Manual usuario
Simatic S7-200 CPU 224

Índice

PRESENTACIÓN

El presenta documento es un compendio de distintos manuales de usuario que versan sobre los automatismos ***Siemens***. Por tanto, todo lo se presenta en este "mini-manual" ha sido elaborado por los ingenieros de Siemens.

El objeto de este archivo, no es otro que resumir algunos conceptos, órdenes de programación, etc.. que se han considerado más relevantes como introducción al mundo de la automatización, debido en gran manera a las amplias posibilidades que presenta los automatismos.

Los manuales de los que se parte, y en los que encontrarás información más detallada, son:

- **Manual del sistema de automatización S7-200**. Presente en la documentación técnica facilitada por el fabricante al adquirir el autómata.
- **Curso nivel básico Simatic S5**.
- **Programación de sistemas de mando con STEP5**.
- **Simatic S5. Aparato de automatización programable en memoria. S5-110A**.
- **El S7-200 en una hora**.
- **El S7-200 en dos horas**.

Estos dos últimos están disponibles en Internet.

CONCEPTO Y DESCRIPCIÓN DE AUTOMATISMO

1.- Necesidades y usos del PLC

Comencemos definiendo un *proceso industrial* como una operación o secuencia de operaciones en las que las variables a controlar (temperaturas, desplazamientos, tiempos, etc...) están debidamente definidas.

La gran mayoría de los procesos industriales requieren algún tipo de control. La necesaria automatización de estas funciones de control puede ser llevada a cabo de muy diferentes formas: a base de cuadros de relés, contactores, etc...

Lamentablemente, cualquier modificación en este tipo de sistemas de control suponía gran esfuerzo técnico y económico, y más todavía si estos cambios eran frecuentes. Además debemos tener en cuenta que la mayoría de estos elementos son dispositivos mecánicos y poseen una vida limitada que requiere una estricta manutención. Por otra parte, estos sistemas suponen un conexionado complejo cuando existen gran cantidad de elementos, lo que implica un enorme esfuerzo de diseño, mantenimiento...

Con el objetivo de solucionar, o al menos reducir, estos inconvenientes se elaboraron los autómatas, que permiten cambiar la funcionalidad del control del proceso industrial sin más que cambiar el programa, ya que gran parte de los componentes necesarios como relés auxiliares, temporizadores, etc... se encuentran implementados en la programación interna de él. Además, en los casos en que las modificaciones superen la capacidad del sistema, es posible agregar módulos de ampliación que permitan cumplir con las nuevas exigencias.

Este automatismo fácilmente programable para tareas de control, y concebido para ser utilizado en ambientes industriales, es lo que se conoce como **PLC**, acrónimo de *Programmable Logic Controller*, es decir, *Controlador Lógico Programable*. A él se conectan los captadores (finales de carrera, pulsadores, etc...) por una parte, y los actuadores (bobinas de contactores, lámparas, pequeños receptores, etc...) por otra.

Los autómatas programables no sólo tienen aplicación industrial, si no que también se emplean para automatizar procesos en el hogar (puerta de un garaje, luces de la casa, etc...), entre otros.

Entre las características de los PLC's destacan:

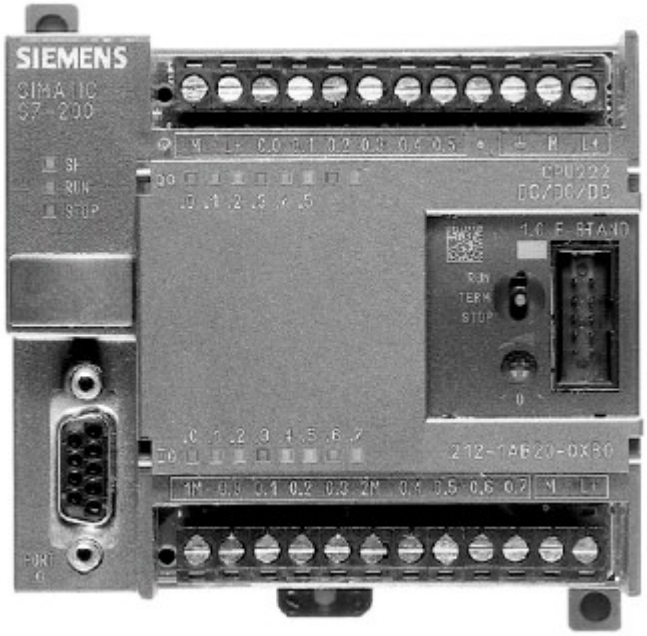
- Fácilmente programables por la mayoría de los técnicos.
- Facilidad en la modificación de programas.
- Comunicación con otros PLC's, pudiendo enviar y recibir señales.
- Tiempo de vida largo.
- Pueden trabajar sin problemas en todo tipo de ambientes industriales.

Actualmente los PC's están comenzando a reemplazar al PLC en algunas aplicaciones. Por lo cual, no sería de extrañar que en un futuro no muy lejano el PLC desapareciera frente al cada vez más potente PC, debido a las posibilidades que los ordenadores pueden proporcionar.

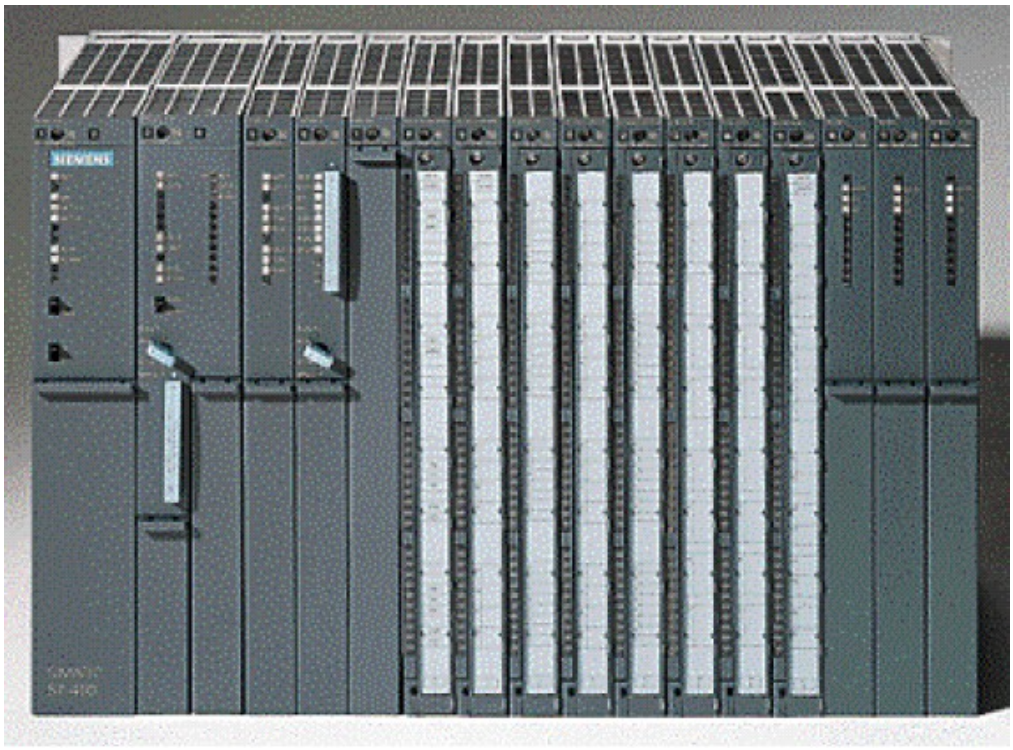
2.- Estructura externa

Existen dos estructuras básicas para los autómatas programables:

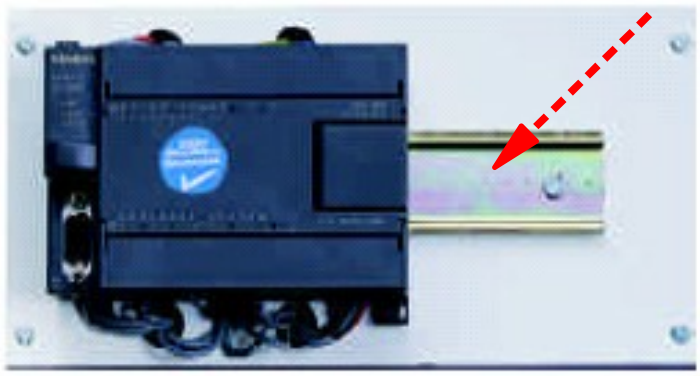
- **Compacta**: consiste en una única pieza en la que se integran todos los elementos.



- **Modular**: en los que la CPU, la fuente de alimentación, las entradas, las salidas, etc..., son cada una un módulo que se elige en función de la aplicación requerida.



Para el caso de una estructura modular se dispone de la posibilidad de fijar los distintos módulos en raíles normalizados, para que el conjunto sea compacto y resistente.



3.- Arquitectura

Los elementos esenciales, que todo autómata programable posee como mínimo, son:

- **Sección de entradas**: se trata de líneas de entrada, las cuales pueden ser digitales o analógicas.
  A estas líneas conectaremos los sensores (captadores).
- **Sección de salidas**: son una serie de líneas de salida, que también pueden ser de carácter digital o analógico.
  A estas líneas conectaremos los actuadores.
- **Unidad central de proceso** (**CPU**): se encarga de procesar el programa que el usuario ha introducido.
  La CPU toma, una a una, las instrucciones programadas por el usuario y las va ejecutando, cuando llega al final de la secuencia de instrucciones programadas, la CPU vuelve al principio y sigue ejecutándolas de manera cíclica.
  Para ello, dispone de diversas zonas de memoria, registros, e instrucciones de programa. Adicionalmente, en determinados modelos, podemos disponer de funciones ya integradas en la CPU; como reguladores PID, control de posición, etc...

A parte de éstos podemos disponer de los siguientes elementos:

- **Unidad de alimentación** (algunas CPU's la llevan incluida).
- **Consola de programación**: que nos permitirá introducir, modificar y supervisar el programa de usuario. Tiende a desaparecer, debido a que la mayoría se programan a partir del PC mediante programas específicos facilitados por cada fabricante; o programados directamente desde el propio autómata.
- **Dispositivos periféricos**: como nuevas unidades de E/S, más memoria, unidades de comunicación en red, etc...
- **Interfaces**: facilitan la comunicación del autómata con otros dispositivos (como un PC), autómatas, etc...

3.1.- CPU

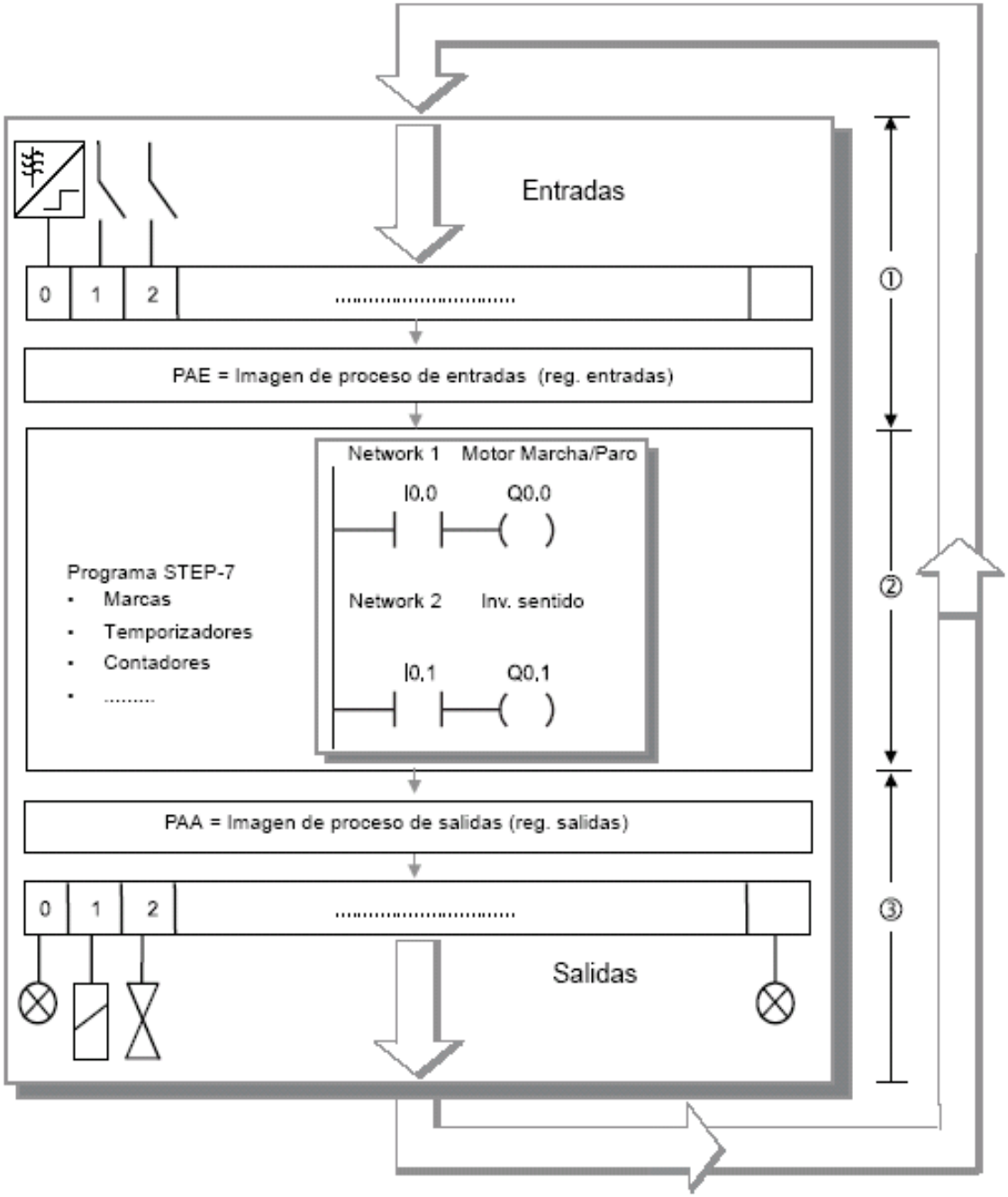
Es el corazón del autómata programable. Sus funciones son:

- Ejecutar el programa de usuario.
- Vigilar que el tiempo de ejecución del programa de usuario no excede un determinado tiempo máximo (tiempo de ciclo máximo). A esta función se le suele denominar *Watchdog* (perro guardián).
- Crear una imagen de las entradas, ya que el programa de usuario no accede directamente a

dichas entradas.

- Renovar el estado de las salidas, en función de la imagen de las mismas, obtenida al final del ciclo de ejecución del programa de usuario.
- Chequear del sistema.

Para ello el autómata va a poseer un ciclo de trabajo, que ejecutará de forma continua:



Durante el funcionamiento cíclico, primero se leen los estados en las entradas, memorizándose en la imagen de proceso de las entradas (*PAE*). Con estas informaciones trabaja luego el programa de control cuando se ejecuta.

De acuerdo a la lógica definida en el programa se modifica el estado de las salidas depositadas en la imagen de proceso de las salidas (*PAA*). En la última etapa del ciclo, los estados memorizados en

la PAA se transfieren a las salidas físicas. Seguidamente comienza de nuevo el ciclo.

Un ciclo dura normalmente entre 3 y 10 ms. La duración depende del número y tipo de instrucciones (operaciones) utilizadas. El ciclo consta de dos partes principales:

1. Tiempo del sistema operativo, normalmente 1 ms; corresponde con las fases 1 y 3.
2. Tiempo para ejecutar las instrucciones; corresponde con la fase 2.

Por otro lado, el ciclo sólo se ejecuta cuando el PLC se encuentra en estado *RUN*.

Supongamos, por ejemplo, el sistema de calefacción de una vivienda: la variable a considerar sería la temperatura, el actuador podría ser un calefactor y el sensor correspondiente un termostato. Según la estructura mostrada en el esquema, el comportamiento del sistema sería el siguiente: el PLC leería permanentemente la entrada correspondiente al sensor se temperatura, cuando la temperatura fuera menor a la programada, conectaría el calefactor y lo desconectaría cuando fuera mayor o igual a la deseada, etc...

La sencillez del ejemplo anterior, apenas permite apreciar las ventajas que la incorporación de un PLC al control de un proceso industrial pueda brindar, sin embargo si consideramos que el mismo PLC puede controlar simultáneamente varios procesos, además coordinarlos con otros, visualizar los distintos estados, alarmas, etc... y que además presenta la posibilidad de reprogramación para poder adaptarse a posibles cambios en el diseño con facilidad, se comprende la importancia que tienen hoy en día los PLC en la automatización industrial

3.2.- Memoria

Dentro de la CPU dispondremos de un área de memoria, la cual emplearemos para diversas funciones:

- **Memoria del programa de usuario**: aquí introduciremos el programa que el autómata va a ejecutar cíclicamente.
- **Memoria de la tabla de datos**: se suele subdividir en zonas según el tipo de datos (como marcas de memoria, temporizadores, contadores, etc...).
- **Memoria del sistema**: aquí se encuentra el programa en código máquina que monitoriza el sistema (programa del sistema o firmware). Este programa es ejecutado directamente por el microprocesador/microcontrolador que posea el autómata.
- **Memoria de almacenamiento**: se trata de memoria externa que empleamos para almacenar el programa de usuario, y en ciertos casos parte de la memoria de la tabla de datos. Suele ser de uno de los siguientes tipos: EPROM, EEPROM, o FLASH.

Cada autómata hace subdivisiones específicas según el modelo y fabricante.

3.3.- Unidades de entrada y salida

Podemos disponer de dos tipos de módulos de entrada y/o salida:

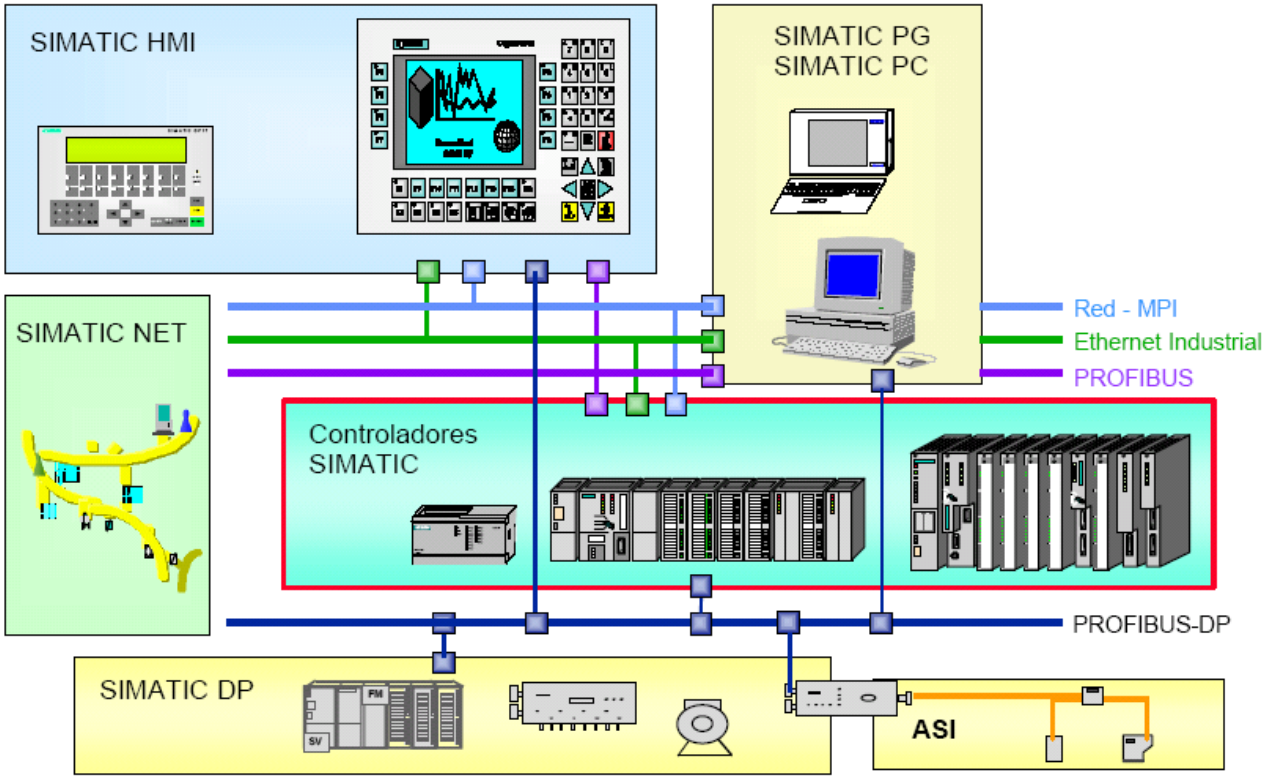
- **Digitales**. Se basan en el principio de todo o nada, es decir o no conducen señal alguna o poseen un nivel mínimo de tensión. Estas E/S se manejan a nivel de bit dentro del programa de usuario.
- **Analógicas**. Pueden poseer cualquier valor dentro de un rango determinado especificado por el fabricante. Estas señales se manejan a nivel de byte o palabra (8/16 bits) dentro del programa de usuario.

Las E/S son leídas y escritas dependiendo del modelo y del fabricante, es decir, pueden estar incluidas sus imágenes dentro del área de memoria o ser manejadas a través de instrucciones específicas de E/S.

3.4.- Interfaces

Todo autómata, salvo casos excepcionales, posee la virtud de poder comunicarse con otros dispositivos (como un PC).

Lo normal es que posea una E/S serie del tipo RS-232 (puerto serie). A través de esta línea se pueden manejar todas las características internas del autómata, incluida la programación del mismo, y suele emplearse para monitorizar el proceso.



3.5.- Unidades de programación

La programación del autómata puede realizarse, generalmente, empleando alguno de los siguientes elementos:

- **Consola de programación**: suele tener la forma de calculadora.
- **PC**: es el modo más empleado en la actualidad. Permite programar desde un ordenador personal estándar, con todo lo que ello supone: herramientas más potentes, posibilidad de almacenamiento, impresión, transferencia de datos, monitorización mediante software SCADA, etc...

Cada autómata, dependiendo del modelo y fabricante, posee una conexión a uno o a varios de los elementos anteriores.

3.6.- Periféricos

El autómata programable, en la mayoría de los casos, puede ser ampliado. Las ampliaciones abarcan un gran abanico de posibilidades: módulos auxiliares de E/S (analógicas, digitales, etc...), memoria adicional, conexión con otros autómatas, etc...

Cada fabricante facilita las posibilidades de ampliación de sus modelos, los cuales pueden variar incluso entre modelos de la misma serie.

