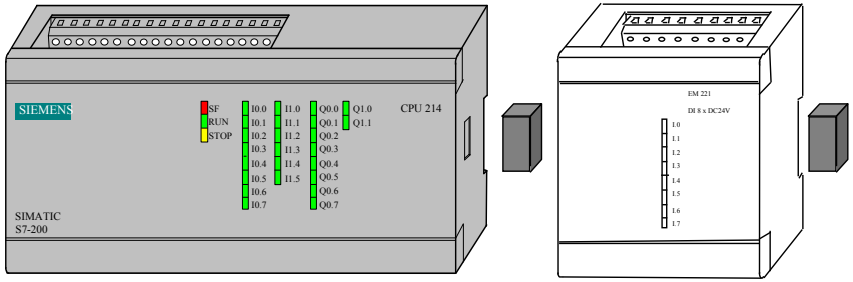
Nociones básicas:

- **Direccionamiento.**
- **Ejecución del programa**
- **Programación**
- **Funciones lógicas.**
- **Función memoria.**
- **Bloques de Función**
- **Resumen.**

Nociones básicas: Ejemplo de direccionamiento.



CPU 214

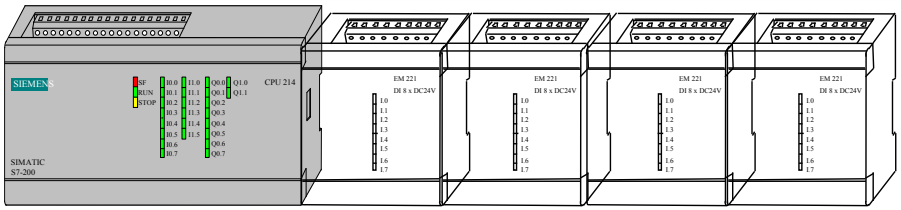
16DI/10DO Integradas

Imagen del proceso de I/Q:

64DI/64DQ

16 AI/16 AQ

Nociones básicas: Ejemplo de direccionamiento.



CPU 224

14 ED/10 SD Integradas

Modulo 4 ED/ 4SD

Modulo 8 ED

Modulo 3 AI/ 1 AQ

Modulo 8 SD

Imagen del proceso de I/Q:

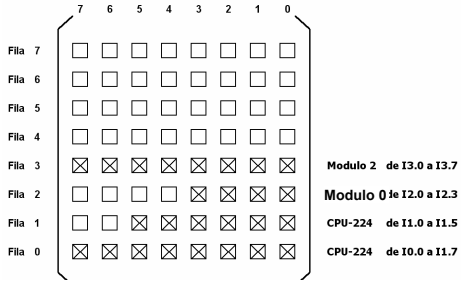
128 ED → de I0.1 a I15.7
128 SD → de Q0.1 a Q15.7
30 AI → de AIW0 a AIW62
30 AQ → de QIW0 a QIW62

	Módulo 0	Módulo 1	Módulo 2	Módulo 3	Módulo 4
CPU-224	4 entradas / 4 salidas	8 entradas	3 AI / 1 AQ	8 salidas	3 AI / 1 AQ

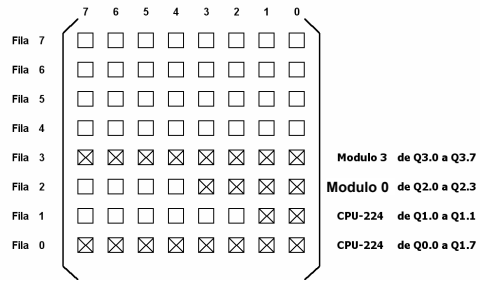
Imagen del proceso de las entradas y salidas asignada a E/S físicas:

I0.0	A0.0	I2.0	Q2.0	I3.0	AIW0	AQW0	Q3.0	AIW8	AQW4
I0.1	A0.1	I2.1	Q2.1	I3.1	AIW2		Q3.1	AIW10	
I0.2	A0.2	I2.2	Q2.2	I3.2	AIW4		Q3.2	AIW12	
I0.3	A0.3	I2.3	Q2.3	I3.3			Q3.3		
I0.4	A0.4			I3.4			Q3.4		
I0.5	A0.5			I3.5			Q3.5		
I0.6	Q0.6			I3.6			Q3.6		
I0.7	Q0.7			I3.7			Q3.7		
I1.0	Q1.0								
I1.1	Q1.1								
I1.2									
I1.3									
I1.4									
I1.5									

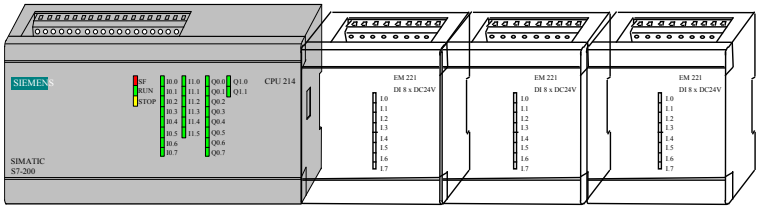
MATRIZ I



MATRIZ Q



Ejercicio de direccionamiento



CPU 224

14 ED/10 SD
Integradas

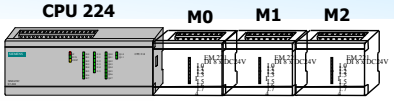
Modulo **Modulo** **Modulo**

8 ED **8 SD** **4 ED/4 SD**

Imagen del proceso de I/Q:

128 ED → de I0.1 a I15.7
128 SD → de Q0.1 a Q15.7
30 AI → de AIW0 a AIW62
30 AQ → de QIW0 a QIW62

Ejercicio de direccionamiento



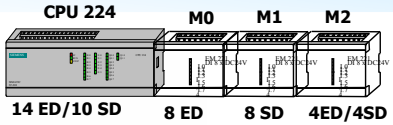
14 ED/10 SD **8 ED** **8 SD** **4ED/4SD**

MATRIZ I

	7	6	5	4	3	2	1	0
Fila 7	☐	☐	☐	☐	☐	☐	☐	☐
Fila 6	☐	☐	☐	☐	☐	☐	☐	☐
Fila 5	☐	☐	☐	☐	☐	☐	☐	☐
Fila 4	☐	☐	☐	☐	☐	☐	☐	☐
Fila 3	☐	☐	☐	☐	☒	☒	☒	☒
Fila 2	☒	☒	☒	☒	☒	☒	☒	☒
Fila 1	☐	☐	☒	☒	☒	☒	☒	☒
Fila 0	☒	☒	☒	☒	☒	☒	☒	☒

Modulo 2 de I3.0 a I3.3
Modulo 0 de I2.0 a I2.7
CPU-224 de I1.0 a I1.5
CPU-224 de I0.0 a I1.7

Ejercicio de direccionamiento



14 ED/10 SD **8 ED** **8 SD** **4ED/4SD**

MATRIZ Q

	7	6	5	4	3	2	1	0
Fila 7	☐	☐	☐	☐	☐	☐	☐	☐
Fila 6	☐	☐	☐	☐	☐	☐	☐	☐
Fila 5	☐	☐	☐	☐	☐	☐	☐	☐
Fila 4	☐	☐	☐	☐	☐	☐	☐	☐
Fila 3	☐	☐	☐	☐	☒	☒	☒	☒
Fila 2	☒	☒	☒	☒	☒	☒	☒	☒
Fila 1	☐	☐	☐	☐	☐	☐	☒	☒
Fila 0	☒	☒	☒	☒	☒	☒	☒	☒

Modulo 1 de Q3.0 a Q3.3
Modulo 0 de Q2.0 a Q2.7
CPU-224 de Q1.0 a Q1.1
CPU-224 de Q0.0 a Q1.7

Ejercicio de direccionamiento

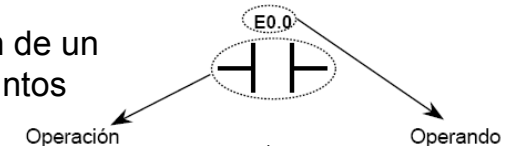
	Módulo 0	Módulo 1	Módulo 3
CPU 224	4 entradas / 4 salidas	8 entradas	8 salidas

Imagen del proceso de las entradas y salidas asignada a E/S físicas:

I0.0	A0.0		I2.0	Q2.0		I3.0	Q3.0
I0.1	A0.1		I2.1	Q2.1		I3.1	Q3.1
I0.2	A0.2		I2.2	Q2.2		I3.2	Q3.2
I0.3	A0.3		I2.3	Q2.3		I3.3	Q3.3
I0.4	A0.4					I3.4	Q3.4
I0.5	A0.5					I3.5	Q3.5
I0.6	Q0.6					I3.6	Q3.6
I0.7	Q0.7					I3.7	Q3.7
I1.0	Q1.0						
I1.1	Q1.1						
I1.2							
I1.3							
I1.4							
I1.5							

Matrices definidas:

• Los operandos de las instrucciones se componen de un dato que puede ser de distintos tipos.



- Los tipos de datos posibles dependerá de la matriz y de la forma de guardar los datos:

I	**entrada**
Q	**salida**
M	**marca**
T	**temporizador**
C	**contador**
SM	**marcas internas**

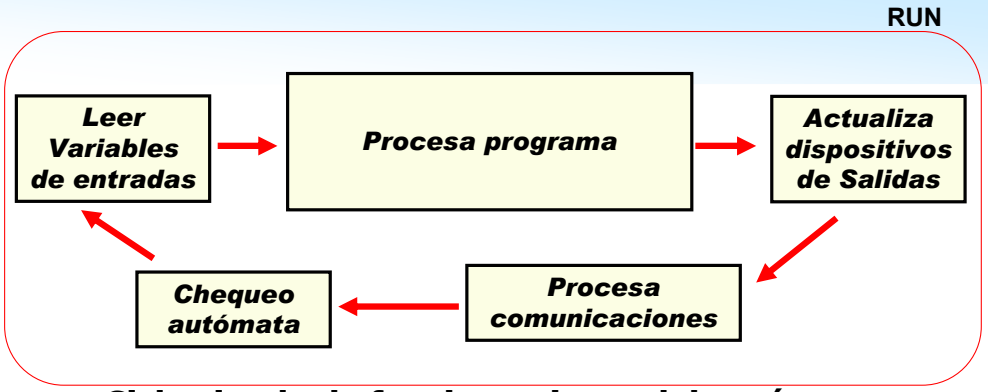
Nociones básicas:

- **Direccionamiento.**
- **Ejecución del programa**
- **Programación**
- **Funciones lógicas.**
- **Función memoria.**
- **Bloques de Función**
- **Resumen.**

Nociones básicas: Ejecución del programa

Resumen de conceptos

- **Ejecución cíclica del programa,**
- **Imagen del proceso:** PAE, PAA.
- **Tiempo de ciclo, Tiempo de respuesta.**
- **Ejecución Lineal**.
- **Ejecución Estructurada:** profundidad de anidamiento, ventajas.



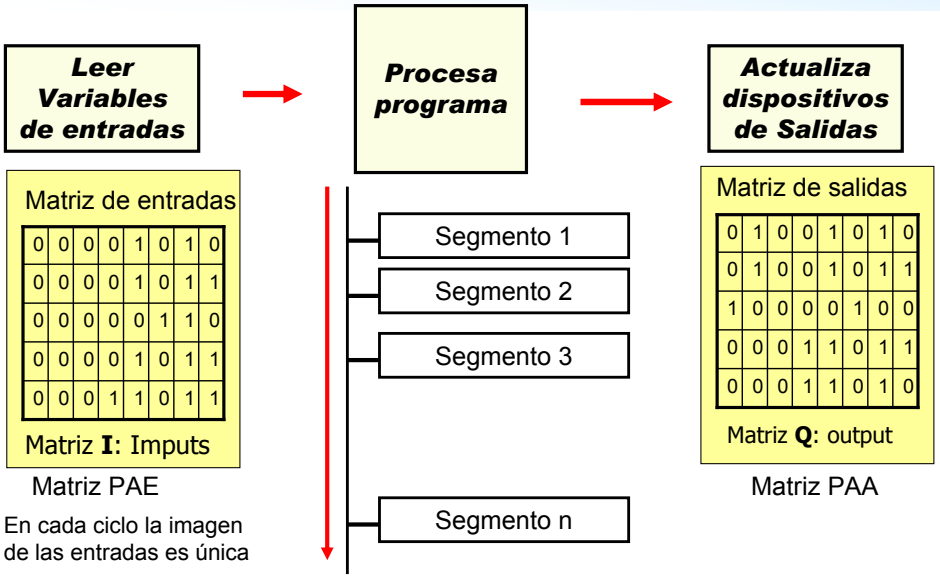
Ciclo simple de funcionamiento del autómata

Ciclo de SCAN

Tiempo de ciclo: tiempo que tarda en ejecutarse el ciclo de SCAN

Velocidad de ejecución/instrucción: tiempo que tarda en ejecutarse una función lógica

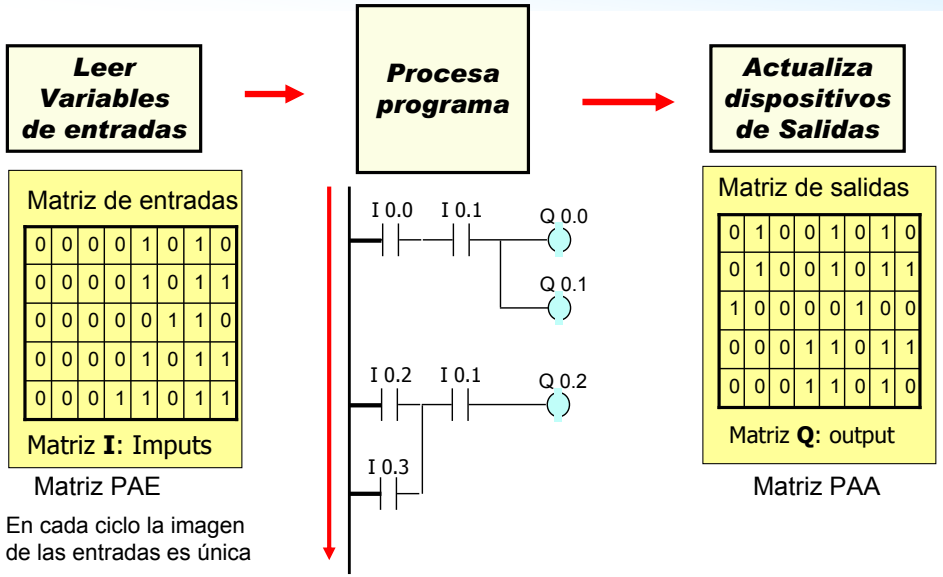
Nociones básicas: Ejecución del programa



En cada ciclo la imagen de las entradas es única

Introducción a la programación

Nociones básicas: Ejecución del programa

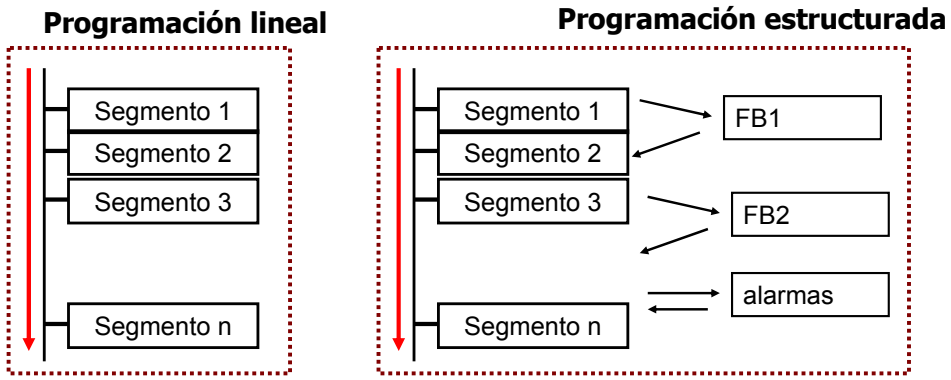


En cada ciclo la imagen de las entradas es única

Introducción a la programación

Nociones básicas: Ejecución del programa

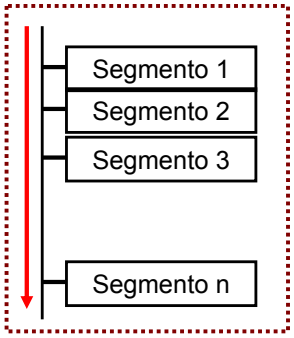
Programación lineal o estructurada



Introducción a la programación

Nociones básicas: Ejecución del programa

Programación lineal o estructurada



Se recomienda cuando los programas son simples.

