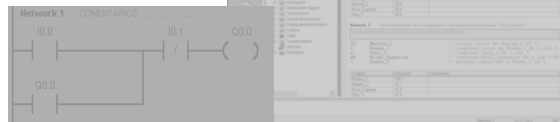


Programación de Automatas

STEP 7 Esquema de Contactos (KOP)



ISA-UMH

1

ÍNDICE

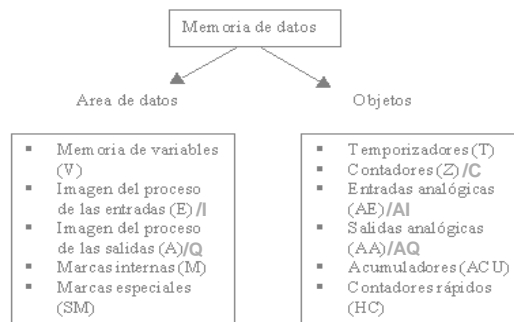
- Introducción a STEP 7: KOP
 - Distribución de la memoria
 - Lenguaje KOP: Diagrama de contactos
 - Operaciones básicas: contactos y salidas
 - Operaciones con temporizadores
 - Operaciones con contadores
 - Operaciones de comparación
 - Operaciones de transferencia
 - Operaciones aritméticas
 - Tabla de Símbolos
 - Ejemplos

Distribución de la memoria

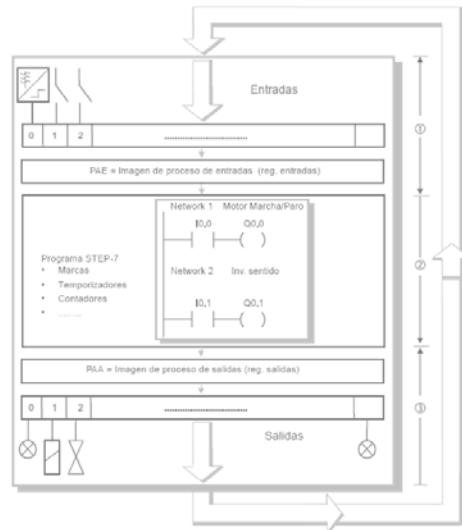
- Memoria de programa
 - La memoria de programa contiene las operaciones de esquema de contactos (KOP) o de lista de asignación (AWL), que ejecuta el autómata programable para la aplicación deseada.
- Memoria de parámetros
 - La memoria de parámetros permite almacenar determinados parámetros configurables, tales como contraseñas, direcciones de estaciones e informaciones sobre las áreas remanentes

Distribución de la memoria

- Memoria de datos
 - La memoria de datos es el área de trabajo a la que accede el programa de aplicación (también denominado programa de usuario).



Ciclo de ejecución del autómata



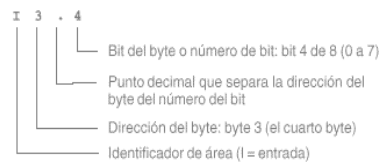
ISA-UMH Lenguajes de Programación STEP7

Direccionamiento de la Memoria

■ Acceso a un bit

- “Identificador de area” “dirección del byte” . “nº del bit”

Ejemplo I 0.0 el bit 0 del byte 0 de las entradas



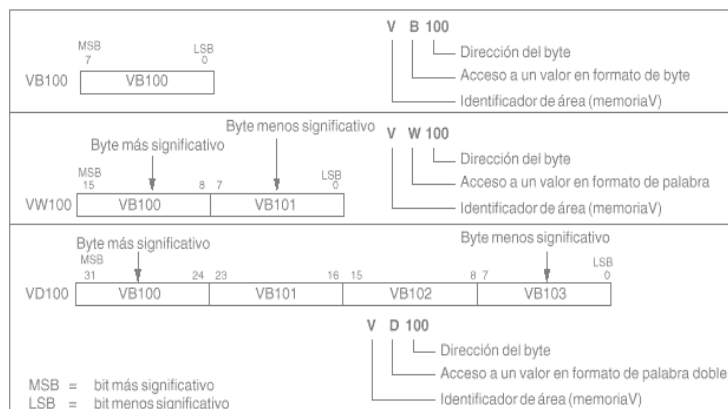
MSB = bit más significativo
LSB = bit menos significativo

	MSB	7	6	5	4	3	2	1	0	LSB
I 0										
I 1										
I 2										
I 3										
I 4										
I 5										
I 6										
I 7										
I 8										
I 9										
I 10										
I 11										
I 12										
I 13										
I 14										
I 15										

ISA-UMH Lenguajes de Programación STEP7

Direccionamiento de la Memoria

- Se puede acceder a diversas áreas de la memoria de la CPU (V, I, Q, M, SM) en formato byte, palabra y palabra doble



Direccionamiento de la Memoria

- Direccionamiento de la imagen del proceso de las entradas (I/E)**
 - Formato:
 - Bit I [módulo].[direcc. del bit] IO.1
 - Byte/word/double I [tamaño][direcc. del byte inicial] IB4
- Direccionamiento de la imagen del proceso de las salidas (Q/A)**
 - Formato:
 - Bit Q [módulo].[direcc. del bit] Q1.1
 - Byte/word/double Q [tamaño][direcc. del byte inicial] QB5
- Direccionamiento del área de marcas (M)**
 - Las marcas internas (área de marcas M) se pueden utilizar como relés de control para almacenar el estado intermedio de una operación u otras informaciones de control
 - Formato:
 - Bit M [direcc. del byte].[direcc. del bit] M26.7
 - Byte/word/double M [tamaño][direcc. del byte inicial] MD20

Direccionamiento de la Memoria

■ Direccionamiento de las marcas especiales (SM)

- Las marcas especiales permiten intercambiar datos entre la CPU y el programa. Dichas marcas se pueden utilizar para seleccionar y controlar algunas funciones especiales de la CPU S7-200, tales como:
 - Un bit que se activa sólo en el primer ciclo. **SM0.1**
 - Un bit que está siempre activado (autómata en marcha) **SM0.0**
 - Bits que se activan y se desactivan en determinados intervalos. **SM0.4 SM0.5**
 - Bits que muestran el estado de operaciones matemáticas y de otras operaciones. **SM1.0 (bit sp: saltos condicionales)**
- Formato:

● Bit	SM [direcc. del byte].[direcc. del bit]	SM0.1
● Byte, palabra, p. Doble	SM [tamaño][direcc. del byte inicial]	SMB86

■ Direccionamiento de la memoria de variables (V)

- Formato:

● Bit	V [direcc. del byte].[direcc. del bit]	V10.2
● Byte, palabra, p. Doble	V [tamaño][direcc. del byte inicial]	VW100

Direccionamiento de la Memoria

■ Direccionamiento de las entradas analógicas (AI)