4.- Lenguajes de programación

Los primeros autómatas programables surgieron debido a la necesidad de sustituir los enormes cuadros de maniobra construidos con contactores y relés. Por lo tanto, la comunicación hombre-máquina debía ser similar a la utilizada hasta ese momento. El lenguaje utilizado, debería ser

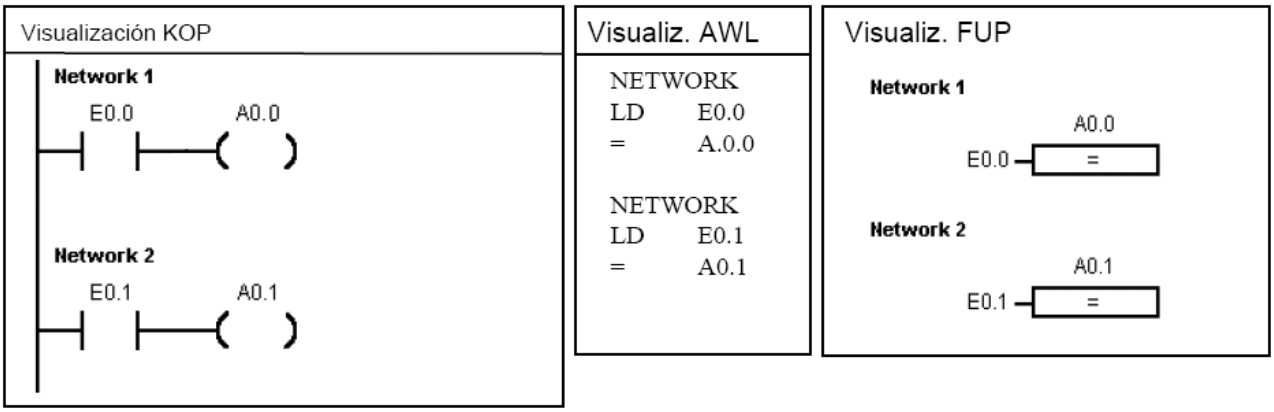
interpretado, con facilidad, por los mismos técnicos electricistas que anteriormente estaban en contacto con la instalación.

Con el tiempo estos lenguajes evolucionaron de tal forma que algunos de ellos ya no tenían nada que ver con el típico plano eléctrico a relés, además de haber evolucionado siguiendo caminos distintos. Todo esto unido al incremento en la complejidad de los procesos a automatizar, no hizo más que complicar el uso de aquello que se creo con una finalidad bien distinta.

Con el fin de subsanar este problema la dirección del IEC (estándar internacional) ha elaborado el estándar IEC 1131-3 para la programación de PLC's, con la idea de desarrollar el estándar adecuado para un gran abanico de aplicaciones.

Los lenguajes gráficos y textuales definidos en el estándar son una fuerte base para entornos de programación potente en PLC's. Los lenguajes más significativos son:

- **Lenguaje de contactos** (**KOP**): es el que más similitudes tiene con el utilizado por un electricista al elaborar cuadros de automatismos.
- **Lenguaje por lista de instrucciones** (**AWL**): consiste en elaborar una lista de instrucciones.
- **Plano de funciones lógicas** (**FUP**): resulta especialmente cómodo de utilizar cuando estamos habituados a trabajar con circuitos de puertas lógicas, ya que la simbología usada en ambos es equivalente.
- **GRAFCET**: es el llamado Gráfico de Orden Etapa-Transición. Ha sido especialmente diseñado para resolver problemas de automatismos secuenciales. Las acciones son asociadas a las etapas y las condiciones a cumplir a las transiciones. Este lenguaje resulta enormemente sencillo de interpretar por operarios sin conocimientos de automatismos eléctricos.



INTRODUCCIÓN

1.- Concepto de automatismo

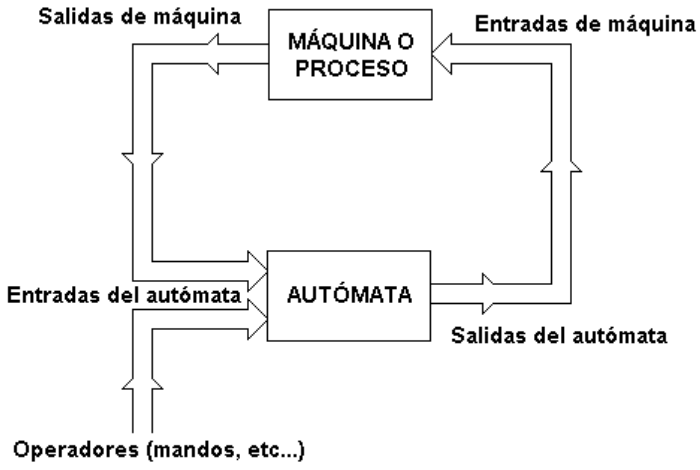
Como se ha dicho anteriormente, un automatismo es un dispositivo que permite a las máquinas o procesos evolucionar con la mínima intervención del hombre y que puede:

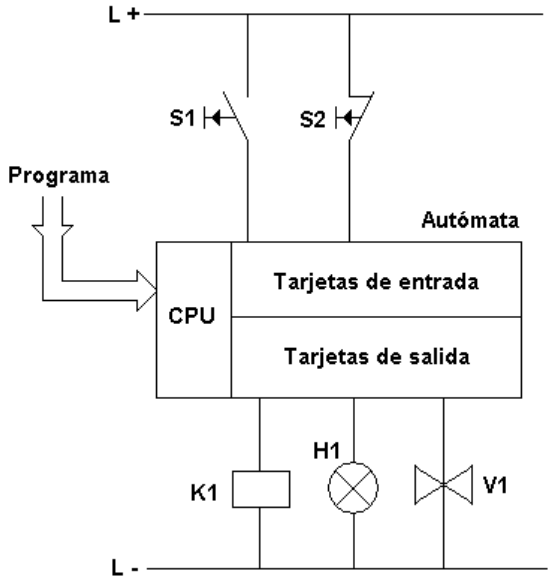
- Encargarse de las tareas repetitivas, peligrosas o trabajosas.
- Controlar la seguridad del personal y de las instalaciones.
- Incrementar la producción y la productividad y economizar materia y energía.
- Incrementar la flexibilidad de las instalaciones para modificar los productos o los ritmos de fabricación.

Un automatismo industrial se concibe generalmente para mandar una máquina o un grupo de máquinas. A estas máquinas se le llama **parte operativa** del proceso, mientras que al conjunto de los componentes del automatismo que suministran las informaciones que sirven para pilotar esta parte operativa se llama **parte de mando**. Es la conjunción de ambas partes lo que constituye el automatismo completo.

Entre el autómata y la máquina se canjean informaciones que frecuentemente son variables binarias (estado de un interruptor...), aunque pueden intervenir igualmente informaciones analógicas (medida de una temperatura.), que serán en ese caso convertidas en un conjunto de señales binarias interpretables por el autómata.

Todo proceso recibe informaciones que se llaman **entradas**, y suministra informaciones que se llaman **salidas**. Si consideramos una máquina cualquiera, ella recibe órdenes del autómata. Estas órdenes, que constituyen las salidas del autómata, son las entradas de la máquina, la cual ejecuta acciones y devuelve informaciones al autómata en función del resultado de sus actuaciones. Estas informaciones que constituyen las salidas de la máquina forman parte de las entradas del autómata, que se complementan con el conjunto de instrucciones transmitidas por el operador al autómata.





En lo sucesivo llamaremos entrada a una entrada del autómata y salida a una salida del autómata. La distinción entre variables de entrada y variables de salida, será de esencial importancia a la hora de analizar un proceso y debe realizarse siempre con mucho cuidado.

A nivel de entradas, conviene señalar, que las informaciones necesarias para que el autómata

ejecute sus instrucciones, las suministran los captadores, sensores, etc... Entre las cualidades que debemos exigir a estos dispositivos podemos citar: tiempo de respuesta, precisión, sensibilidad, inmunidad a perturbaciones, robustez...

En lo referente a salidas, las informaciones suministradas por el autómata a la máquina (o procesos) corresponden a los instantes en los que una acción debe empezar. Por tanto, nos interesa elaborar un sistema que elabore informaciones que cambien de valor en los instantes deseados (ni antes, ni después).

Deberemos prestar atención a la potencia requerida por los diversos dispositivos, pues a menudo el autómata no es capaz de suministrarla, por lo que es necesario recurrir a periféricos que realicen esta labor.

2.- Variables de estado

Partamos de un ejemplo cotidiano: el mando de una ascensor.

Supongamos que haya una llamada desde el tercer piso. Si la cabina se encuentra en le quinto, debe descender, si se encuentra en la planta baja, debe subir; si está desplazándose entre dos plantas debe continuar su movimiento, pero el automatismo debe registrar la llamada procedente de la tercera planta. Concluyendo, podemos decir que: la orden a aplicar a la cabina depende de la situación, del **estado**, en el que se encuentra el ascensor en el momento de la llamada.

Del ejemplo se extrae que será muy importante conocer en cada instante el estado de un automatismo para conocer su respuesta cuando un mando actúe sobre él, es decir, cuando una variable de entrada cambie de valor.

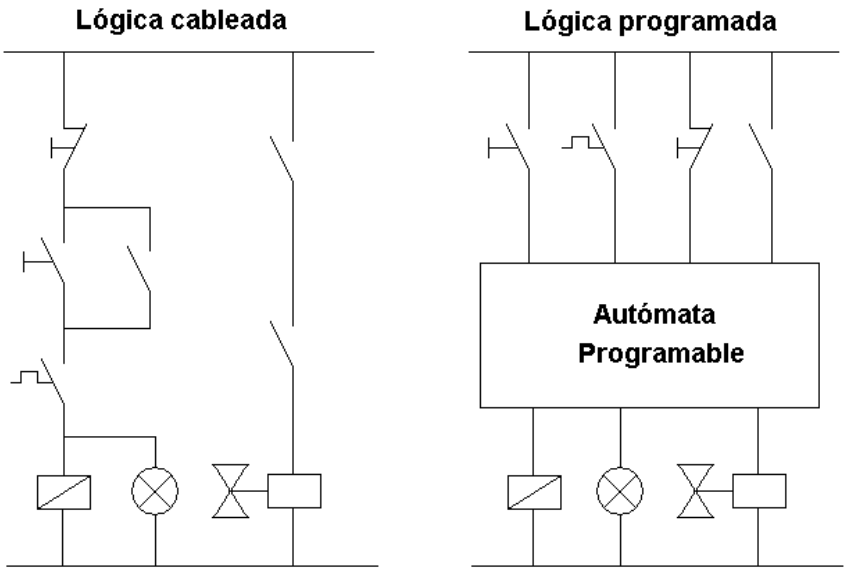
Por tanto, para caracterizar el estado de un automatismo, en ocasiones, no basta con conocer solamente el valor de las variables de entrada, pues como bien ilustra el ejemplo del ascensor: no es el hecho de que haya una llamada en la tercera planta el único determinante del movimiento de la cabina... Necesitamos, además, conocer el estado de un conjunto de variables (**variables de estado**) que nos permitan "prever" cual será la evolución del automatismo en función de los cambios ocurridos en las variables de entrada.

En la elección del conjunto de variables que permitan describir el comportamiento deseado, reside la complejidad de la programación, y no suele ser única.

3.- Cableado vs. programa

Vamos a diferenciar entre lógica cableada y lógica programada. Mientras un mando con relés o contactores representa la lógica cableada; un autómata programable representa la lógica programada.

- **Lógica cableada**: el programa de mando queda determinado a través de la unión entre los diferentes elementos, tales como bobinas de accionamiento, contactos de interruptores, etc... La modificación del programa supone una transformación del cableado.
- **Lógica programada**: el programa de mando y el cableado son independientes. Los contactos de los captadores y las bobinas de accionamiento se conectan a las entradas-salidas del autómata. El programa de mando, se escribe en la memoria del autómata, quedando fijada la secuencia en que deben ser consultados los contactos, la forma en que deben realizarse las combinaciones (AND u OR) y la asignación de los resultados a las salidas, es decir, el accionamiento de las bobinas.
  En el caso de ser necesario realizar una variación del programa, no hay que modificar el cableado del autómata, sino solamente el contenido del programa.



4.- Señal binaria, estado de señal

El autómata consulta el valor de sus las entradas según dos estados:

- Existe tensión.
- No existe tensión.

A partir de estos datos y según el programa:

- Activa o...
- desactiva...

...los "aparatos" conectados a sus salidas.

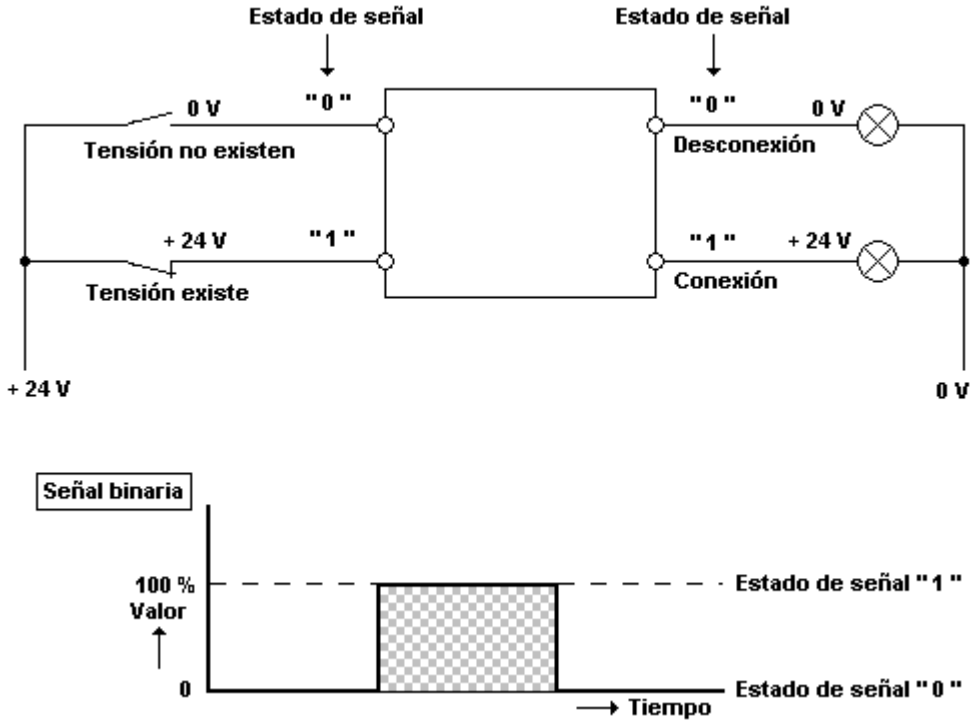
En ambos casos nos encontramos con un clara y diferenciada situación de los estados, conocida como:

- **Estado de señal "0" → No existe tensión → Desactivado**.
- **Estado de señal "1" → Existe tensión → Activado**.

Estos dos estados de señal son los dos valores diferentes que puede tomar una **señal binaria*** (señal de valor doble).

Veamos esto con un ejemplo muy sencillo: imaginemos un interruptor de luz, éste sólo tiene el efecto "luz encendida" ó "luz apagada. Es decir el valor del interruptor que responde a la cuestión "¿Está la luz encendida?" o está activada (luz encendida) o está desactivada (luz apagada). Dicho con otras palabras, el interruptor de luz tiene un ancho de información de 1 **bit** (señal binaria). En este caso no se considera el estado "Luz apagada, pero bombilla fundida".

* El concepto inglés Bit (**Bi**nari Digi**t**) es aceptado como la unidad técnica de información más pequeña que existe.



4.1.- Contactos abiertos y cerrados

Con anterioridad se dijo que el autómata consultaba el valor de sus entradas, es decir, si existe tensión ("1") o no existe tensión ("0"). Sin tener en cuenta si el contacto asociado a la entrada era cerrado o abierto...

Sin embargo, para la elaboración del programa si que deberíamos conocer las funciones técnicas del "contacto":

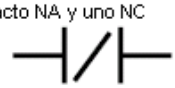
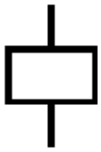
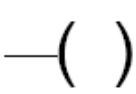
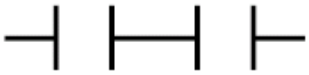
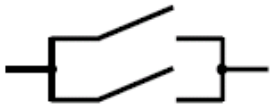
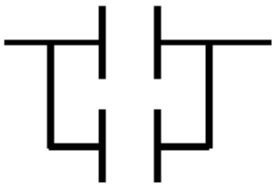
- Si en una entrada hay conectado un contacto abierto, se aplicará el estado de señal "1" en la entrada cuando se accione el contacto.
- Por el contrario, si a la entrada nos encontramos con un contacto cerrado, se aplicará el estado de señal "0" en la entrada cuando se accione el contacto.

El autómata no tiene posibilidad de determinar si en una entrada hay conectado un contacto cerrado o abierto. Solo puede consultar o reconocer los estados de señal "1" ó "0".

Nos es indiferente si un estado se ha alcanzado a través de un contacto abierto o cerrado. Lo único importante es la elección del tipo de contactos, sobretodo teniendo en cuenta las normativas de seguridad...

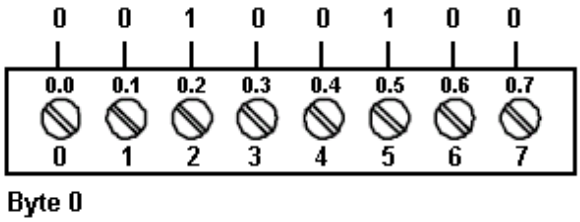
Tipo contacto	Estados	¿Existe tensión?	Estado de señal
 Abierto	Accionado	SI	1
	No accionado 	NO	0
 Cerrado	Accionado 	NO	0
	No accionado 	SI	1

Por tanto, cualquier combinación de contactos tiene su equivalente lógica, es decir, tiene como resultado un “0 (corte de corriente)” ó un “1 (paso de corriente)”. En la siguiente tabla se muestra la correspondencia de símbolos eléctricos (o combinaciones de ellos) con la simbología KOP reconocida por el autómata:

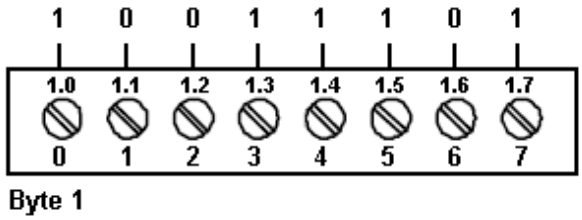
	Consulta: ¿Circula corriente? Si sí, entonces el resultado de esta pregunta es verdadero. (Consulta: ¿"1"?)	
	Consulta: ¿No circula corriente? Si sí (no hay corriente), entonces el resultado de esta pregunta es ver-dadero. (Consulta: ¿"0"?)	P. ej. finales de carrera, conmutadores, en los que existe un contacto NA y uno NC 
	Bobina: Si la bobina se alimenta con un valor "verdadero" (corriente) entonces se activa (La bobina se excita).	 Las salidas pueden ser "entradas"
	Conexión en serie: (Combinación Y). Para que circule la corriente deberán estar cerrados el primer Y el segundo interruptor.	
	Conexión en paralelo (Combinación O). Para que circule la corriente deberá estar cerrado el primer interruptor O el segundo.	

4.1.1.- Conceptos de bit, byte y palabra

- **Bit**. Unidad del símbolo binario, solamente puede tomar los valores "0" y "1".
  En ocasiones, el bit es insuficiente para definir determinados aspectos de una automatización. Debiendo recurrir a conjuntos formados por varios símbolos binarios (byte).
- **Byte**. Conjunto de 8 símbolos binarios, es decir, el byte tiene una longitud de 8 bits, cada uno de lo cuales puede tomar cualquier valor entre 0 y 1.



Byte 0

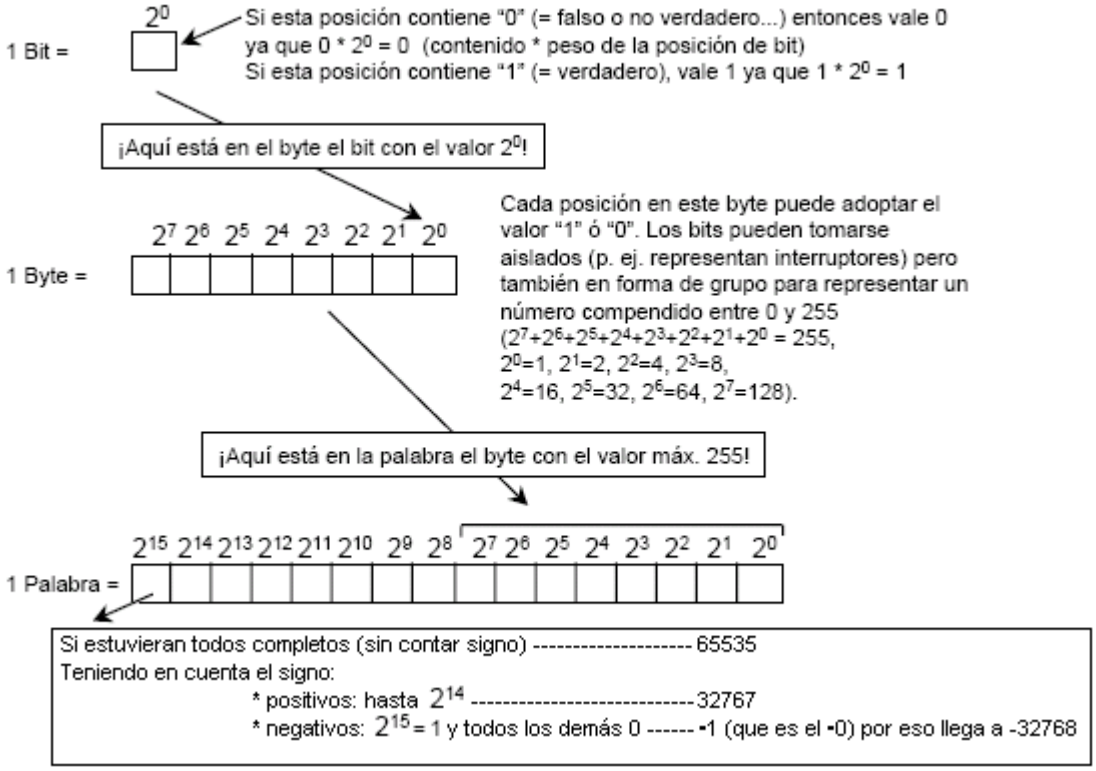


Byte 1

- **Palabra**. En un PLC los bits se asocian en grupos.
  Con se ha dicho, 8 bits se denominan byte. Y cada bit en dicho grupo está exactamente definido por una posición propia que tiene una dirección específica.
  Un byte tiene una dirección de byte y direcciones de bit 0...7.

Un grupo de 2 bytes se denomina palabra.

Este sistema de numeración se denomina binario y tiene como base 2.



En un PLC una palabra permite representar valores numéricos de -32768 a +32767.

Se ha convenido que el bit con el peso 2^{15} señaliza números negativos (si aparece un "1" en la posición 2^{15}, el número en cuestión es negativo).