Se dejan para pequeños autómatas

Nociones básicas: Ejecución del programa

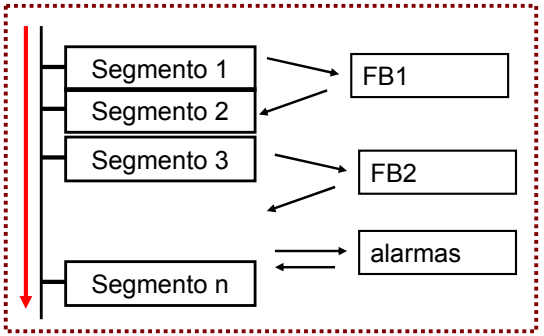
Programación lineal o estructurada

Las funciones complejas de automatización se pueden procesar mejor si se dividen en tareas más pequeñas→ bloques.

La secuencia y el anidamiento de las llamadas de bloques se denomina jerarquía de llamadas.

La profundidad de anidamiento admisible depende del tipo de CPU.

Programación estructurada



Bloque principal

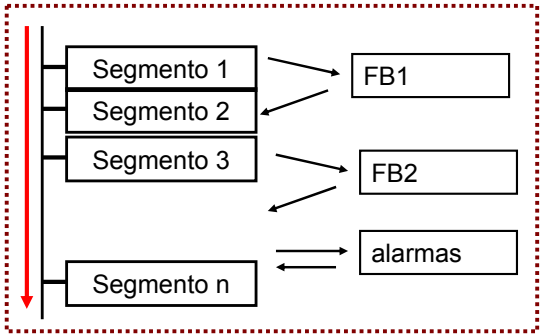
Nociones básicas: Ejecución del programa

Programación lineal o estructurada

Cada bloque (tarea) tiene el aspecto de un programa autónomo y puede ser analizada y programada por separado.

Existen varios tipos de bloques:
- **Bloques de organización.**
- **Los cíclicos.**
- **Los de interrupción.**
- **Los temporizados.**
- **Bloques de función.**
- **etc.**

Programación estructurada



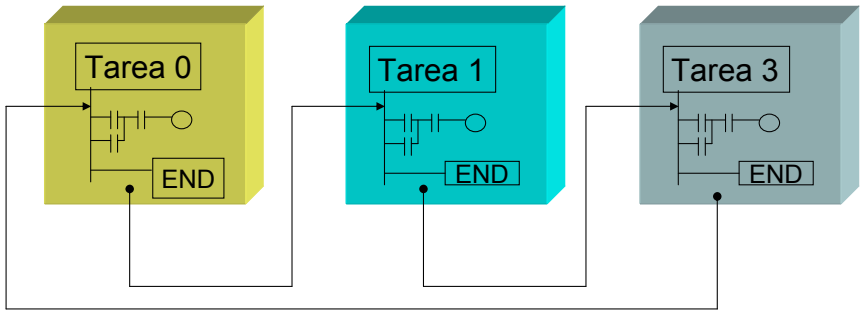
Una función es un bloque lógico "sin memoria".
Los bloques de función son bloques programables.

Nociones básicas: Ejecución del programa

Programación lineal o estructurada

Orden de Ejecución de los bloques Cíclicos. Ejemplo

- Ejemplo de aplicación de cuatro tareas cíclicas (0, 1, 2 y 3) en la que la tarea 2 no está activada. El orden de ejecución de las tareas sería el siguiente:

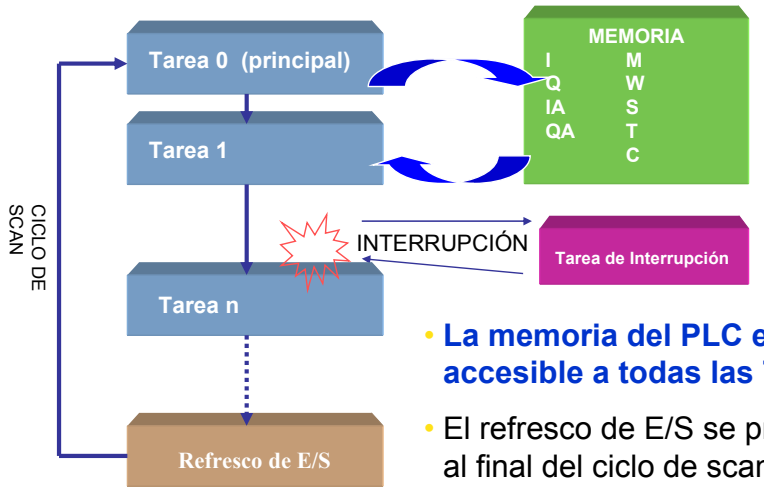


- **En caso de ser activada la tarea 2, ésta se ejecutaría una vez se haya ejecutado la instrucción END de la tarea 1.**

Nociones básicas: Ejecución del programa

Programación lineal o estructurada

Ejecución de los bloques:

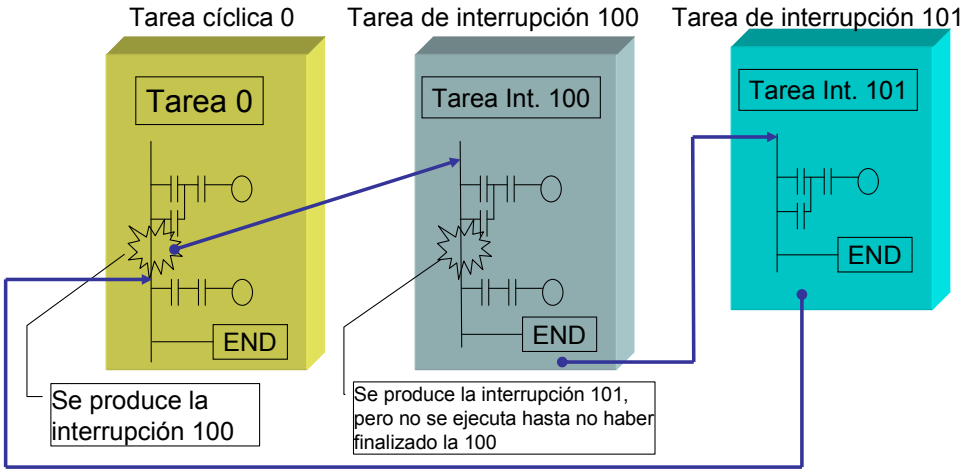


- **La memoria del PLC es accesible a todas las Tareas.**
- El refresco de E/S se produce al final del ciclo de scan.

Nociones básicas: Ejecución del programa

Programación lineal o estructurada

Prioridad de las Tareas (bloques, subrutinas) de Interrupción:



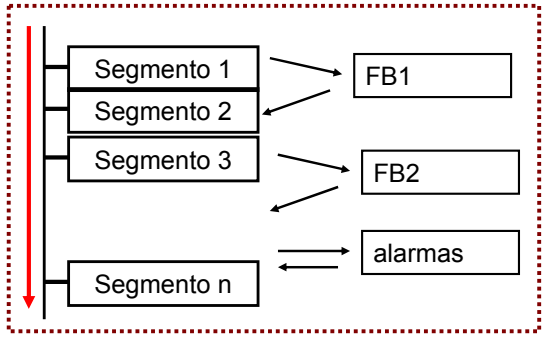
Nociones básicas: Ejecución del programa

Programación lineal o estructurada

Con la estructuración del programa el usuario puede:

- **Realizar programas intensos con gran claridad.**
- **Estandarizar partes del programa.**
- **Modificar programas en poco tiempo.**
- **Comprobar programas por partes.**

Programación estructurada



Bloque principal

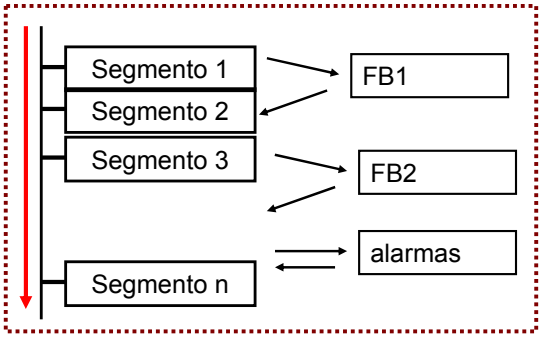
Nociones básicas: Ejecución del programa

Programación lineal o estructurada

La utilización de subrutinas produce los siguiente beneficios:

- Se reduce la longitud total del programa.
- El tiempo de ciclo también se acorta. La subrutina se puede invocar de forma condicional.
- Las subrutinas se pueden transportar fácilmente a otros proyectos.

Programación estructurada



Bloque principal

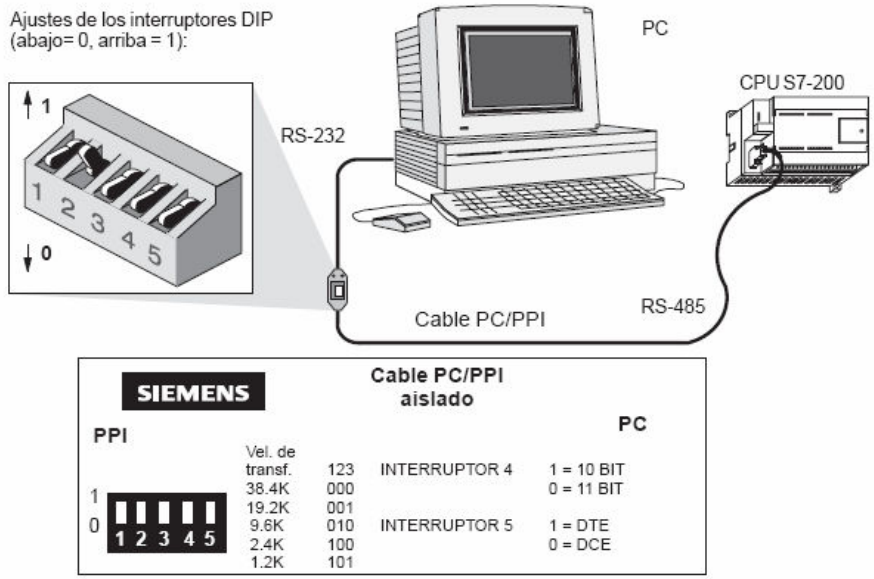
Nociones básicas: Ejecución del programa

Resumen de conceptos

- **Ejecución cíclica del programa,**
- **Imagen del proceso:** PAE, PAA.
- **Tiempo de ciclo, Tiempo de respuesta.**
- **Ejecución Lineal**.
- **Ejecución Estructurada:** profundidad de anidamiento, ventajas.

Nociones básicas:

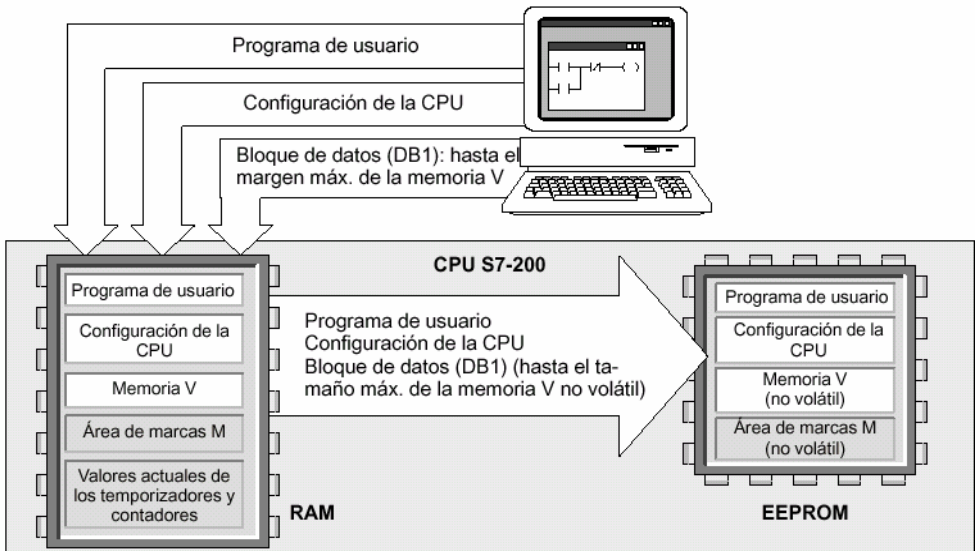
- **Direccionamiento.**
- **Ejecución del programa**
- **Programación**
- **Funciones lógicas.**
- **Función memoria.**
- **Resumen.**



Cargar programas en la CPU y en la PG/PC

- **El programa comprende tres elementos**: el programa de usuario, el bloque de datos (opcional) y la configuración de la CPU (opcional).
- Cargando el programa en la CPU se almacenan dichos elementos en la memoria RAM (de la CPU).
- La CPU también copia automáticamente el programa de usuario, el bloque de datos (DB1) y la configuración de la CPU en la EEPROM no volátil para que se almacenen allí.