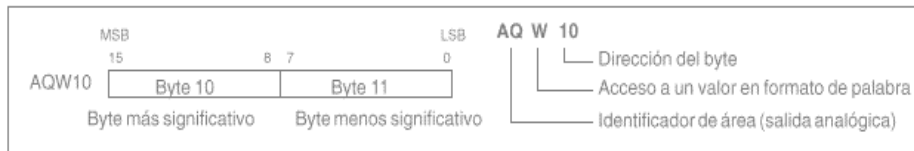
- La CPU S7-200 convierte valores reales analógicos (p.ej. temperatura, tensión, etc). en valores digitales en formato de palabra (de 16 bits).
 - Puesto que las entradas analógicas son palabras que comienzan siempre en bytes pares (p.ej. 0, 2, 4, etc)., es preciso utilizar direcciones con bytes pares (p.ej. AIW0, AIW2, AIW4, etc)
- Formato:

● AIW [dirección del byte inicial]	AIW4
------------------------------------	------



Direccionamiento de la Memoria

- Direccionamiento de las salidas analógicas (AQ)
 - La CPU S7-200 convierte valores digitales en formato de palabra (de 16 bits) en valores reales analógicos (p.ej. corriente o voltaje), proporcionales al valor digital.
 - Puesto que las salidas analógicas son palabras que comienzan siempre en bytes pares (p.ej. 0, 2, 4, etc.), es preciso utilizar direcciones con bytes pares (p.ej. AQW0, AQW2, AQW4, etc.) para acceder a las mismas.
 - Formato:
 - AQW [dirección del byte inicial] AQW4



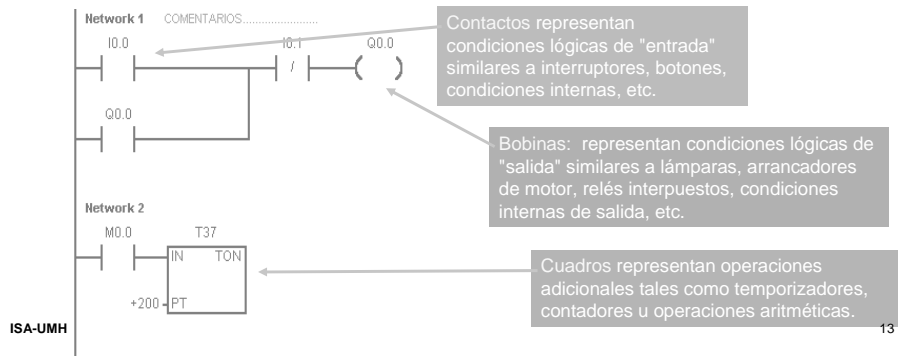
ÍNDICE

- Introducción a SETP 7: KOP
 - Distribución de la memoria
 - Lenguaje KOP: Diagrama de contactos
 - Operaciones básicas: contactos y salidas
 - Operaciones con temporizadores
 - Operaciones con contadores
 - Operaciones de comparación
 - Operaciones de transferencia
 - Operaciones aritméticas
 - Tabla de Símbolos
 - Ejemplos

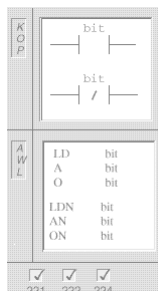
Lenguaje KOP

■ Esquema de Contactos KOP

- la lógica se divide en unidades pequeñas y de fácil comprensión llamadas "segmentos" o "networks"
- El programa se ejecuta segmento por segmento, de izquierda a derecha y luego de arriba a abajo.
- Tras alcanzar la CPU el final del programa, comienza nuevamente en la primera operación del mismo



Operaciones con Contactos

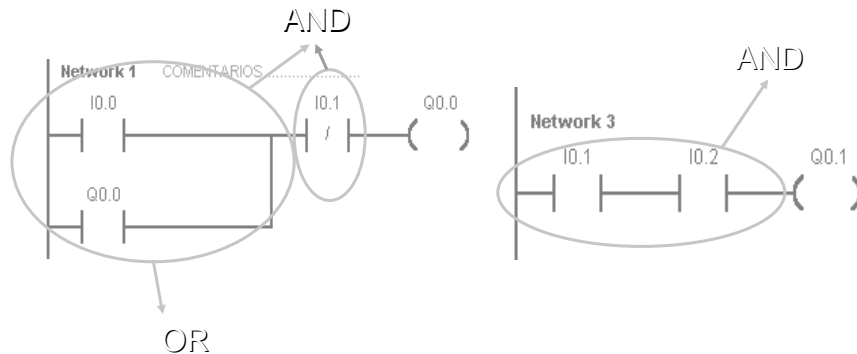


■ Contactos estándar

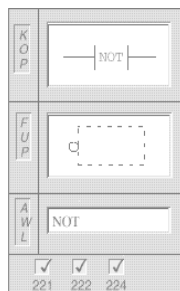
- El **contacto abierto** (—|—) se cierra (se activa) si el valor binario de la dirección $n = 1$.
- El **contacto cerrado** (—|/—) se cierra (se activa) si el valor binario de la dirección $n = 0$
- Operandos:
 - n : I, Q, M, SM, T, C, V, S

Operaciones con Contactos

- La operación AND se implementa mediante contactos en serie
- La operación OR se implementa mediante contactos en paralelo



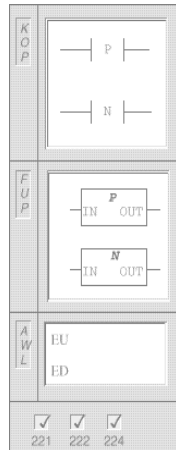
Operaciones con Contactos



■ NOT

- El contacto **NOT** invierte el sentido de circulación de la corriente. La corriente se detiene al alcanzar el contacto NOT. Si no logra alcanzar el contacto, entonces hace circular la corriente.
- En otras palabras, si al contacto **NOT** llega un "0" entonces sale un "1", y si llega un "1" sale un "0".
- Operandos:
 - ninguno

Operaciones con Contactos

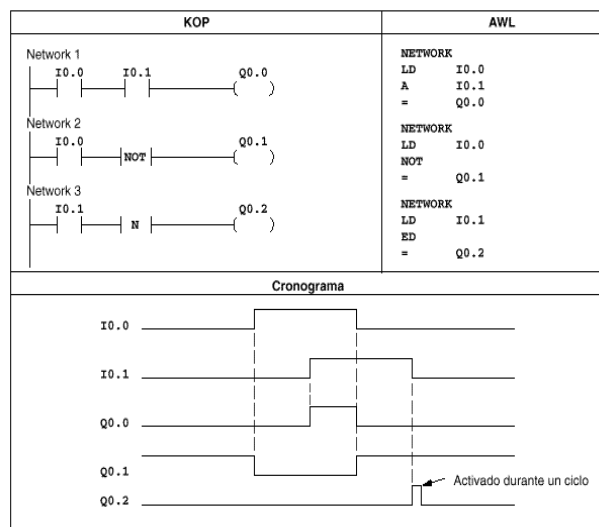


■ Detectar flanco positivo y negativo

- El contacto **Detectar flanco positivo** permite que fluya la corriente durante un ciclo cada vez que se produce un cambio de 0 a 1 (de "off" a "on")
- El contacto **Detectar flanco negativo** permite que fluya la corriente durante un ciclo cada vez que se produce un cambio de 1 a 0 (de "on" a "off")
- Operandos:
 - ninguno