4.2.- Direccionamiento de entradas y salidas

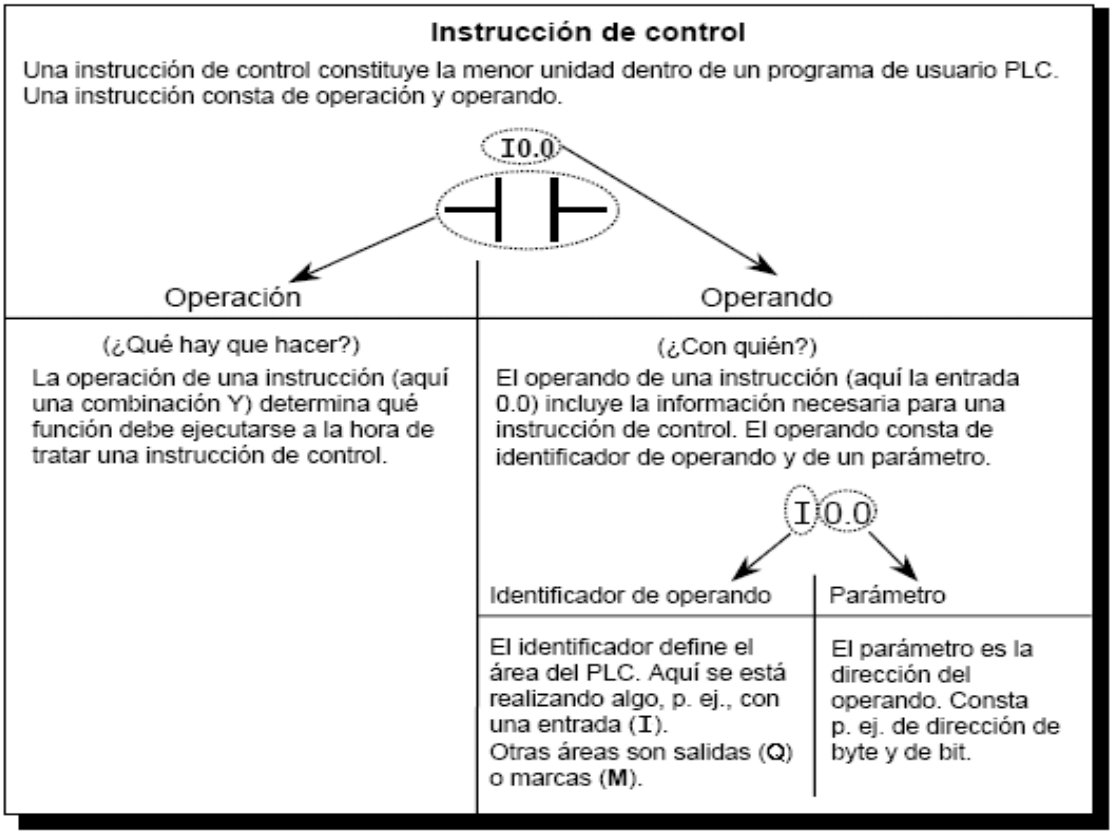
Una vez entendida la diferencia entre "0" y "1" (concepto de bit) y la "estructura del byte", debemos conocer como el autómata denomina a cada una de sus entradas y salidas.

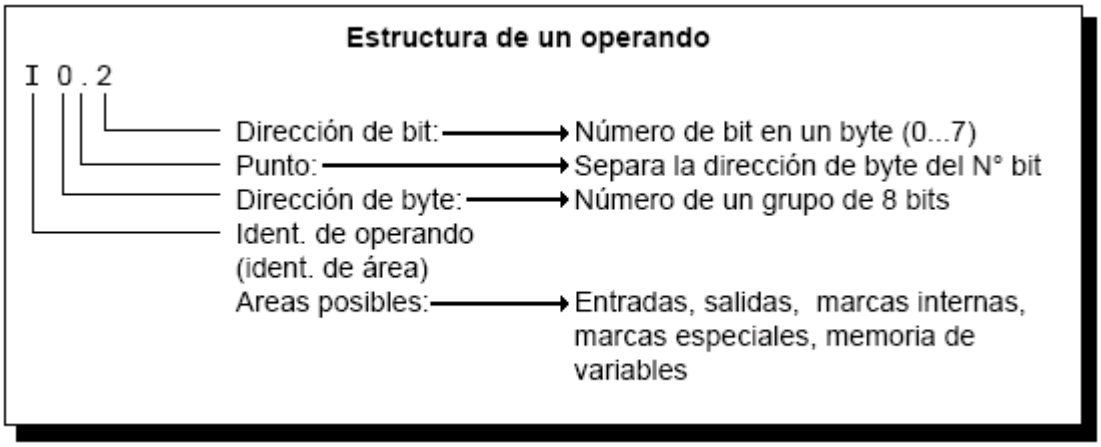
En primer lugar el autómata utiliza un operando distintivo:

- **I** para denominar entradas (algunos lenguajes utilizan la *E*).
- **Q** para denominar salidas (algunos lenguajes utilizan la *A*).

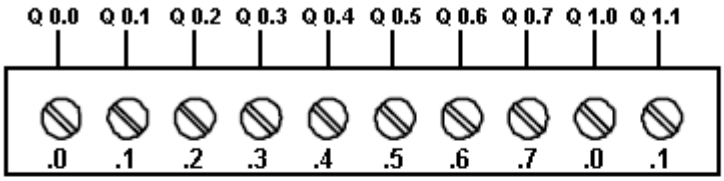
Junto con el distintivo de entrada o salida aparece el parámetro 0.4, 1.2 ó 4.7. El parámetro consiste en una combinación:

- **0.**, **1.** ó **4.** → byte.
- **.4**, **.2** ó **.7** → bit.



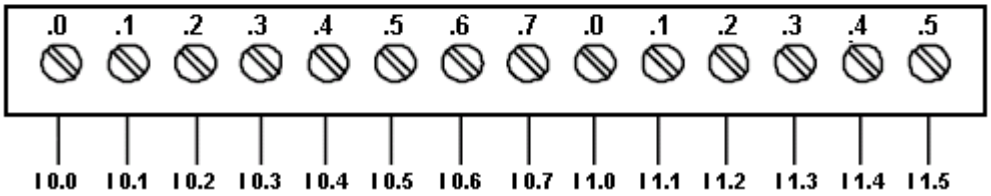


En el caso del autómata objeto de estudio, que presenta 14 entradas y 10 salidas:



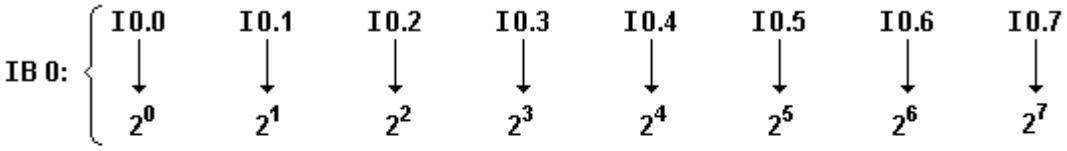
Salidas " Q "

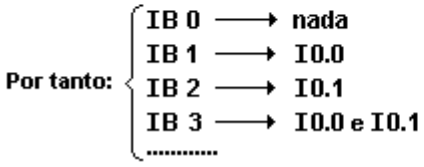
Entradas "I"



4.2.1.- Direccionamiento de bytes

El direccionamiento de bytes es similar al de bits, pero en este caso solo se utiliza el identificador de parámetro, seguido de la letra B (byte) más la dirección de byte. De este modo podemos acceder a distintos bits con una sola "llamada":

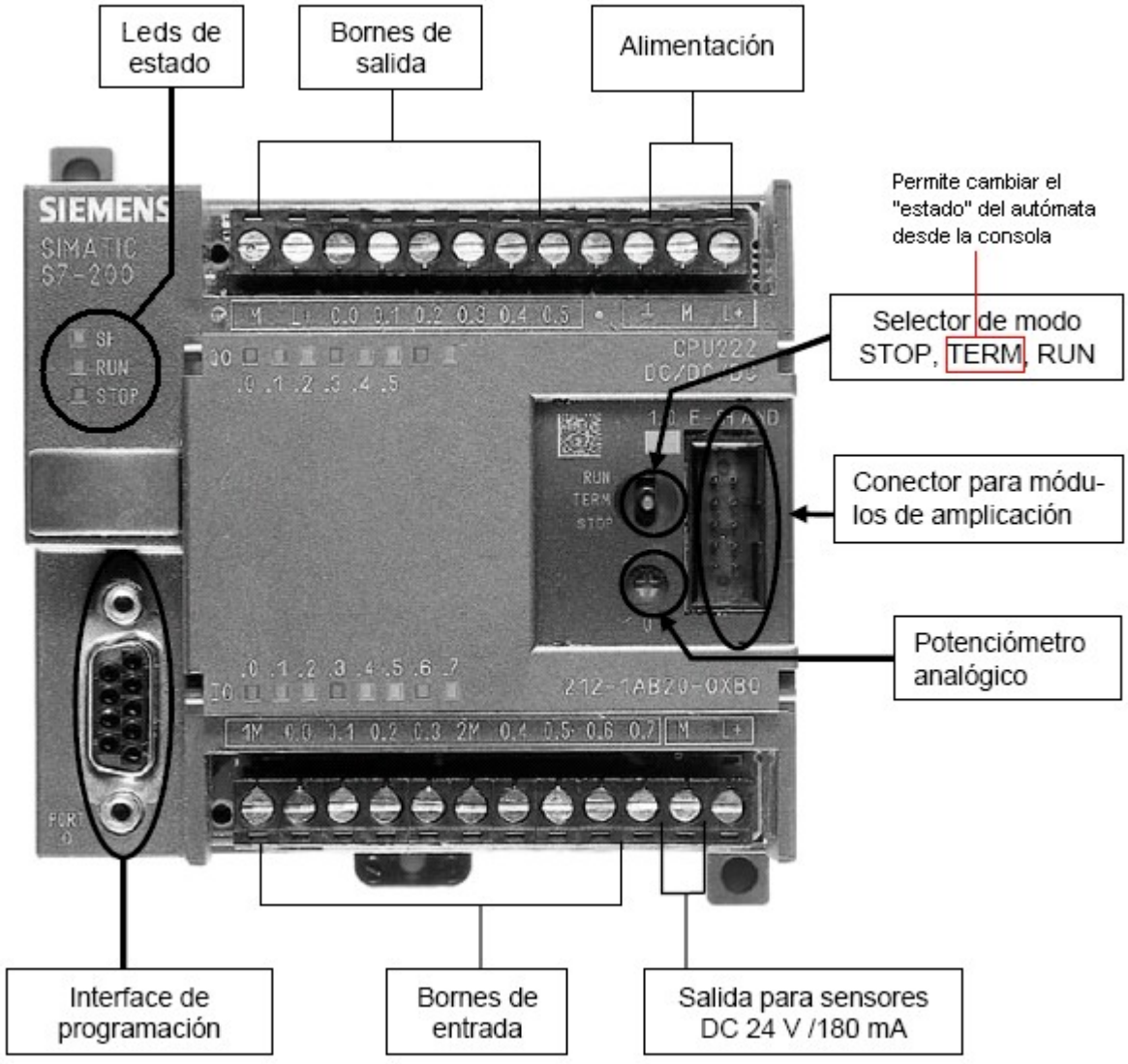




S7-200 CPU 224

1.- Constitución del PLC

En la figura podemos observar la apariencia externa que presenta un autómata de la familia S7-200. En este caso se trata de una CPU-222, la cual presenta algunas diferencias respecto de la CPU-224, con la que trabajaremos. Pese a ello, la distribución de componentes es exactamente la misma, variando la cantidad de E/S, potenciómetros analógicos, etc...



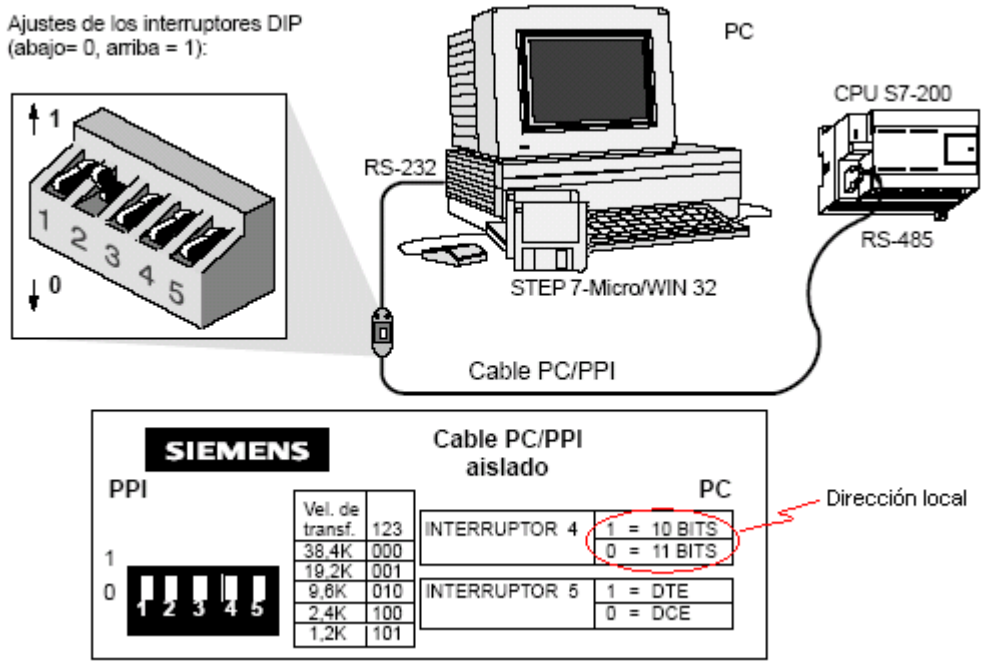
2.- Configuración de la comunicación (cable PC/PPI)

Vamos a configurar la comunicación entre la CPU S7-224 y el PC, utilizando para ello el cable PC/PPI. La configuración se realizará con un solo maestro y sin ningún otro equipo de hardware instalado (como p. ej. un módem o una unidad de programación).

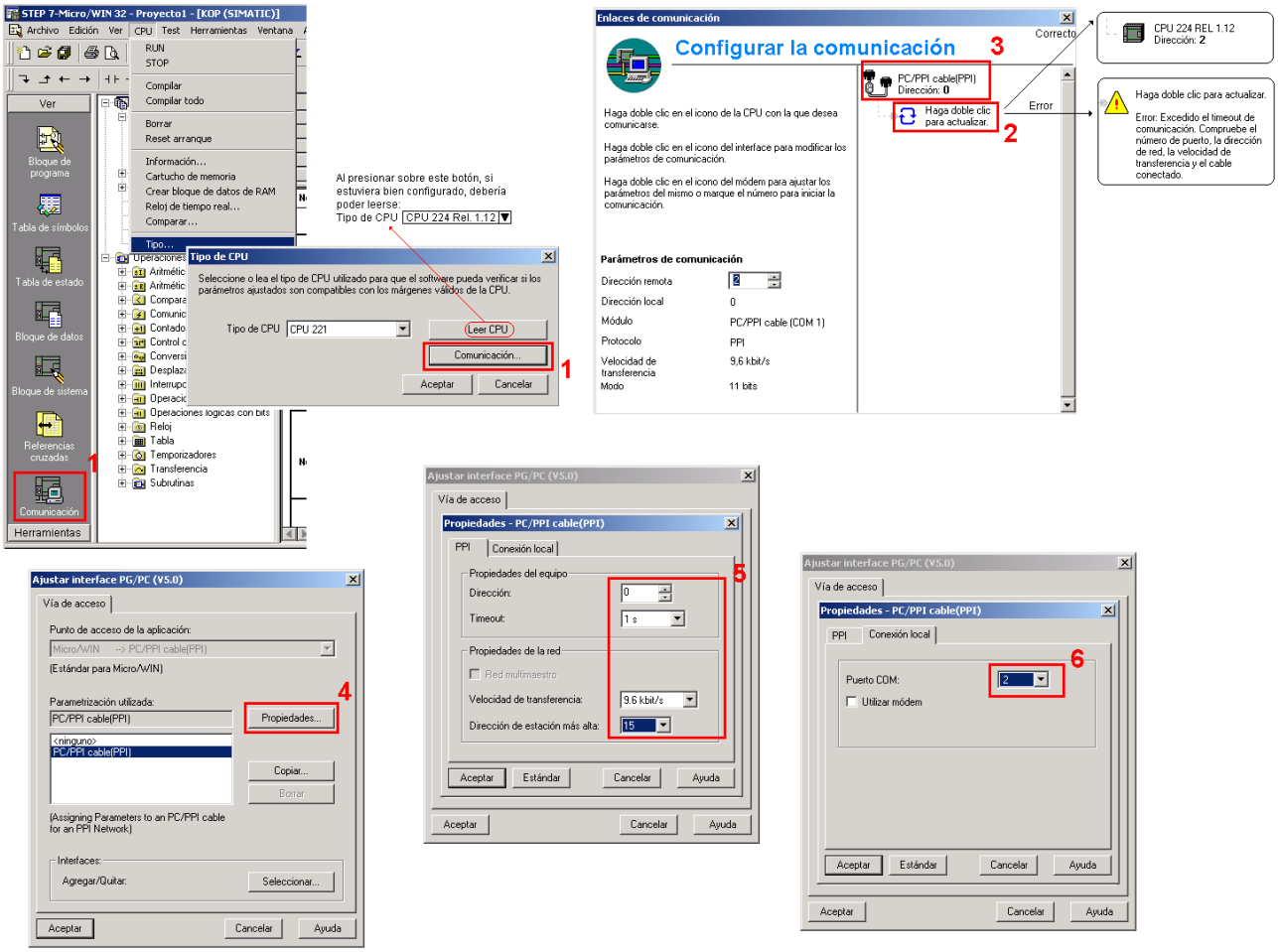
2.1.- Conectar el PC a la CPU

Para establecer una conexión correcta entre los dos componentes, deberemos realizar:

1. Ajuste los interruptores DIP del cable PC/PPI a la **velocidad de transferencia** asistida por su PC. Seleccione también las pociones "**11 bits**" y "**DCE**".
2. Conecte el extremo RS-232 ("PC") del cable PC/PPI al puerto de comunicaciones de su PC (COM1 ó COM2).
3. Conecte el extremo RS-485 ("PPI") del cable PC/PPI al puerto de comunicaciones de la CPU.



2.2.- Ajustar el interface



1. Hacer clic sobre el icono de comunicación en la barra de navegación. O en su lugar seleccionar la opción "**Tipo**" dentro del menú "**CPU**". La CPU que debería aparecer es:

   CPU 224 Rel. 1.12

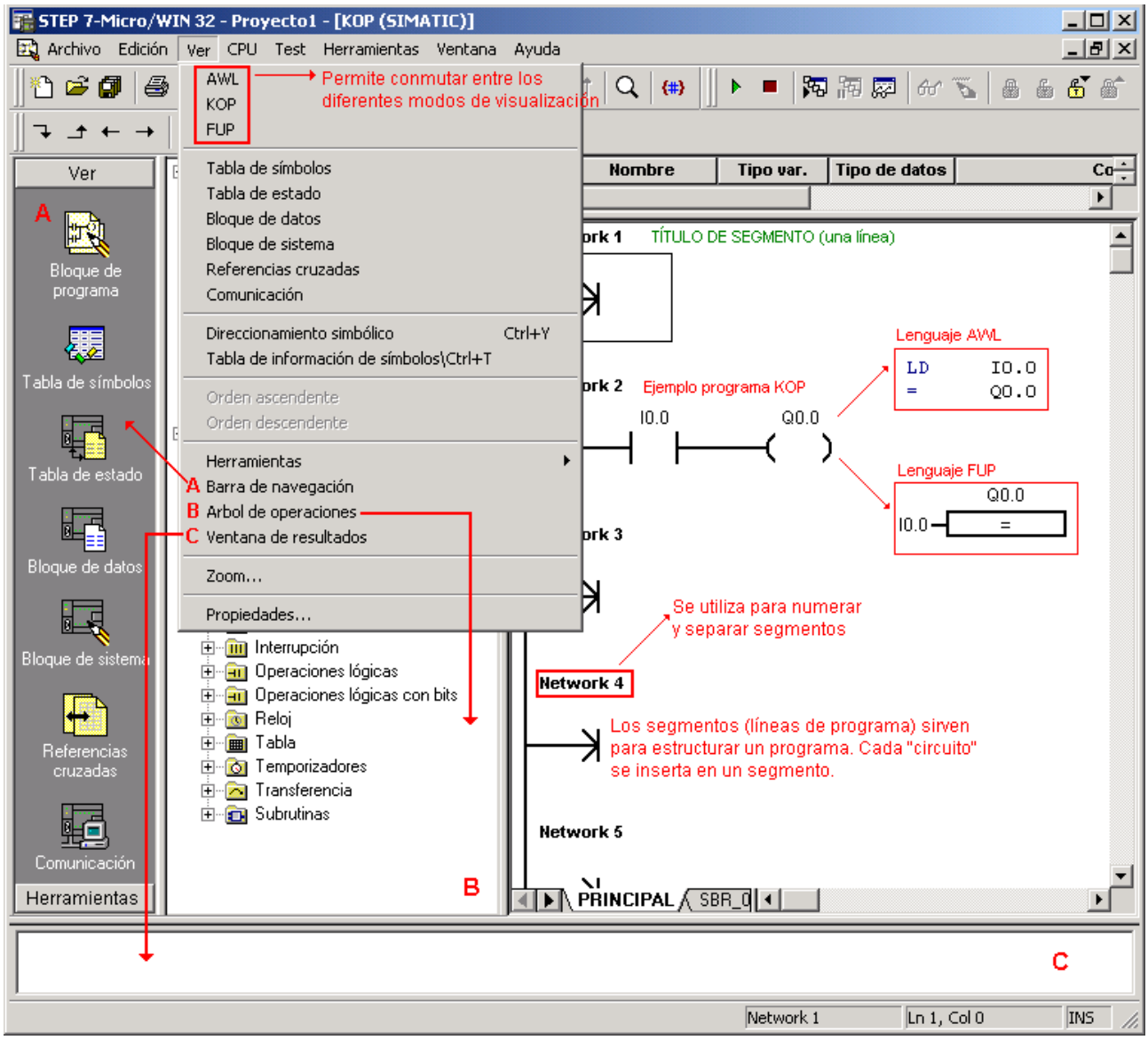
   En caso contrario, comprobar los valores de configuración ajustados para la comunicación dentro de la ventana "**Configurar la comunicación**".
2. Hacer doble clic en el campo destinado a **actualizar** la comunicación. Con ello, la CPU conectada debería reconocerse y registrarse automáticamente.
3. Si la CPU no es reconocida o aparece una información relativa a que no es posible establecer la comunicación, deberemos hacer doble clic en el campo **Cable PPI**.
4. En la opción Puerto PG/PC, seleccione Cable PC/PPI y presione el botón "**Propiedades**".
5. En la carpeta **PPI**, ajuste:
   - *Dirección de CPU* → 0.
   - *Timeout* → 1 s.
   - *Velocidad de transferencia* → 9'6 kbits/s.
   - *Dirección de estación más alta* → 15.
6. En la carpeta **Conexión Local**, seleccionaremos el puerto (interface) en el que hayamos conectado el cable PC/PPI.
   Confirmaremos los cambios realizados en cada ventana pulsando Aceptar.

Finalmente, volveremos a realizar doble clic en el campo destinado a **Actualizar** la comunicación. Con ello la CPU debería reconocerse y registrase automáticamente (esta operación puede durar algunos segundos), en caso contrario, repetiremos los pasos desde el punto 2 realizando las modificaciones oportunas hasta que reconozca la CPU. Cierre seguidamente la ventana, presionando el **aspa** de la parte superior derecha.

3.- V3.1 STEP 7 MicroWin

A continuación pasaremos a explicar algunas de las opciones del software utilizado para "programar" el autómata.

