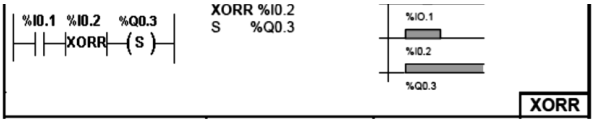
Las instrucciones O exclusiva pueden realizarse también con contactos e instrucciones comunes. A continuación detallamos la forma de realizarlo para ilustrar la lógica de la instrucción.



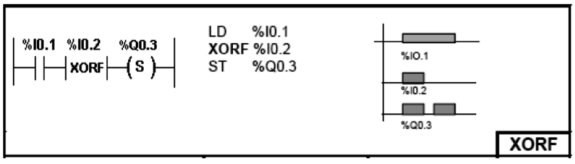
- Suma Lógica Exclusiva Negada



- Suma lógica Exclusiva Flanco Ascendente



- Suma lógica Exclusiva Flanco Descendente



Operandos permitidos

La siguiente tabla enumera los tipos de instrucciones XOR y operandos permitidos.

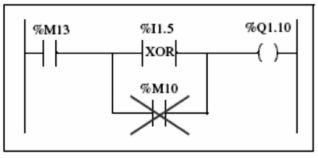
Lista de instrucciones	Operandos permitidos
XOR	%I,%Q,%M,%S,%X,%BLK.x,%•:Xk
XORN	%I,%Q,%M,%S,%X,%BLK.x,%•:Xk
XORR	%I
XORF	%I

Casos especiales

Precauciones especiales para utilizar instrucciones XOR en programas de Ladder Logic:

- No inserte contactos XOR en la primera posición de un escalón.
- No inserte contactos XOR de forma paralela con otros elementos de Ladder Logic (consulte el siguiente ejemplo)

Como se muestra en el siguiente ejemplo, la inserción de un elemento de forma paralela con el contacto XOR generará un error de validación.



Instrucción NOT (N)

La instrucción NOT (N) niega el resultado booleario de la instrucción anterior.

Ejemplo

A continuación se muestra un ejemplo de uso de la instrucción NOT.

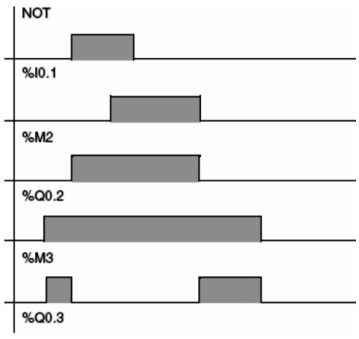
```
LD    %I0.1
OR    %M2
ST    %Q0.2
N
AND   %M3
ST    %Q0.3
```

Operandos permitidos

No aplicable.

Cronograma.

El siguiente diagrama muestra la temporización de la instrucción NOT.



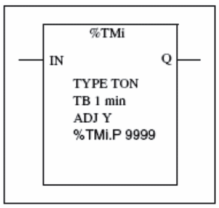
Bloque de función del temporizador (%TMi)

Cada uno de los temporizadores puede configurarse de una de las tres formas propuestas por la normativa IEC61131.

Los 3 tipos propuestos son:

1. TON (temporizador de retardo a la conexión): utilice este tipo de temporizador para controlar las acciones de retardo a la conexión.
2. TOF (temporizador de retardo a la desconexión): utilice este tipo de temporizador para controlar las acciones de retardo a la desconexión.
3. TP (pulso de temporizador): utilice este tipo de temporizador para generar pulsos de duración determinada.

Los retardos o períodos de pulsos se pueden programar y modificar utilizando TwidoSoft. A continuación se muestra una ilustración del bloque de función del temporizador:



Bloque de función del temporizador

Parámetros

El bloque de función del temporizador presenta los siguientes parámetros:

Parámetro	Etiqueta	Valor
Número de Temporizador	%TMi	Controlador compacto 0 a 63. Controladores modulares 0 a 127.
Tipo	TON	Retardo a la conexión (predeterminado)
	TOF	Retardo a la desconexión.
	TP	Pulso (monoestable)
Base de tiempo	TB	1 min (predeterminado), 1 s, 100 ms, 10 ms, 1 ms. (para TM0 y TM1)
Valor actual	%TMi.V	Palabra que aumenta de 0 a %TMi.P cuando el temporizador está en funcionamiento. Se puede leer y comprobar, pero no se puede escribir desde el programa. %TMi.V se puede modificar utilizando el editor de datos.
Valor preestablecido	%TMi.P	0 - 9999. Palabra que se puede leer, comprobar escribir desde el programa. El valor predeterminado es 9999. El período o retardo generado es igual a %TMi.P x TB
Editor de datos	Y/N	Y: Sí, el valor preestablecido %TMi.P puede modificarse utilizando el editor de datos. N: No, el valor preestablecido %TMi.P no se puede modificar.
Establecimiento de entrada (o instrucción)	IN	Inicia el temporizador en flanco ascendente (tipos TON o TP) o en flanco descendente (tipo TOF)
Salida del temporizador	Q	El bit asociado %TMi.Q se establece en 1 dependiendo de la función realizada: TON, TOF o TP.1.

El tiempo t de temporización se calcula de la siguiente forma:

t = BT x %TMi.P

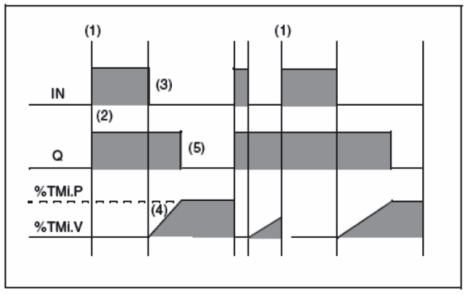
RECUERDE que...
Cuanto mayor sea el valor preestablecido, mayor será la precisión del temporizador.

Tipo de temporizador TOF

El tipo de temporizador TOF (temporizador de retardo a la desconexión) se utiliza para controlar las acciones de retardo a la desconexión. Este retardo se puede programar con TwidoSoft.

Cronograma

El siguiente cronograma ilustra el funcionamiento del temporizador de tipo TOF.



Operación

En la siguiente tabla se describe el funcionamiento del temporizador de tipo TOF.

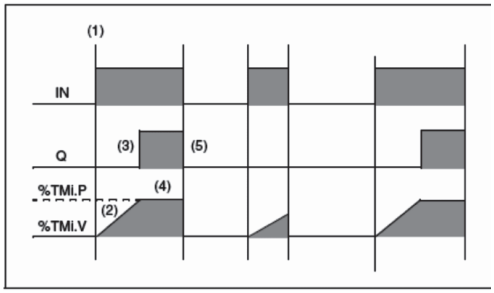
Fase	Descripción
1	El valor actual %TMi.V se establece en 0 en un flanco ascendente en la entrada IN, aun cuando el temporizador se encuentre en ejecución.
2	El bit de salida %TMi.Q se establece en 1 cuando se detecte un flanco ascendente en la entrada N.
3	El temporizador inicia en el flanco descendente de la entrada IN.
4	El valor actual %TMi.V aumenta a %TMi.P en incrementos de una unidad por pulso de la base de tiempo TB.
5	El bit de salida %TMi.Q se restablece a 0 cuando el valor actual llega a %TMi.P.

Tipo de temporizador TON

El tipo de temporizador TON (temporizador de retardo a la conexión) se utiliza para controlar las acciones de retardo a la conexión. Este retardo se puede programar con TwidoSoft.

Cronograma

El siguiente cronograma ilustra el funcionamiento del temporizador de tipo TON.

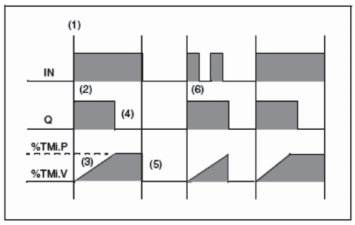


Operación
En la siguiente tabla se describe el funcionamiento del temporizador de tipo TON.