Cargar programas en la CPU y en la PG/PC



Cargar programas en la CPU y en la PG/PC

Cualquier CPU tiene dos modos de operación:

- **STOP:** La CPU no ejecuta el programa. Cuando está en modo STOP, es posible cargar programas o configurar la CPU.
- **RUN:** La CPU ejecuta el programa. Cuando está en modo RUN, no es posible cargar programas ni configurar la CPU.

El diodo luminoso (LED) en la parte frontal de la CPU indica el modo de operación actual.
Para poder cargar un programa en la memoria de la CPU es preciso cambiar a modo STOP**.**

SELECTOR

- Si el selector se pone en **STOP**, se detendrá la ejecución del programa.
- Si el selector se pone en **RUN**, se iniciará la ejecución del programa.
- Si el selector se pone en **TERM** (terminal), no cambiará el modo de operación de la CPU. Será posible cambiarlo utilizando el software de programación (STEP 7-Micro/WIN).

Cargar programas en la CPU y en la PG/PC

- En modo STOP, la CPU se encuentra en un estado semiactivo. El programa de usuario no se ejecuta, pero las entradas se actualizan.
- Las condiciones de interrupción se inhiben. Si ocurren interrupciones de comunicación, la CPU recibe los mensajes y ejecuta las peticiones según sea necesario.
- Mientras la CPU está en modo STOP, los cambios de los valores de las E/S se efectúan en la imagen del proceso, con excepción de la función "Forzar" que tiene prioridad sobre los cambios de los valores de E/S en la imagen del proceso.
- Estando la CPU en modo STOP, el usuario puede cargar el programa en la CPU y en la PG/el PC, así como borrar la memoria.

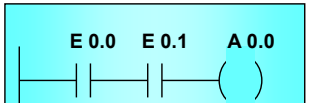
Cargar programas en la CPU y en la PG/PC

- En modo RUN, la CPU lee las entradas, ejecuta el programa, escribe en las salidas, procesa las peticiones de comunicación, actualiza los módulos inteligentes, ejecuta tareas auxiliares internas y gestiona las condiciones de interrupción.
- La CPU no soporta tiempos de ciclos fijos de ejecución en modo RUN.
- Estas acciones (con excepción de las interrupciones de usuario) se gestionan conforme a su prioridad en el orden en que van ocurriendo
- Al principio de cada ciclo se leen los valores actuales de los bits de entrada. Dichos valores se escriben luego en la imagen del proceso de las entradas.
- Los bits de entrada que no tengan una entrada física correspondiente, pero que se encuentren en el mismo byte que otras entradas físicas, se ponen a 0 en la imagen del proceso cada vez que se actualicen las entradas, a menos que se hayan forzado.

Lenguajes de Programación: Seleccion

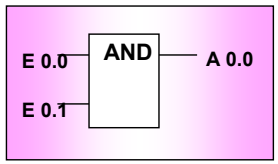
Las CPUs se deben de poder programar en los lenguajes de programación descritos en la norma IEC 61131 y se debe de poder pasar de un lenguaje a otro fácilmente:

LAD (KOP)



El esquema de contactos (KOP) es un lenguaje de programación gráfico con componentes similares a los elementos de un esquema de circuitos.

FUP



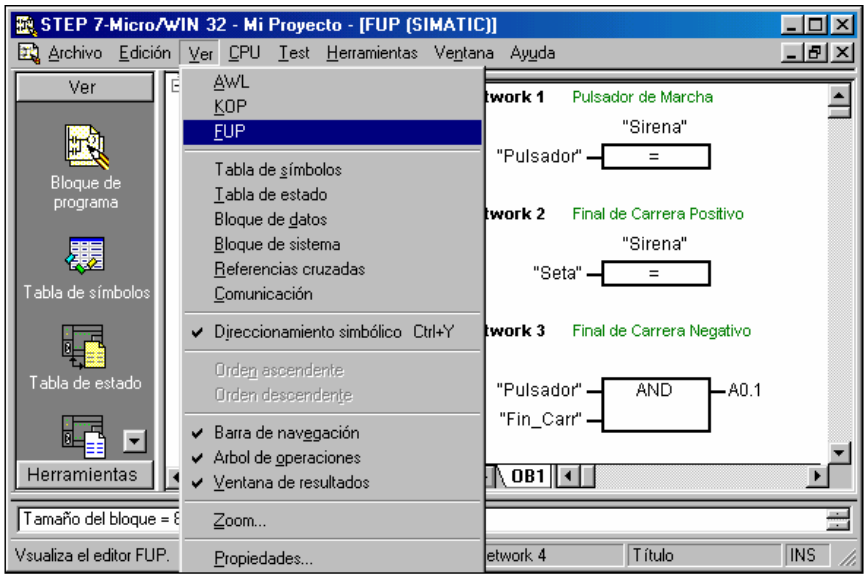
El Esquema de Funciones Lógicas utiliza “cajas” para cada función. El símbolo que se encuentra dentro de la caja indica su función (p.e. & --> operación AND).

AWL

```
LD   E 0.0
A    E 0.1
=    A0.0
```

La lista de instrucciones (AWL) comprende un juego de operaciones nemotécnicas que representan las funciones de la CPU.

Lenguajes de Programación: Seleccion



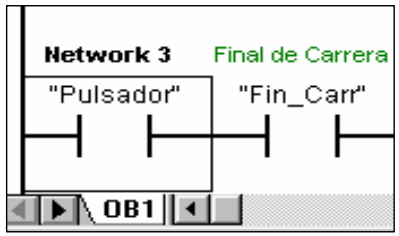
Lenguajes de Programación: Programacion en **LAD** (KOP)

- El esquema de contactos **(LAD)** es un **lenguaje** de programación **gráfico** con componentes similares a los elementos de un esquema de circuitos.
- Al programar con **LAD**, se crean y se disponen componentes gráficos que conforman un segmento de operaciones lógicas.
- **Para crear programas se dispone de los siguientes elementos**:
  - **Contactos** : Representan un interruptor por el que la corriente puede circular.
  - **Bobinas** Representan un relé o una salida excitada por la corriente.
  - **Cuadros** Representan una función (por ejemplo, un temporizador, un contador o una operación aritmética) que se ejecuta cuando la corriente llega al cuadro.

Lenguajes de Programación: Programacion en **LAD** (KOP)

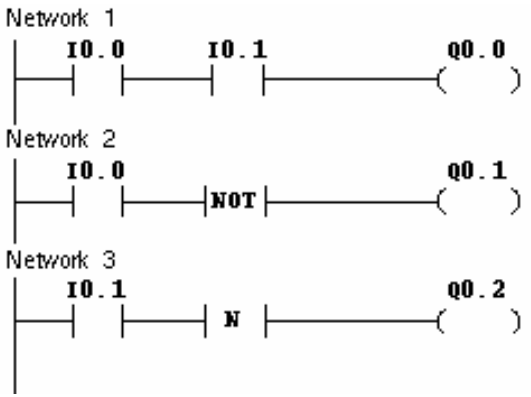
- **Contactos**

- La corriente circula por un contacto normalmente abierto sólo cuando el contacto está cerrado (es decir, cuando su valor lógico es "1").
- De forma similar, la corriente circula por un contacto normalmente cerrado o negado (NOT) sólo cuando dicho contacto está abierto (es decir, cuando su valor lógico es "0").



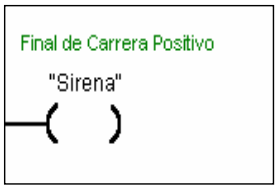
Lenguajes de Programación: Programacion en **LAD** (KOP)

- **Contactos**
- 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.
- Ejemplo



Lenguajes de Programación: Programacion en **LAD** (KOP)

- **Bobinas**
- Cuando se ejecuta la operación Asignar, el bit de salida se activa en la imagen del proceso.



Lenguajes de Programación: Programacion en **LAD** (KOP)

TIPOS de DATOS

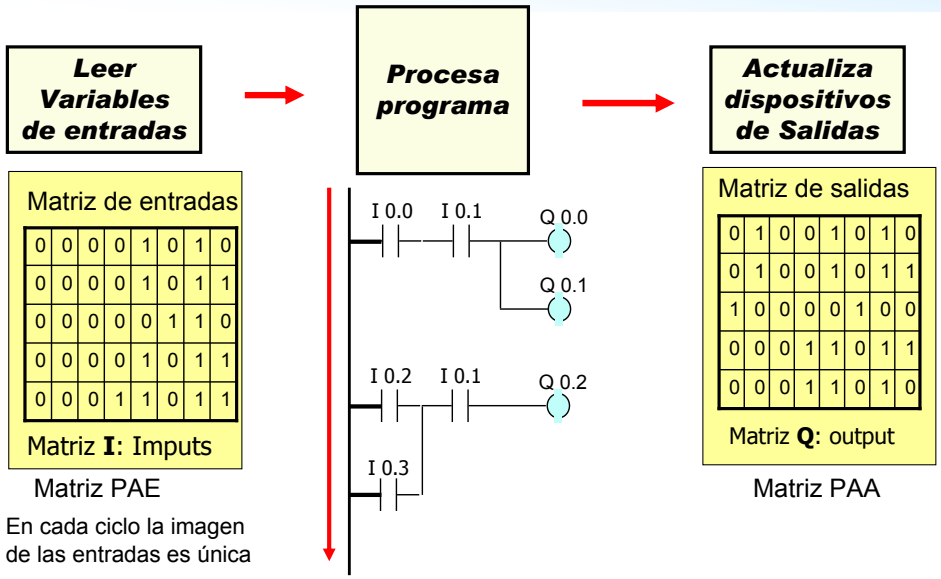
- Los operandos de las instrucciones se componen de un dato que puede ser de distintos tipos.
- Los tipos de datos posibles son:

I	**entrada**
Q	**salida**
M	**marca**
T	**temporizador**
C	**contador**
SM	**marcas internas**

Lenguajes de Programación: Programacion en **LAD** (KOP)

TIPOS de DATOS

- Cada uno de estos tipos se pueden direccionar en 4 posibles modos (salvo T y C):
  - Por defecto: Bit.
  - B: byte (8 bits).
  - W: palabra (16 bits).
  - D: palabra doble (32 bits).



TIPOS de DATOS

Descripción	SIMATIC	Internacional
Esquema de contactos	KOP	LAD
Diagrama de funciones	FUP	FBD
Lista de instrucciones	AWL	STL
Entrada	E	I
Salida	A	Q
Memoria de variables	V	V
Marcas	M	M
Entrada analógica	AE	AI
Salida analógica	AA	AQ
Temporizador	T	T
Contador	Z	C
Contador rápido	HC	HC
SCR	S	S
Marcas especiales	SM	SM
Acumulador	AC	AC
Memoria de variables locales	L	L

TIPOS de DATOS : marcas de memoria

Matriz M

- Cuando realicemos nuestro programa y operemos a nivel de bit en operaciones lógicas (and, or, etc.) puede que nos aparezca la necesidad de almacenar el resultado lógico que tengamos en un determinado momento.
- Para ello se dispones de matrices internas (M) de un numero de filas que depende del automata, que podemos direccionar como:
  - **Marcas → M 0.0 a 255.7**
  - **Byte de marcas → MB 0 a 255**
  - **Palabra de marcas → MW 0 a 254**
  - **Palabra doble de marcas MD 0 a 252**

TIPOS de DATOS: Marcas de Memoria de solo lectura

Matriz SM

- El byte de marcas SMB0 (SM0.0 - SM0.7) contiene ocho bits de estado que proporcionan informaciones sobre el programa de usuario.
- Dichos bits permiten llamar diversas funciones del programa.
- Por ejemplo, SM0.1 está activado (puesto a 1) sólo en el primer ciclo. Con esa marca especial es posible llamar una subrutina de inicialización.

Depende del autómata

TIPOS de DATOS: Marcas de Memoria de solo lectura

Matriz SM

Marcas especiales	Descripción
SM0.0	**Marca Funcionamiento continuo (puesta a '1')**
SM0.1	**Marca de primer ciclo (puesta a '1' en el primer ciclo; después se pone a '0')**
SM0.2	**Datos remanentes perdidos - sólo vale para el primer ciclo ('0' = datos presentes; '1' = datos perdidos)**
SM0.3	**Marca de conexión (se pone a '1' en el primer ciclo tras la conexión; después se pone a '0')**
SM0.4	**Reloj de 60 segundos (cambia: 30 segundos a '0', luego 30 segundos a '1')**
SM0.5	**Reloj de 1 segundo (cambia: 0,5 segundos a 0', luego 0,5 segundos a '1')**
SM0.6	**Reloj de ciclo ('1' lógico en ciclos alternos)**
SM0.7	**Posición del selector de modos de operación ('0' TERM; '1' RUN)**

Depende del autómata

TIPOS de DATOS: Marcas de Memoria de solo lectura

Matriz SM

- Estado del programa
- Estado de las operaciones
- Búfer de recepción de caracteres en modo Freeport
- Error de paridad en modo Freeport
- Desbordamiento de la cola de espera (interrupciones)
- Bits de estado de E/S
- Identificador (ID) de la CPU (SMB 6.0)
- Registro de errores e ID de los módulos de ampliación
- Palabras de estado del tiempo de ciclo
- Valores de los potenciómetros analógicos

Depende del autómata

Introducción a la programación

TIPOS de DATOS: Marcas de Memoria de solo lectura

Matriz SM

- Interface 0 para comunicación Freeport
- Interface 1 para comunicación Freeport
- Escribir en EEPROM
- Intervalos de interrupciones temporizadas
- Bytes de programación de los contadores rápidos
- Bytes de programación de la salida de impulsos
- Interface 0 - recepción de mensajes
- Interface 1 - recepción de mensajes
- Estado del protocolo estándar DP

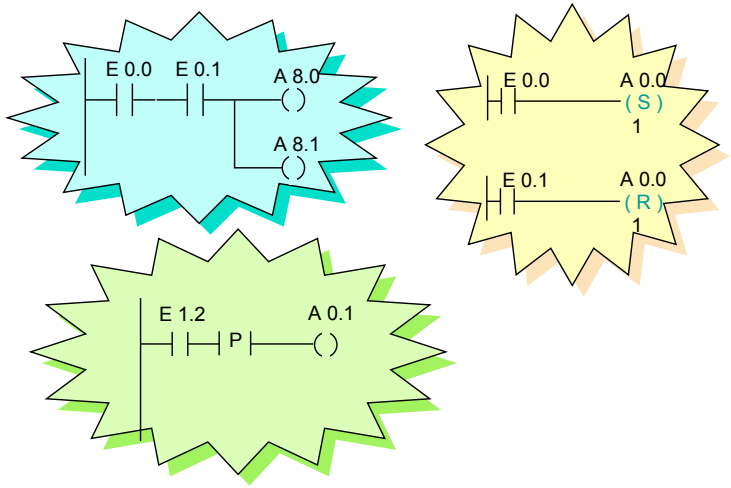
Depende del autómata

Introducción a la programación

Nociones básicas:

- **Direccionamiento.**
- **Ejecución del programa**
- **Programación**
- **Funciones lógicas.**
- **Función memoria.**
- **Bloques de Función**
- **Resumen.**

Operaciones Lógicas con Bits

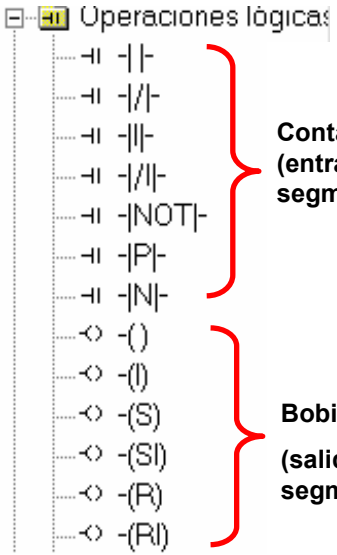


Funciones binarias

AND
OR
XOR
SET
RESET

Forman parte de todos los programas de los autómatas programables. Constituyen las normas para calcular los estados de señal de los actuadores en función de los estados de los sensores.