3.1.- Aspecto general



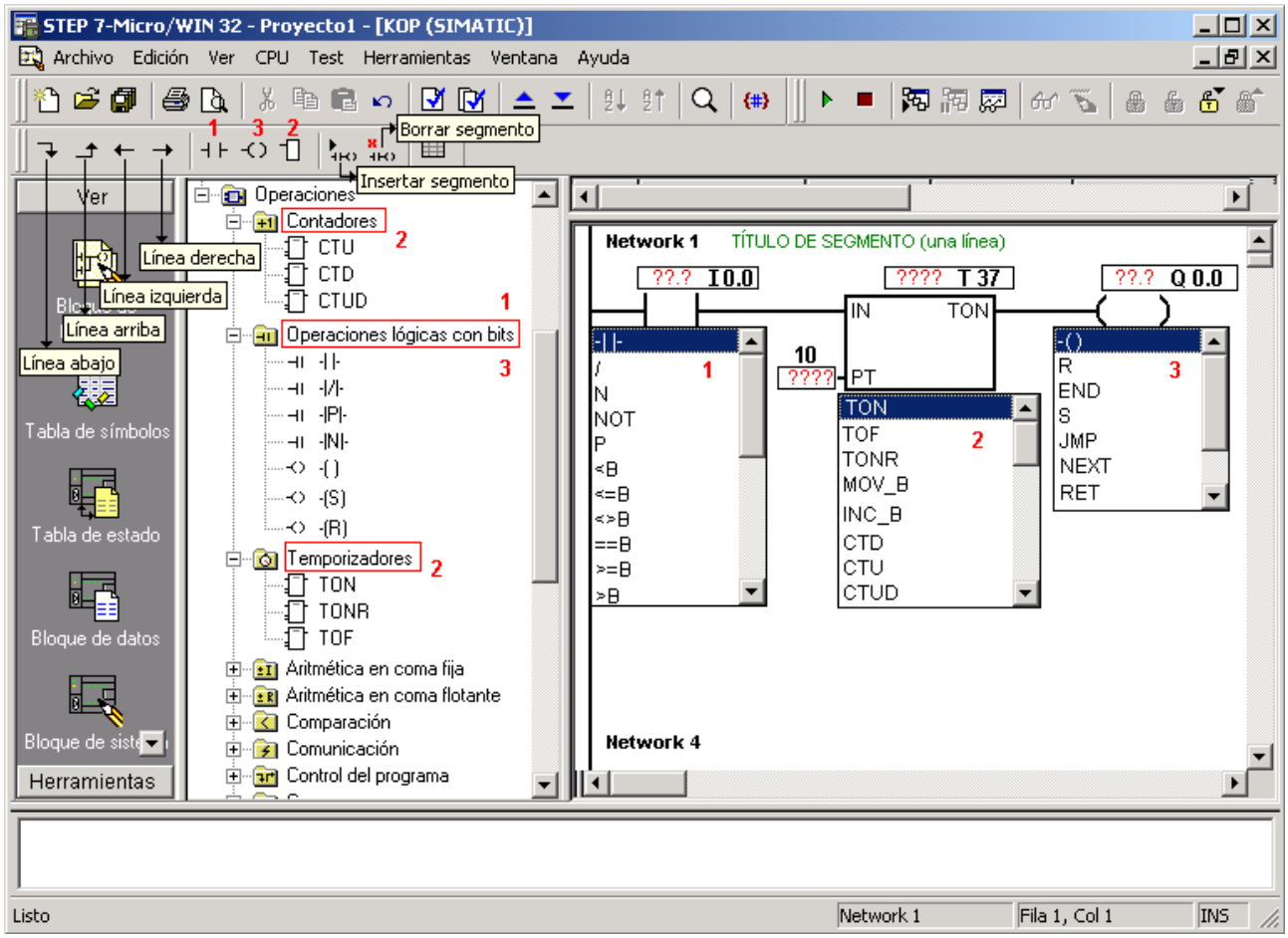
Como se desprende de la figura, la pantalla se divide en 4 partes principalmente (además de los menús e iconos de acceso rápido):

- **Barra de navegación**: nos permite acceder a las opciones más comunes de forma rápida.
- **Árbol de operaciones**: en donde se sitúan todas las órdenes de programación aceptadas por el autómata.

- **Ventana de resultados**: en la que se visualiza el estado de la compilación del programa, errores, etc...
- **Ventana de programación**: situada a la parte derecha y dividida por Networks (líneas de programación). En este lugar elaboraremos el programa que ha de gobernar al PLC.
  Su aspecto varía según el lenguaje elegido (KOP, AWL ó FUP) y que podremos seleccionar a través de las teclas que llevan sus mismos nombres.
  Hay que señalar que el programa es capaz de traducir a cualquiera de estos lenguajes, es decir: si p. ej. estamos programando en AWL y seleccionamos el lenguaje KOP, se realizará automáticamente una traducción del programa de AWL a KOP...

3.2.- Introducir órdenes

A partir de ahora todas las explicaciones versarán sobre el lenguaje KOP, por tratarse del lenguajes más intuitivo debido a su carácter eléctrico.



El programa presenta varias maneras de introducir contactos, bobinas o cuadros:

- Desde el *Árbol de direcciones*, abriendo las distintas carpetas existentes dentro de **Operaciones**.
- O bien a través de los iconos que aparecen como marcados en el dibujo como:
  - **1 (contactos)** → para insertar entradas.
  - **2 (bobinas)** → para insertar salidas.
  - **3 (cuadros)** → para insertar funciones ya programadas (contadores, temporizadores, etc...).

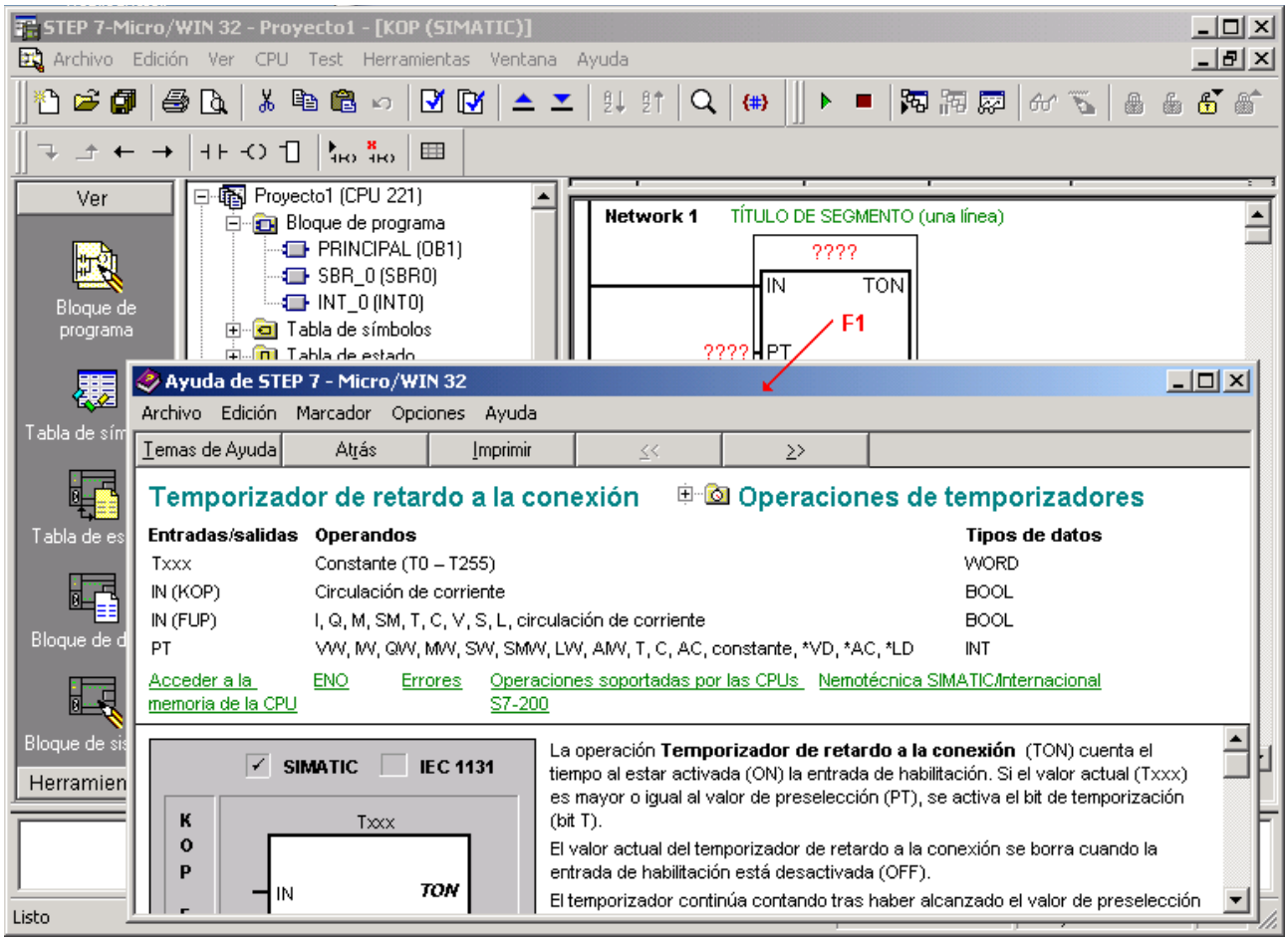
Una vez introducido el elemento seleccionado, deberemos darle nombre: para ello deberemos colocarnos en los interrogantes situados en la parte superior del elemento y teclear la estructura explicada con anterioridad para entradas y salidas (el resto de elementos serán explicados más adelante).

Para realizar combinaciones (serie, paralelo, mixto...) de funciones/elementos deberemos utilizar "**las líneas**", que permiten realizar ramificaciones a partir de una única línea.

3.3.- Ayuda

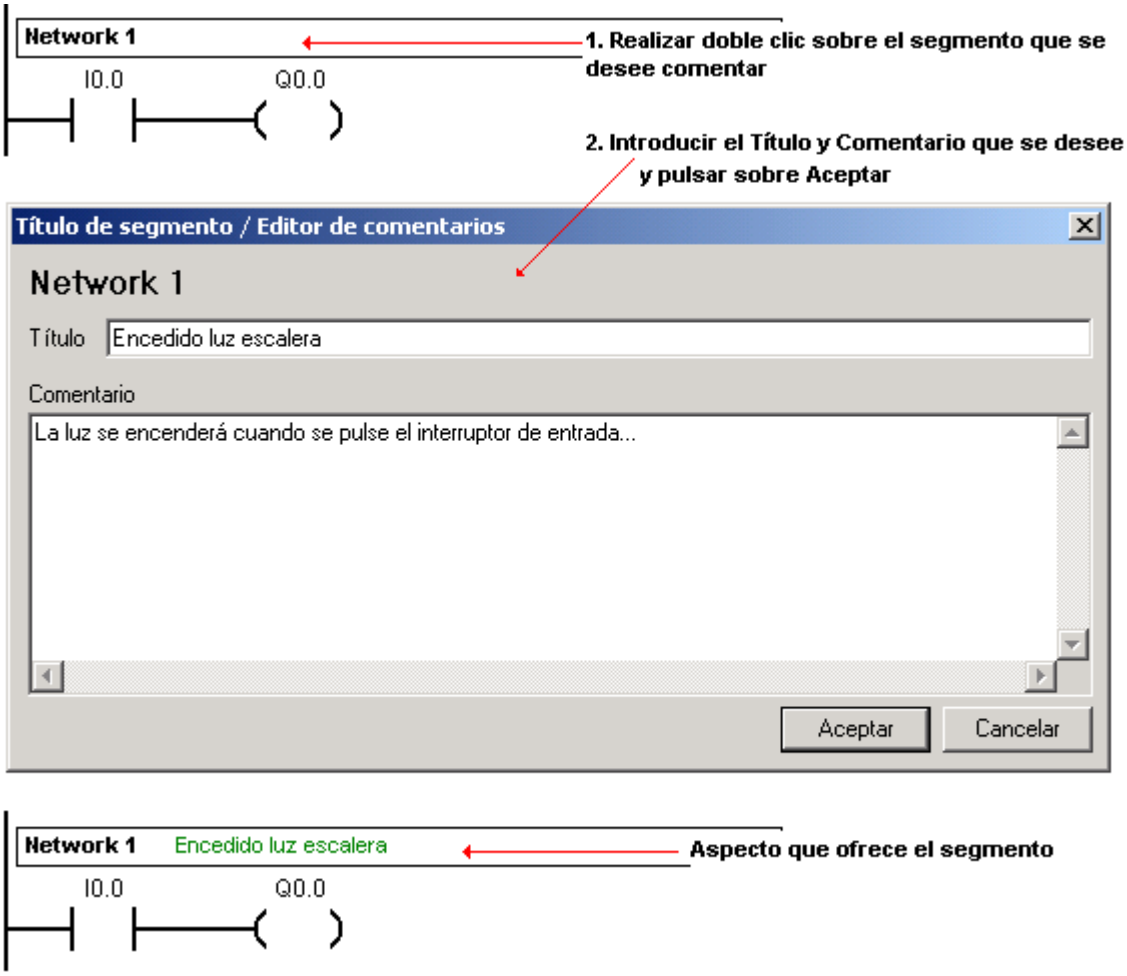
Como cualquier programa, que se precie, disponemos de menús de ayuda de cualquier elemento.

Para acceder a él, basta con seleccionar el objeto del que se quiere obtener la ayuda y presionar F1 sobre el teclado:



3.4.- Introducir comentarios

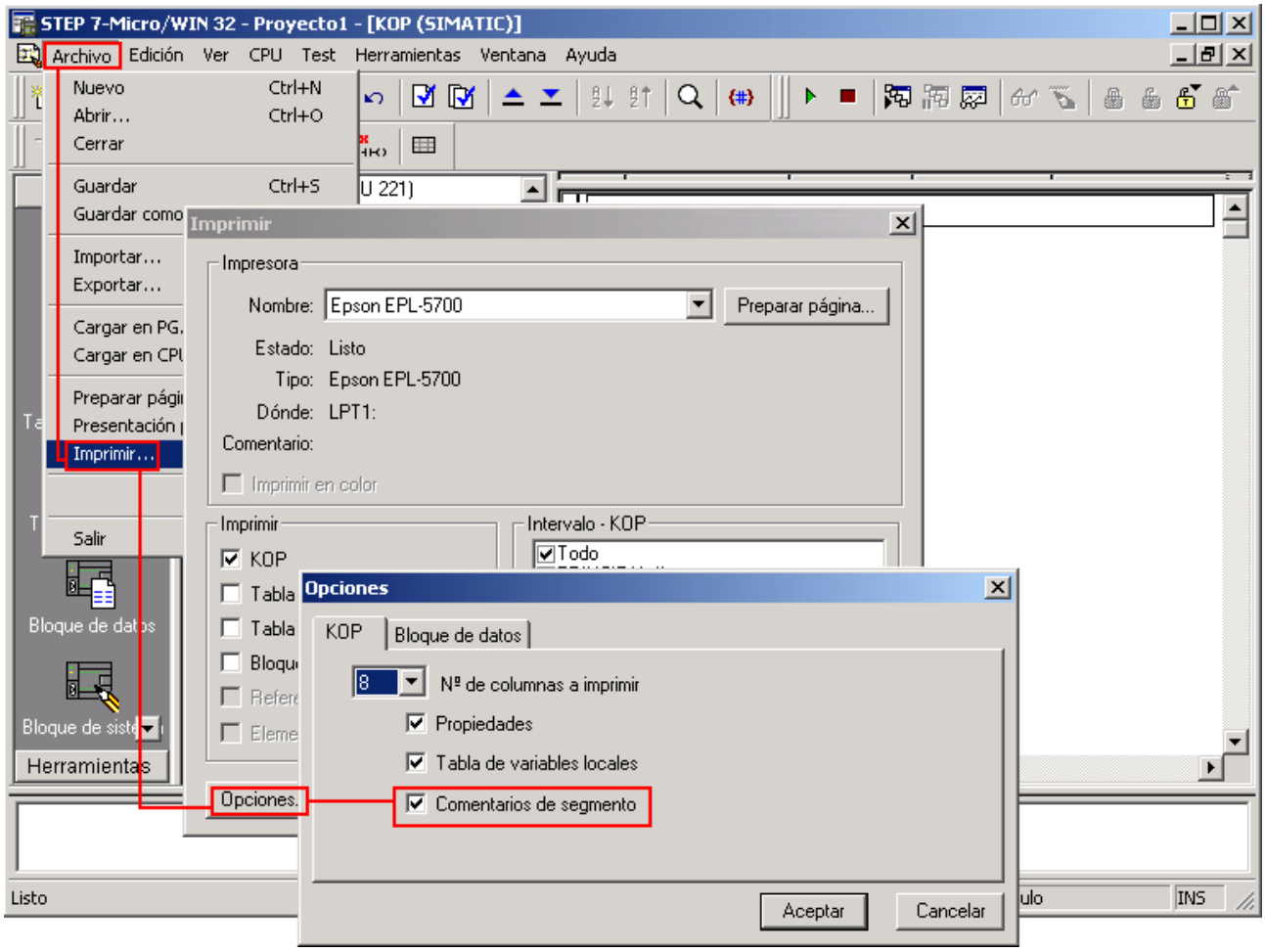
Podemos introducir comentarios dentro de cada segmento que faciliten la interpretación del programa:



El editor de comentarios se dividen en:

- **Título del segmento**. Se visualiza en pantalla.
- **Comentario**. No aparece en pantalla, para poderlo observar deberemos:
  - Realizar doble clic sobre el segmento/Network correspondiente.
  - O bien imprimir el programa, especificando que se impriman dichos comentarios.

Para imprimir los comentarios introducidos:



3.5.- Direccionamiento simbólico

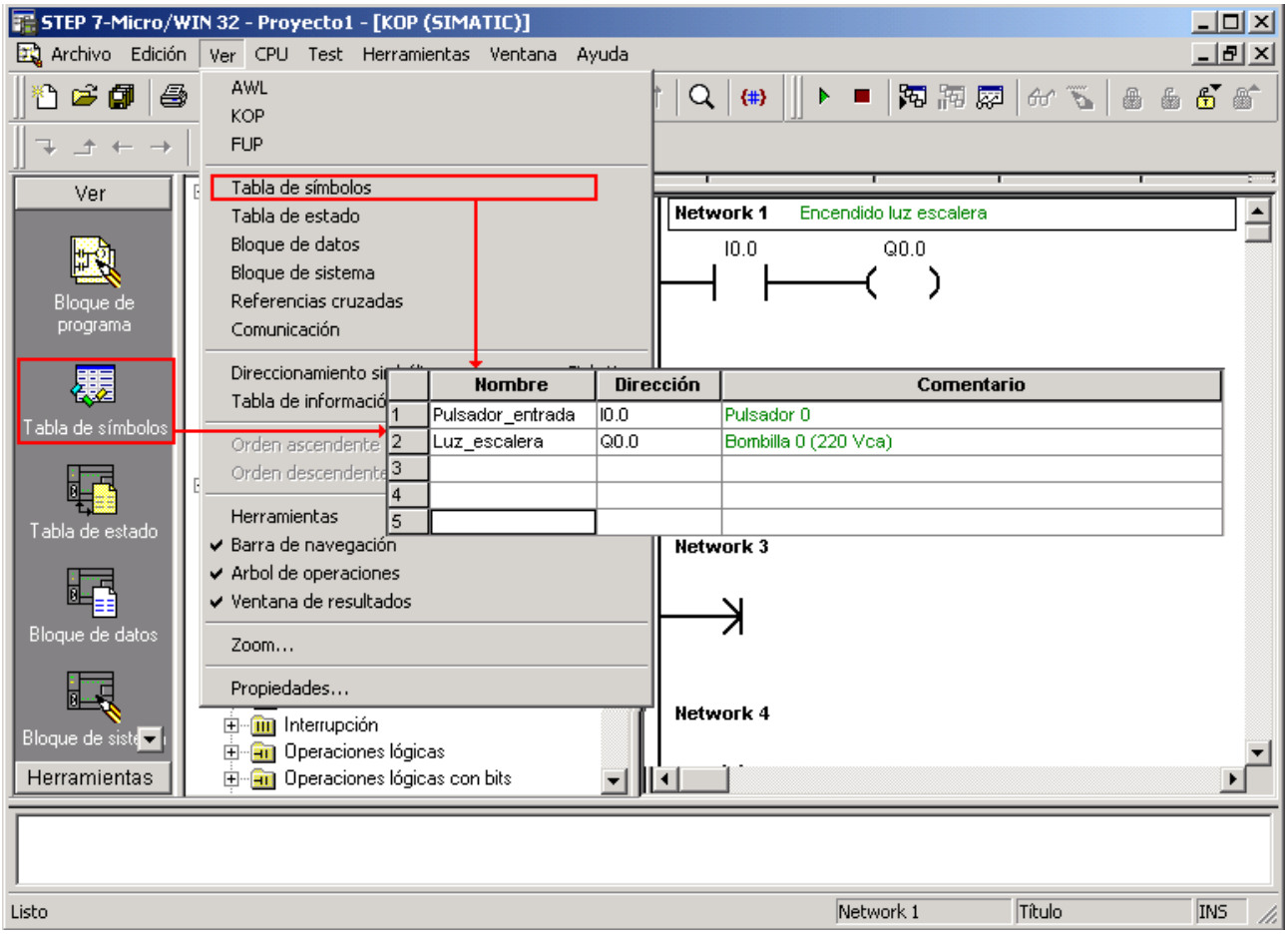
Hasta ahora hemos editado el programa del PLC utilizando operandos en el "idioma del PLC" (I 0.0, Q 0.0, etc...). Sin embargo, con un programa muy largo, este tipo de operandos dificulta su lectura y comprensión. Sería muy útil poder trabajar con las denominaciones de los interruptores o con un texto explícito, es decir, en lugar de I 0.0 utilizar "pulsador de marcha"...

Para ello, hemos de recurrir al direccionamiento simbólico, al cual podemos acceder a través de la *Barra de navegación* o bien recurriendo a las opciones del menú *Ver*, seleccionando en ambos casos la opción **Tabla de símbolos**.

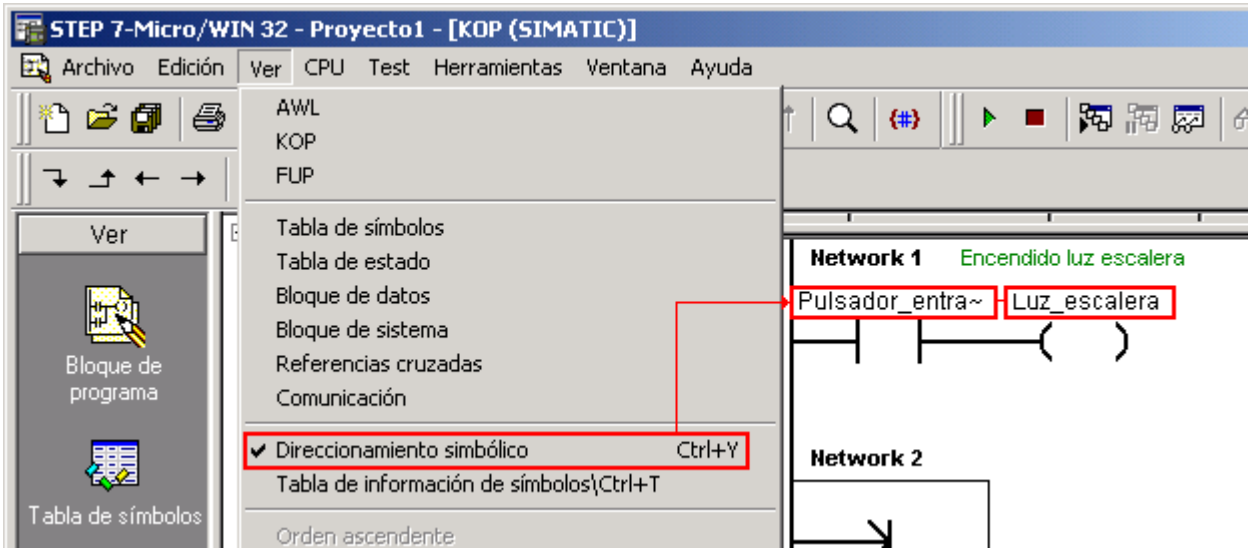
Con ello obtendremos una ventana para editar la tabla de símbolos:

- Bajo "**nombre**" introduciremos lo que luego se visualizará como texto explícito.
- Bajo "**direcciones**" se introducen los operandos que deben ser sustituidos por los nombres simbólicos.
- Bajo "**comentario**" podemos introducir un texto explicativo.