Fase	Descripción
1	El temporizador inicia en el flanco ascendente de la entrada IN.
2	El valor actual %TMi.V aumenta de 0 a %TMi.P en incrementos de una unidad por pulso de la base de tiempo TB.
3	El bit de salida %TMi.Q se establece en 1 cuando el valor actual llega a %TMi.P.
4	El bit de salida %TMi.Q permanece en 1 mientras la entrada IN esté en 1.
5	Si se detecta un flanco descendente en la entrada IN, el temporizador se detiene, aun cuando el temporizador no haya alcanzado el valor %TMi.P, y %TMi.V se establece en 0.

Tipo de temporizador TP
El tipo de temporizador TP (pulso de temporizador) se utiliza para generar pulsos de duración determinada. Este retardo se puede programar con TwidoSoft.

Cronograma
El siguiente cronograma ilustra el funcionamiento del temporizador de tipo TP:



Operación
En la siguiente tabla se describe el funcionamiento del temporizador de tipo TP.

Fase	Descripción
1	El temporizador se inicia en el flanco ascendente de la entrada IN. El valor actual %TMi.V se establece en 0 si el temporizador todavía no se ha iniciado.
2	El bit de salida %TMi.Q se establece en 1 cuando se inicia el temporizador.
3	El valor actual %TMi.V del temporizador aumenta de 0 a %TMi.P en incrementos de una unidad por pulso de la base de tiempo TB
4	El bit de salida %TMi.Q se establece en 0 cuando el valor actual llega a %TMi.P
5	El valor actual %TMi.V se establece en 0 cuando %TMi.V es igual a %TMi.P y la entrada IN vuelve a 0
6	Este temporizador no se puede restablecer. Una vez %TMi.V es igual a %TMi.P y la entrada IN es 0, %TMi.V se establecerá en 0

7.4

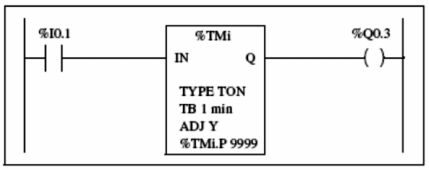
Programación y configuración de temporizadores

Los bloques de función del temporizador (%TMi) se programan de la misma manera, independientemente del modo en que vayan a utilizarse. La función del temporizador (TON, TOF o TP) se selecciona durante la configuración.

Ejemplo

La siguiente ilustración es un bloque de función del temporizador con ejemplos de programación.

Bloque de función del temporizador



Programación reversible

```
BLK    %TM1
LD     %I0.1
IN
OUT_BLK
LD     Q
ST     %Q0.3
END_BLK
```

Programación no reversible

```
LD     %I0.1
IN     %TM1
LD     %TM1.Q
ST     %Q0.3
```

Configuración

Durante la configuración deben introducirse los siguientes parámetros:

- Tipo de temporizador: TON, TOF o TP.
- Tiempo base (TB): 1 min., 1s, 100 ms., 10 m.s o 1 ms.
- Valor preestablecido (%TMi.P): 0 a 9999.
- Ajuste: Sí o No (S o N)

Casos especiales

La siguiente tabla contiene una lista de casos especiales de programación y configuración de temporizadores.

Caso especial	Descripción
Efecto de un reinicio en frío (%S0=1)	Fuerza el valor actual a 0. Establece la salida %TMi.Q en 0. El valor preestablecido se restablece al valor definido durante la configuración.
Efecto de un reinicio en caliente (%S1=1)	No tiene ningún efecto en los valores actuales y presentes del temporizador. El valor actual no varía durante un corte de alimentación.
Efecto de una detención del controlador	No inmovilizará el valor actual.
Efecto de un salto del programa	Un salto sobre el bloque del temporizador no mantendrá el temporizador. El temporizador continúa aumentando hasta que alcanza el valor preestablecido (%TMi.P). En este punto, el bit de finalización (%TMi.Q) asignado a la salida Q del bloque del temporizador cambia de estado; sin embargo, la salida asociada cableada directamente a la salida del bloque no se activa y el controlador no la explora.
Comprobación por bit %TMi.Q (bit de finalización)	Es recomendable realizar una prueba del bit. %TMi.Q una única vez en el programa.
Efecto de modificar el valor preestablecido %TMi.P	Modificar el valor presente mediante una instrucción o ajustando el valor sólo tiene efecto cuando se vuelve a activar el temporizador.

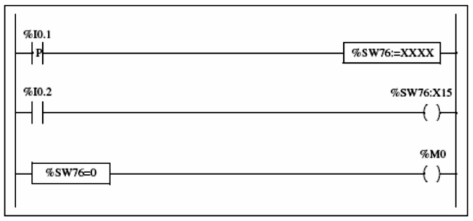
Temporizadores con un tiempo base de 1 ms

El tiempo base de 1 ms sólo está disponible en temporizadores %TM0 y %TM1. Las cuatro palabras del sistema %SW76, %SW77, %SW78 y SW79 se pueden utilizar como "relojes de arena". El sistema hace que estas cuatro palabras disminuyan individualmente cada milisegundo si tienen un valor positivo. Se pueden conseguir varias temporizaciones, cargando de manera sucesiva una de estas palabras o realizando comprobaciones de los valores inmediatos. Si el valor de uno de estas cuatro palabras es menor que 0, no se modificará. Es posible inmovilizar un temporizador estableciendo el bit 15 correspondiente en 1 y cancelar la inmovilización restableciéndolo en 0.

Ejemplo

A continuación se muestra un ejemplo de programación de un bloque de función del temporizador.

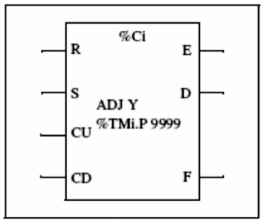
```
LDR  %I0.1          (Ejecución del temporizador en el flanco ascendente de %I0.1)
[%SW76:=XXXX] (XXXX = valor necesario)
LD   %I0.2          (gestión opcional de inmovilización, la entrada I0.2
                    queda inmovilizada)
ST   %SW76:X15
LD   [%SW76=0]  (restablecimiento final del temporizador)
ST   %M0
.....
```



7.5

Bloque de función del contador progresivo/regresivo (%Ci)

El bloque de función del contador (%Ci) proporciona un recuento de eventos progresivo o regresivo. Estas dos operaciones pueden realizarse de forma simultánea.
A continuación se muestra una ilustración del bloque de función del contador progresivo/regresivo.



Up/down counter function block

Parámetros

El bloque de función del contador tiene los siguientes parámetros:

Parámetro	Etiqueta	Valor
Número de contador	%Ci	0 a 31
Valor actual	%Ci.V	La palabra aumenta o disminuye con arreglo a las entradas (o instrucciones) CU y CD. El programa puede leerla y comprobarla, pero no escribirla. Utilice el editor de datos para modificar %Ci.V.
Valor preestablecido	%Ci.P	0-%Ci.P-9999. La palabra puede leerse comprobarse y escribirse (valor preestablecido: 9999)

Parámetro	Etiqueta	Valor
Editar utilizando el editor de datos	S/N	• S: Sí, el valor preestablecido puede modificarse utilizando el editor de datos. • N: No, el valor preestablecido no puede modificarse utilizando el editor de datos.
Restablecer entrada (o instrucción)	R	En estado 1: %Ci.V = 0.
Establecer entrada (o instrucción)	S	En estado 1: %Ci.V = %Ci.P.
Entrada de conteo progresivo (o instrucción)	CU	Incrementos %Ci.V en un flanco ascendente.
Entrada de conteo regresivo (o instrucción)	CD	Disminuciones %Ci.V en un flanco ascendente.
Salida de trasgresión por debajo de rango	E (Vacío)	El bit asociado %Ci.E=1, cuando el contador regresivo %Ci.V cambia de 0 a 9999 (establecido a 1 cuando %Ci.V alcanza 9999 y se restablece a 0 si el contador continúa con el conteo regresivo).
Salida predeterminada alcanzada	D (Hecho)	El bit asociado %Ci.D=1, cuando %Ci.V=%Ci.P.
Salida de desborde	F (Llena)	El bit asociado %Ci.F=1, cuando %Ci.V cambia de 9999 a 0 (establecido a 1 cuando %Ci.V alcanza 0 y se restablece a 0 si el contador continúa con el conteo progresivo).