Operaciones con Contactos

■ Ejemplos:



Ejemplo: Control del panel de mando de un motor

■ Entradas/Salidas

● Entradas:

- Interruptor on/off -> I0.0 (On-> 24V, Off-> 0V)
- Palanca Giro Positivo -> I0.1 (Giro-> 24V, Paro-> 0V)
- Palanca Giro Negativo -> I0.2 (Giro-> 24V, Paro-> 0V)

● Salidas:

- Lámpara Funcionamiento -> Q0.0 (Encendida-> 24V, Apagada-> 0V)
- Lámpara Sentido Positivo -> Q0.1 (Encendida-> 24V, Apagada-> 0V)
- Lámpara Sentido Negativo -> Q0.2 (Encendida-> 24V, Apagada-> 0V)
- Contactor giro positivo motor -> Q0.3 (Giro-> cerrado, Paro-> abierto)
- Contactor giro negativo motor -> Q0.4 (Giro-> cerrado, Paro-> abierto)

ISA-UMH Lenguajes de Programación STEP7



Ejemplo: Control del panel de mando de un motor

■ Funcionamiento:

- El Interruptor on/off pone en marcha o para el sistema y activa la lámpara de funcionamiento
- La Palanca Giro Positivo hace girar el motor en sentido positivo y se enciende la lámpara indicadora
- La Palanca Giro Negativo hace girar el motor en sentido negativo y se enciende la lámpara indicadora
- Si se accionan ambas palancas al mismo tiempo no gira el motor y se activan las dos lámparas indicadoras de giro.

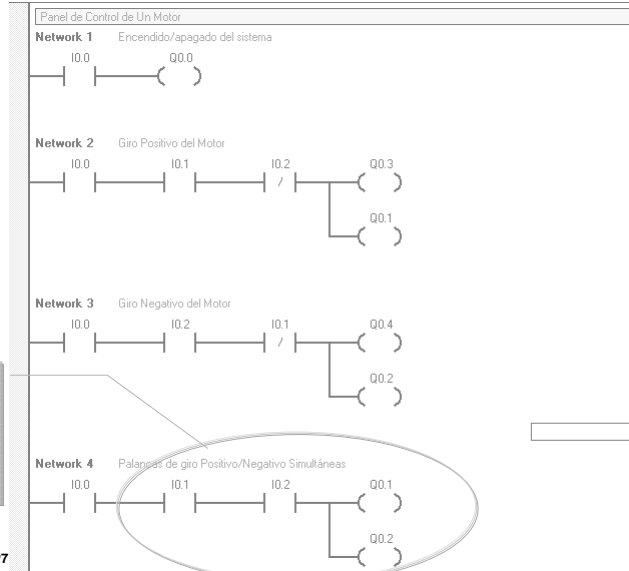
ISA-UMH Lenguajes de Programación STEP7



Ejemplo: Control del panel de mando de un motor

■ Solución 1:

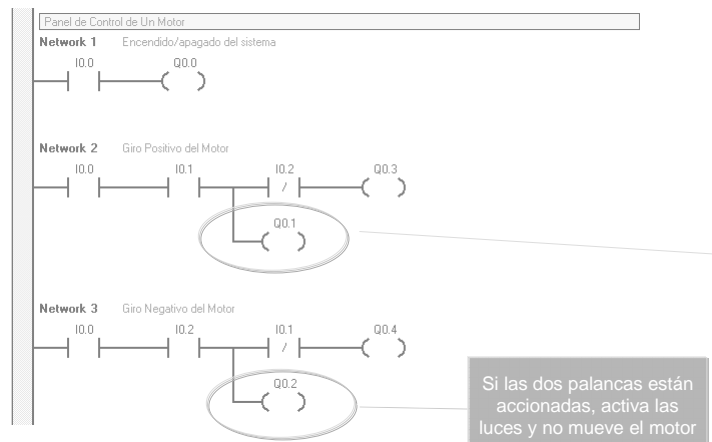
Problema:
Si no se activan las
dos palancas las
luces se apagan



ISA-UMH Lenguajes de Programación STEP7

Ejemplo: Control del panel de mando de un motor

■ Solución 2:



ISA-UMH Lenguajes de Programación STEP7

Ejemplo: Control del panel de mando de un motor

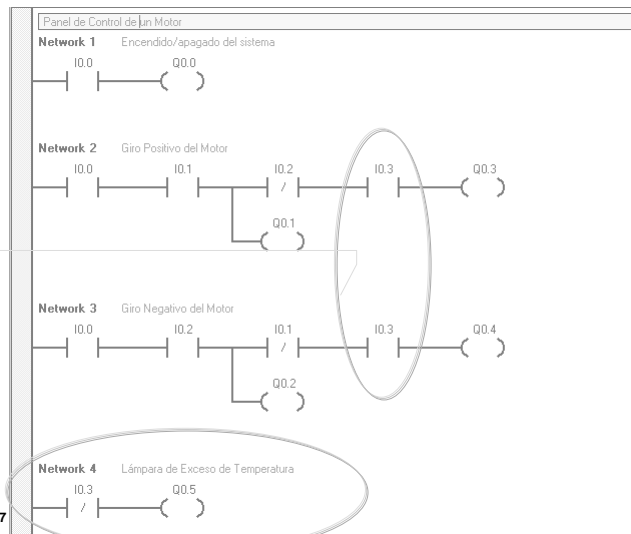
- **Funcionamiento Adicional: Sensor de Temperatura del devanado del motor**
 - Entradas:
 - Sensor Temperatura -> I0.3 (OK-> 24V, Exceso T^a-> 0V)
 - Salidas:
 - Lámpara Exceso T^a -> Q0.5 (Encendida-> 24V, Apagada (OK)-> 0V)
- **Funcionamiento:**
 - Si la temperatura es excesiva para el motor

ISA-UMH Lenguajes de Programación STEP7

Ejemplo: Control del panel de mando de un motor

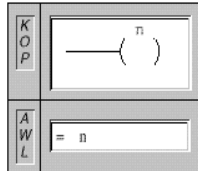
- **Funcionamiento Adicional: Sensor de Temperatura del devanado del motor**

Si la T^a es excesiva (0V) se para el motor



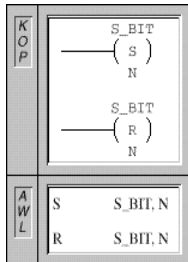
ISA-UMH Lenguajes de Programación STEP7

Operaciones con Salidas



■ Asignar

- Al ejecutar la operación Asignar se activa/desactiva el parámetro indicado (n).
- Operandos:
 - n: I, Q, M, SM, T, C, V, S