Para que tenga efecto, no deberemos olvidar guardar el trabajo realizado.



Finalmente, debemos activar el direccionamiento simbólico. Para ello, a través del menú *Ver* seleccionaremos la opción **Direccionamiento simbólico**:



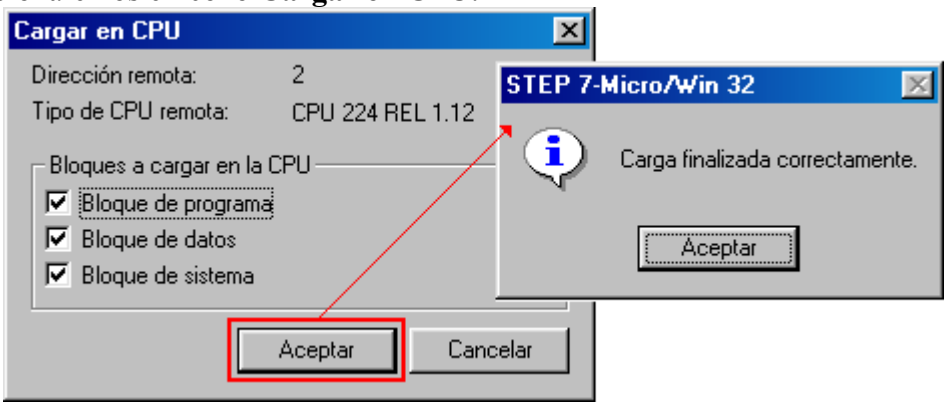
3.6.- Compilar-ejecutar...

A continuación explicaremos la secuencia a seguir para una correcta transmisión y ejecución del programa diseñado:

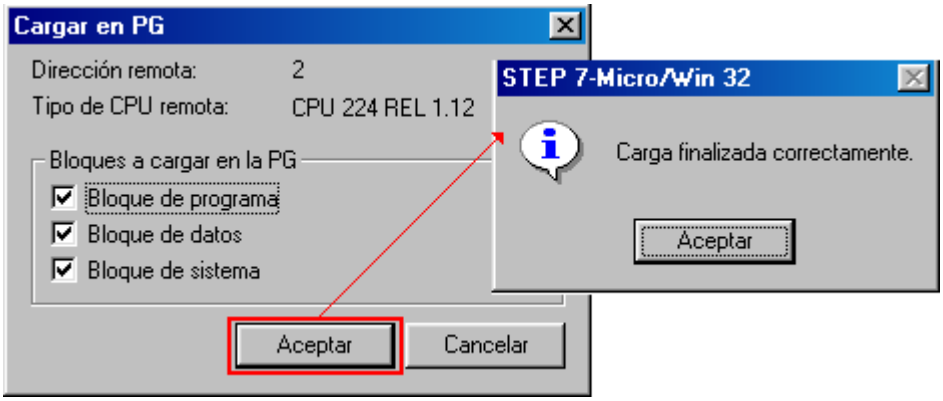
1. En primer lugar **compila**remos el programa, con la finalidad de depurar posibles "errores ortográficos". El resultado de la compilación aparecerá en la Ventana de resultados

Si existe algún error deberemos subsanarlo, en caso contrario pasamos al siguiente punto...

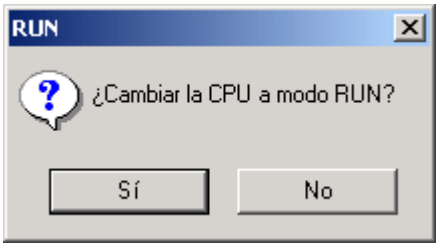
2. Llegados a este punto debemos transferir el programa elaborado al autómata, para ello seleccionaremos el icono **Cargar en CPU**.



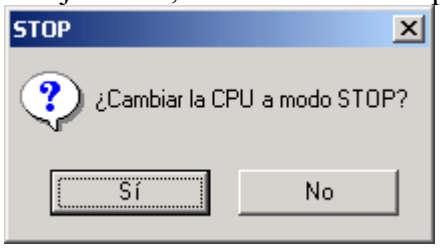
La opción **Cargar en PG** realiza el proceso contrario, es decir, carga el programa que tiene el autómata en memoria al MicroWin.



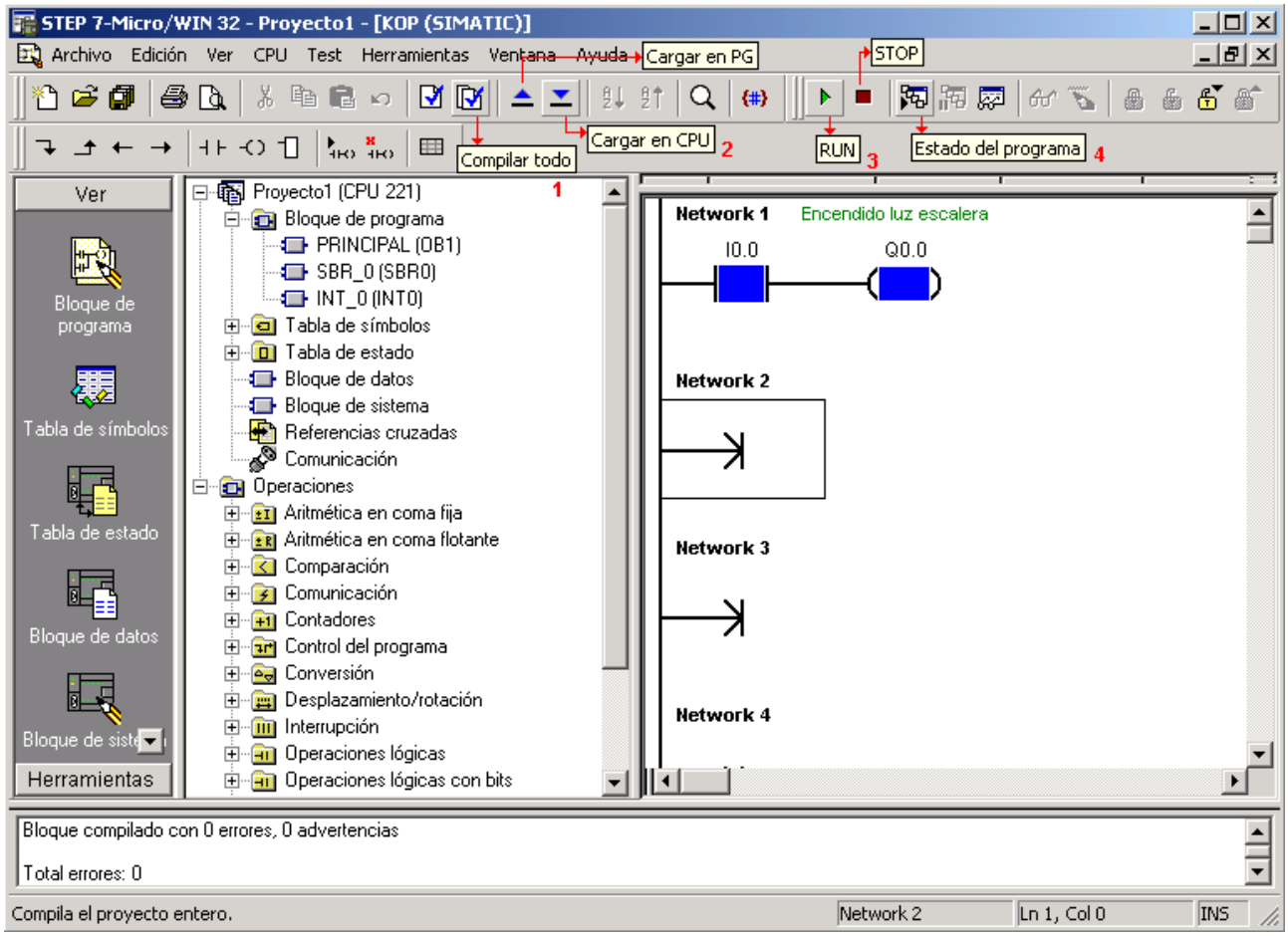
3. Por fin podemos ejecutar el programa, mediante la opción **RUN**, y observar su funcionamiento real a través del PLC. Debemos recordar que el autómata debe tener su selector en posición *TERM*.



Cuando queramos detener la ejecución, será suficiente con presionar el icono **STOP**.



4. Existe la posibilidad de visualizar el desarrollo del programa a través del MicroWin y de este modo poder depurar y perfeccionar el código elaborado). Esto es posible mediante la opción **Estado del programa**, de este modo cuando se active un contacto su interior aparecerá de color azul.
   Debemos tener cuidado con esta opción, pues cuando se encuentra activada no permite realizar ninguna modificación al programa.



Cualquier modificación realiza al programa, para que surja efecto, deberá ser transferida de nuevo al autómata

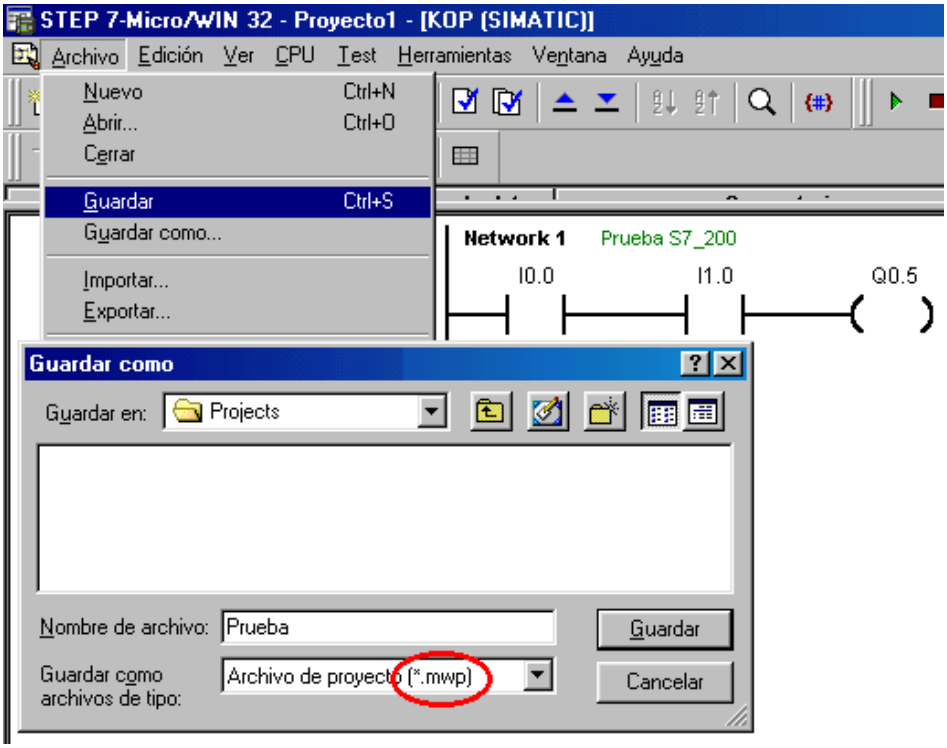
4.- Simulador S7_200

El problema que plantea el programa anterior reside en el hecho de que no permite simular el programa diseñado a no ser que conectemos una autómata.

Para subsanar este hecho utilizaremos un simulador, desde el cual podamos probar nuestros diseños sin necesidad de tener un PLC. A continuación se detallan los pasos a seguir:

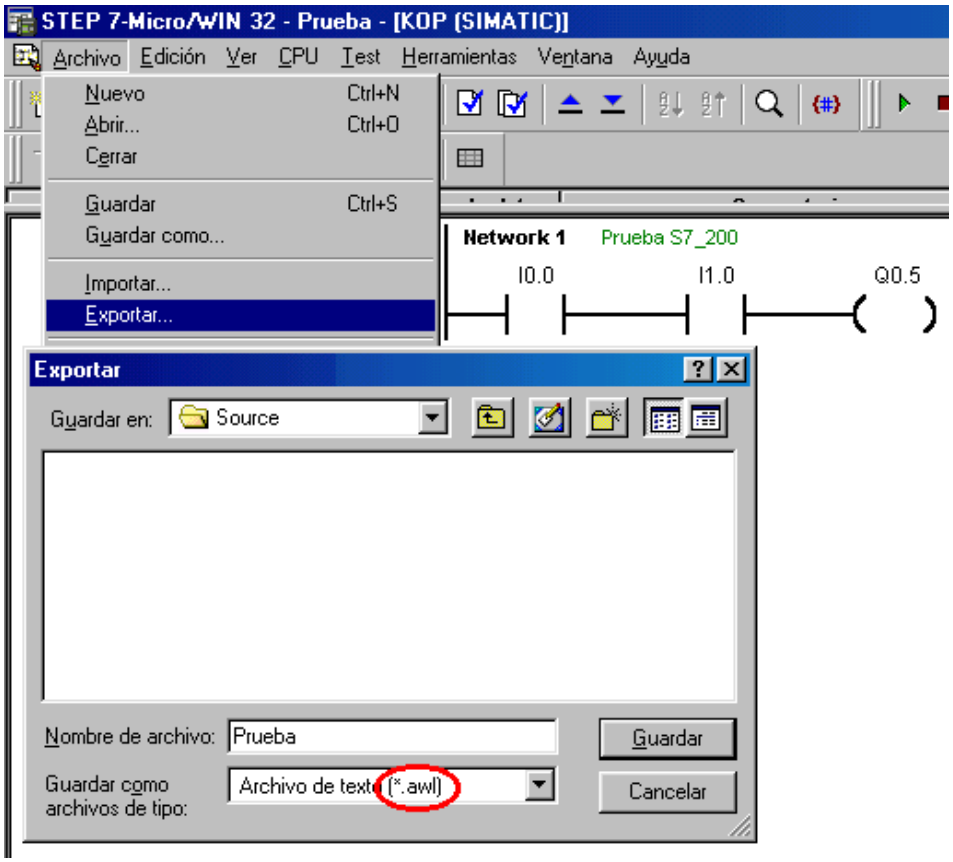
4.1.- Adecuar el archivo

Una vez diseñado y compilado el programa (ver apartado 3.6), para asegurarnos de que no existen errores, lo guardaremos... al *guardar* se crea un archivo de extensión MWP con el nombre que le indiquemos, por ejemplo Prueba.mwp



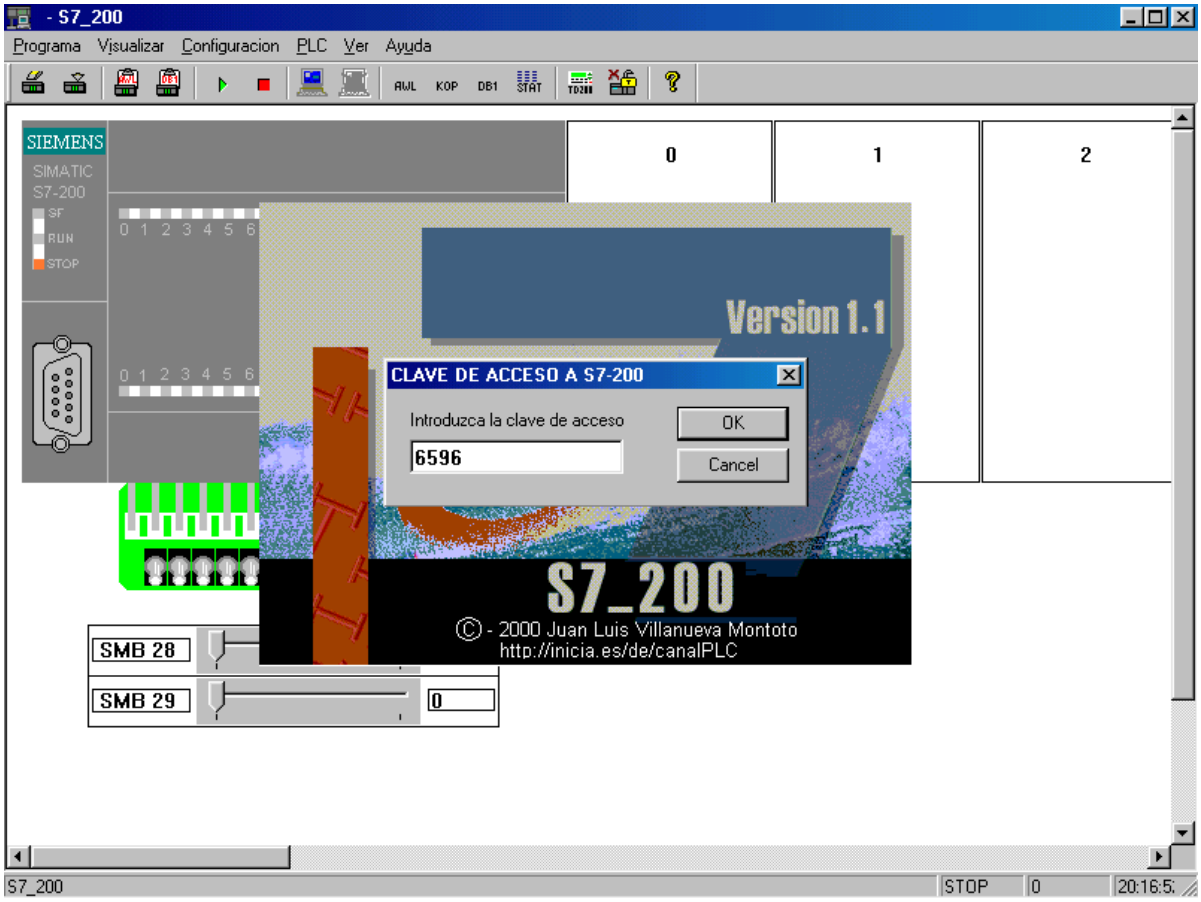
Este archivo no es adecuado, ya que el simulador sólo acepta archivos con extensión AWL.

Por ello, una vez guardado, deberemos *exportar* el programa para conseguir un archivo con extensión AWL, que es la extensión aceptada por el simulador. Podemos darle, por ejemplo, el nombre Prueba.awl



4.2.- Ejecutar el simulador

Cada vez que ejecutemos el simulador, nos pedirá una contraseña que deberemos introducir de forma correcta para habilitar sus funciones...

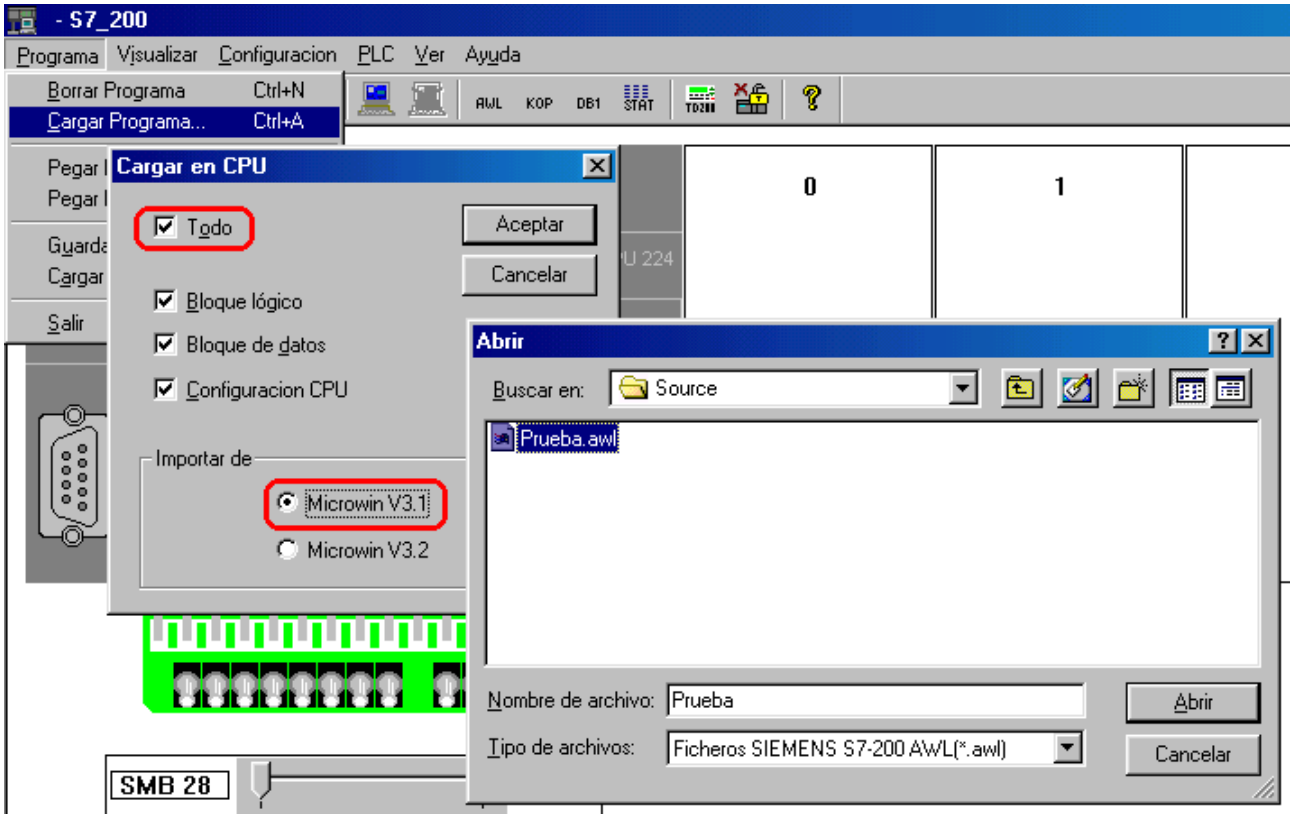


4.3.- Configurar el tipo de CPU

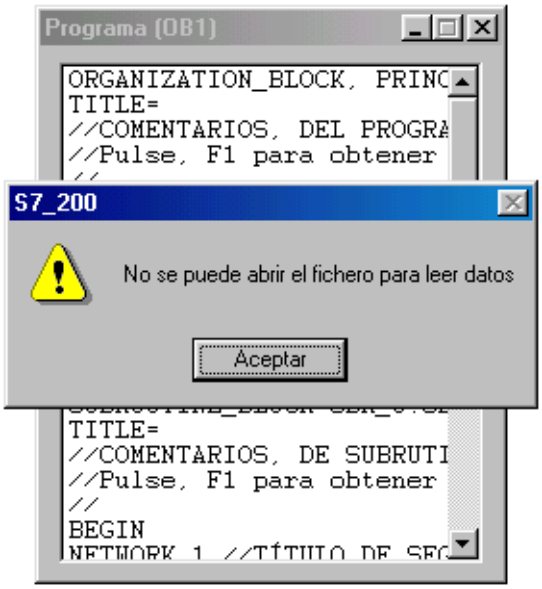
Antes de cargar ningún programa, deberemos configurar correctamente el tipo de autómata... en nuestro caso, recordemos que se trata de la CPU 224.

4.4.- Cargar el programa

Ahora ya podemos cargar el programa que queremos simular... deberemos tener en cuenta la versión del MircoWin utilizada para el diseño del programa.