Operación

La siguiente tabla describe las fases principales de la operación del contador progresivo/regresivo.

Operación	Acción	Resultado
Conteo progresivo	Aparece un flanco ascendente en la CU de entrada de conteo progresivo (o se activa la CU de instrucción).	El valor actual de %Ci.V aumenta en una unidad.
	El valor actual de %Ci.V es igual al valor %Ci.P preestablecido.	El bit %Ci.D de salida "preestablecida alcanzada" asignado a la salida D cambia a estado 1.
	El valor actual %Ci.V cambia de 9999 a 0.	El bit de salida %Ci.F (desborde de conteo progresivo) cambia a estado 1.
	Si el contador continúa con el conteo progresivo.	El bit de salida %Ci.F (desborde de conteo progresivo) se restablece a 0.
Conteo regresivo	Aparece un flanco ascendente en el CD de entrada de conteo regresivo (o se activa el CD de instrucción).	El valor actual de %Ci.V disminuye en una unidad.
	El valor actual %Ci.V cambia de 0 a 9999.	El bit de salida %Ci.E (trasgresión por debajo de rango) cambia a estado1.
	Si el contador continúa con el conteo regresivo.	El bit de salida %Ci.E (trasgresión por debajo de rango) se restablece como 0.

Operación	Acción	Resultado
Conteo progresivo/ regresivo	Para utilizar simultáneamente las funciones de conteo progresivo y regresivo (o para activar las instrucciones CD y CU), deben controlarse las dos entradas correspondientes CU y CD. Estas dos entradas se examinan sucesivamente. Si ambas están en 1, el valor actual permanece intacto.	
Restablecer	La entrada R se establece a estado 1 (o la instrucción R se activa)	El valor actual %Ci.V se fuerza a 0. Las salidas %Ci.E, %Ci.D y %Ci.F están a 0. La entrada restablecida tiene prioridad.
Establecer	Si la entrada S está en estado 1 (o se activa la instrucción S) y la entrada restablecida está a 0 (o la instrucción R está inactiva).	El valor actual %Ci.V toma el valor %Ci.P y la salida %Ci.D se establece a 1.

Casos especiales

La siguiente tabla contiene una lista de casos especiales de programación y configuración de contadores.

Caso especial	Descripción
Efecto de un reinicio en frío (%S0=1)	• El valor actual %Ci se establece a 0. • Los bits de salida %Ci.E, %Ci.D y %Ci.F se establecen a 0. • El valor preestablecido se inicializa con el valor definido durante la configuración.
Efecto de un reinicio en caliente (%S1=1) de una detención del controlador	No tiene ningún efecto sobre el valor actual del contador (%Ci.V)
Efecto de modificar el valor preestablecido %Ci.P	La modificación del valor preestablecido mediante una instrucción o ajustándolo entra en vigor cuando la aplicación procesa el bloque (activación de una de las entradas)

7.6 Programación y configuración de contadores

El siguiente ejemplo es un contador que proporciona un conteo de elementos hasta 5000. Cada pulso de entrada %I1.2 (cuando el bit interno %M0 está en 1) incrementa el contador %C8 hasta su valor preestablecido final (bit %C8.D=1)
El contador se restablece mediante la entrada %I1.1.

Ejemplo

Modelo de programación:
La siguiente ilustración es un bloque de función del contador con ejemplos de programaciones reversibles y no reversibles.

Bloque de función del controlador

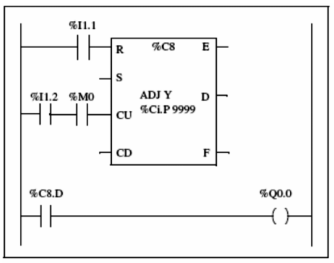


Diagrama Ladder Logic

```
BLK   %C8
LD    %I1.1
R
LD    %I1.2
AND   %M0
CU
END_BLK
LD    %C8.D
ST    %Q0.0
```

Programación reversible

```
LD    %I1.1
R     %C8
LD    %I1.2
AND   %M0
CU    %C8
LD    %C8.D
ST    %Q0.0
```

Programación no reversible

Configuración:
Deben introducirse los siguientes parámetros durante la configuración:

- *Valor preestablecido (%Ci.P): establecido a 5000 en este ejemplo*
- *Ajuste: Sí*

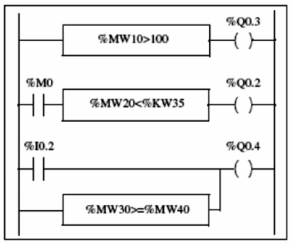
Instrucciones de comparación

Las instrucciones de comparación se utilizan para comparar dos operandos. La siguiente tabla enumera los tipos de instrucciones de comparación.

Fase	Descripción
>	Prueba si el operando 1 es mayor que el operando 2
>=	Prueba si el operando 1 es mayor o igual que el operando 2
<	Prueba si el operando 1 es menor que el operando 2
<=	Prueba si el operando 1 es menor o igual que el operando 2
=	Prueba si el operando 1 es igual que el operando 2
<>	Prueba si el operando 1 es diferente del operando 2

Estructura

La comparación se ejecuta entre corchetes siguiendo las instrucciones LD, AND y OR. El resultado es 1 cuando la comparación solicitada es verdadera. Ejemplos de instrucciones de comparación:




```
LD    [%MW10 > 100]
ST    %Q0.3

LD    %M0
AND   [%MW20 < %KW35]
ST    %Q0.2

LD    %I0.2
OR    [%MW30>=%MW40]
ST    %Q0.4
```

8 Grafcet

Introducción

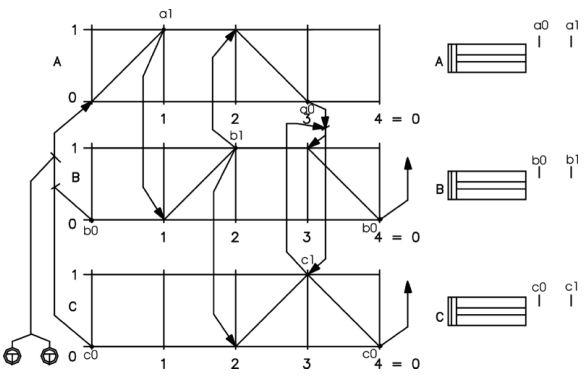
Los primeros métodos para el desarrollo de automatismos eran puramente intuitivos, llevados a términos por expertos y desarrollados basándose en la experiencia.
Todo automatismo secuencial o concurrente se puede estructurar en una serie de etapas que representan estados o subestados del sistema, en los cuales se realiza una o más acciones, así como transiciones, que son las condiciones que deben darse para pasar de una etapa a otra.
El Grafcet es un diagrama funcional que describe la evolución del proceso que se pretende automatizar, indicando las acciones que hay que realizar sobre el proceso y qué informaciones provocan estas acciones.
Partiendo de él se pueden obtener las secuencias que ha de realizar el PLC. Su empleo para resolver tareas de automatización facilita el diálogo entre personas con niveles de formación técnica diferente, tanto en el momento del análisis del proceso a automatizar, como posteriormente en el mantenimiento y reparación de averías.
El GRAFCET surge en Francia a mediados de los años 70, debido a la colaboración de algunos fabricantes de autómatas, como Telemecanique y Aper con dos organismos oficiales, AFCET (Asociación francesa para la cibernética, economía y técnica) y ADEPA (Agencia nacional para el desarrollo de la producción automatizada) Homologado en Francia, Alemania, y posteriormente por la comisión Electrónica Internacional (IEC 848, año 1988)