Operaciones lógicas con bits (en lenguaje LAD)



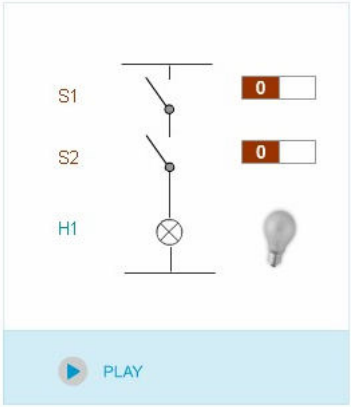
Contactos (entrada al segmento)

Bobinas (salida del segmento)

| | El contacto abierto directo se cierra (se activa) si la entrada física (bit) es 1.

|/| El contacto cerrado directo se cierra (se activa) si la entrada física (bit) es 0.

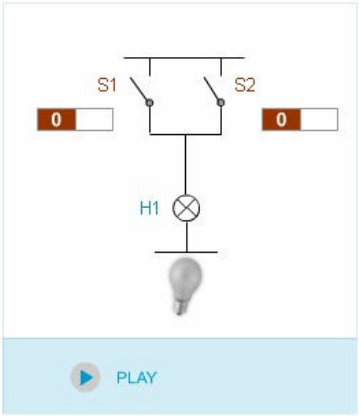
Operador AND/Y



Si observamos el esquema del circuito, podemos ver que la lámpara H1 solo se enciende cuando S1 y S2 están cerrados

Así, el operador AND/Y ofrece como resultado el estado de la señal 1 cuando todos los operandos consultados muestran el estado de señal 1.

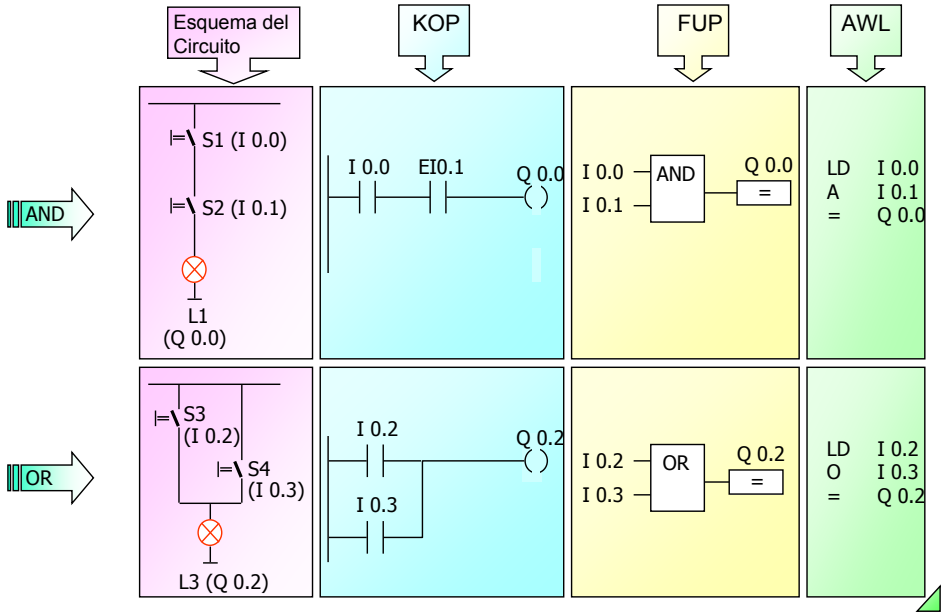
Operador OR/O



La dependencia de las condiciones de salida con relación a las de entrada se denomina operador OR/O. Para que la lámpara H1 se encienda, deben estar cerrados los interruptores S1 o bien S2.

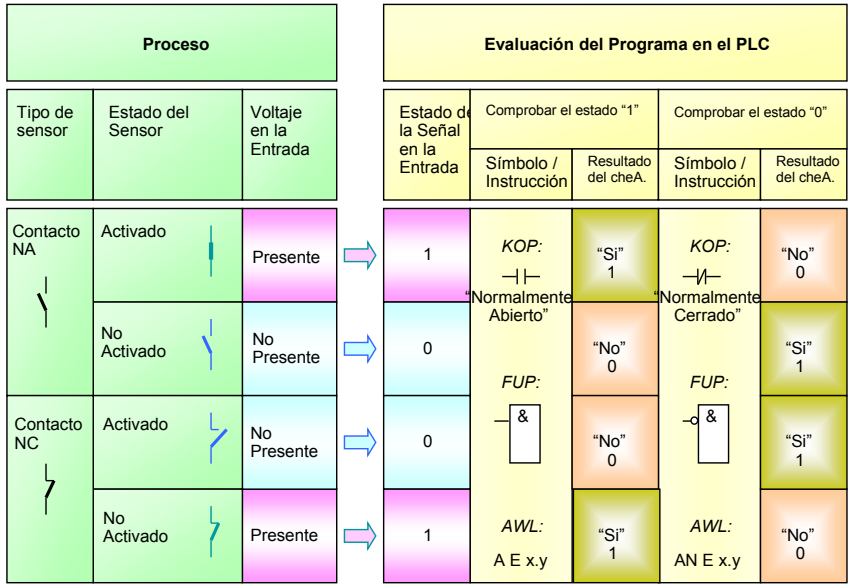
Obtenemos el estado de la señal 1 cuando al menos uno de los operandos consultados lleva el estado de señal 1.

Operaciones Lógicas a Nivel de Bit: **AND**, **OR** con distintas representaciones



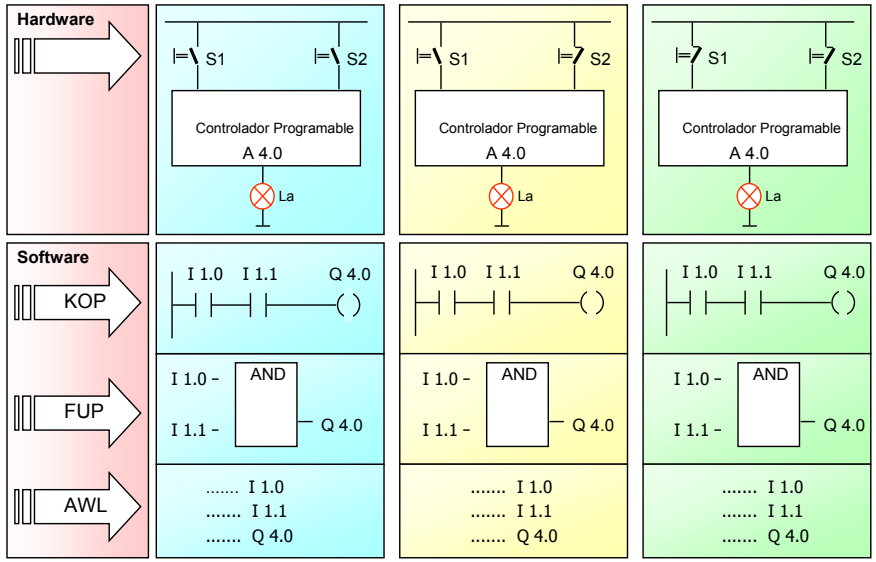
Operaciones Lógicas con Bits

Contactos NA y NC. Sensores y Símbolos



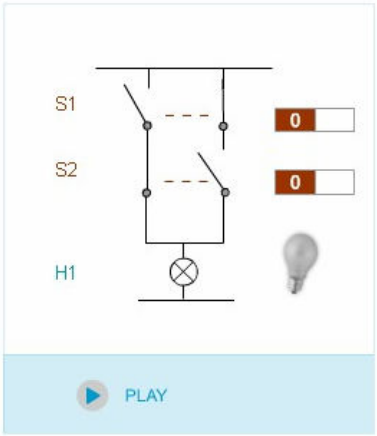
Operaciones Lógicas con Bits

Ejercicio: Realizar las modificaciones necesarias a los programas en el esquema para obtener la siguiente funcionalidad: Cuando el interruptor S1 esté activado y el S2 desactivado, la bombilla debería estar iluminada en los tres ejemplos.



Operaciones Lógicas con Bits

Operador XOR

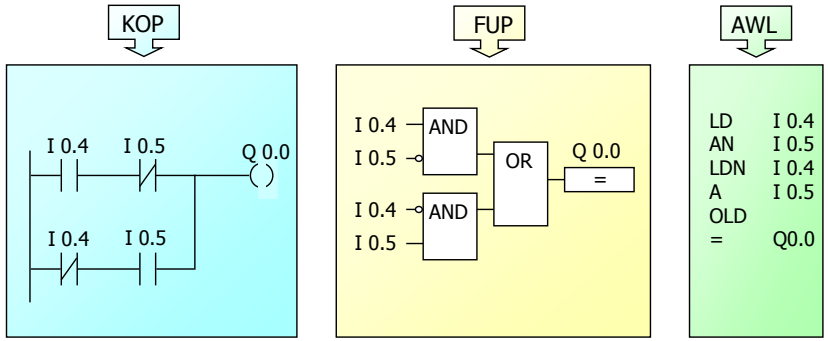


Para que la lámpara H1 se encienda, deben estar cerrados únicamente el interruptor S1 o únicamente el interruptor S2. Esta dependencia de las condiciones de salida con relación a la entrada se denomina operador O exclusiva (XOR).

El operador XOR ofrece como resultado el estado de señal 1 cuando exactamente uno de los operandos consultados lleva el estado de señal 1.

Operaciones Lógicas con Bits

Operaciones Lógicas a Nivel de Bit: OR - Exclusiva



Regla La regla de una operación XOR de dos operandos es la siguiente:
La Salida se pone a "1" cuando los estados de las dos entradas son distintos.

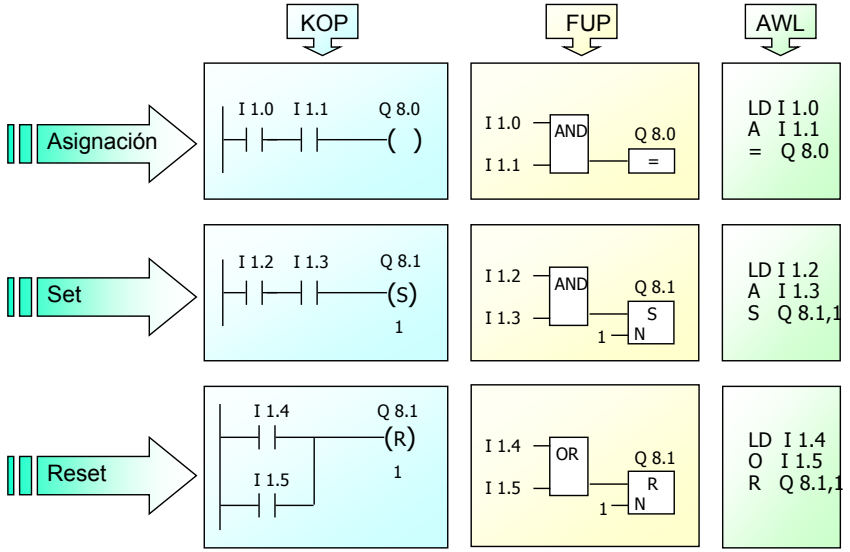
Operaciones Lógicas con Bits

Resultado de Operación Lógica, Primera Comprobación. Ejemplos

	Ejemplo 1				Ejemplo 2				Ejemplo 3			
	Estado de la Señal	Resultado de la Comprobación	RLO	Primera Comprobación	Estado de la Señal	Resultado de la Comprobación	RLO	Primera Comprobación	Estado de la Señal	Resultado de la Comprobación	RLO	Primera Comprobación
LD I 1.0	0				1				1			
AN I 1.1	0				1				0			
A M 4.0	0				1				1			
= Q8.0												
= Q 8.1												
LD I 2.0	0				1				0			