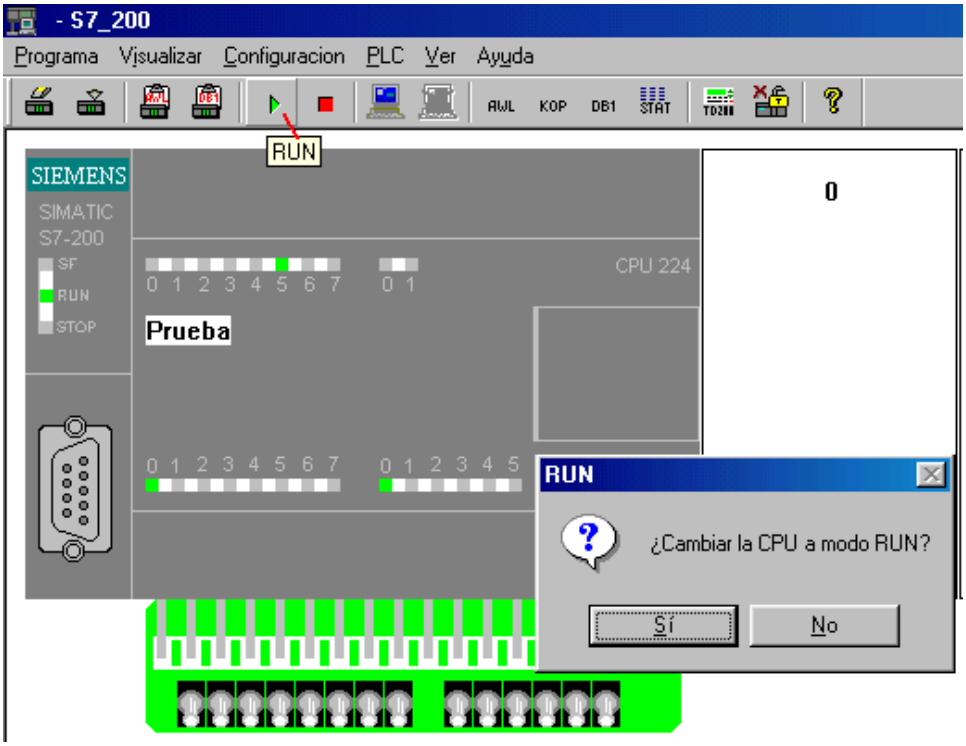


Una vez seleccionado y abierto el programa a simular, aparecerá un mensaje de error. Pero, que no os preocupe porque se ha cargado correctamente...



4.5.- RUN y simular

Finalmente ya solo nos queda poner en RUN el simulador y "jugar" con la botonera...

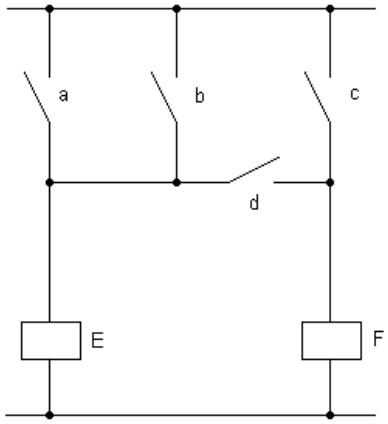


5.- Ejercicios

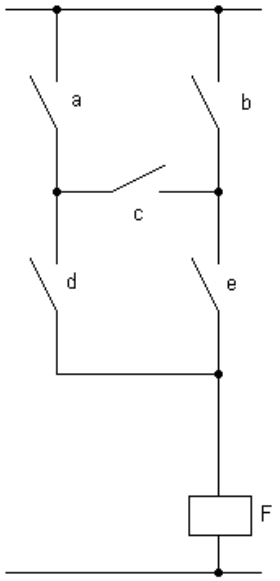
Transformar los siguientes ejercicios a la lógica programada que es capaz de interpretar el autómata.

Simula y comprueba su funcionamiento.

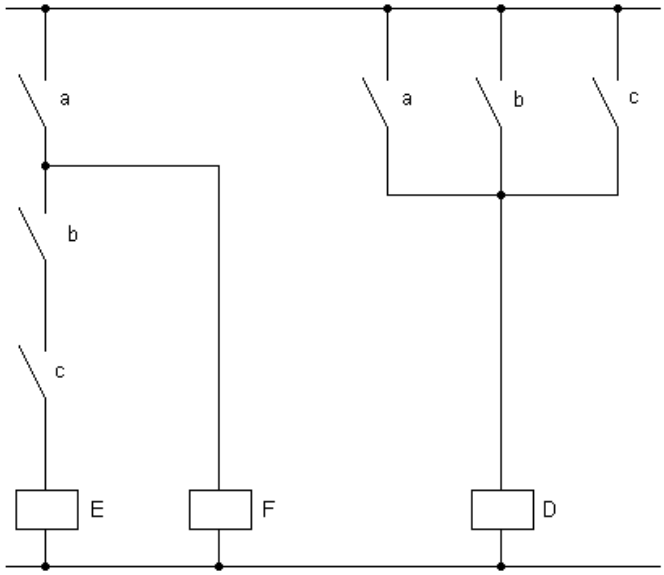
5.1.- Circuito en puente simple



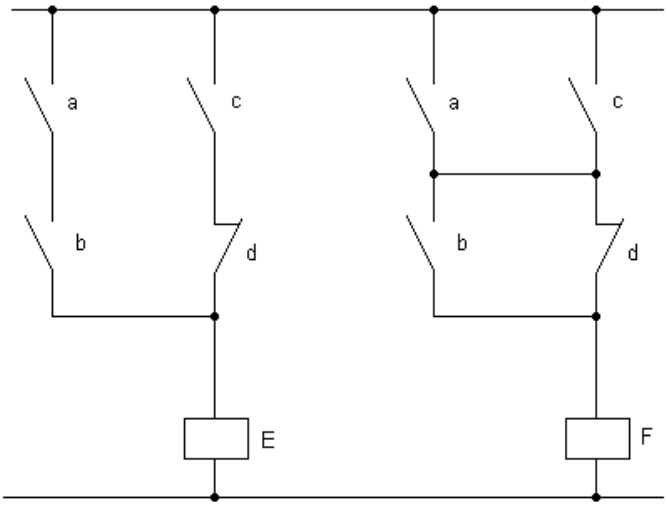
5.2.- Circuito en puente complicado



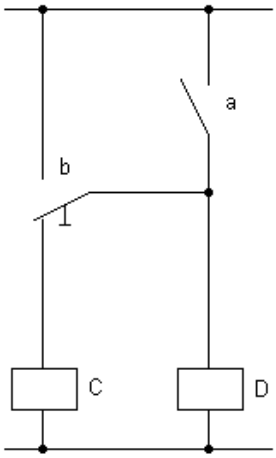
5.3.- Serie-paralelo



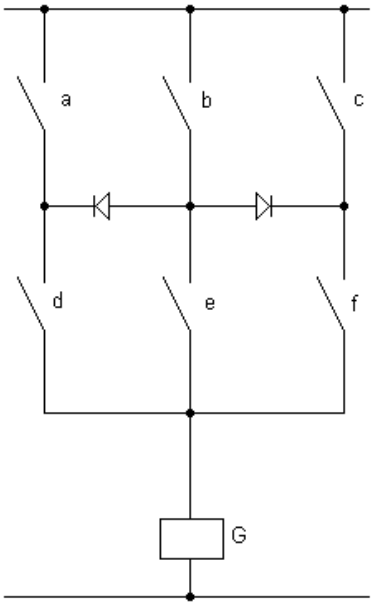
5.4.- Contactos NC



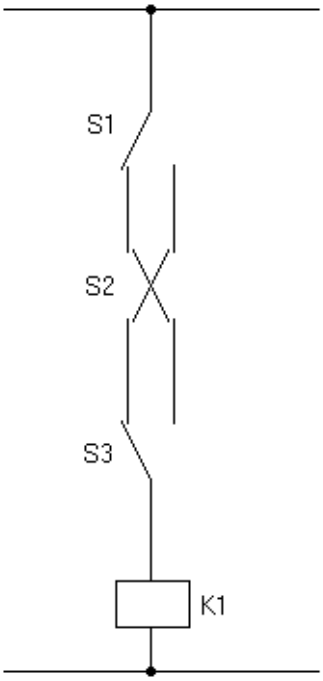
5.5.- Conmutador



5.6.- Circuito con diodos

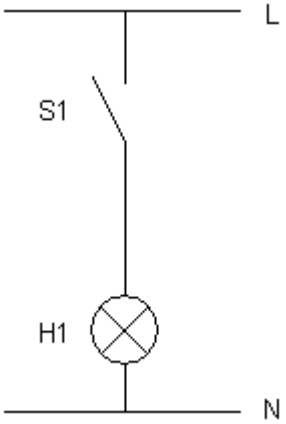


5.7.- Circuito "cruzamiento"

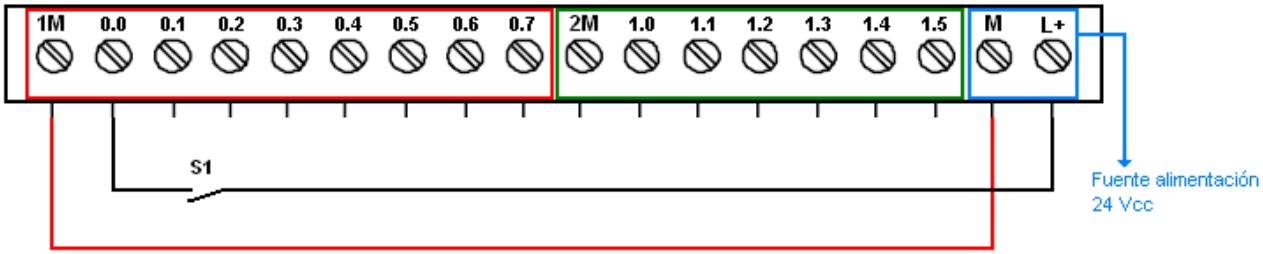


6.- Conexionado entradas-salidas

Para una mejor comprensión distinguiremos entre el bornero de entradas del bornero de salidas, pues cada uno posee distinta estructuración... supongamos el siguiente montaje muy sencillo:



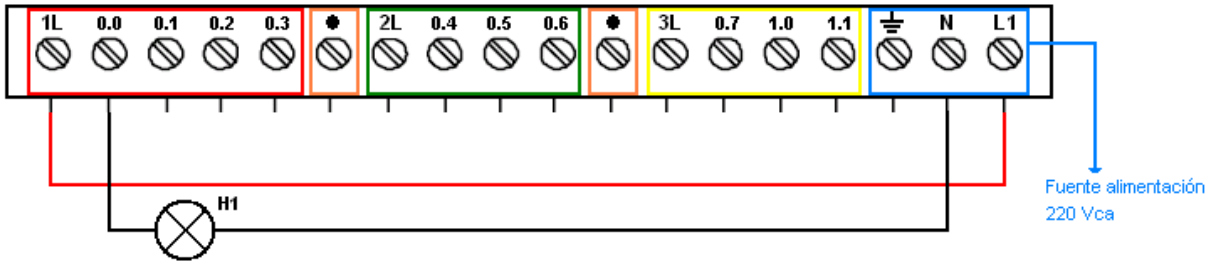
6.1.- Bornero de entradas



Deberemos pontear 1M con M para "alimentar" su "conjunto", es una especie de permiso de conexión (para "activar" el conjunto 2M, deberemos pontear 2M con M).

En el caso de que el sensor necesitara alimentación, deberemos alimentarlo también a la tensión correspondiente.

6.2.- Bornero de salidas



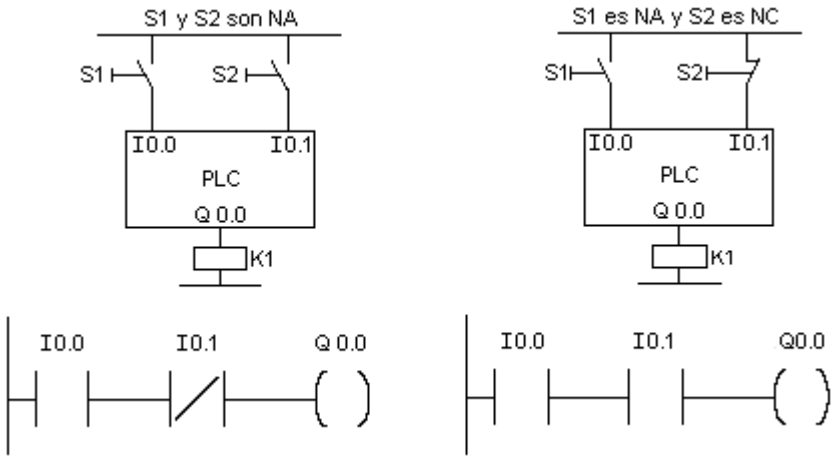
Para "activar/alimentar" cada "conjunto", deberemos conectar el positivo de alimentación (L1 para 220 Vca) a 1L, 2L ó 3L según convenga.

Al realizar este conexionado, tendré todas las salidas asociadas a 1L (desde 0.0 hasta 0.3) alimentadas a 220 V en alterna, por tanto deberé tener cuidado de no conectar a estas salidas ningún dispositivo que funcione a cualquier otra tensión. Debiendo utilizar en este caso una salida de otro conjunto (p. ej. 0.4 perteneciente a 2L), alimentando dicho conjunto a la tensión apropiada.

Las salidas identificadas con un punto no tiene conexión, es decir, están deshabilitadas...

6.3.- Conexión elementos NA-NC

La naturaleza de los elementos que utilicemos en el montaje influye en gran manera en el diseño del programa. Supongamos, por ejemplo, un sencillo circuito con un botón de marcha que accione un elemento y otro de paro que los desactive...



Según el dibujo anterior, dependiendo de la naturaleza del botón de paro utilizaremos una programación u otra:

- **Paro NA**. Deberemos programarlo cerrado para que permita el paso de corriente en su estado de reposo e interrumpa la circulación en el momento se accione.
- **Paro NC**. Se programará abierto, pues será su propia naturaleza la que cierre el contacto durante el estado de reposo, mientras que al presionarlo los contactos se separarán impidiendo el paso de corriente.

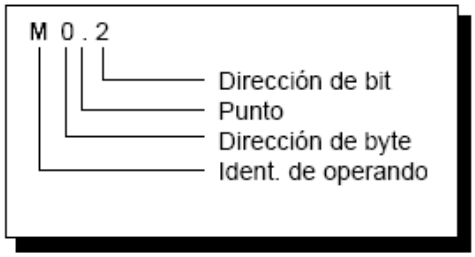
OPERACIONES SIMATIC

Este apartado describe el juego de operaciones SIMATIC para los sistemas de automatización S7-200.

1.- Marcas

Hasta ahora solamente habíamos hablado de entradas (I) y de salidas (Q). Vamos a añadir un nuevo término llamado **Marca**, cuyo identificador de operando es: **M**.

Al igual que las entradas y salidas, junto con el identificador de operando necesita de un parámetro. Éste tiene exactamente la misma estructura que las entradas y salidas:



Consideraciones:

- Las marcas se utilizan como la memoria de una calculadora de bolsillo, para guardar resultados intermedios.
- Las marcas se utilizan cuando el resultado intermedio de un segmento debe procesarse en otros segmentos o para guardar estados sucesivos evaluados.
- En PLC's, las marcas se utilizan como salidas; su efecto es similar a los relés o contactores auxiliares utilizados en la técnica convencional. Una marca puede utilizarse todas las veces que se desee como contacto NA o NC.
- Si se corta la alimentación se pierde el estado de la marca.
  Para evitar esto existe la función de "remanencia" (Set).

1.1.- Marcas especiales

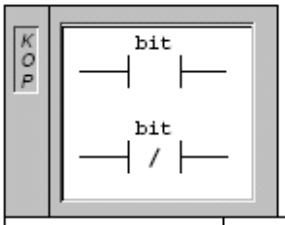
Las **marcas especiales** (**SM**) ofrecen una serie de funciones de estado y control. Sirven para intercambiar informaciones entre la CPU y el programa, pudiéndose utilizar en formato de bits, bytes, palabras o palabras dobles.

A continuación se presentan algunas marcas especiales:

Marcas	Descripción (sólo lectura)
SM 0.1	Se activa en el primer ciclo. Se utiliza p. ej. Para llamar una subrutina de inicialización.
SM 0.4	Ofrece un reloj que está activado durante 20 segundos y desactivado otros 30 segundos, siendo el tiempo de ciclo de 1 minuto. Ofrece un retardo fácil de utilizar o un tiempo de reloj de 1 minuto.
SM 0.5	Ofrece un reloj que está activado durante 0'5 segundos y desactivado otros 0'5 segundos, siendo el tiempo de ciclo de 1 segundo. Ofrece un reloj que está activado 0'5 segundos y desactivado 0'5 segundos, siendo su tiempo de reloj de 1 minuto.
SM 0.6	Ofrece un reloj que está activado un ciclo y desactivado en el ciclo siguiente. Se puede utilizar como entrada de contaje de ciclos.
SMB 28	Lectura de los potenciómetros analógicos.
SMB 29	

2.- Operaciones lógicas con bits

2.1.- Contactos estándar



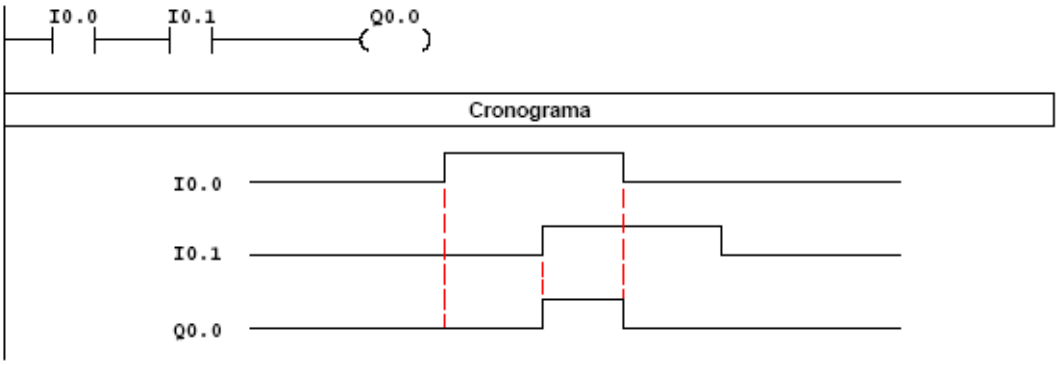
Entradas/salidas	Operandos	Tipos de datos
Bit	I, Q, M, SM, T, C, V, S, L	BOOL

Estas operaciones leen el valor direccionado de la memoria o de la imagen del proceso si el tipo de datos es I o Q.

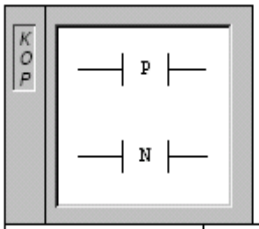
Su forma de proceder es:

- El **contacto normalmente abierto** se cierra (ON) si el bit es igual a 1.
- El **contacto normalmente cerrado** se cierra (ON) si el bit es igual a 0.

Para combinaciones AND y OR se pueden utilizar siete entradas como máximo.



2.2.- Detectar flanco positivo y negativo

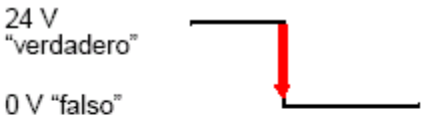


Entradas/salidas	Operandos	Tipos de datos
	I, Q, M, SM, T, C, V, S, L, circulación de corriente	BOOL

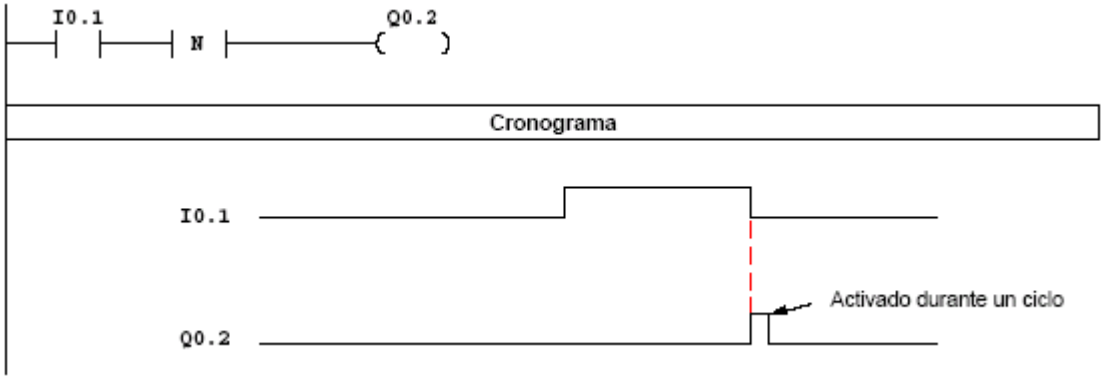
Forma de actuar:

- El contacto **detectar flanco positivo** permite que la corriente circule durante un ciclo cada vez que se produce un cambio de 0 a 1 (de “off” a “on”).
  La transición de un contacto (entrada, salida...) de “abierto” a “cerrado” o de “falso” a “verdadero” se designa como flanco creciente o positivo.

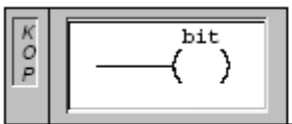
- El contacto **detectar flanco negativo** permite que la corriente circule durante un ciclo cada vez que se produce un cambio de señal de 1 a 0 (de "on" a "off").
  La transición de "cerrado" a "abierto" o de "verdadero" a "falso" se designa como flanco decreciente o negativo.



Se colocan después de un contacto estándar, realizando su función sobre este (solamente sobre el que le antecede).

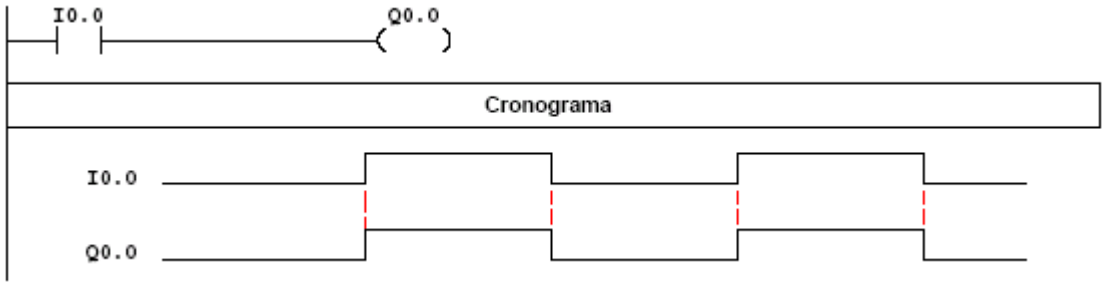


2.3.- Asignar

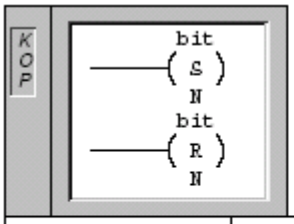


Entradas/salidas	Operandos	Tipos de datos
Bit	I, Q, M, SM, T, C, V, S, L	BOOL

Cuando se ejecuta la operación **asignar**, el bit de salida se activa en la imagen del proceso. El bit indicado se ajusta de forma equivalente a la circulación de la corriente.