8.1 Elementos Gráficos

Descripción de los pasos

Antes de describir los pasos recordemos los diagramas de Espacio-Fase.
En este tipo de diagramas se representa la secuencia de acción que se quiere automatizar. Para describirlo en forma precisa se representa la secuencia de acción de los actuadores (cilindros, motores, luces, sirenas, etc.) y el encadenamiento de las señales de mando (finales de carrera, pulsadores, llaves, o sensores de cualquier tipo)
Se utilizan para ello dos ejes coordenados. En el eje vertical se representa el estado de los actuadores del sistema, utilizando los valores binarios (0 y 1)
Se adoptará valor “0” para indicar la posición de reposo de los actuadores (motores detenidos y cilindros con vástagos retraídos, etc.)
Y con valor “1” para indicar el estado del elemento actuado (motor en marcha, cilindro con su vástago extendido, etc.)
En el eje horizontal se indicarán las fases o pasos en que se subdivide el ciclo de trabajo. Estos pasos o fases están caracterizados por la modificación o cambio del estado de un elemento constitutivo del mando.

Ejemplo



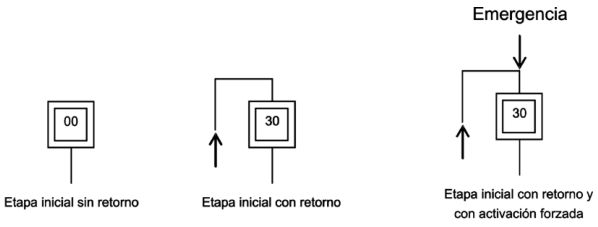
Modelo de aplicación del diagrama Espacio-Fase:

En este diagrama de espacio fase se puede ver como las señales del comando bimanual "Y" b0 "Y" c0 permiten el arranque de la secuencia. Se puede ver también que primero se extiende el vástago del cilindro "A". Una vez que el cilindro "A" llega a su final de carrera extendido se activa a1, que es quien da señal para que se extienda el vástago del cilindro "B". Cuando "B" llegue a su final de carrera extendido b1, esta señal da la orden para que se retraiga el vástago del cilindro "A" y se extienda el vástago del cilindro "C". Una vez que "A" activó su final de carrera retraído a0, "Y" "C" activó su final de carrera extendido c1 se retraen los vástagos de los cilindros "B" y "C". El ciclo podrá recomenzar sólo cuando los finales de carrera b0 y c0 se activen y se presione el comando bimanual.

Se puede ver entonces que:

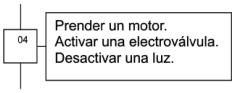
***1° fase**: se extiende el vástago del cilindro "A".*
***2° fase**: se extiende el vástago del cilindro "B".*
***3° fase**: se retrae el vástago del cilindro "A" y simultáneamente se extiende el vástago del cilindro "C".*
***4° fase**: se retraen los vástagos de los cilindros "B" y "C" simultáneamente.*

Como se dijo antes el Grafcet es un diagrama funcional que al igual que el diagrama de espacio - fase describe la evolución del proceso. El Grafcet también está formado por etapas o pasos, y cada una de ellas llevará asociada una o varias acciones a realizar sobre el proceso.
Las etapas o pasos representan cada uno de los estados del sistema.
Las etapas se representan con cuadrados y con un número o una letra y un subíndice. En cualquier caso el número indica el orden que ocupa el paso dentro del Grafcet.
Para distinguir el comienzo del Grafcet, la primera etapa se representa con un doble cuadrado.
Los pasos o etapas iniciales de un sistema se activan al iniciar el GRAFCET. Una vez que se han iniciado, tienen el mismo tratamiento que las otras etapas. Un sistema debe tener como mínimo un paso o etapa inicial.

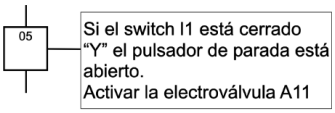


Los pasos o etapas representan los estados estables del sistema. Se representan mediante un cuadrado numerado, los pasos o etapas deben estar numerados; aún que no necesariamente de forma correlativa, no puede haber dos pasos o etapas con el mismo número, las etapas o pasos pueden estar activos o inactivos. Al representar el estado del GRAFCET en un momento dado, se puede indicar que una etapa está activa, con un punto de color o coloreando todo el cuadrado.

En las etapas, puede o no haber acciones asociadas. Estas acciones asociadas con cada etapa se representan con un rectángulo, donde se indica el tipo de acción a realizarse, puede ser que una etapa pueda llevar asociada varias acciones.



Más aún la activación de una salida puede estar sujeta a una condición lógica. Está condición lógica puede por ejemplo ser función de señales de entrada, de variables internas, o del estado activo o inactivo de otros pasos del GRAFCET.



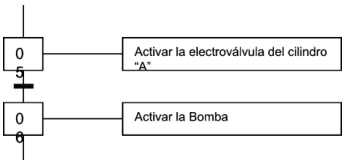
Entonces cuando el paso 05 esté activo será necesario que: el switch I1 esté cerrado y el pulsador de parada está abierto para activar la electroválvula A11.

Líneas de evolución

Las líneas de evolución unen entre si las etapas que representan actividades consecutivas. Las líneas se entenderán siempre orientadas de arriba abajo, a menos que se represente una flecha en sentido contrario. Dos líneas de evolución que se crucen debe de interpretarse, en principio que no están unidas

Transiciones

En Grafcet, el proceso se descompone en una serie de, pasos que son activados uno tras otro. Por tanto, tendrá que existir una condición que al validarse permita pasar de un paso a otro. A esta condición se la llama transición.



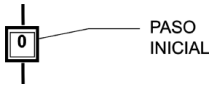
En la figura anterior, hay dos etapas y una transición entre ellas; para que el proceso evolucione del paso 5 al paso 6, es necesario que el paso 5 esté activo y que la transición entre los dos pasos sea válida.

Que la transición sea válida implica que la señal o conjunto de señales que se agrupa en esa transición están presentes o activas. Sólo entonces se produce la activación del paso 6 y la desactivación del 5.

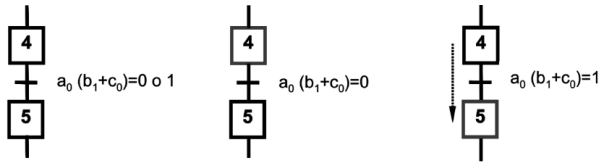
Ese conjunto de señales que forman la transición es información que proviene del exterior (ordenes del operador, de contadores, de temporizadores, de finales de carreras, etc.) y/o de variables auxiliares y/o del estado activo o inactivo de algunos pasos. En un Grafcet de una sola rama sólo puede existir un paso activo; por lo tanto se produce la activación de la etapa 6 y la desactivación de la etapa 5.

8.2 Reglas de Evolución

- El proceso se descompone en etapas, que serán activadas en forma secuencial.
- Una o varias acciones se asocian a cada etapa. Estas acciones sólo están activas cuando la etapa está activa.
- Una etapa se hace activa cuando la precedente lo está y la transición entre ambas ha sido activada.
- La activación de una transición implica la activación de la etapa siguiente y la desactivación de la precedente.
- El paso o etapa inicial estará en un comienzo incondicionalmente activo y será marcado con un doble recuadro.