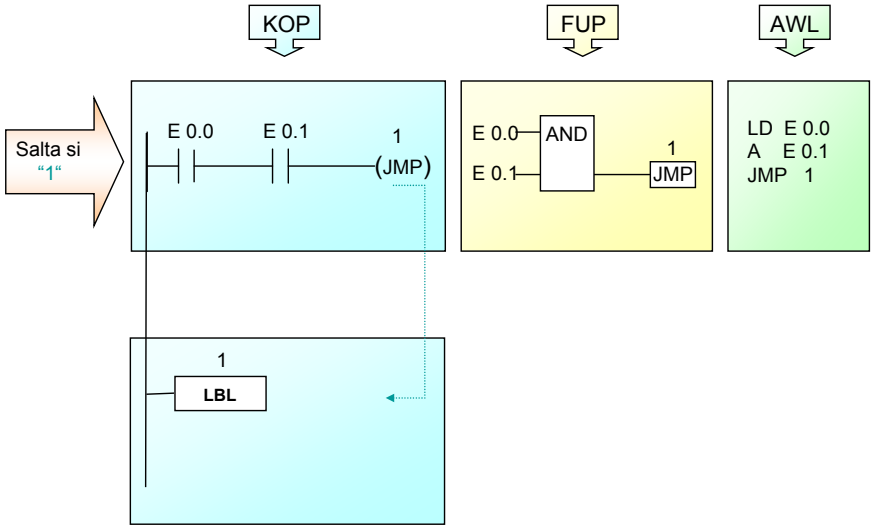
Operaciones Lógicas con Bits

Asignación, Set (Función memoria), Reset:



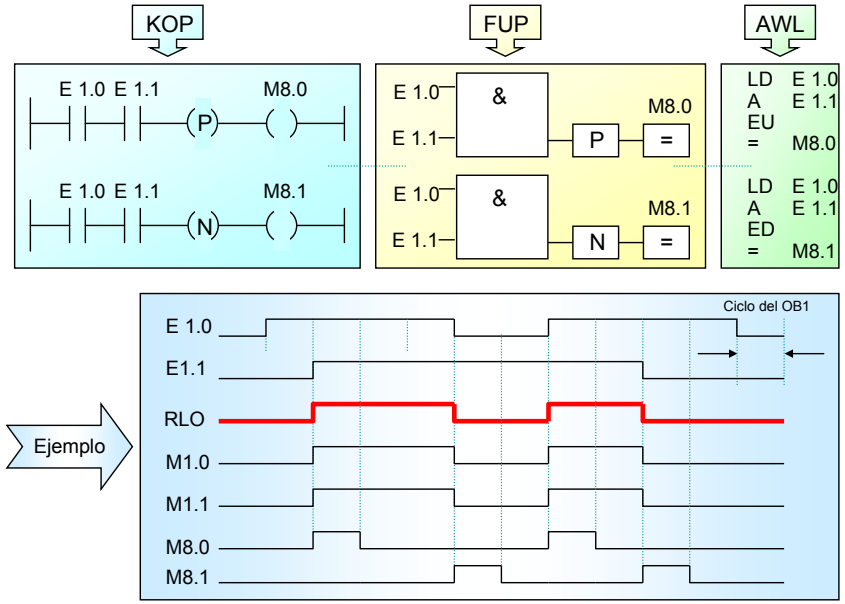
Operaciones Lógicas con Bits

Salto Incondicional: (programacion lineal)



Operaciones Lógicas con Bits

Funciones de Detección de Flanco: **Flanco positivo, Flanco negativo**

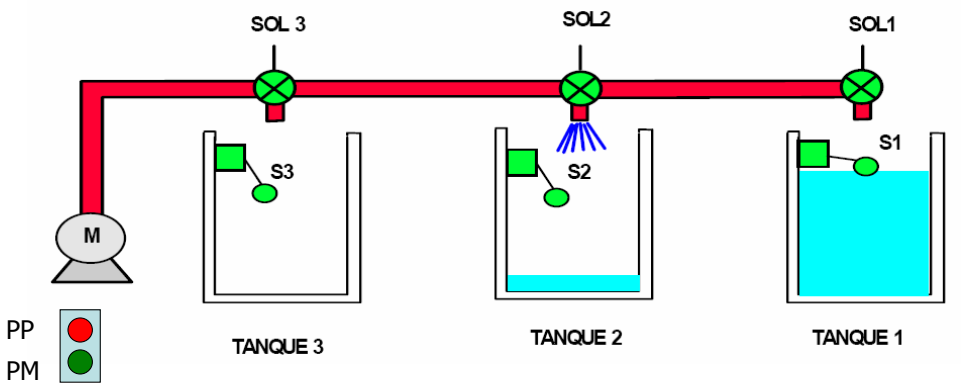


Ejemplo: Problema de llenado de tanques en secuencia:

Realizar el programa para que un PLC controle el llenado de tres tanques de agua en forma secuencial como se describe a continuación:

Al oprimir **PM** arranca la bomba **M** y se abre la válvula de llenado **SOL1** para el tanque 1, al terminar el llenado se cierra **SOL1** y se abre la válvula **SOL2** para llenar el tanque 2, al terminar se cierra **SOL2** y se abre la válvula **SOL3** para llenar el tercer tanque.

Al finalizar se para la bomba.



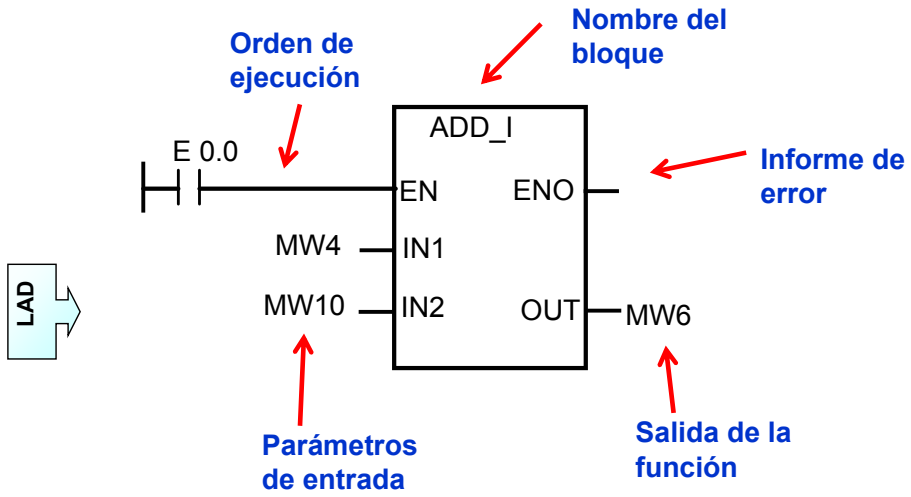
Nociones básicas:

- **Direccionamiento.**
- **Ejecución del programa**
- **Programación**
- **Funciones lógicas.**
- **Función memoria.**
- **Bloques de Función**
- **Resumen.**

Nociones básicas:

- **Direccionamiento.**
- **Ejecución del programa**
- **Programación**
- **Funciones lógicas.**
- **Función memoria.**
- **Bloques de Función**
- **Resumen.**

Bloques de Funciones Básicas:



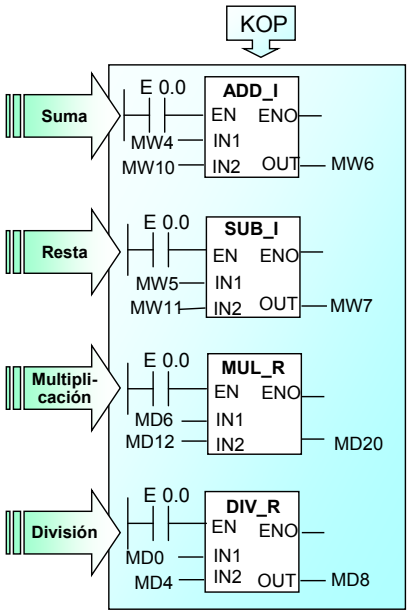
Funciones Aritméticas Básicas

Todas las instrucciones tienen el mismo formato:

EN La instrucción es ejecutada si el RLO=1 en la entrada de habilitación EN.

ENO Si el resultado se sale del rango permitido de valores para ese tipo de datos, la salida de habilitación ENO=0. Esto impide que aquellas operaciones que dependan de la salida ENO puedan ser ejecutadas.

OUT El resultado de la operación matemática se almacena en la salida indicada en la salida OUT.



Funciones Aritméticas Básicas

Instrucciones

Suma:

ADD_I	Suma de enteros
ADD_DI	Suma de doble entero
ADD_R	Suma de real

Resta:

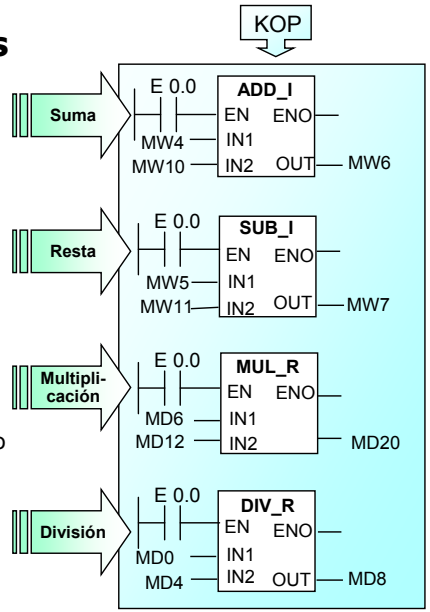
SUB_I	Resta de enteros
SUB_DI	Resta de doble entero
SUB_R	Resta de real

Multiplicación:

MUL_I	Multiplicación de enteros
MUL_DI	Multiplicación de doble entero
MUL_R	Multiplicación de real

Division:

DIV_I	División de enteros
DIV_DI	División de doble entero
DIV_R	División de real



Funciones Aritméticas Básicas

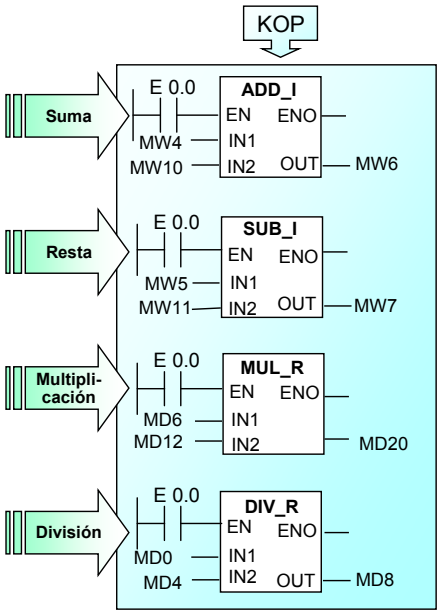
Esas instrucciones afectan a los siguientes bits de marcas especiales:

SM1.0 El resultado de la operación ha sido Cero

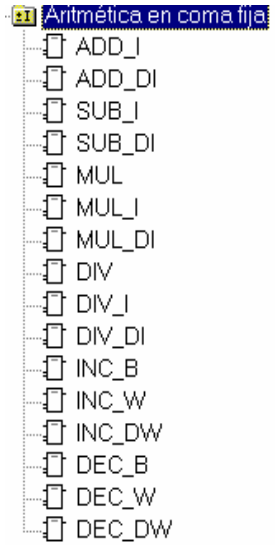
SM1.1 Overflow

SM1.2 El resultado de la operación ha sido Negativo

SM1.3 Division por cero



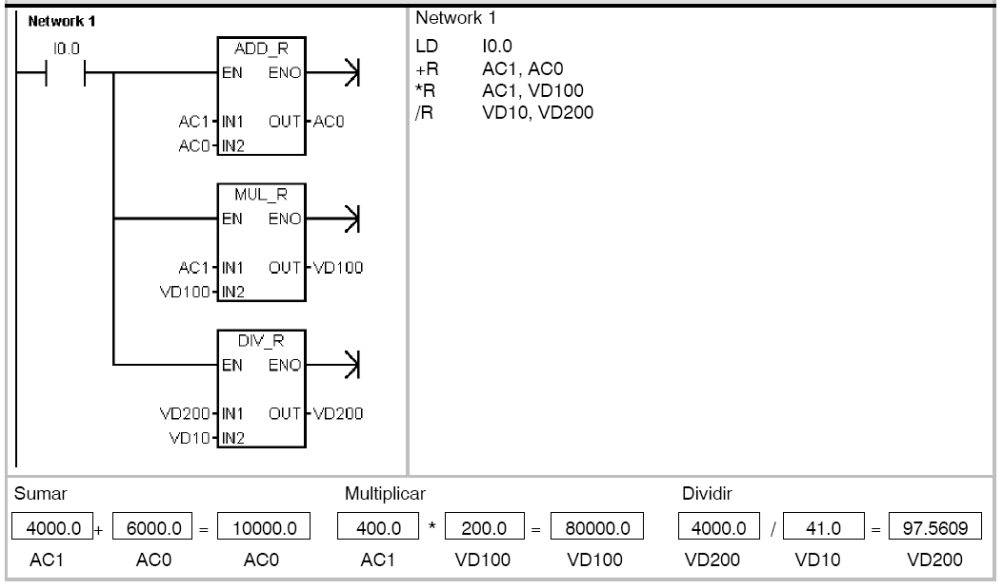
Aritmética en coma fija y en coma flotante



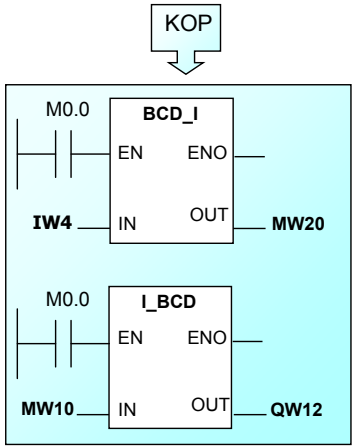
Funciones en coma flotante:

SIN, COS, TAN, LN, EXP

Ejemplo: Operaciones aritméticas con números reales



Instrucciones de Conversión: BCD <-> Entero



EN Si el RLO=1 en la entrada de habilitación EN, la conversión se ejecuta.

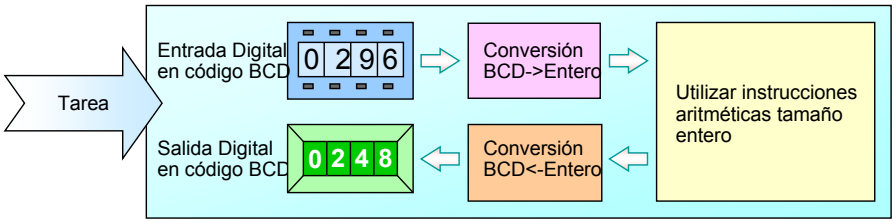
IN Cuando EN=1, el valor en la entrada IN es leido para la conversión.

OUT El resultado de la conversión se almacena en la dirección dada por la salida OUT.

BCD_I La función "BCD a Entero" lee el contenido en el parámetro IN como un número de cuatro dígitos en BCD (rango: 0 a 9999) y lo convierte en un valor entero 16 bits.