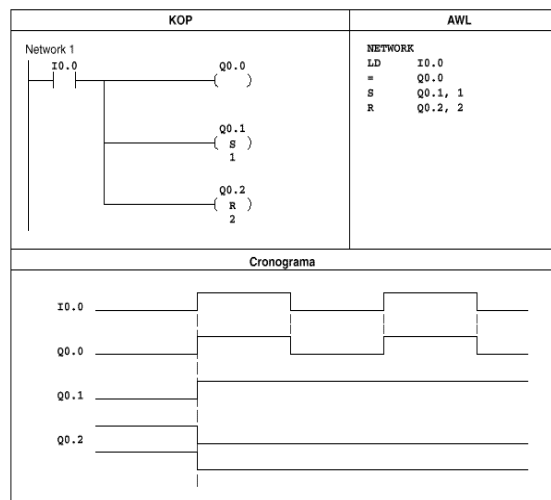


■ Poner a 1 (SET), Poner a cero (RESET)

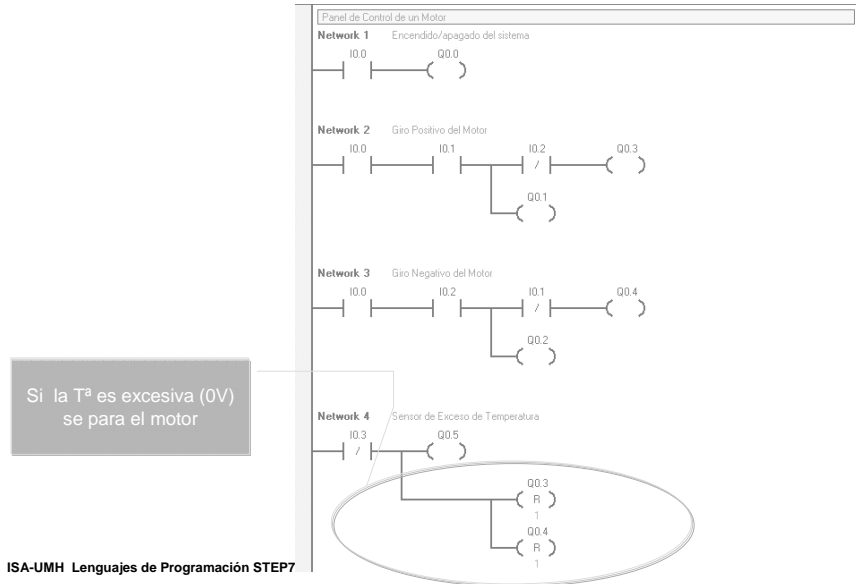
- Al ejecutar las operaciones **Poner a 1** y **Poner a 0**, se activa (se pone a 1) o se desactiva (se pone a 0) el número indicado de entradas y/o salidas (N) a partir de S_BIT, respectivamente.
- Operandos:
 - S_BIT: I, Q, M, SM, T, C, V, S
 - N: IB, QB, MB, SMB, VB, AC, constante (1-255)

Operaciones con Salidas

■ Ejemplo:



Ejemplo SET/RESET: Control del panel de mando de un motor



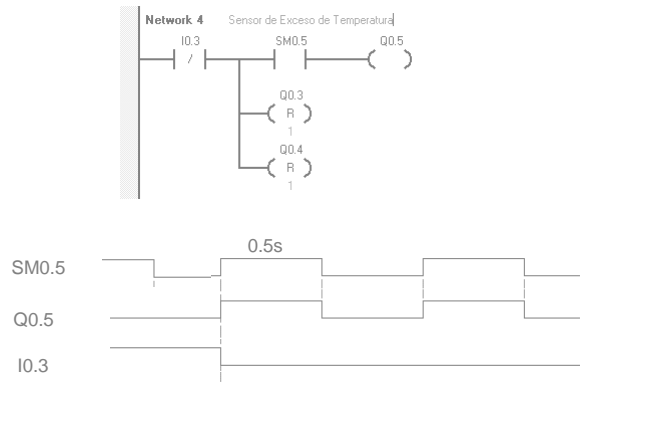
Marcas

- Equivalen a las variables en otros lenguajes
- Las marcas internas (área de marcas M) se pueden utilizar como relés de control para almacenar el estado intermedio de una operación u otras informaciones de control
 - Permiten almacenar información entre diferentes ciclos de ejecución del autómata
- Direccionamiento del área de marcas (**M**)
 - Formato:

● Bit	M [direcc. del byte].[direcc. del bit]	M26.7
● Byte/word/double	M [tamaño][direcc. del byte inicial]	MD20
- Las marcas especiales representan estados internos del firmware del autómata:
 - Un bit que se activa sólo en el primer ciclo. **SM0.1**
 - Bits que se activan de forma periódica. **SM0.5**
 - Bits que muestran el estado de operaciones matemáticas y de otras operaciones. **SM0.0 (bit sp: saltos condicionales)**