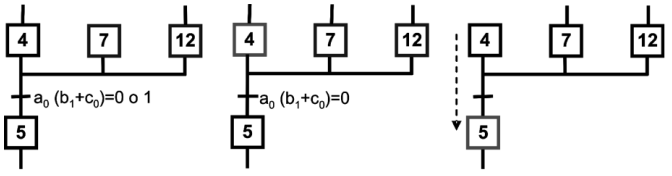
- Una transición puede estar habilitada o no.
- Estará habilitada cuando el paso inmediato anterior esté activo.
- La transición no podrá saltar si no está habilitada o si las señales asociadas en la condición lógica que la generó no son verdaderas.
- El salto de una transición lleva a la activación de todos los pasos que le siguen inmediatamente y desactiva todos los pasos inmediatamente anteriores a ella.



Transición inhabilitada
Como el paso 4 está inactivo la transición está inhabilitada

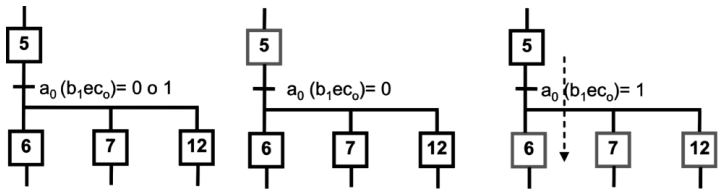
Transición habilitada
Como el paso 4 está activo la transición está habilitada, pero no podrá saltar porque que la condición lógica no es verdadera.

Salto de una transición
Como el paso 4 está activo la transición está habilitada la condición lógica es verdadera entonces es posible el salto.



Transición habilitada
Los pasos 4, el 7 y el 12 están activos entonces la transición está habilitada, pero no podrá saltar porque que la condición lógica no es verdadera.

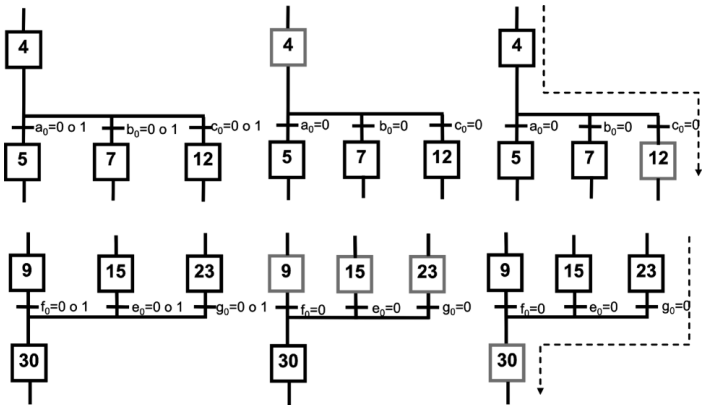
Salto de una transición
Los pasos 4,7y 12 están activos entonces la transición está habilitada. Como la condición lógica es verdadera entonces es posible el salto.



Transición inhabilitada
Como el paso 5 está inactivo la transición está inhabilitada

Transición habilitada
El paso 5 está activo entonces la transición está habilitada, pero no podrá saltar porque que la condición lógica no es verdadera

Salto de una transición
El paso 5 está activo entonces la transición está habilitada. Como la condición lógica es verdadera entonces es posible el salto y la activación simultánea de los pasos siguientes



8.3 Descripción de las instrucciones Grafcet para el autómata Twido

Instrucciones Grafcet

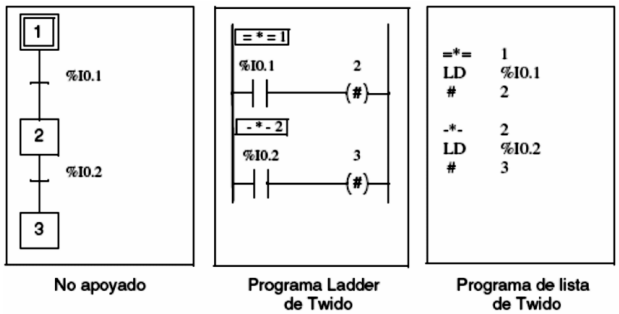
La tabla que aparece a continuación enumera todas las instrucciones y objetos necesarios para programar un diagrama Grafcet con un PLC Twido.

Representación gráfica (1)	Transcripción en lenguaje TwidoSoft	Función
Paso inicial	=*= i	Comenzar paso inicial (2)
Paso de transición	# i	Activar paso i tras desactivar el paso actual
	-*- i	Comenzar el paso i y validar la transición asociada (2)
	#	Desactivar el paso actual sin activar ningún otro paso
	#Di	Desactivar el paso i y el paso actual
	=*= POST	Iniciar procesamiento posterior y finalizar procesamiento secuencial
	%Xi	Se puede comprobar y escribir el bit asociado con el paso i (el número máximo de pasos depende del controlador).
Xi	LD %Xi, LDN %Xi AND %Xi, ANDN %Xi, OR %Xi, ORN %Xi XOR %Xi, XORN %Xi	Comprobar actividad del paso i
Xi (S)	S %Xi	Activar paso i
Xi (R)	R %Xi	Desactivar paso i

1. No apoya Grafcet gráfico.
2. El primer paso =*=i o -*-i escrito indica el inicio del procesamiento secuencial y, por lo tanto, el final del procesamiento previo.

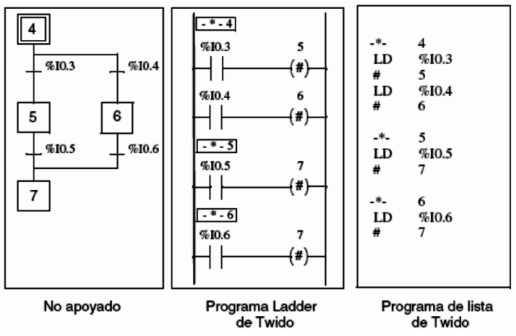
Ejemplos de Grafcet

Secuencia lineal:

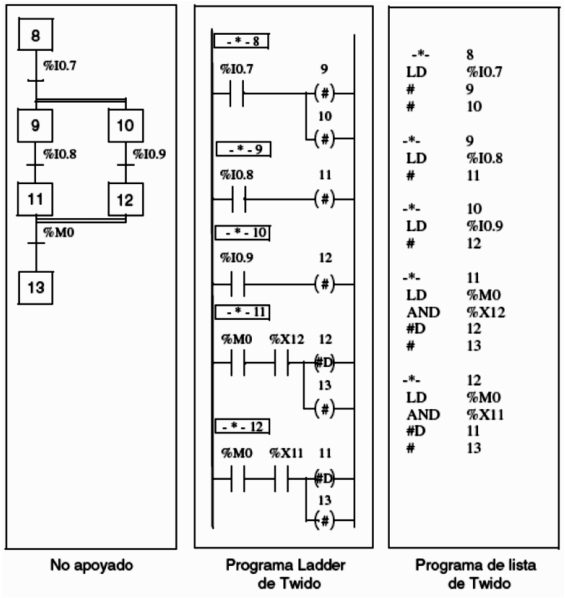


No apoyado | Programa Ladder de Twido | Programa de lista de Twido

Secuencia alternativa:



Secuencias simultáneas:



RECUERDE que...
Para que un diagrama Grafcet funcione, debe haber al menos un paso activo utilizando la instrucción *=i (paso inicial) o el diagrama debe ubicarse antes durante el procesamiento previo utilizando el bit de sistema %S23 y la instrucción S %Xi.