I_BCD La función "Entero a BCD" lee el contenido del parámetro IN como un valor entero 16 bits y lo convierte en un número de cuatro dígitos en BCD (rango: 0 a 9999). Si el valor se sale de este rango (overflow) => ENO = 0.

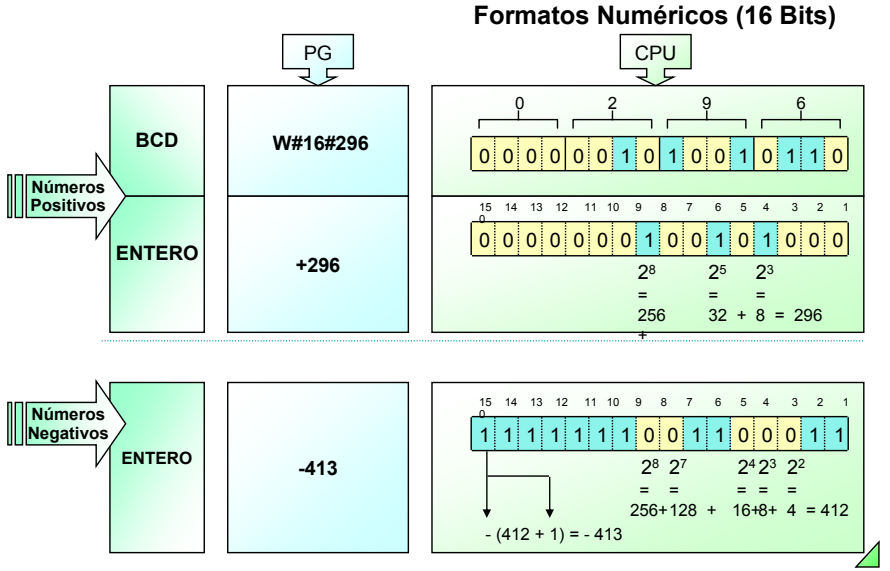
Instrucciones de Conversión: BCD <-> Entero



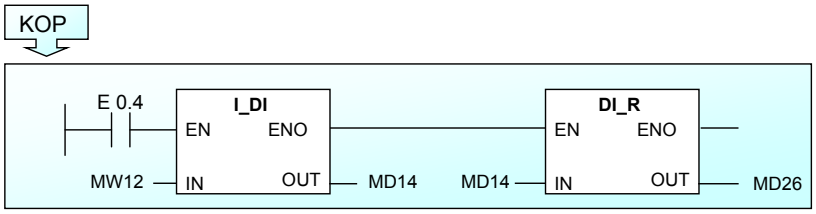
Ejemplo

- Un programa de usuario debe llevar a cabo funciones matemáticas utilizando valores a través de unos pulsadores, y mostrar los resultados en un display digital.
- Las funciones matemáticas no pueden realizarse en código BCD, por lo que hay que realizar un cambio de formato.

Instrucciones de Conversión: BCD <-> Entero



Instrucciones de Conversión: I -> DI -> REAL

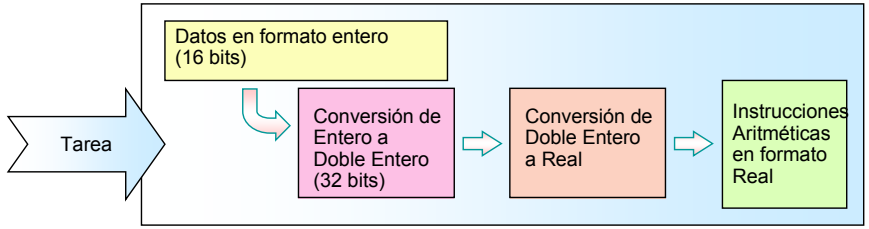


I_DE / ITD Convierte de Entero a Doble Entero.

DI_R / DTR Convierte de Doble Entero a Real.

Introducción a la programación: Bloques de función

Instrucciones de Conversión: I -> DI -> REAL



Ejemplo

•Un programa de usuario que trabaje con enteros también necesita realizar divisiones, con lo que pasamos ahora a números Reales.

•Por lo tanto es necesario utilizar una conversión, primero de Entero a Doble Entero, y después a Real.

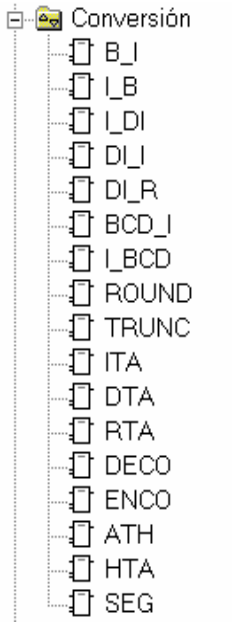
Introducción a la programación: Bloques de función

CONVERSIONES

B_I= byte a entero

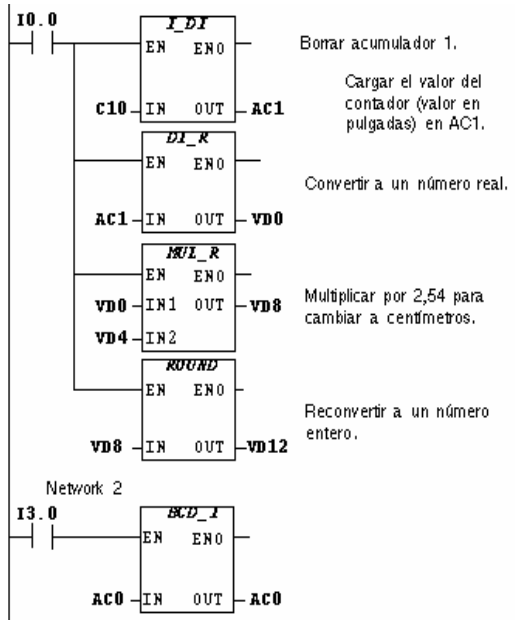
Round = redondea por encima

Trunc = trunca



Introducción a la programación: Bloques de función

CONVERSIONES: Ejemplo



Instrucciones de Temporizador

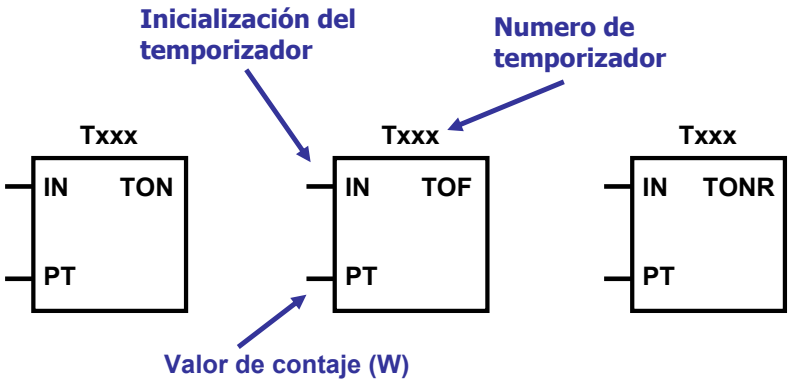
Instrucciones de Temporizador

Tipo de temporizador	Denominación
Temporizador de retardo a la conexión	**→ TON**
Temporizador de retardo a la desconexión.....	**→ TOF**
Temporizador de retardo a la conexión memorizado	**→ TONR**

Nº de temporizadores: 256 (T0 a T255)

Temporizador	Resolución	Valor máximo	Nº de temporizador
TONR	1 ms	32,767 s	T0, T64
	10 ms	327,67 s	T1-T4, T65-T68
	100 ms	3276,7 s	T5-T31, T69-T95
TON, TOF	1 ms	32,767 s	T32, T96
	10 ms	327,67 s	T33-T36, T97-T100
	100 ms	3276,7 s	T37-T63, T101-T255

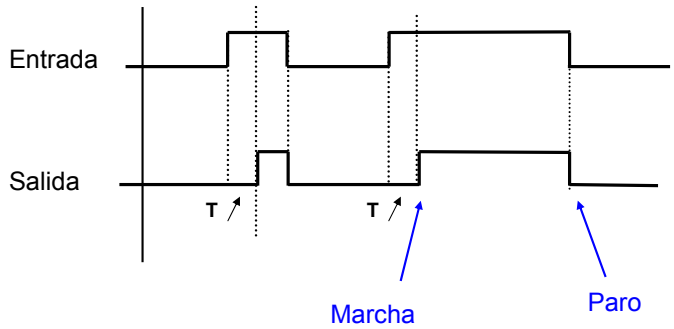
Instrucciones de Temporizador



- **El Nº nos da la unidad de temporización: 1, 10, 100 ms.**
- **El valor de contaje seleccionado las veces que cuenta la unidad de tiempo.**
- **Se puede acceder al estado del temporizador (BIT) como al valor de contaje (INT).**

Instrucciones de Temporizador

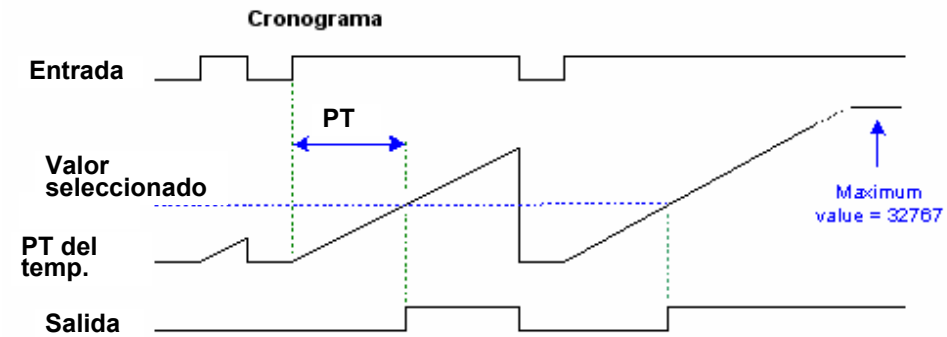
Instrucción TON : Temporizador de retardo a la conexión

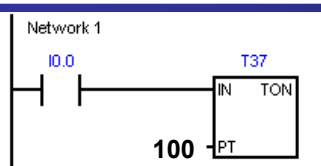


- **SE GENERA UNA SEÑAL DE RETARDO TRAS LA ACTIVACION DE LA SEÑAL DE ENTRADA**

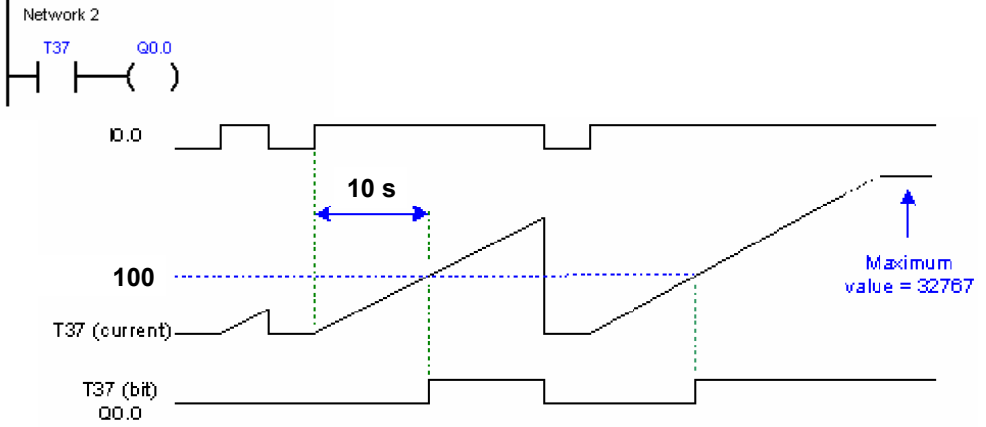
Instrucciones de Temporizador

Instrucción TON : Temporizador de retardo a la conexión



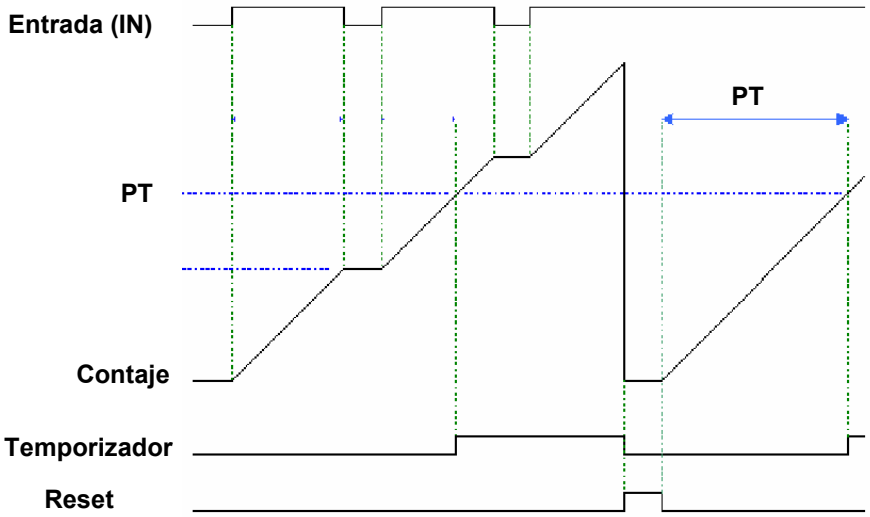


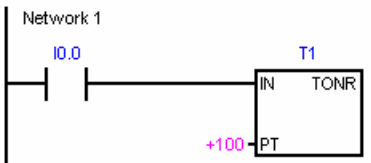
Ejemplo: **Temporizador de retardo a la conexión de 10 segundos**