Ejemplo Marcas Especiales: Control del panel de mando de un motor

- Parpadeo del LED de temperatura excesiva
 - Uso de la marca especial SM0.5



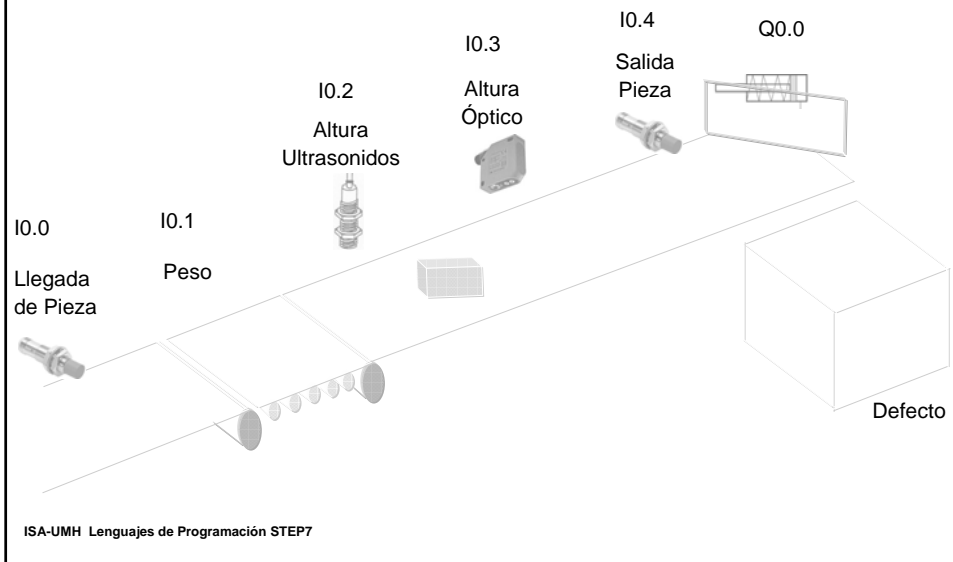
ISA-UMH Lenguajes de Programación STEP7

Ejemplo: uso de Marcas/Biestables

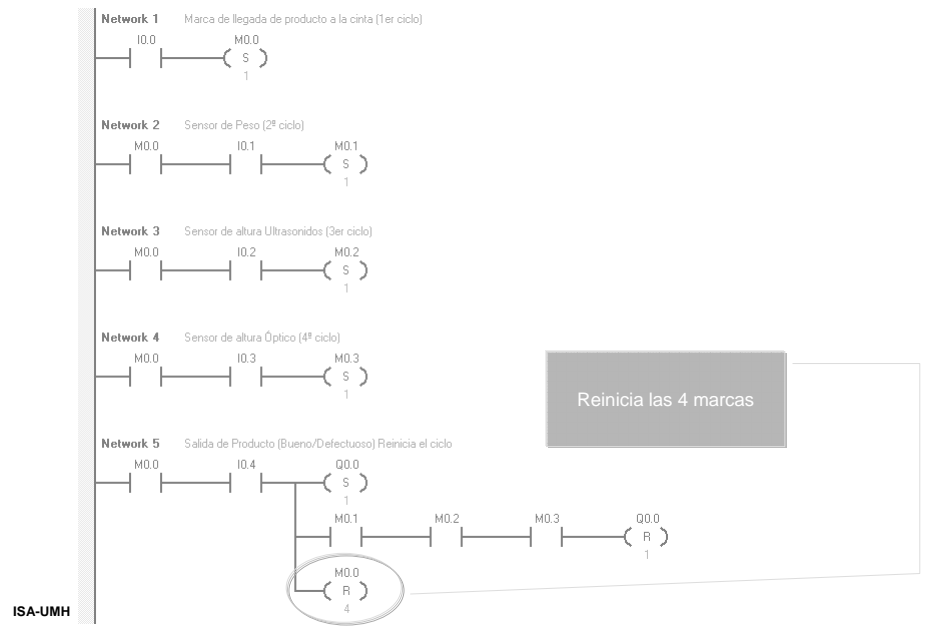
- Cinta transportadora:
 - Se realizan tres medidas sucesivas (en diferentes ciclos de ejecución) de una pieza para determinar si es defectuosa:
 - Entradas:
 - Llegada de pieza entrada I0.0 (Pieza->24V, No pieza->0V)
 - Peso de pieza (célula de carga) I0.1 (OK->24V, Mal->0V)
 - Altura (ultrasonidos) I0.2 (OK->24V, Mal->0V)
 - Altura (óptico) I0.3 (OK->24V, Mal->0V)
 - Llegada de pieza salida I0.4 (Pieza->24V, No pieza->0V)
 - Salidas:
 - Trampilla neumática Q0.0 (Bien->abierto(0V), Mal->Cerrado(24V))

ISA-UMH Lenguajes de Programación STEP7

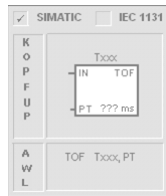
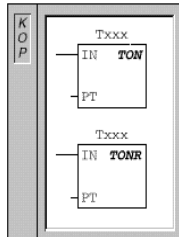
Cinta Transportadora



Ejemplo: uso de Marcas/Biestables



Operaciones con Temporizadores

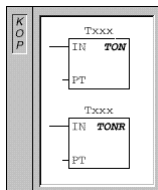


ISA-UMH Lenguajes de Programación STEP7

- Temporizador de retardo a la conexión (TON)
- Temporizador de retardo a la conexión memorizado (TONR)
 - Empiezan a contar hasta el valor máximo al ser habilitadas. Si el valor actual (Txxx) es mayor o igual al valor de preselección (PT), se activa el bit de temporización.
 - Cuando se inhibe la operación,
 - el temporizador de retardo a la conexión se pone a 0,
 - el temporizador de retardo a la conexión memorizado se detiene pero no se pone a 0
 - Ambos temporizadores se detienen al alcanzar el valor máximo.
- Temporizador de retardo a la desconexión (TOF):
 - Empieza a contar cuando IN está a nivel bajo. Al pasar a nivel alto se resetea
 - Cuando IN se activa, el bit de temporización es 1. Al superar el valor PT el estado pasa a valor 0

33

Operaciones con Temporizadores



- Operandos:

Txxx:

	TON / TOF	TONR
1 ms	T32, T96	T0, T64
10 ms	T33 a T36	T1 a T4
	T97 a T100	T65 a T68
100 ms	T37 a T63	T5 a T31
	T101 a T255	T69 a T95

PT: VW, T, C, IW, QW, MW, SMW, AIW, constante,

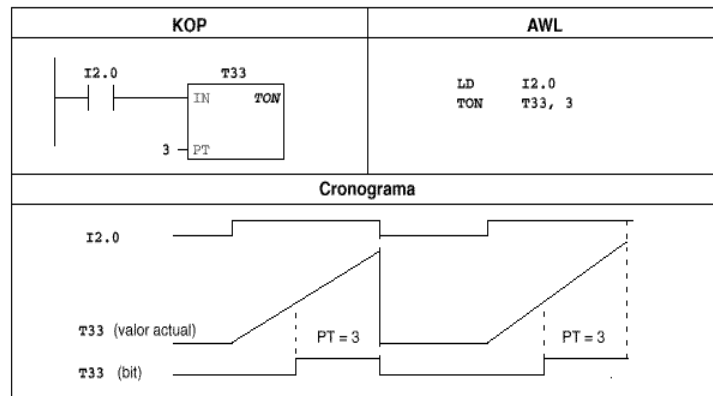
- Por ejemplo, el valor de conteo 50 en un temporizador de 100 milisegundos (ms) equivale a 5000 ms = 5 seg.

ISA-UMH Lenguajes de Programación STEP7

34

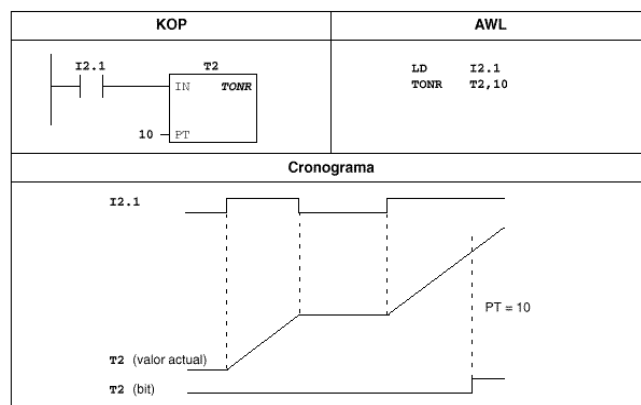
Operaciones con Temporizadores

■ TON



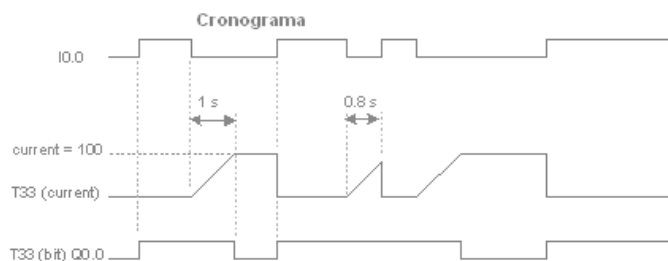
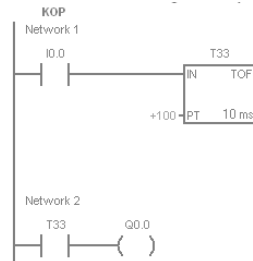
Operaciones con Temporizadores