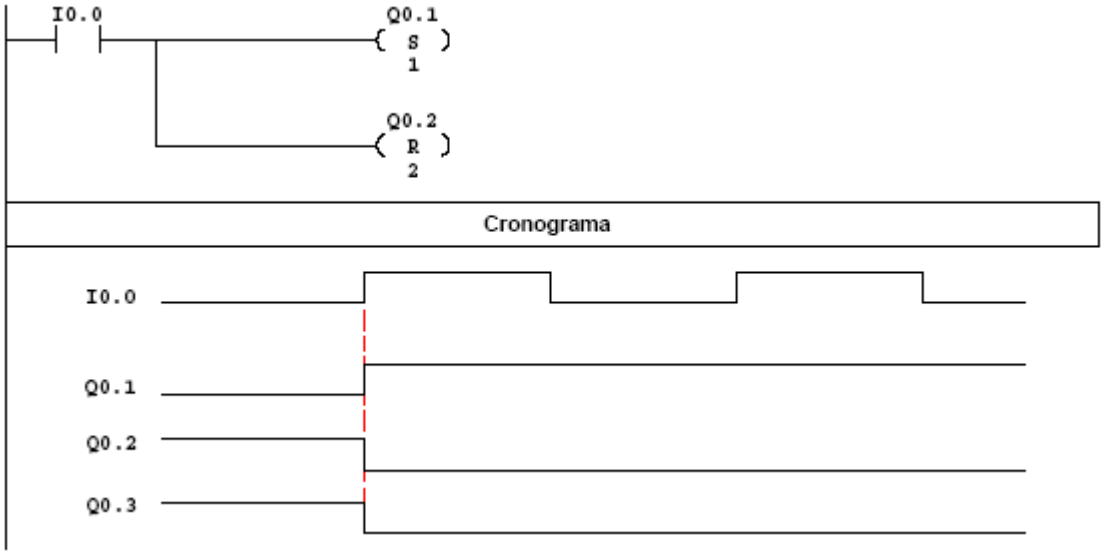
2.4.- Poner a 1, poner a 0 (N bits)



Entradas/salidas	Operandos	Tipos de datos
Bit	I, Q, M, SM, T, C, V, S, L	BOOL
N	VB, IB, QB, MB, SMB, SB, LB, AC, constante, *VD, *AC, *LD	BYTE

Cuando se ejecutan las operaciones **poner a 1** y **poner a 0**, se activa (se pone a 1) o se desactiva (se pone a 0) el número indicado de salidas (N) a partir del valor indicado por el bit o por el parámetro OUT.

El margen de entradas y/o salidas que se pueden activar o desactivar está comprendido entre 1 y 255. Con la operación poner a 0, si el bit indicado es un bit T (bit de temporización) o un bit C (bit de contaje), se desactivará el bit de temporización/contaje y se borrará el valor actual del temporizador/contador.



Consideraciones:

- Se utilizan con frecuencia para mantener permanentemente activadas o desactivadas entradas, salidas o marcas cuando se active brevemente (por impulso) o un contacto antepuesto.
- Una salida o marca “puesta a 1” permanece en ese estado hasta que sea borrada por la

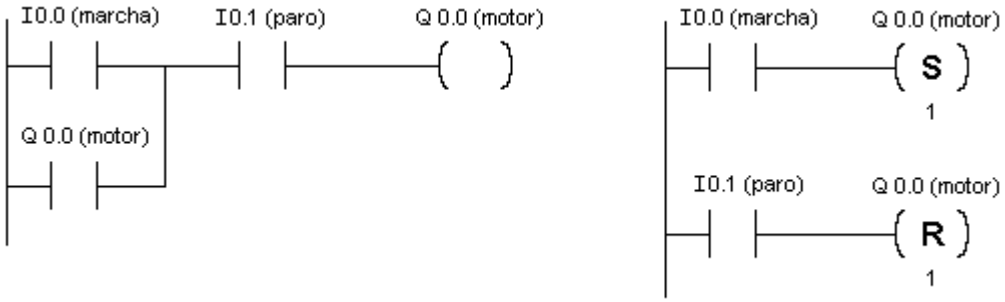
instrucción - (R).

- Si en la bobina de poner a 1 y en su bobina asociada de poner a 0 de una salida se aplica la señal "1", tiene prioridad la operación que está después en el programa.
- No aguantan el paso de Run a Stop y viceversa, es decir, no permanecen grabadas.

2.4.1.- Ejemplo "enclavamiento"

Hasta este momento, habíamos considerado las entradas como interruptores, es decir, la salida permanece activada mientras la entrada esté cerrada (1 o nivel alto), pero qué ocurre cuando utilizamos pulsadores y queremos que la salida quede "activada permanentemente".

En estos casos, que representan la mayoría de las ocasiones, deberemos enclavar la salida o recurrir a la opción SET.



No debemos olvidar que todo Set lleva asociado un Reset, a no ser que queramos mantener activada la salida siempre.

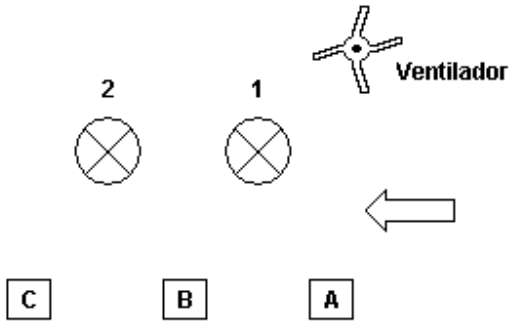
2.4.2.- Ejercicio "cruzamiento"

Diseña un programa que simule el funcionamiento del esquema eléctrico de un cruzamiento, es decir, encendido de una lámpara desde tres interruptores.

2.4.3.- Ejercicio "telerruptor"

Diseña un programa que realice la misma función que el programa del punto anterior, pero considerando las entradas como pulsadores.

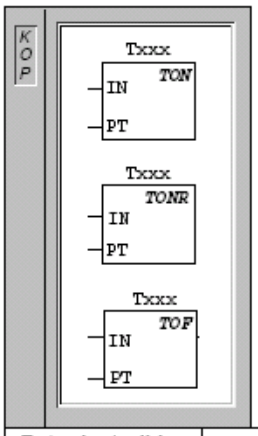
2.4.4.- Ejercicio "pasillo automatizado"



El sentido del pasillo es el marcado por la flecha. Cuando la fotocélula A detecta una presencia, enciende la bombilla 1 y el ventilador. Cuando la fotocélula B detecte presencia. Apagará la bombilla 1 y encenderá la bombilla 2. Finalmente la fotocélula C apagará todo el sistema.

Este proceso sólo se iniciará con un pulsador de marcha y se desconectará con un pulsador de paro.

3.- Operaciones de temporalización

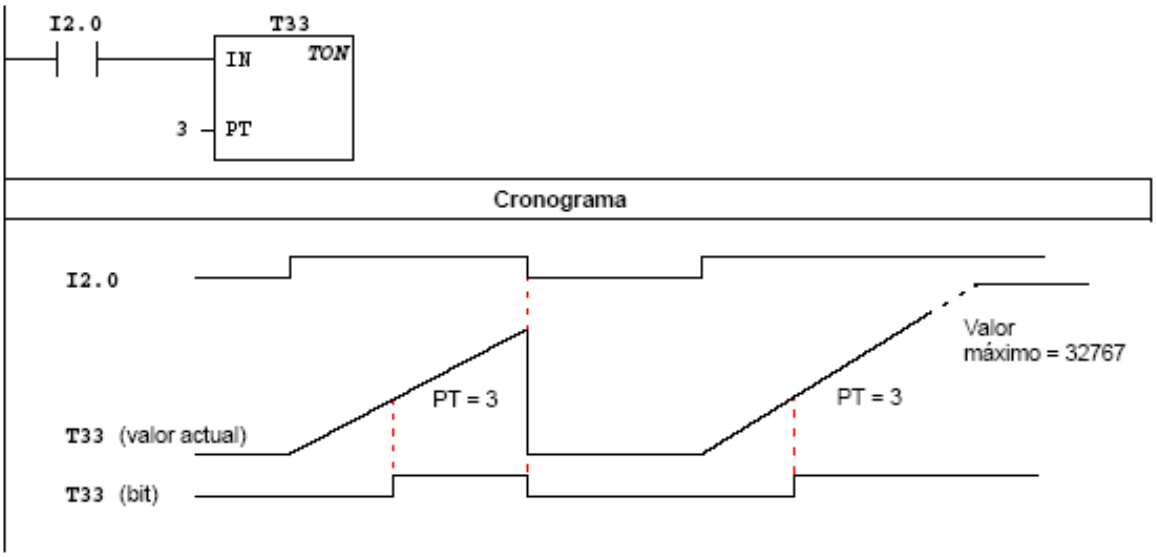


Entradas/salidas	Operandos	Tipos de datos
Txxx	Constante	WORD
IN	I, Q, M, SM, T, C, V, S, L, circulación de corriente	BOOL
PT	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, constante, *VD, *AC, *LD	INT

Dentro de la temporización hemos de diferenciar entre tres tipos de "relojes":

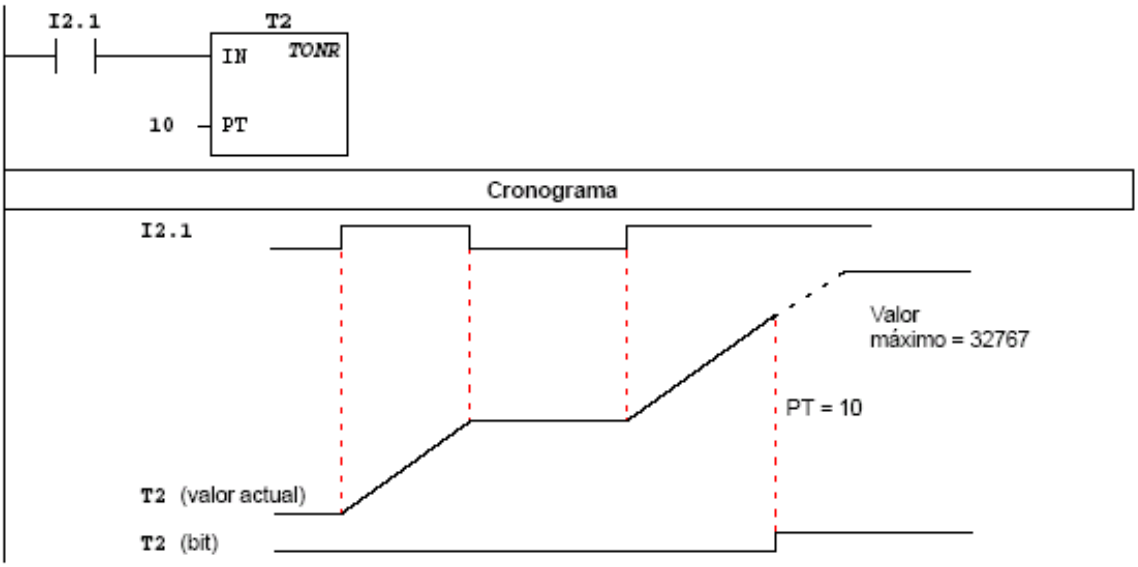
- **Temporizador de retardo a la conexión (TON).**
- **Temporizador de retardo a la conexión memorizado (TONR).**
- **Temporizador de retardo a la desconexión (TOF).**

Las operaciones **temporizador de retardo a la conexión** y **temporizador de retardo a la conexión memorizado** cuentan el tiempo al estar activada (ON) la entrada de habilitación. Si el valor actual (Txxx) es mayor o igual al valor de preselección (PT), se activa el bit de temporización (bit T). Cuando la entrada de habilitación está desconectada (OFF), el valor actual se borra en el caso del temporizador de retardo a la conexión. En cambio, se conserva en el temporizador de retardo a la conexión memorizado. Éste último sirve para acumular varios períodos de tiempo de la entrada en ON. Para borrar el valor actual del temporizador de retardo a la conexión memorizado se utiliza la operación poner a 0 (Reset).



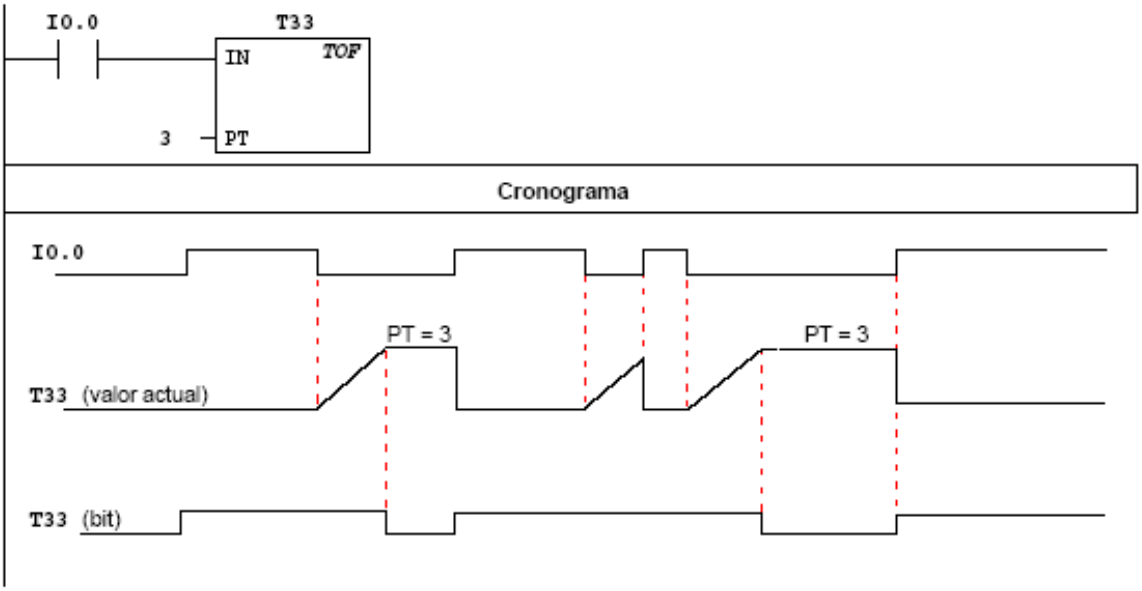
Tanto el temporizador de retardo a la conexión como el temporizador de retardo a la conexión

memorizado continúan contando tras haberse alcanzado el valor de preselección y paran de contar al alcanzar el valor máximo de 32767.



El **temporizador de retardo a la desconexión** se utiliza para retardar la puesta a 0 (OFF) de una salida durante un período determinado tras haberse desactivado (OFF) una entrada. Cuando la entrada de habilitación se activa (ON), el bit de temporización se activa (ON) inmediatamente y el valor actual se pone a 0. Cuando la entrada se desactiva (OFF), el temporizador cuenta hasta que el tiempo transcurrido alcance el valor de preselección. Una vez alcanzado éste, el bit de temporización se desactiva (OFF) y el valor actual detiene el contaje. Si la entrada está desactivada (OFF) durante un tiempo inferior al valor de preselección, el bit de temporización permanece activado (ON). Para que la operación TOF comience a contar se debe producir un cambio de ON a OFF.

Si un temporizador TOF se encuentra dentro de una sección SCR y ésta se encuentra desactivada, el valor actual se pone a 0, el bit de temporización se desactiva (OFF) y el valor actual no cuenta.



Estos temporizadores tienen tres resoluciones. La resolución viene determinada por el número del temporizador:

Tipo temporizador	Resolución	Valor máximo	N.º temporizador
TONR	1 ms	32'767 s (0'546 min)	T0, T64
	10 ms	327'67 s (0'546 min)	T1 a T4, T65 a T68
	100 ms	3276'7 s (0'546 min)	T5 a T31, T69 a T95
TON, TOF	1 ms	32'767 s (0'546 min)	T32, T96
	10 ms	327'67 s (0'546 min)	T33 a T36, T97 a T100
	100 ms	3276'7 s (0'546 min)	T37 a T63, T101 a T255

El valor actual resulta del valor de contaje multiplicado por la base de tiempo. Por ejemplo, el valor de contaje 50 en un temporizador de 10 ms equivale a 500 ms.

No se pueden compartir números iguales para los temporizadores TOF y TON. Por ejemplo, no puede haber tanto un TON T32 como un TOF T32.

3.1.- Ejercicio "base de tiempos"

Utilizando tres resoluciones distintas, elabora tres temporizadores de 5 segundos para cada tipo de temporizador.

3.2.- Ejercicio "coche fantástico"

Realizar, utilizando los bits de la entrada I 0., la secuencia de encendido y apagado de leds del coche de la popular serie de televisión "El coche fantástico". La temporalización entre bit y bit ha de ser de 1 segundo.

3.3.- Ejercicio "intermitente"

Realizar un programa que simule el funcionamiento de un intermitente.

3.4.- Ejercicio "inversor de giro"

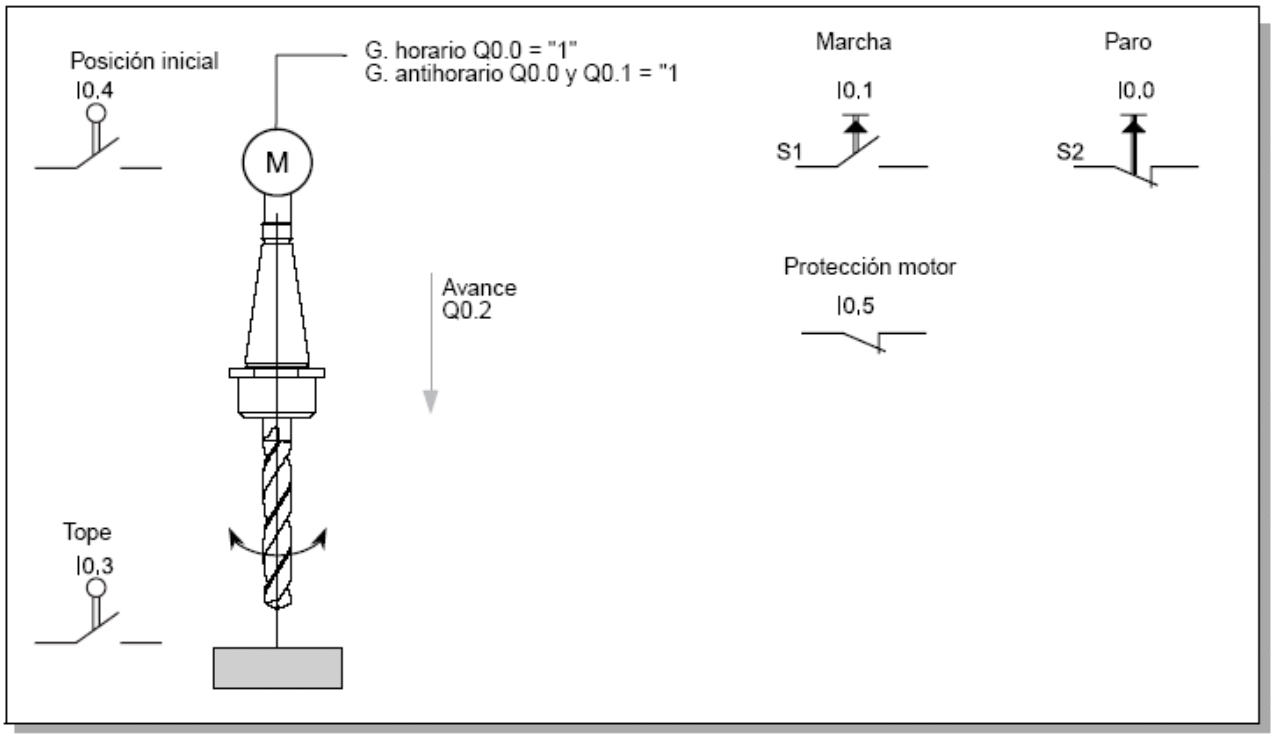
Elaborar el esquema de fuerza y de mando de una inversión de giro.

A continuación elabora su aplicación con un autómata programable (programa y conexionado del PLC).

Entradas	I 0.0	Relé térmico
	I 0.1	Pulsador de paro
	I 0.2	Pulsador de marcha I
	I 0.3	Pulsador de marcha II
Salidas	Q 0.0	Contactor giro I
	Q 0.1	Contactor giro II
	Q 0.3	Luz intermitente giro motor

Simula el funcionamiento de una puerta de garaje.

3.5.- Ejercicio "taladro"

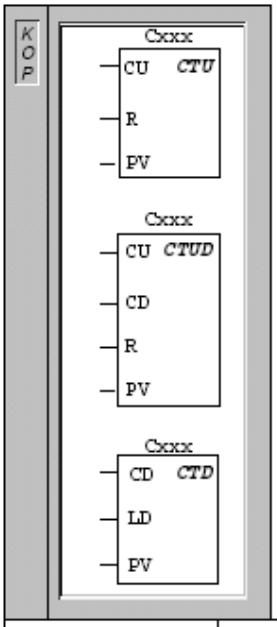


Con S1 se pone en marcha con giro horario el motor de una taladradora automática. Tras 3 segundos se conecta el avance.

Cuando se alcanza el tope en I 0.3, se desconecta el avance. Un resorte lleva la máquina a la posición inicial. Para ello el accionamiento gira en sentido antihorario (Q 0.0 y Q 0.1 están a "1").

Una vez alcanzada la posición inicial I 0.4 = "1", el accionamiento sigue funcionando otro segundo hasta que se desconecta la máquina. Con paro es siempre posible desconectar la máquina (se activa con I 0.0 = "0").