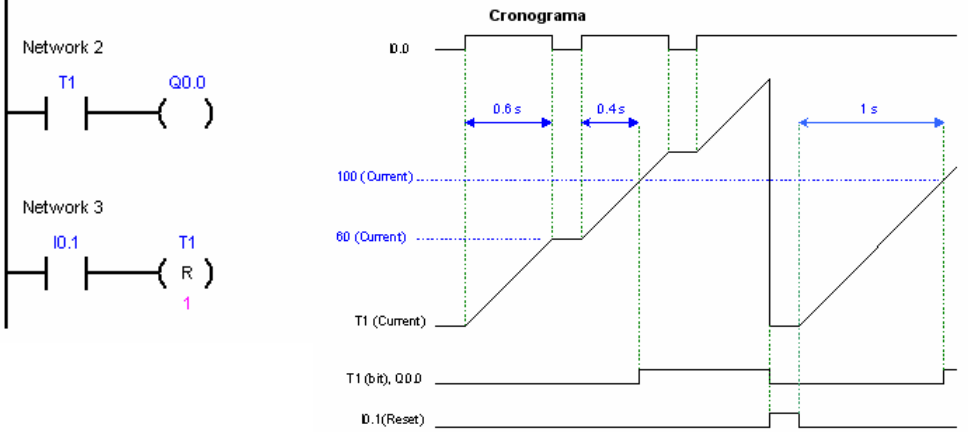
Instrucciones de Temporizador

Instrucción TONR: Temporizador de retardo a la conexión memorizado



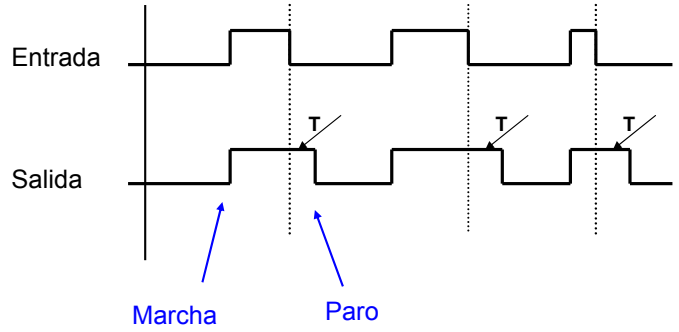


TONR. Temporizador de retardo a la conexión memorizado



Instrucciones de Temporizador

Instrucción TOF: Temporizador de retardo a la desconexión

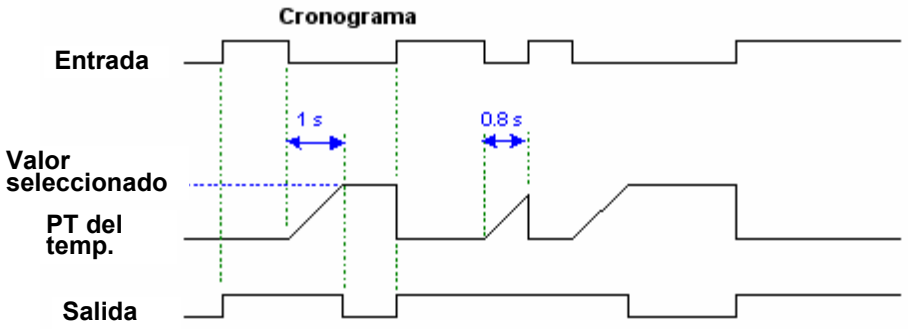


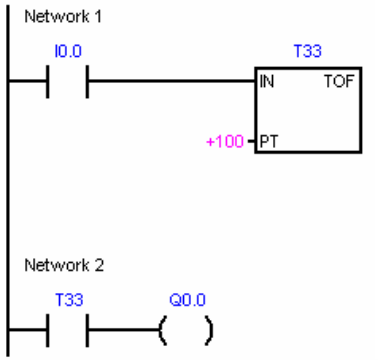
- **SE GENERA UNA SEÑAL DE RETARDO TRAS LA CAÍDA DE LA SEÑAL DE ENTRADA**

Introducción a la programación: Bloques de función

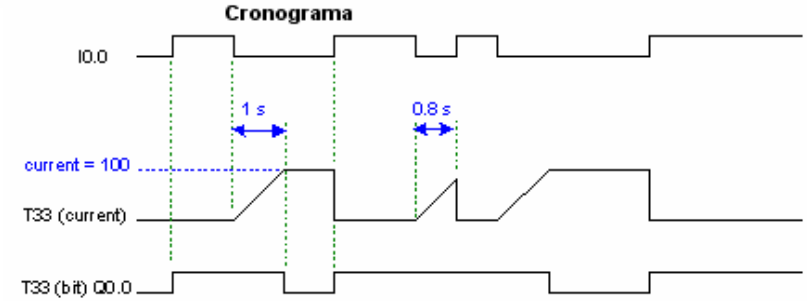
Instrucciones de Temporizador

Instrucción TOF: Temporizador de retardo a la desconexión





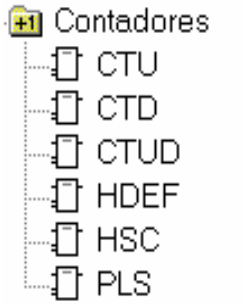
TOF → Temporizador de retardo a la desconexión (TOF)

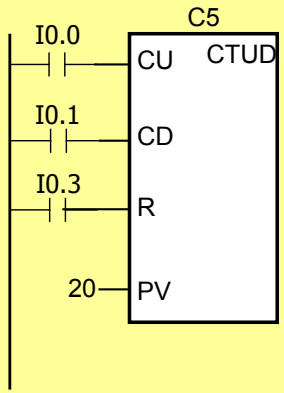


Introducción a la programación: Bloques de función

Instrucciones de Contadores

Instrucciones de Contadores





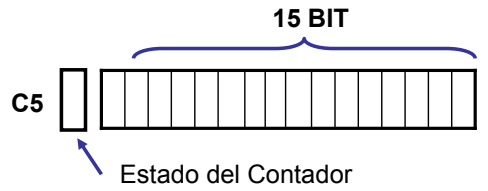
- CTU= Contador hacia delante
- CTD= Contador hacia detrás
- CTUD= Contador hacia delante y hacia detrás

- Si se accede tipo BIT, nos da si se activa el temporizador, y se hace tipo PALABRA (W) nos da el valor

Introducción a la programación: Bloques de función

Instrucciones de Contadores

Valor de Contaje: Se reserva una palabra (16 bits) en la memoria de datos del sistema para cada contador. El valor de contaje se almacena en código binario (rango: -32767 a +32767).



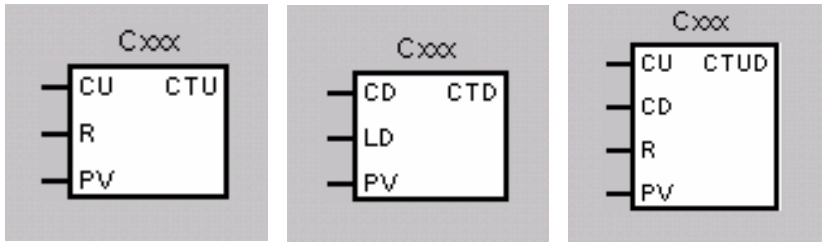
Esta zona de la memoria se identifica con la letra "C" seguida de un número "XXX" que debe estar comprendido en el intervalo [0..255]. Se puede decir por tanto, que el usuario dispone de hasta 256 contadores distintos.

El valor actual (VA) del número de eventos producidos se almacena en una variable del tipo CXXX cuyo tamaño es una palabra (una WORD) y cuyo tipo es INT.

Por cada contador CXXX, el PLC ofrece al usuario una variable de tipo BIT también identificado como CXXX que tomará el valor "1" ó "0"

Introducción a la programación: Bloques de función

Instrucciones de Contadores



Contar Ascendente: Cuando en la entrada "CU" cambia de “0” a “1”, el contaje se incrementa en una unidad (límite superior = 32767).

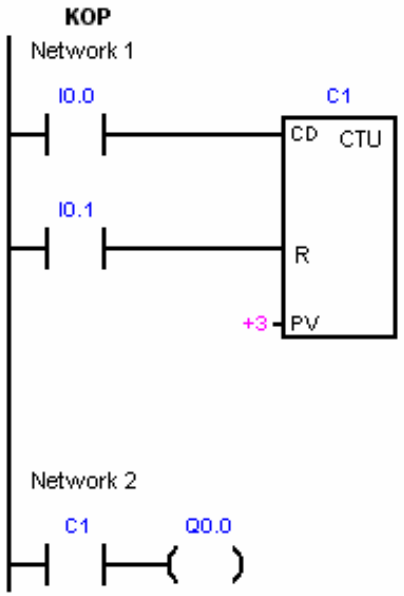
Contar Descendente: Cuando en la entrada "CD" cambia de “0” a “1”, el contaje se decrementa en una unidad (límite inferior = -32768).

Set del Contador: Cuando el valor de contaje es mayor o igual al valor de preselección “PV”, la salida lógica del contador “Cxxx” es puesto a “1“.

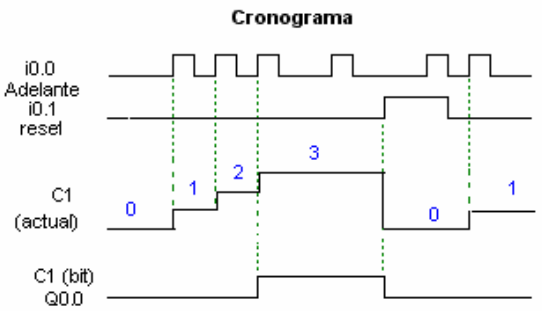
Reset del Contador: Cuando el RLO de la entrada “R” es puesta a “1“, el valor de contaje es puesto a “0“. El contador no puede volver a contar mientras que la entrada “R“ esté puesta a “1“.

Introducción a la programación: Bloques de función

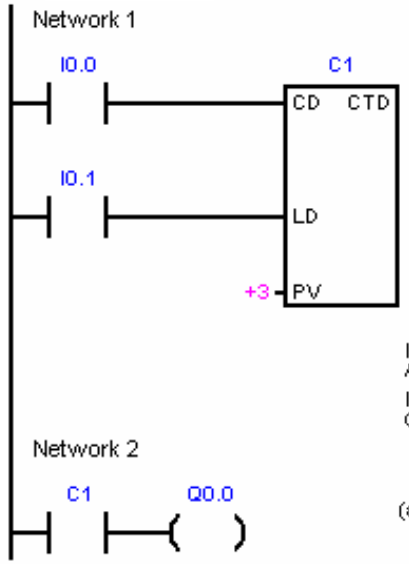
Instrucciones de Contadores



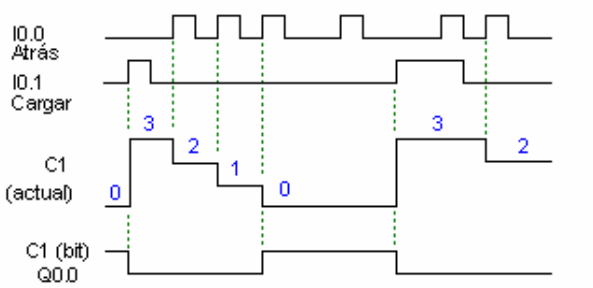
Ejemplo de una operación de contaje hacia delante (CTU):



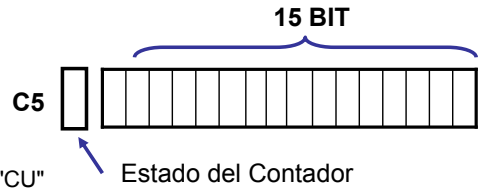
Instrucciones de Contadores



Ejemplo de una operación de contaje hacia atrás (CTD):



Instrucciones de Contadores

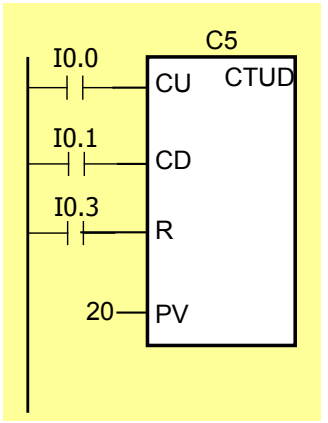


Contar Ascendente: Cuando en la entrada "CU" cambia de "0" a "1", el contaje se incrementa en una unidad (límite superior = 32767).

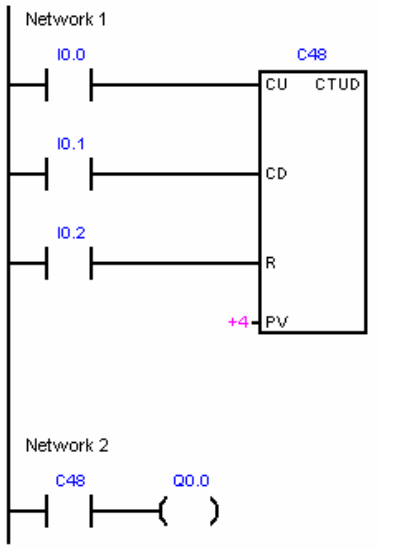
Contar Descendente: Cuando en la entrada "CD" cambia de "0" a "1", el contaje se decrementa en una unidad (límite inferior = -32768).

Set del Contador: Cuando el valor de contaje es mayor o igual al valor de preselección "PV", la salida lógica del contador "Cxxx" es puesto a "1".

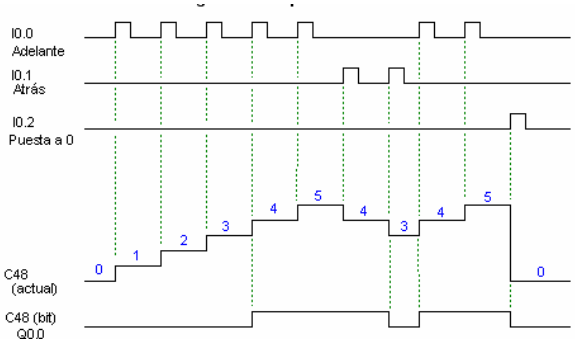
Reset del Contador: Cuando el RLO de la entrada "R" es puesta a "1", el valor de contaje es puesto a "0". El contador no puede volver a contar mientras que la entrada "R" esté puesta a "1".



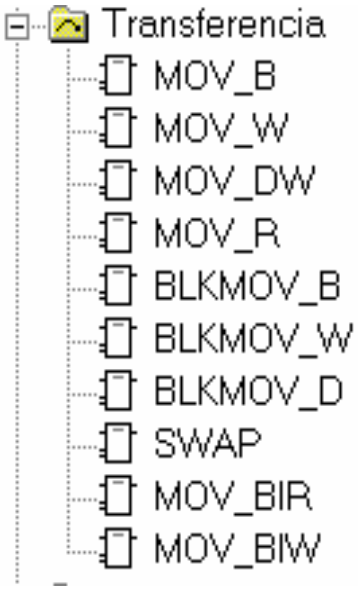
Instrucciones de Contadores



Ejemplo de una operación de contaje hacia adelante/atrás (CTUD):

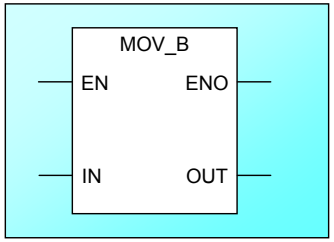


Instrucciones de Transferencia de datos



Instrucciones de Transferencia de datos:

Instrucción: **MOV_B**
MOV_W
MOV_DW
MOV_R

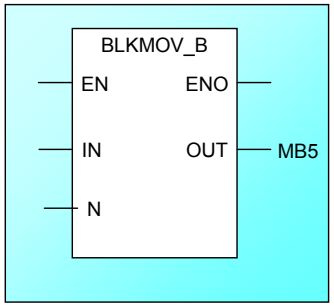


MOVE
Si la entrada EN está activa, el valor de la entrada "IN" se copia en la dirección de salida "OUT". "ENO" tiene el mismo estado de señal que "EN".