■ TONR



Operaciones con Temporizadores

■ TOF



ISA-UMH

37

Operaciones con Temporizadores

- Direccionamiento del área de temporizadores (T)
- Hay dos variables asociadas a los temporizadores:
 - Valor actual: En este número entero de 16 bits con signo se deposita el valor de tiempo contado por el temporizador.
 - Bit del temporizador (bit T): Este bit se activa (se pone a 1) cuando el valor actual del temporizador es mayor o igual al valor predeterminado. (Este último se introduce como parte de la operación).
- A estas dos variables se accede
 - Formato: T [número del temporizador] Ej. T24
- Las operaciones con operandos en formato de bit acceden al bit del temporizador, en tanto que las operaciones con operandos en formato de palabra acceden al valor actual.
- Una operación de RESET sobre un temporizador inicializa el contador

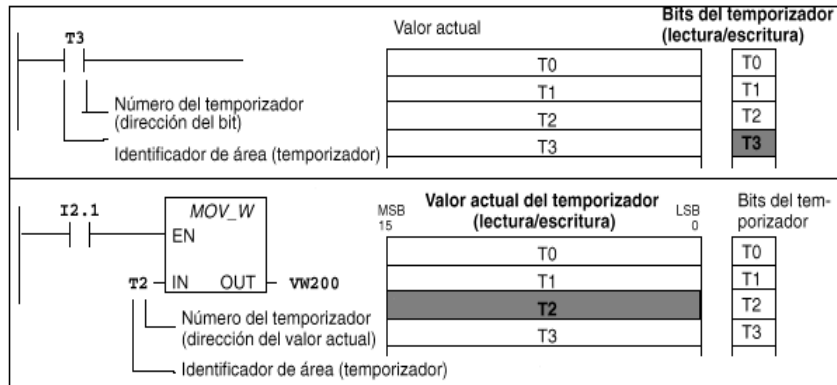


ISA-UMH Lenguajes de Programación STEP7

38

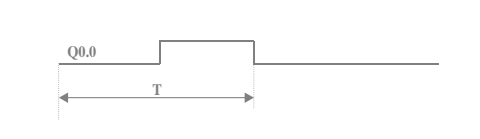
Operaciones con Temporizadores

■ Direccionamiento:

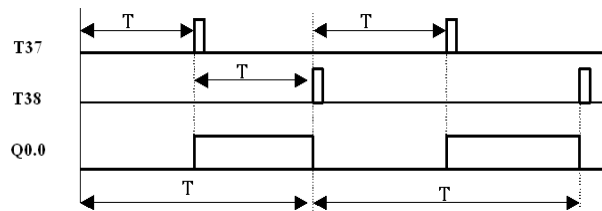


Ejemplo uso Temporizadores

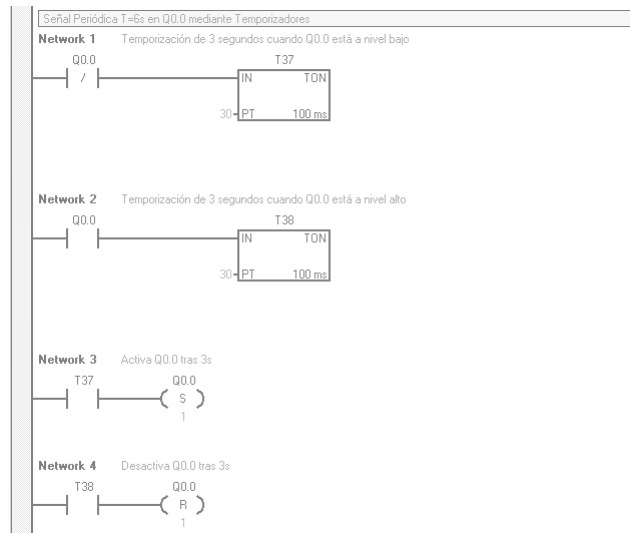
- Realizar el programa de control que obtenga en la salida Q0.0 una señal periódica de período 6 segundos.



- Para conseguir una señal periódica se utilizan dos temporizadores con retardo a la conexión TON, T37 y T38.



Ejemplo uso Temporizadores



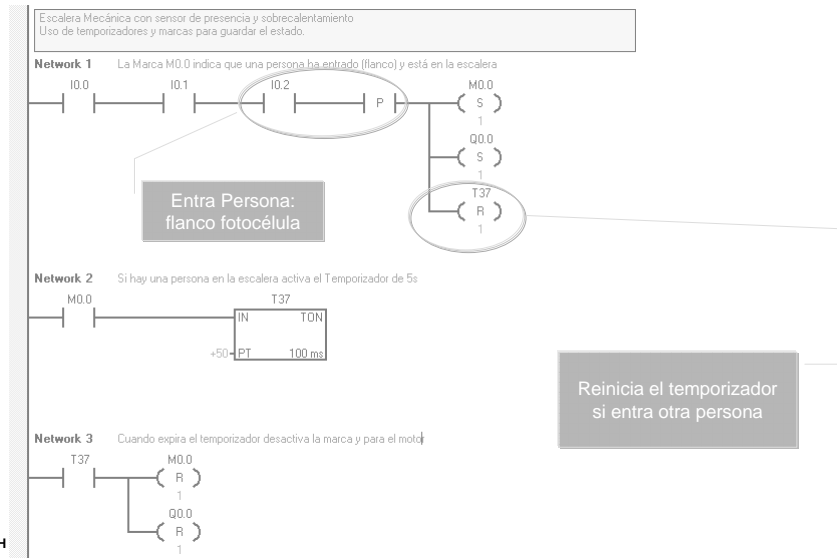
Ejemplo uso Temporizadores:

Automatización de una escalera mecánica

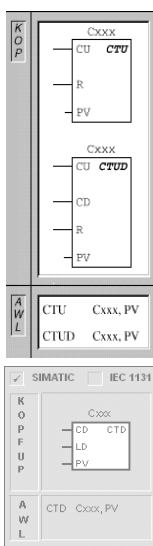
- El control del motor de una escalera automática consta de un interruptor de encendido y apagado (ON/OFF), un sensor de temperatura para detectar sobrecalentamientos y una célula fotoeléctrica a la entrada de la misma para detectar el paso de personas.
- Se desea diseñar el control de funcionamiento de la misma teniendo en cuenta que el tiempo estimado en recorrer todo el trayecto es 5 seg.
- *Nota: Cada vez que detecte a una persona reiniciará la temporización. Nadie a mitad del recorrido.*
- *En caso de sobrecalentamiento se debe acabar el ciclo de temporización.*
- Señales involucradas:
 - I0.0 ON/OFF
 - I0.1 Protección del motor ("1" funcionamiento correcto)
 - I0.2 Fotocélula (Detección paso personas por flanco positivo)
 - Q0.0 Acciona el motor

Ejemplo uso Temporizadores:

Automatización de una escalera mecánica



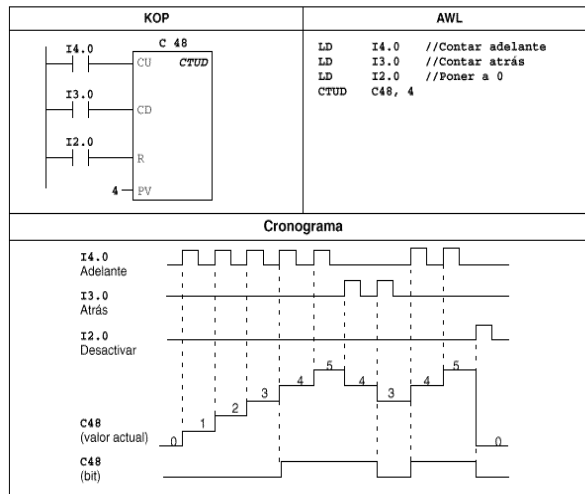
Operaciones con Contadores



■ Contar adelante, Contar adelante/atrás

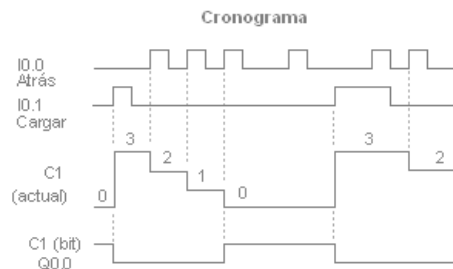
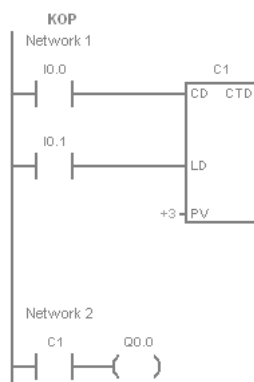
- La operación **Contar adelante** (CTU)
 - empieza a contar hasta el valor máximo cuando se produce un flanco positivo en la entrada de contaje adelante (CU).
 - Si el valor actual (Cxxx) es mayor o igual al valor de preselección (PV), se activa el bit de contaje (Cxxx).
 - El contador se inicializa (0) al activarse la entrada de desactivación (R).
- La operación **Contar adelante/atrás** (CTUD)
 - empieza a contar adelante o atrás cuando se produce un flanco positivo en la entrada de contaje adelante (CU) o atrás (CD).
- La operación **Contar atrás** (CTD).
 - PV es el valor inicial. Se decrementa con un flanco negativo en CD. Se activa el bit de contaje si el contador se hace 0. Al activarse LD se inicializa a PV.
- Operandos:
 - Cxxx: 0 a 255
 - PV: VW, T, C, IW, QW, MW, SMW, AC, AIW, constante

Operaciones con Contadores



Operaciones con Contadores

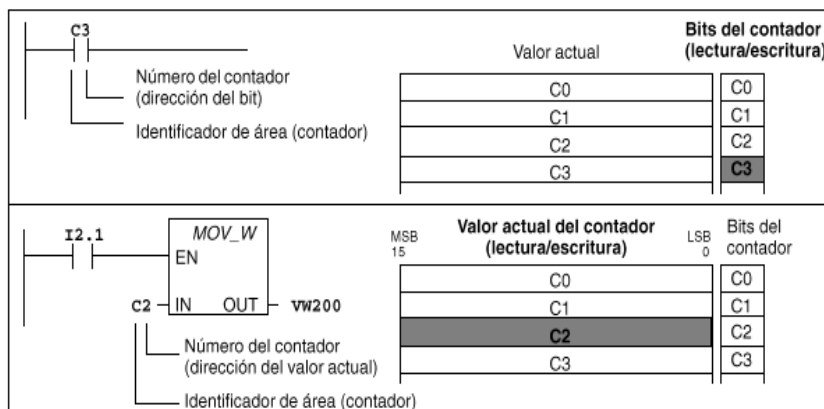
■ Contador Atrás (CD)



Operaciones con Contadores

- Direccionamiento de los contadores (C)
 - Hay dos variables asociadas a los contadores:
 - Valor actual: En este número entero de 16 bits con signo se deposita el valor de conteo acumulado.
 - Bit del contador (bit C): Este bit se activa (se pone a 1) cuando el valor actual del contador es mayor o igual al valor predeterminado. (Este último se introduce como parte de la operación).
 - A estas dos variables se accede utilizando la dirección del contador (C + número del contador).
 - Dependiendo de la operación utilizada, se accede al bit del contador o al valor actual.
 - Formato: C [número del contador] Ej. C20
- Una operación (bobina) de RESET sobre un contador inicializa el valor del contador
- El contador puede inicializarse a cualquier valor utilizando operaciones de transferencia (MOV_W)

Operaciones con Contadores



Operaciones de Comparación

- Las operaciones disponibles permiten comparar bytes (B), enteros de 2 bytes (I), enteros dobles de 4 bytes (D), reales (R), y cadenas de texto (S).

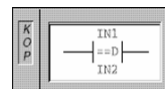
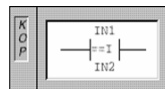
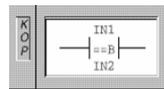
Elemento en KOP	Descripción
	RLO = 1 IN1 == IN2 RLO = 0 IN1 != IN2
	RLO = 1 IN1 != IN2 RLO = 0 IN1 == IN2
	RLO = 1 IN1 >= IN2 RLO = 0 IN1 < IN2
	RLO = 1 IN1 <= IN2 RLO = 0 IN1 > IN2
	RLO = 1 IN1 > IN2 RLO = 0 IN1 <= IN2
	RLO = 1 IN1 < IN2 RLO = 0 IN1 >= IN2

- Las comparaciones de bytes no llevan signo.
- Las comparaciones de palabras y palabras dobles sí que llevan signo (el bit más significativo indica el signo: 0 = + y 1 = -)

Hex: 7FFF > 8000

Bin: 0111111111111111 > 1000000000000000]