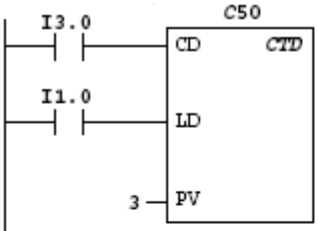
4.- Operaciones con contadores

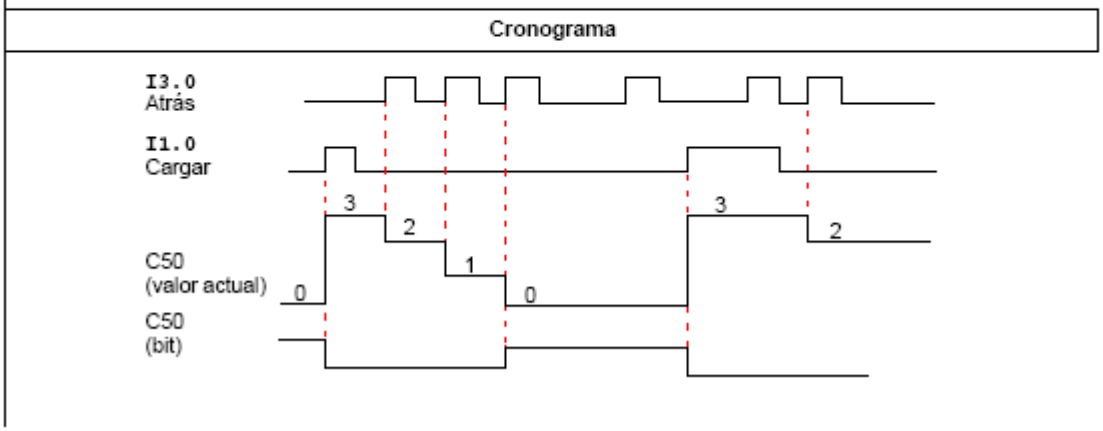


Entradas/salidas	Operandos	Tipos de datos
Cxxx	Constante	WORD
CU, CD, R, LD	I, Q, M, SM, T, C, V, S, L, circulación de corriente	BOOL
PV	VW, IW, QW, MW, SMW, LW, SW, AIW, AC, T, C, constante, *VD, *AC, *LD	INT

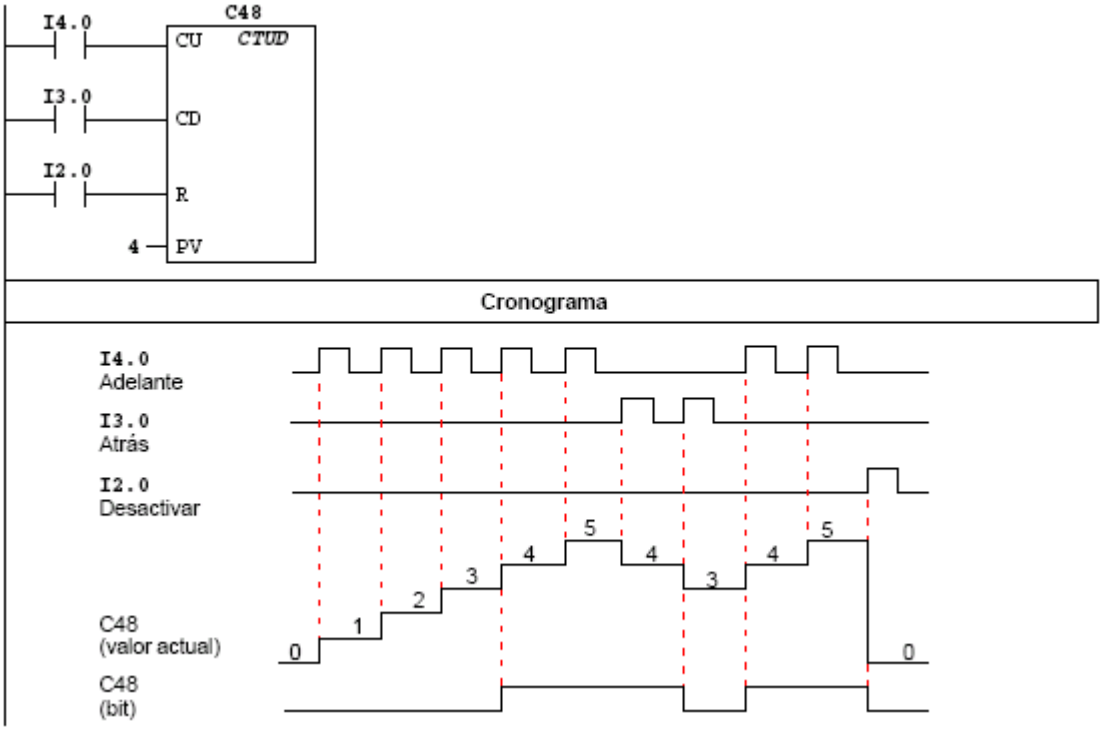
Dentro de los contadores, encontramos 3 tipos:

- **Contar adelante** (**CTU**). Empieza a contar hasta el valor máximo cuando se produce un flanco positivo en la entrada de contaje adelante (CU). Si el valor actual (Cxxx) es mayor o igual al valor de preselección (PV), se activa el bit de contaje (Cxxx). El contador se inicializa al activarse la entrada de desactivación (R) y para de contar cuando alcanza PV.





- **Contar adelante/atrás (CTUD)**. Empieza a contar adelante cuando se produce un flanco positivo en la entrada de contaje adelante (CU). Por el contrario, empieza a contar atrás cuando se produce un flanco positivo en la entrada de contaje atrás (CD). Si el valor actual (Cxxx) es mayor o igual al valor de preselección (PV), se activa el bit de contaje (Cxxx). El contador se inicializa al activarse la entrada de desactivación (R).
  El contador adelante/atrás acepta valores negativos.



- **Contar atrás (CTD)**. Empieza a contar atrás desde el valor de preselección cuando se produce un flanco positivo en la entrada de contaje atrás (CD). Si el valor actual es igual a cero, se activa el bit de contaje (Cxxx). El contador desactiva el bit de contaje (Cxxx) y carga el valor actual con el valor de preselección (PV) cuando se activa la entrada de carga (LD).
  El contador atrás se detiene al alcanzar el valor cero.

Los márgenes de contaje para todos van desde Cxxx = C0 hasta C255.

Puesto que cada contador dispone sólo de un valor actual, no se podrá asignar un mismo número a varios contadores (los contadores adelante, adelante/atrás y atrás acceden a un mismo valor actual).

4.1.- Ejercicio "impulsos"

Realizar un programa que: después de 5 impulsos de la entrada I 0.0 active Q 0.0. Tras 3 impulsos de I 0.0 (estando activado Q 0.0) desactive Q 0.0 y active Q 0.1. Pasados 5 impulsos, estando activado Q 0.1, se desactive Q 0.1 y active Q 0.0... y así sucesivamente.

4.2.- Ejercicio "control de acceso"

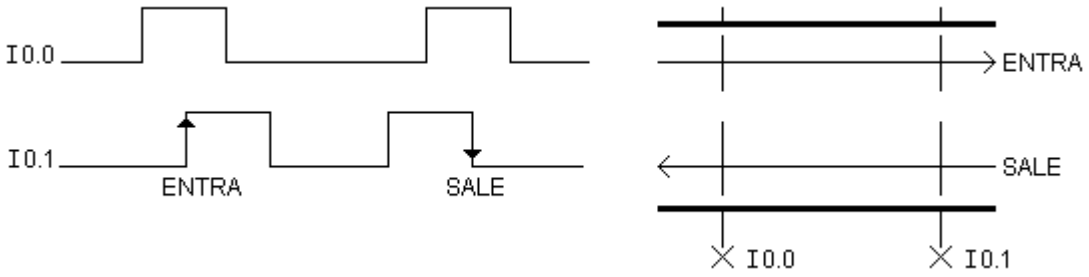
Un recinto tiene un límite de personas que entran y salen por la misma puerta.

Para el control de acceso se dispone de dos barreras fotoeléctricas conectadas a las entradas I 0.0 e I 0.1, tal como se muestra en la figura. Cuando se supera el número de personas en el interior se activa la correspondiente señalización por medio de las salida Q 0.0.

Se detecta que una persona entra cuando la entrada I 0.0 está a nivel alto y se da un flanco ascendente en I 0.1.

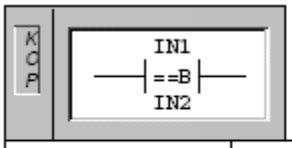
Se detecta que una persona sale cuando la entrada I 0.0 está a nivel alto y se da un flanco descendente en I 0.1.

el control de número de personas se cuenta por medio de un contador ascendente/descendente. Se dispone además de la entrada I 0.2 para resetear en cualquier momento el contador.



5.- Operaciones de comparación

5.1.- Comparar byte



Entradas/salidas	Operandos	Tipos de datos
Entradas	IB, QB, MB, SMB, VB, SB, LB, AC, constante, *VD, *AC,*LD	BYTE

La operación **comparar byte** se utiliza para comparar dos valores: **IN1** e **IN2**. Las comparaciones incluyen:

- IN1 = IN2
- IN1 >= IN2
- IN1 <= IN2
- IN1 > IN2
- IN1 < IN2
- IN1 <> IN2

Las comparaciones de bytes no llevan signo.

El contacto se activa si la comparación es verdadera.

5.1.1.- Ejercicio “potenciómetro analógico”

Realiza un programa que active la salida Q 0.0 cuando los dos potenciómetros analógicos de que dispone el PLC tengan el mismo valor; active Q 0.1 cuando uno de ellos sea mayor o igual a 100; y active Q 0.2 mientras el otro se menor a 70.

5.1.2.- Ejercicio “regular la temperatura de una habitación con un calefactor eléctrico”

Consideraremos la sonda a través de la cual se obtiene la temperatura el potenciómetro 1. Mientras que la temperatura de la habitación la marcará el potenciómetro 2.

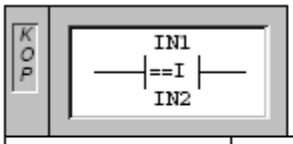
Existen 2 interruptores de control: el primero es para activar / desactivar el sistema. El segundo controla el modo de control (regulación / ventilación):

- *Modo regulación*: si la temperatura actual <= a 3ºC, arranca el ventilador. Si la temperatura actual >= 2ºC, para el calefactor.
- *Modo ventilación*: arranca el ventilador en caso de estar en modo ventilación.

Entradas / salidas:

- I 0.0: selector ON / OFF
- I 0.1: selector modo (0 regulación, 1 ventilación).
- Q 0.0: calefactor.
- Q 0.1: ventilador

5.2.- Comparar entero



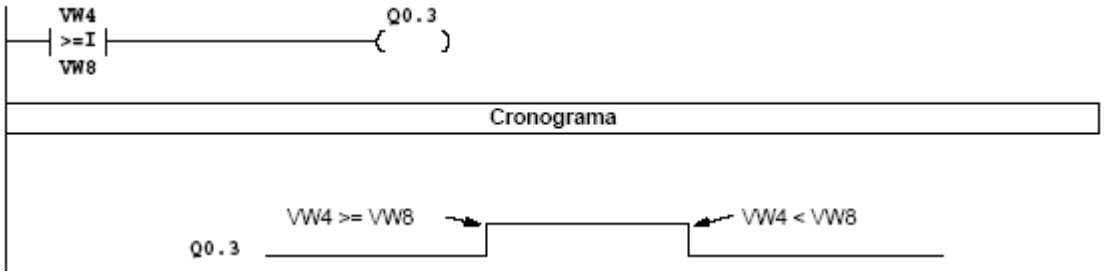
Entradas/salidas	Operandos	Tipos de datos
Entradas	IW, QW, MW, SW, SMW, T, C, VW, LW, AIW, AC, constante, *VD, *AC,*LD	INT

La operación **comparar entero** se utiliza para comparar dos valores: **IN1** e **IN2**. Las comparaciones incluyen:

- IN1 = IN2
- IN1 >= IN2
- IN1 <= IN2
- IN1 > IN2
- IN1 < IN2
- IN1 <> IN2

Las comparaciones de enteros llevan signo (16#7FFF > 16#8000).

El contacto se activa si la comparación es verdadera.



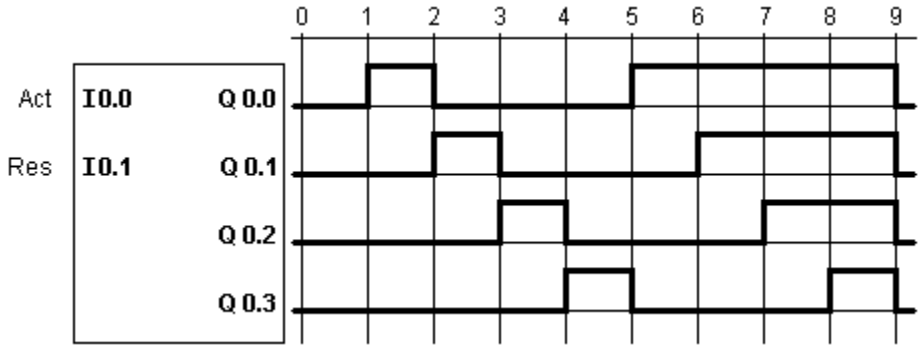
5.2.1.- Ejercicio "programador cíclico"

Al activar la entrada I 0.0 a nivel alto se desarrolla la secuencia especificada en la figura.

Si la señal de entrada I 0.0 pasa a nivel bajo la secuencia se detiene, pudiéndose continuar en el punto de partida al volver al nivel alto.

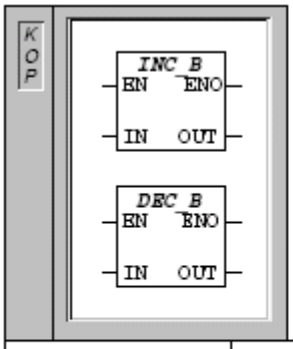
Si la señal de Reset está a nivel alto se desactivarán todas las salidas. La secuencia se repetirá una vez finalizada de forma cíclica.

Para modificar el tiempo de la secuencia, basta con modificar la fase de tiempos del temporizador.



6.- Operaciones aritméticas con enteros

6.1.- Incrementar y decrementar byte



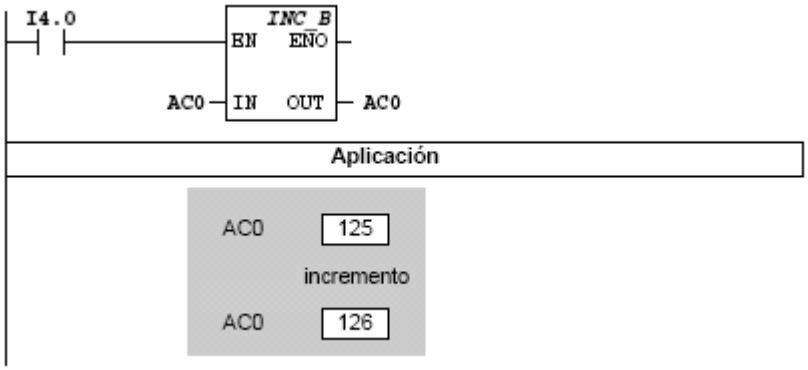
Entradas/salidas	Operandos	Tipos de datos
IN	VB, IB, QB, MB, SB, SMB, LB, AC, constante, *VD, *AC, *LD	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE

Las operaciones **incrementar byte** y **decrementar byte** suman/restan 1 al byte de entrada (IN) y depositan el resultado en la variable indicada por OUT.

Su forma de operar es la siguiente:

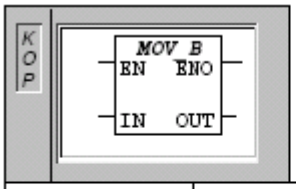
IN + 1 = OUT
IN - 1 = OUT

Estas operaciones no llevan signo.



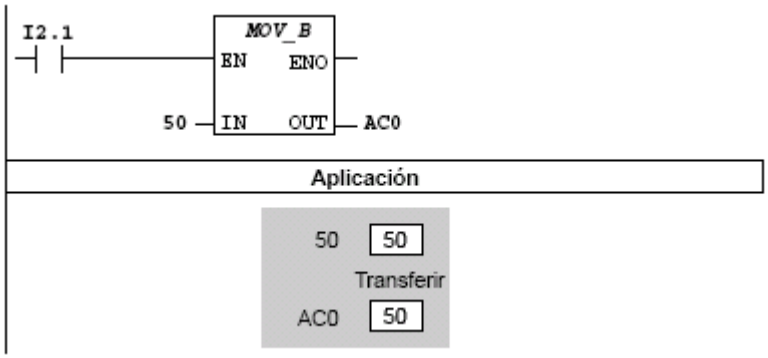
7.- Operaciones de transferencia

7.1.- Transferir byte



Transferir...	Entradas/salidas	Operandos	Tipos de datos
Byte	IN	VB, IB, QB, MB, SB, SMB, LB, AC, constante, *VD, *AC, *LD	BYTE
	OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE

La operación **transferir byte** transfiere el byte de entrada (IN) al byte de salida (OUT). El byte de entrada permanece inalterado.



7.1.1.- Ejercicio "contador"

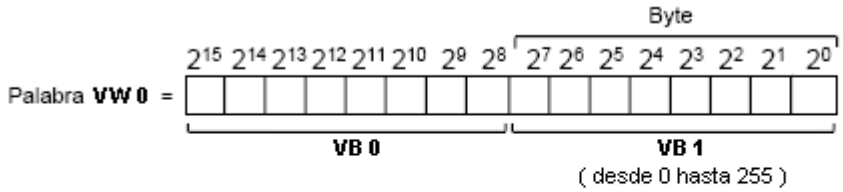
Realizar un contador CTUD sin utilizar la operación propiamente dicha.

Elaborar el ejercicio "impulsos" sin utilizar contadores.

7.1.2.- Ejercicio "intermitente variable"

Una señal intermitente se regula a través del potenciómetro analógico integrado en el PLC.

El valor del potenciómetro se carga en el byte variable VB1, ya que es la parte baja de la palabra VW0.



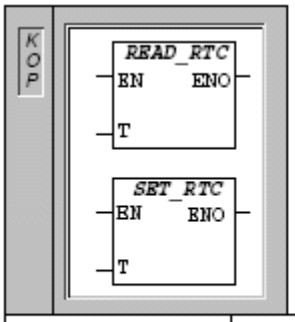
256 ⟶ 0.0
257 ⟶ 0.0 y 1.0
258 ⟶ 0.0 y 1.1
259 ⟶ 0.0 , 1.0 y 1.1

512 ⟶ 0.1
513 ⟶ 0.1 y 1.0
514 ⟶ 0.1 y 1.1
515 ⟶ 0.1 , 1.0 y 1.1

768 ⟶ 0.0 y 0.1
769 ⟶ 0.0 , 0.1 y 1.0
770 ⟶ 0.0 , 0.1 y 1.1
771 ⟶ 0.0 , 0.1, 1.0 y 1.1

La palabra VW 0 será el tiempo de preselección en los temporizadores.

8.- Operaciones de reloj



Entradas/salidas	Operandos	Tipos de datos
T	VB, IB, QB, MB, SMB, SB, LB, *VD, *AC, *LD	BYTE

La operación **leer reloj de tiempo real** lee la hora y fecha actuales del reloj y carga ambas en un búfer de 8 bytes (que comienza en la dirección T).

La operación **ajustar reloj de tiempo real** escribe en el reloj la hora y fecha actuales que están cargadas en un búfer de 8 bytes (que comienza en la dirección T).

T	T+1	T+2	T+3	T+4	T+5	T+6	T+7
Año	Mes	Día	Hora	Minuto	Segundo	0	Día de la semana

El reloj de tiempo real se inicializa con la siguiente fecha y hora tras un corte de alimentación prolongado o una pérdida de memoria:

- **Fecha**: 01-Ene-90
- **Hora**: 00:00:00
- **Día de la semana**: Domingo

El reloj de tiempo real de la CPU S7-200 utiliza sólo los dos dígitos menos significativos para representar el año. Por tanto, el año 2000 se representa como ”00”.

Todos los valores de la fecha y la hora se deben codificar en BCD (p. ej., 16#97 para el año 1997). Utilice los siguientes formatos de datos:

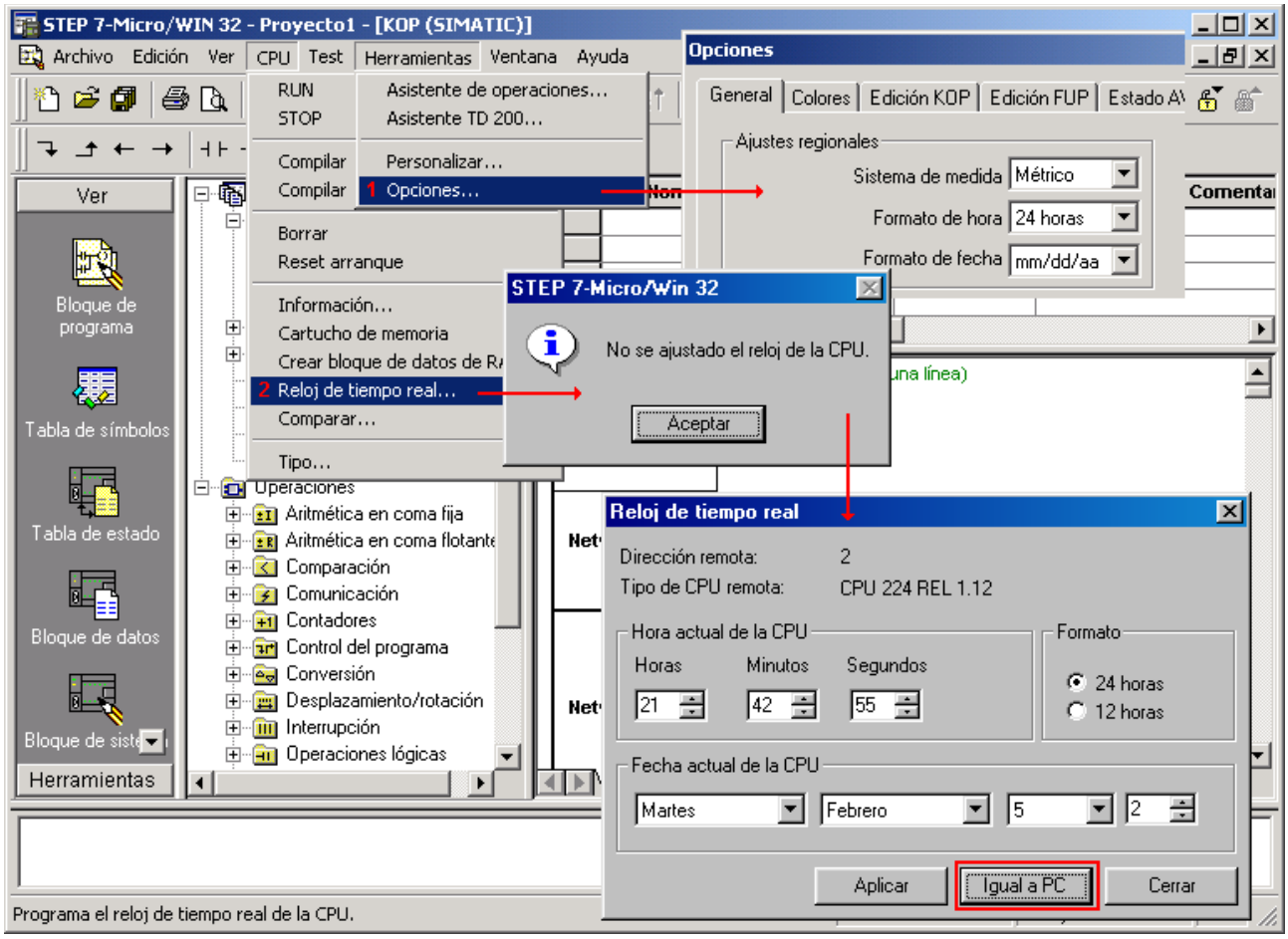
Año/Mes	aamm	aa - 0 a 99	mm - 1 a 12
Día/Hora	ddhh	dd - 1 a 31	hh - 0 a 23
Minutos/Segundos	mmss	mm - 0 a 59	ss - 0 a 59
Día de la semana	d	d - 0 a 7	1 = domingo
			0 = desactiva el día de la semana (permanece 0)

La CPU S7-200 no comprueba si el día de la semana coincide con la fecha. Así puede ocurrir que se acepten fechas no válidas, p. ej. el 30 de febrero. Asegúrese de que los datos introducidos sean correctos.

El sistema de automatización S7-200 no utiliza la información relativa al año de ninguna forma y no es afectado por el cambio de siglo (en el año 2000). No obstante, si en los programas de usuario se utilizan operaciones aritméticas o de comparación con el valor del año, se deberá tener en cuenta la representación de dos dígitos y el cambio de siglo.

Los años bisiestos se tratan correctamente hasta el año 2096.

Si queremos ajustar el reloj desde el MicroWin, deberemos seguir los pasos representados en la figura siguiente:



8.1.- Ejercicio "reloj"

Elabora un programa que active Q 0.0 durante las próximas fiestas de San Jorge.

Deberá comenzar el día 21 a las 21 horas y terminar el día 24 a las 21 horas y 30 minutos.

Puedes utilizar a partir del byte VB 400.

8.2.- Ejercicio "iluminación interior de escalera y exterior de una cabaña"

Iluminación interior: si se pulsa cualquiera de los dos botones deben encenderse las luces de la escalera durante 2 minutos.

Iluminación exterior: las luces se encienden durante 3 minutos en caso que:

- Interruptor control exterior activado.
- Detector infrarrojos activado.
- Lunes a viernes de 17h a 21h.
- Sábado a domingo de 17h a 23h.

Entradas / salidas:

- I 0.0: botón escaleras en planta inferior.
- I 0.1: botón escaleras en planta superior.
- I 0.2: detector de infrarrojos en exterior.
- I 0.3: interruptor control exterior.