8.4 Descripción de la estructura del programa Grafcet

Un programa Grafcet de TwidoSoft consta de tres partes:

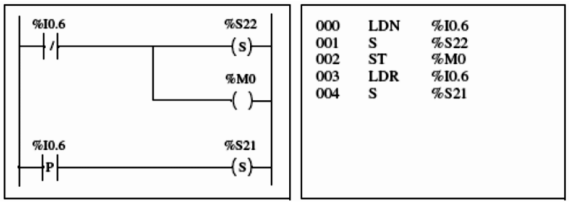
- Procesamiento previo
- Procesamiento secuencial
- Procesamiento posterior

Procesamiento previo

El procesamiento previo consta de las siguientes partes:
- Recuperación de la alimentación
- Errores
- Cambios de modo de funcionamiento
- Pasos Grafcet de ubicación previa
- Entrada lógica

Ejemplo

En el ejemplo de ubicación previa que aparece a continuación (área anterior al primer paso Grafcet), el estado 0 de la entrada %I0.6 solicita que el diagrama Grafcet se restaure estableciendo el bit de sistema %S22 en 1. Esto desactivará los pasos activos. El flanco ascendente de la entrada %I0.6 coloca el diagrama antes del paso X1. Finalmente, la utilización del bit de sistema %S21 fuerza la inicialización de Grafcet.



*El procesamiento previo comienza con la primera línea del programa y finaliza con la primera aparición de una instrucción "= * =" o "- * -". Existen tres bits de sistema designados al control de Grafcet: %S21, %S22 y %S23. La aplicación establece cada uno de estos bits de sistema en 1 (si fuera necesario), normalmente durante el procesamiento previo. El sistema lleva a cabo la función asociada cuando finaliza el procesamiento previo y, entonces, el sistema restaura bit de sistema a 0.*

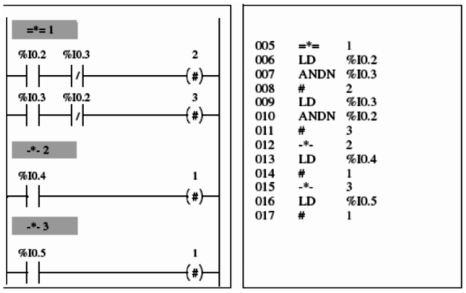
Bit de sistema	Nombre	Descripción
%S21	Iniciación de Grafcet	Todos los pasos activos se desactivan y los pasos iniciales se activan.
%S22	Restablecer Grafcet	Se desactivan todos los pasos.
%S23	Ubicación previa de Grafcet	Este bit se debe establecer en 1 si %Xi han sido escritos de manera explícita por la aplicación durante el procesamiento previo. Si el procesamiento previo mantiene el bit en 1 sin ningún cambio explícito de los objetos %Xi, Grafcet se congela (no se tienen en cuenta las actualizaciones)

Procesamiento secuencial

El procesamiento secuencial se realiza en el diagrama (instrucciones que representan el diagrama)

- Pasos
- Acciones asociadas a los pasos
- Transiciones
- Condiciones de transición

Ejemplo

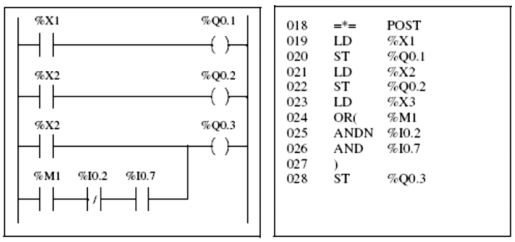


*El procesamiento secuencial termina con la ejecución de la instrucción "= * = POST" o con la finalización del programa.*

Procesamiento posterior
El procesamiento posterior consta de las siguientes partes:

- Comandos del procesamiento secuencial para controlar las salidas
- Dispositivos de bloqueo de seguridad específicos para las salidas

Ejemplo

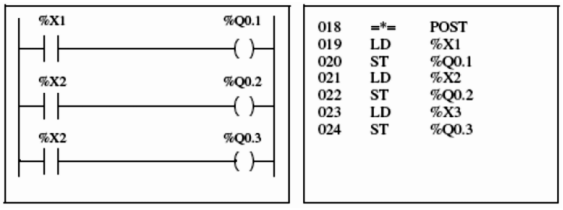


8.5 Acciones asociadas con los pasos Grafcet

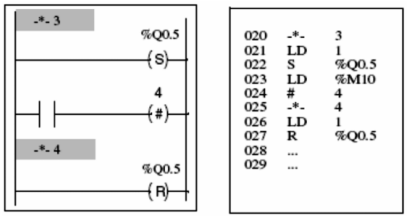
Un programa Grafcet de TwidoSoft ofrece dos modos de programar acciones asociadas con los pasos:

- En la sección de procesamiento posterior
- En las instrucciones de lista o escalones de Ladder Logic de los propios pasos
- Asociación de acciones en el procesamiento posterior

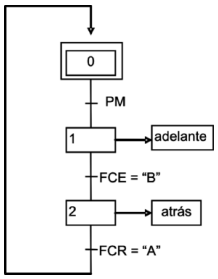
En caso de que existan limitaciones en el modo de seguridad o de ejecución, es preferible programar acciones en la sección de procesamiento posterior de una aplicación Grafcet. Puede utilizar las instrucciones de lista Establecer y Restablecer o conectar bobinas en el programa Ladder Logic para activar los pasos de Grafcet (%Xi)



Ejemplo
Puede programar las acciones asociadas a los pasos dentro de las instrucciones de lista o escalones de Ladder Logic. En este caso, la instrucción de lista o el escalón de Ladder Logic no se examinan a menos que esté activo el paso. Éste es el modo más eficaz, claro y sostenible de utilizar Grafcet.

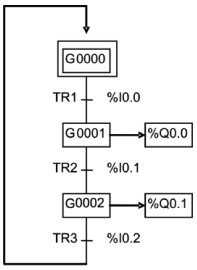


8.6 Grafcet a programar



Conexión de las entradas/salidas:

Entradas:	PM	%I0.0
	FCR	%I0.1 (Final de carrera retraído)
	FCE	%I0.2 (Final de carrera extendido)
Salidas:	A11	%Q0.0 (adelante)
	A10	%Q0.1 (atrás)



8.7 Evaluación

1. ¿En que consiste la programación de un autómata?
2. ¿Qué es el GRAFCET?
3. ¿De que elementos gráficos está compuesto el GRAFCET?
4. ¿Cuándo se hace activa una etapa?
5. ¿Qué diferentes estructuras tiene el GRAFCET?
6. ¿Cuáles son los lenguajes de programación más utilizados?
7. Explicar los conceptos generales del lenguaje de contactos.

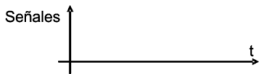
9 Ejercicios de Aplicación

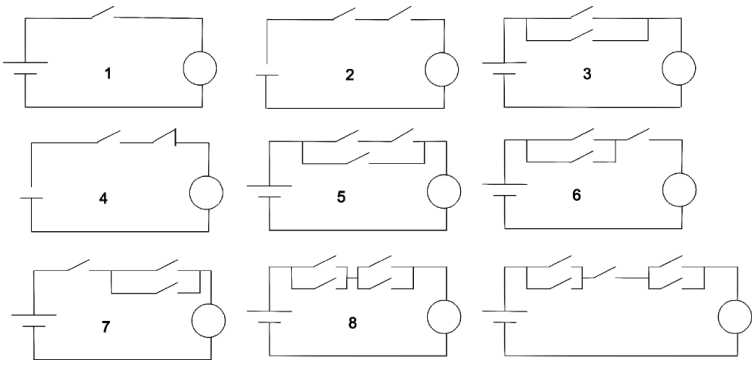
Problemas con compuertas

Objetivo Didáctico: conocer las instrucciones de programación LD, STR, AND, OR.