EN es una entrada booleana que tienen todos los bloques KOP y FUP. El bloque solo se ejecutará si la entrada EN está puesta a "1".

Instrucciones de Transferencia de datos:

Instrucción: **BLKMOV_B**
BLKMOV_W
BLKMOV_D

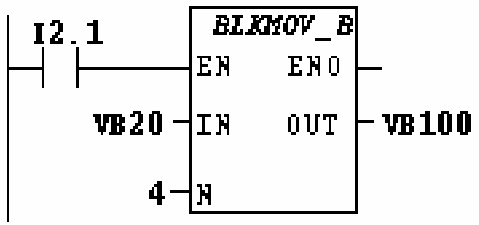


MOVE BLOCK
Si la entrada EN está activa, se copia un bloque de datos de longitu dada por "N" y que empieza en la direccion "IN" en la dirección de salida "OUT". .

N puede estar comprendido entre 1 y 255. Los tipos de datos de entrada y salida pueden variar, pero deben ser del mismo tipo.

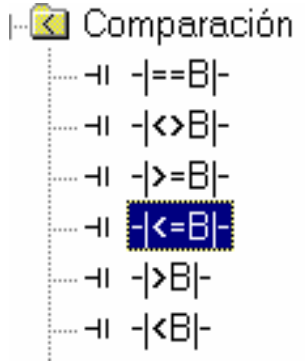
Instrucciones de Transferencia de datos:

Ejemplo:



Transferir
Campo 1 (VB20 a VB23) a campo 2 (VB100 a VB103)

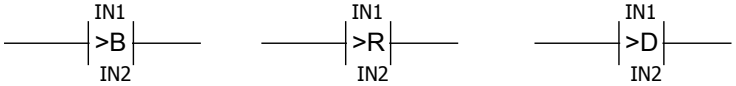
Instrucciones de Comparación



Comparación Se pueden utilizar instrucciones de comparación para comparar parejas de valores numéricos:

- **B** Bytes
- **I** Entero (Números en Coma Fija 16-bit s con signo)
- **D** Doble Entero (Números en Coma Fija 32-bits con signo)
- **R** Real (Número en Coma Flotante 32-bit con signo).

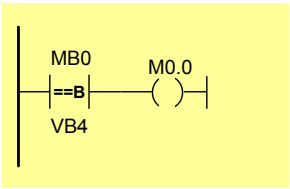
Instrucciones de Comparación:



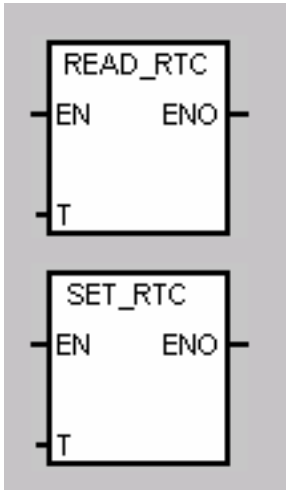
Si el resultado de la comparación es “Verdadero“, el RLO de la instrucción es puesto a "1". En caso de ser “Falso“, se pone a "0".
Las entradas IN1 y IN2 son comparadas de acuerdo con el tipo de comparación seleccionada:

- **==** IN1 igual que IN2
- **<>** IN1 distinto que IN2
- **>** IN1 mayor que IN2
- **<** IN1 menor que IN2
- **>=** IN1 mayor o igual que IN2
- **<=** IN1 menor o igual que IN2.

Ejemplo:



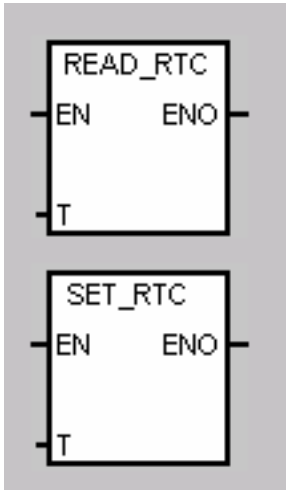
Instrucciones de Reloj en tiempo Real



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).

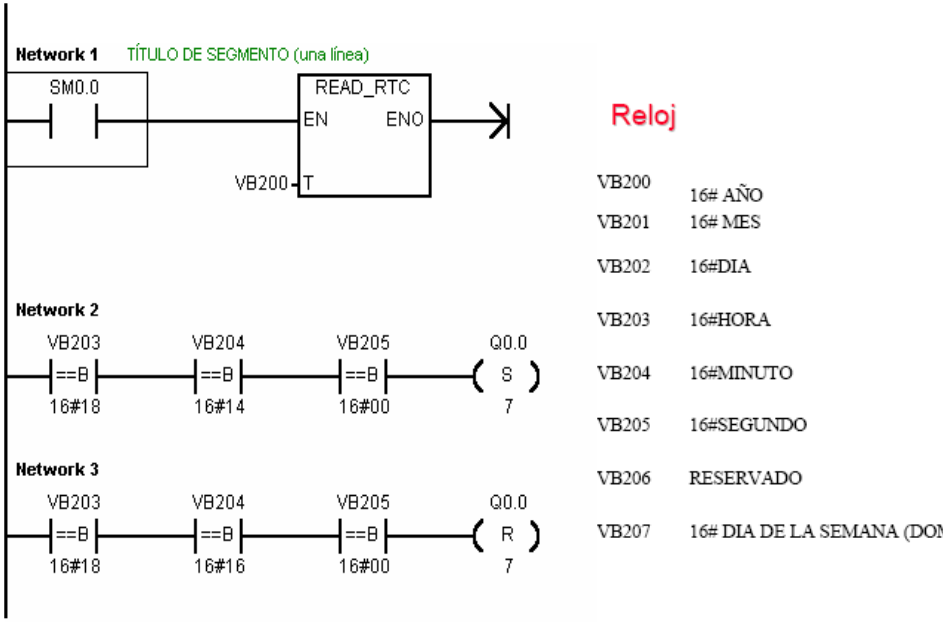
Instrucciones de Reloj en tiempo Real



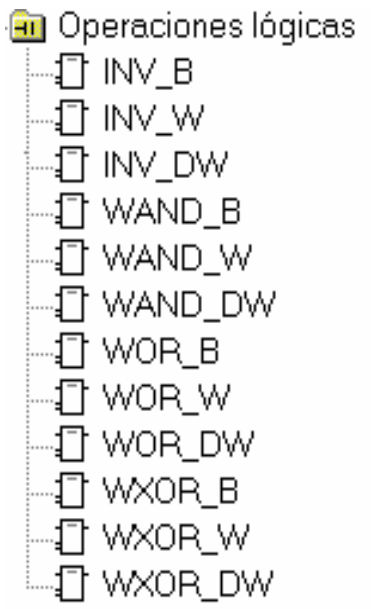
Formato del búfer de tiempo de 8 bytes (T)

Byte T	Descripción	Datos de byte
0	Año (0 a 99)	Año actual (valor BCD)
1	Mes (1 a 12)	Mes actual (valor BCD)
2	Día (1 a 31)	Día actual (valor BCD)
3	Hora (0 a 23)	Hora actual (valor BCD)
4	Minuto (0 a 59)	Minuto actual (valor BCD)
5	Segundo (0 a 59)	Segundo actual (valor BCD)
6	00	Reservado – ajustado siempre a 00
7	Día de la semana (1 a 7)	Día actual de la semana, 1=domingo (valor BCD)

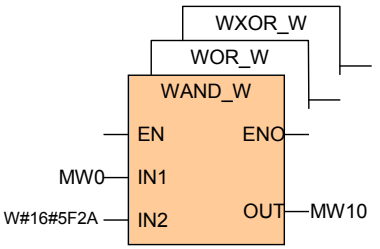
Ejemplo:

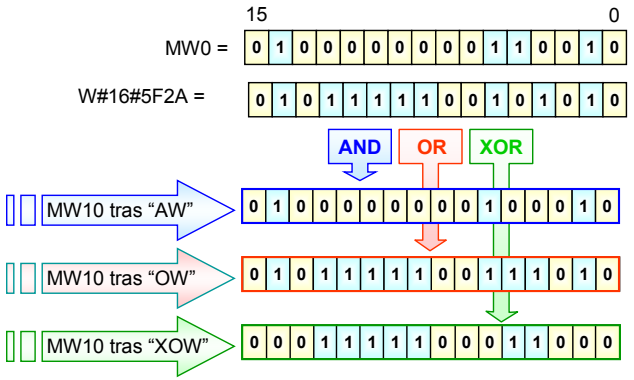


Otras operaciones:

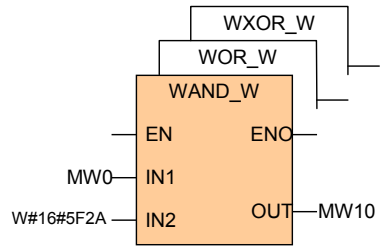


Operaciones Lógicas Digitales





Operaciones Lógicas Digitales



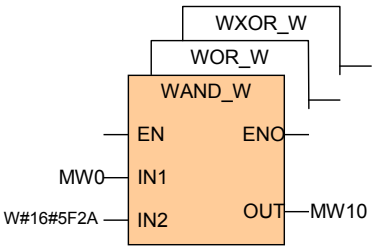
WAND_W

La operación "AND a nivel Palabra" realiza un AND de los dos valores digitales de las entradas IN1 y IN2 bit a bit. El resultado de la operación AND se almacena en la dirección indicada en la salida OUT.
La instrucción se ejecuta cuando EN = 1.

Ejemplo: Poner a 0 la tétrada alta de una doble palabra:

MW0	=	0100	0100	1100	0100
W#16#0FFF	=	0000	1111	1111	1111
MW30	=	0000	0100	1100	0100

Operaciones Lógicas Digitales

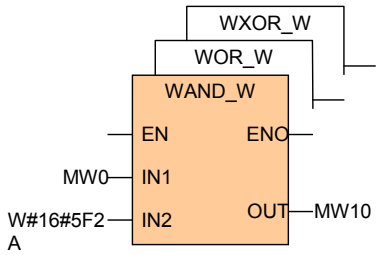


WOR_W
La operación "OR a nivel Palabra" realiza un OR de los dos valores digitales de las entradas IN1 y IN2 bit a bit. El resultado de la operación OR se almacena en la dirección indicada en la salida OUT.
La instrucción se ejecuta cuando EN = 1.

Ejemplo: Poner a "1" el bit de menor peso de la doble palabra MW32:

MW32	=	0100	0010	0110	1010
W#16#0001	=	0000	0000	0000	0001
MW32	=	0100	0010	0110	1011

Operaciones Lógicas Digitales



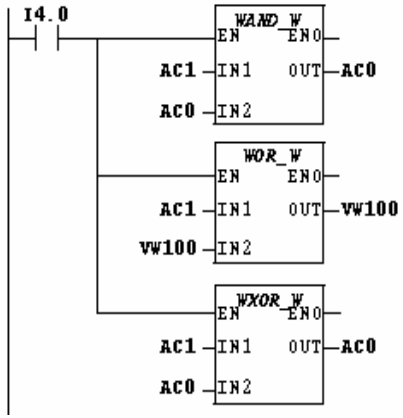
WXOR_W
La operación "XOR a nivel Palabra" realiza un XOR de los dos valores digitales de las entradas IN1 y IN2 bit a bit. El resultado de la operación XOR se almacena en la dirección indicada en la salida OUT.
La instrucción se ejecuta cuando EN = 1.

Ejemplo: Detectar algún cambio en la EW0:

MW0	=	0100	0100	1100	1010
MW28	=	0110	0010	1011	1001
MW24	=	0010	0110	0111	0011

Operaciones Lógicas Digitales

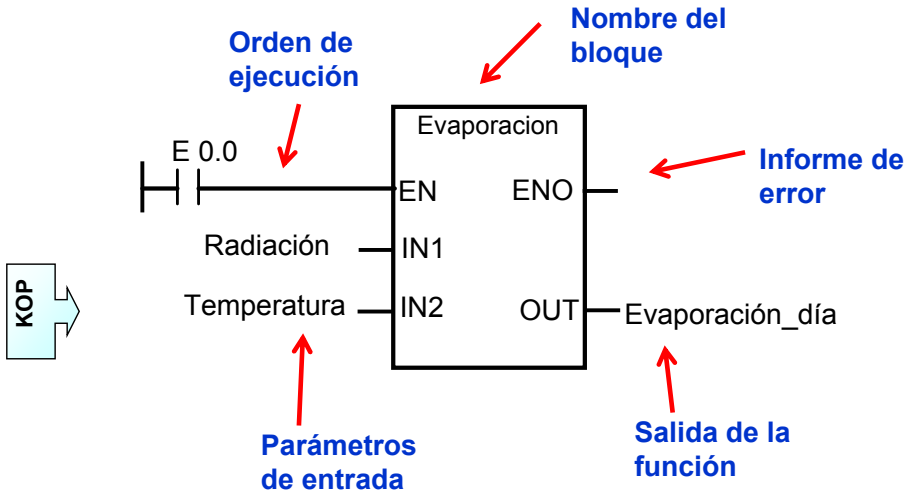
Ejemplo:



Nociones básicas:

- **Direccionamiento.**
- **Ejecución del programa**
- **Programación**
- **Funciones lógicas.**
- **Función memoria.**
- **Bloques de Función**
- **Resumen.**

Bloques de función programables:



Nociones básicas:

- **Direccionamiento.**
- **Ejecución del programa**
- **Programación**
- **Funciones lógicas.**
- **Función memoria.**
- **Bloques de Función**
- **Resumen.**

Introducción a la programación

Resumen

- **Ejecución cíclica del programa,**
- **Imagen del proceso:** PAE, PAA.
- **Tiempo de ciclo, Tiempo de respuesta.**
- **Ejecución Lineal**.
- **Ejecución Estructurada:** profundidad de anidamiento, ventajas.
- **Funciones lógicas: AND, OR, XOR,**
- **Funciones SET, RESET.**
- **Bloques de función.**
- **Bloques de función programables.**