Planteo del Ejercicio

Con preguntas del tipo: ¿qué debería suceder para que la lámpara se encienda?, confeccionar el listado de instrucciones para que el PLC ejecute la acción equivalente al circuito eléctrico.
Confeccionar un diagrama donde se muestre la activación de las llaves y donde se muestre en que condiciones se activará la salida en función del tiempo.





Ejercicio 1

Objetivo: presentar las instrucciones LD y STR dentro de un problema particular. Mostrar que la instrucción STR sólo activa la salida mientras la entrada este presente.

Planteo

Se pretende instalar una luz indicadora para el control de tránsito interno en una planta industrial, y se desea que cumpla con los requisitos siguientes:

- Cuando sea accionado un pulsador, la lámpara deberá encenderse.
- Cuando el accionamiento sea liberado, la lámpara deberá apagarse y permanecer en esta condición.

1. Realizar el diagrama de espacio fase.
2. Confeccionar el listado de instrucciones o realizar la misma programación con lógica de escalera.

Ejercicio 2

- Presentar las instrucciones LD y STR dentro de un problema particular.
- Mostrar que la instrucción STR sólo activa la salida mientras la entrada esté presente.
- Mostrar el funcionamiento de una válvula 5/2 mando eléctrico reacción resorte y de un cilindro de doble efecto.

Planteo

Se desea implementar un desvío sobre una cinta transportadora de alimentos.
El dispositivo deberá trabajar de la siguiente manera:

- Cuando un trabajador presione un pulsador el desvío deberá activarse.
- Cuando el trabajador suelte o deje de presionar el pulsador el desvío deberá cambiar nuevamente la dirección.

Ejercicio 3

Objetivo: reconocer la necesidad de señales simultáneas para la ejecución de una acción.

Planteo

- Se ha pensado en un sistema para el control de arranque de una máquina, la cual es gobernada desde 2 puntos diferentes (1) y (2), con las siguientes condiciones:
- Si en los puntos (1) y (2) no existe accionamiento, la máquina no arrancará.
- Si en el punto (1) hay accionamiento y en el punto (2) no hay, la máquina continuará apagada.
- Si ocurre al revés también la máquina continuará apagada.
- Si los dos puntos (1) y (2) están accionados, la máquina arrancará.

1. Realizar un diagrama eléctrico del problema.
2. Realizar un programa con lógica de escalera y con listado de instrucciones para poder arrancar la máquina.

Ejercicio 4

Objetivo: reconocer la posibilidad de usar distintas señales para la ejecución de una acción (condición “O”)

Planteo

Se desea implementar una alarma luminosa en una máquina como señal de llamado al departamento de mantenimiento y se quiere accionar la misma con dos pulsadores siempre que su accionamiento cumpla con los requisitos siguientes:

- Si ninguno de los pulsadores se activa la alarma estará apagada.
- Si uno de los pulsadores se activa la alarma se encenderá.
- Si los dos pulsadores se activan simultáneamente, la alarma se encenderá.

1. Realizar un diagrama eléctrico del problema. Realizar un programa con lógica de escalera y con listado de instrucciones para solucionar el problema (LD, LD, STR)

Ejercicio 5

Objetivo: reconocer la posibilidad de usar distintas señales para la ejecución de una acción (condición “O”)

Planteo

Se deberá provocar la salida de un cilindro, para lo cual han de emplearse los interruptores 1, 2, y 3. Este movimiento se deberá ejecutar si se cumplen las condiciones siguientes:

- Si el interruptor está accionado y los dos restantes no lo están.
- Si los interruptores 2 y 3 están accionados y el interruptor 1 no lo está.

1. Realizar un diagrama eléctrico del problema. Realizar un programa con lógica de escalera y con listado de instrucciones para poder dar solución al problema planteado.

Ejercicio 6

Objetivo: conocer el uso de enclavamientos, de las instrucciones SET y RESET y manejar el diagrama de espacio fase.

Planteo

En un silo contenedor de granos se ha instalado un cilindro neumático para ejecutar la función de apertura y cierre de la compuerta de suministro. La misma deberá trabajar bajo las siguientes condiciones:

- El control de apertura de la compuerta se realizará por medio de un botón pulsador.
- El control de cierre de la compuerta se realizará con otro pulsador.

1. Realizar un diagrama espacio fase.
2. Realizar un circuito eléctrico que responda con lo pedido. Además, armar un programa con lógica de escalera y listado de instrucciones que cumpla con lo requerido.

Ejercicio 7

Objetivo: reconocer la posibilidad de usar distintas señales para la ejecución de una acción (condición “O”)

Planteo

Ha sido instalado un cilindro neumático de doble efecto para la apertura y cierre de la puerta de una cámara frigorífica. El mismo se tendrá que abrir con dos pulsadores: uno dentro y el otro fuera de la cámara frigorífica con las siguientes condiciones:

- Ya sea dentro o fuera sólo existirá un botón que hará las dos funciones: la de apertura y la de cierre.
- Para abrir la puerta, deberá estar completamente cerrada.
- Para cerrar la puerta, deberá estar completamente abierta.

Ejercicio 8

Objetivo: reconocer la posibilidad de usar señales, y señales negadas para la ejecución de una acción.

Planteo

Una gran máquina ha sido instalada en un piso de difícil acceso. Ella deberá ser encendida y apagada desde dos puntos separados. El operador deberá ser capaz de cambiar el estado en que se encuentra la máquina desde cualquiera de las dos localizaciones. Por ejemplo, si la máquina está funcionando, al cambiar de posición una de las llaves ésta se detendrá.

Ejercicio 9

Objetivo: utilizar un sensor de final de carrera para retorno del cilindro neumático.

Planteo

En una máquina herramienta (taladro) se ha implementado una unidad de avance oleo-neumática, para obtener una velocidad de trabajo lenta.
Se pretende que:

- La señal de avance será enviada por el operador por medio de un pulsador.
- Cuando el vástago alcance su final de carrera, el retorno deberá efectuarse en forma automática.

1. Realizar un diagrama espacio - fase.
2. Realizar un circuito eléctrico que responda con lo pedido. Además, armar un programa con lógica de escalera y listado de instrucciones que cumpla con lo requerido.

Ejercicio 10

Se observa que al mantener el botón de marcha presionado el cilindro cabecea. Se requiere que el movimiento del taladro sea completado aún si se mantiene el pulsador presionado.

Ejercicio 11

Realizar las siguientes secuencias con los cilindros de doble efecto, recordando que se trata de válvulas monoestables:

A+ / B+ / A- / B-
A+ / B+ A- / B-
A+ / A- / B+ / B-
A+ / B+ / B- / A-
B+ /B- /A+ /B+ /B- /A-
GRAFCET. A+ / A-
A+ luz roja / A-
A+ luz roja / A- luz amarilla
A+ luz roja / A- luz amarilla / luz verde cuando esté en reposo
A+ / B+ / A- / B-
A+ / B+ A- / B-
A+ / A- / B+ / B-
A+ / B+ / B- / A-
B+ /B- /A+ /B+ /B- /A-

Para todos los casos realizar el diagrama de espacio fase.

Ejercicio 12

Objetivo: mostrar el uso de la instrucción PUT para realizar ciclos repetitivos.

Planteo

Presionar el botón de marcha para ejecutar el ciclo A+, A- continuamente y detener el ciclo presionando el botón de parada.

Ejercicio 13

Objetivo: reconocer el uso de la instrucción de temporización y de la instrucción de flancos.

Planteo

- Encender una luz durante cinco segundos.
- Prender en el siguiente orden: la luz roja 5 segundos, la amarilla 5 segundos, la verde 5 segundos.
- Prender en el siguiente orden: la luz roja 5 segundos, la roja y la amarilla otros 5 segundos, y luego la roja, la amarilla y la verde otros 5 segundos.

1. Realizar el ejercicio con un equivalente eléctrico, con lógica de escalera y con Grafcet.

Ejercicio 14

Objetivo: reconocer el uso de las divergencias OR y AND.

Planteo

Se pretende seleccionar uno u otro cilindro. Para ello se cuenta con un pulsador de marcha, uno de parada y una llave selectora.
Con el botón de marcha y la llave, en uno activar la electroválvula que extiende el cilindro A.
Con la llave en cero y el botón de parada, activar la electroválvula que extiende el cilindro B.
Los retornos de ambos vástagos de cilindro deben producirse en forma automática una vez que se alcancen su final de carrera extendido (DIVOR RDIV, CVOR)

Ejercicio 15

Objetivo: reconocer otros usos de la divergencia en OR.

Planteo

Se pretende que un cilindro A ejecute un ciclo automático y continuo de salida y entrada de vástago, desde que se pulsa el botón de marcha hasta que se presione el de parada.

Ejercicio 16

Objetivo: uso de la divergencia en OR.

Planteo

Un móvil se desliza a través de un husillo movido por un motor de doble sentido de giro, para lo cual llevará un contactor a0 que lo conecta para que gire a derecha y otro a1 para que gire a la izquierda. El móvil debe realizar un movimiento de vaivén continuado, a partir del momento en que el sistema recibe la orden de impulso de puesta en marcha.
Un pulso de parada deberá detener el motor, pero no en el acto, sino al final del movimiento de vaivén ya iniciado.
Un pulso de emergencia debe producir el retroceso inmediato del móvil a la posición de origen, y el sistema no podrá ponerse en marcha de nuevo con PM si previamente no se ha accionado el pulsador de rearme.

Ejercicio 17

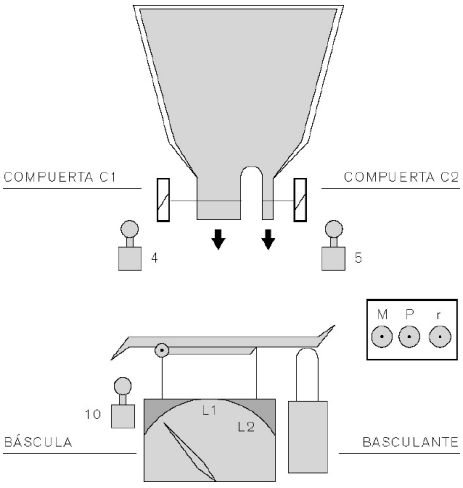
Objetivo: reconocer el uso de la divergencia en AND.

Planteo

Una pulsación de PM (pulsador de marcha) debe provocar la apertura de las dos compuertas. Cuando la aguja de la pesadora llegue a L1 debe desactivarse C1, cerrando la compuerta correspondiente. Cuando la aguja llegue a L2 deberá desactivarse D1, cerrándose la compuerta de afinado. Vaciado el contenido de la pesadora por medio de un basculante, ésta volverá a la posición de reposo sin que el paso de la aguja por delante de L1 provoque efecto alguno. Pulsando de nuevo PM se inicia un nuevo ciclo.

Al accionar el pulsador de emergencia se deberán cerrar las dos compuertas en cualquier momento del ciclo y éste se parará. Para reanudarlo bastará con pulsar el rearme. El ciclo deberá continuar en la fase en que se interrumpió. Si durante el ciclo se pulsase P, no deberá alterarse aquél.

En la figura se ilustra el proceso a automatizar.
Las compuestas son accionadas por cilindros de doble efecto, mientras que el basculante es accionado por un cilindro de simple efecto.



Ejercicio 18

Objetivo: aprender el uso de los contadores, comparador. Descubrir que es posible activar una salida o variable interna a través del comparador con las instrucciones STR, TMR, DIF y PUT.

Planteo

Presionar tres veces el pulsador de marcha para encender la luz roja durante 5 segundos. Volver el contador a cero con el botón de parada.
Lo mismo que en el ejercicio anterior, pero evitar que si se sigue presionado el botón de marcha el contador continúe el conteo. Bloquear el contador cuando éste llega al valor deseado.

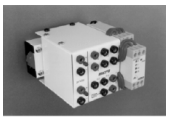
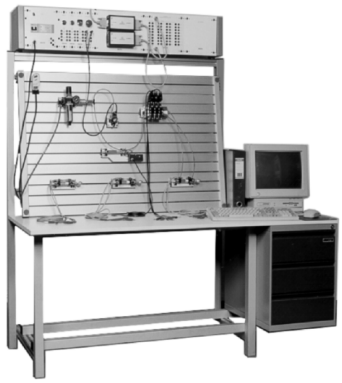
Material didáctico

La Academia realiza y comercializa una variedad de elementos didácticos de gran flexibilidad, fácil montaje y re-ubicación o cambio, con posibilidades de expan-sión con módulos que permiten partir de un modelo básico, y terminar en un podero-so centro de estudio y ensayo.

Paneles serie DIDACTO
Estos paneles están enteramente diseñados por MICRO en un desarrollo compartido por nuestros especialistas de Capacitación y de Ingeniería. Los componentes que se utilizan para su construcción son los mismos que adopta la industria de todo el mundo para la implementación de sus automatismos en una amplia gama de aplicaciones y complejidades.




Se entregan con una base de montaje en estructuras de perfiles de aluminio anodizados, y un exclusivo sistema de fijación de elementos de ajuste manual de un cuarto de vuelta que permita su fácil re-ubicación o cambio, facilitando la tarea didáctica del capacitador y la asimilación de conceptos de los asistentes.

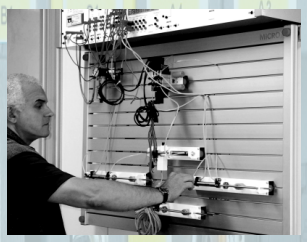




En cuanto a las posibilidades de expansión, se han contemplado diferentes módulos que permiten migrar de un modelo básico y llegar a implementar hasta un poderoso Centro de Estudio y Ensayo que incluya PC, interfaces para accionamiento de actuadores, mobiliario, etc., cubriendo variadas tecnologías complementarias.

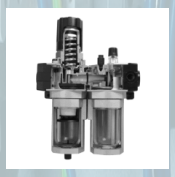
Software

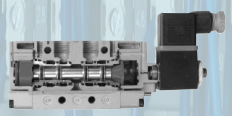
Los softwares utilizados tienen como misión amalgamar la potencialidad de la informática aplicada a la enseñanza de automatización. Puede clasificarse en:

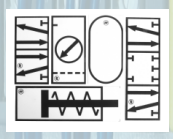
1. Softwares de simulación, que pueden diseñar, ensayar y simular circuitos que incluyan componentes electrónicos, neumáticos e hidráulicos.
2. Softwares de cálculo, información técnica y selección de componentes adecuados para cada requisición técnica.
3. Softwares de presentaciones que, preparados por nuestros ingenieros, optimizan las charlas y las adecuan al medio al que van dirigidas.



Los referencia a los softwares de simulación, y con el fin de hacerlos interactivos, se dispone de interfaces que permiten físicamente hacer actuar a los elementos que son visualizados en el monitor de la computadora.







Material de soporte

LA ACADEMIA dispone de variados elementos didácticos para facilitar la trans-misión efectiva de los conceptos. Entre ellos se cuenta con componentes en corte,simbología para pizarra magnética, manuales, videos, transparencias, etc.

TECNOLOGÍA

América Educa Cursos Online

Cursos

LA ACADEMIA cubre un extenso rango de temarios en los cursos que dicta en sus aulas que, para tal efecto, posee en su edificio central. Pero también atiende los requerimientos de la Industria y las instituciones educativas trasladándose con su laboratorio móvil a las ciudades del interior, y otros países.

Uruguay Estudia

La Academia

TUTORES EN RED