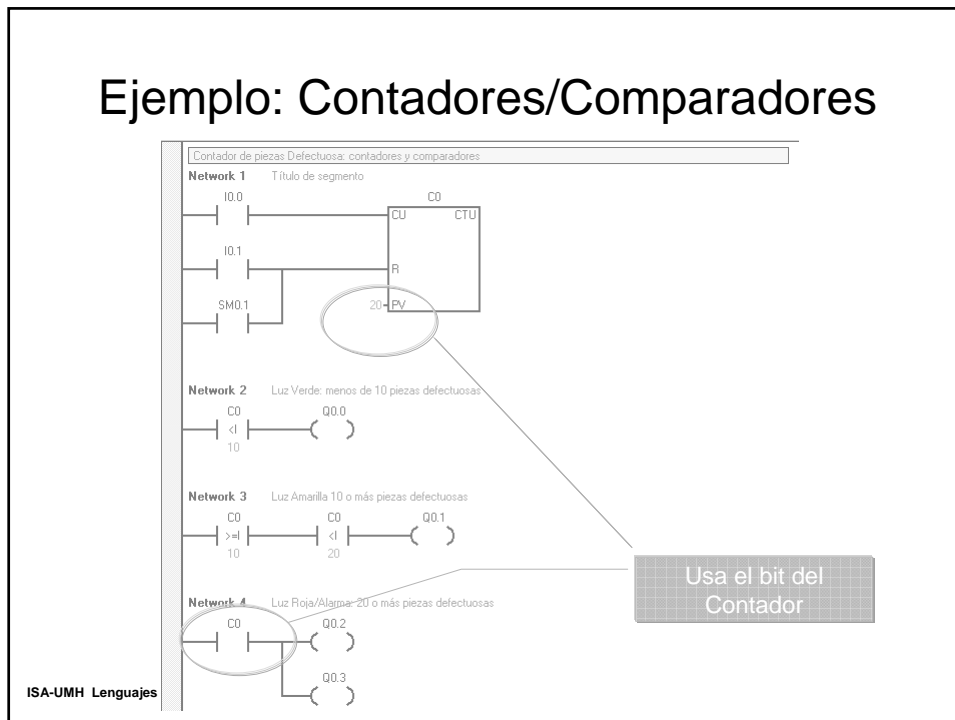
Dec: + 32767 > - 0



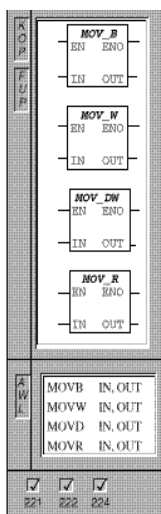
Ejemplo: Contadores/Comparadores

- En un proceso se cuenta el número de piezas defectuosas fabricadas, y se indica el estado de la máquina con tres luces:
 - Luz Verde se activa si hay menos de 10 piezas defectuosas
 - Luz Naranja se activa si hay entre 10 y <20 piezas defectuosas
 - Luz Roja se activa si hay 20 o más piezas defectuosas
- Al producirse más de 20 piezas defectuosas se hace sonar la alarma
 - Conexiones:
 - Q0.0 Luz Verde ON (menos de 10 piezas defectuosas)
 - Q0.1 Luz Naranja ON (10 a 20 piezas defectuosas)
 - Q0.2 Luz Roja ON (20 o más piezas defectuosas)
 - Q0.3 Alarma ON (20 o más piezas defectuosas)
 - I0.0 Sensor pieza defectuosa (0V: pieza OK, 24V: pieza Defectuosa)
 - I0.1 Pulsador de Reset de la Máquina (24V: pulsado)

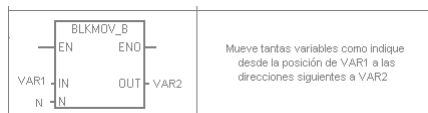
Ejemplo: Contadores/Comparadores



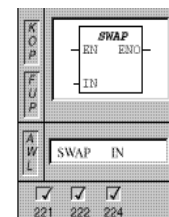
Operaciones de Transferencia



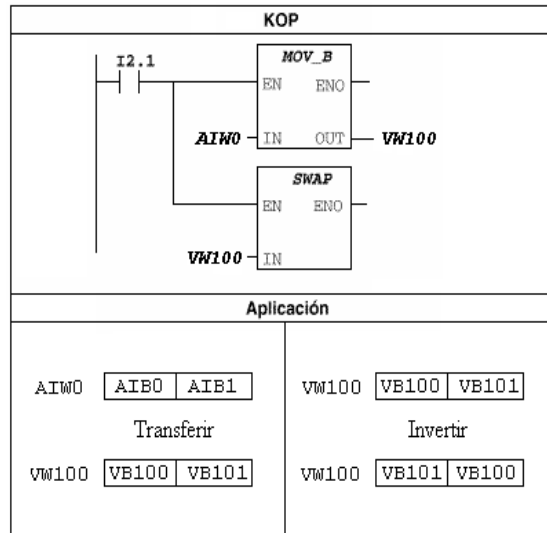
- Transferir byte, Transferir palabra, Transferir palabra doble y Transferir real
 - Las operaciones de transferencia se utilizan para transferir datos de una dirección a otra.
 - La transferencia se produce en cada ciclo de ejecución si la entrada EN está a nivel alto.
 - BLKMOV_? : permite mover array de datos



- La operación SWAP (Invertir bytes) de una palabra intercambia el byte más significativo y el byte menos significativo de una palabra (IN).



Operaciones de Transferencia



Operaciones aritméticas

Elemento en KOP	Descripción
	Si EN activo: $VAR3 = VAR1 + VAR2$
	Si EN activo: $VAR3 = VAR1 - VAR2$
	Si EN activo: $VAR3 = VAR1 * VAR2$
	Si EN activo: $VAR3 = VAR1 / VAR2$

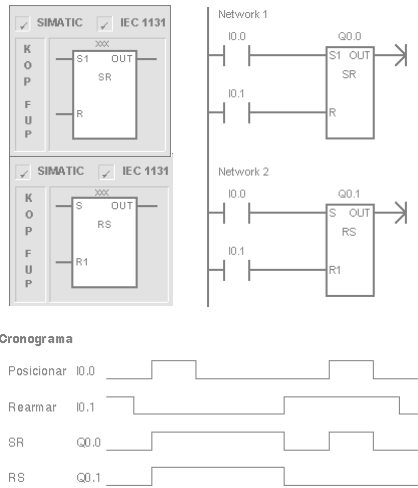
- Operaciones aritméticas entre variables de tipo entero con signo (I 16 bits), entero doble (DI 32 bits) y Reales (R):

- La operación se realiza en cada ciclo de ejecución si la entrada EN está a nivel alto.
- SM1.1 indica errores de desbordamiento o error en los datos de entrada
- SM1.3 indica error de división por 0
- SM1.0 indica si el resultado es cero (*flag Z de la CPU*)
- SM1.2 indica si el resultado es negativo (*flag N de la CPU*)

- EN0

- 1 si el resultado es válido
- 0 si se ha activado un flag de error (SM1.1, SM1.3)

Cuadros Biestables



- **Biestables SR, RS:**
 - Permiten combinar las operaciones de SET y RESET en un solo cuadro
 - Utilizan un Dirección de memoria global para almacenar el estado (Operando)
 - Disponen de una conexión de salida para continuar el esquema
 - Su ejecución se realiza en una sola etapa por lo que puede usarse la dirección de memoria en la condición.
 - Útiles para conmutar el estado de un bit de forma condicionada
- **Diferencias: si ambas entradas están activas:**
 - Biestable SR: prioridad al set
 - Biestable RS: prioridad al reset

ÍNDICE

- **Introducción a SETP 7: KOP**
 - Distribución de la memoria
 - Lenguaje KOP: Diagrama de contactos
 - Operaciones básicas: contactos y salidas
 - Operaciones con temporizadores
 - Operaciones con contadores
 - Operaciones de comparación
 - Operaciones de transferencia
 - Operaciones aritméticas
 - Tabla de Símbolos
 - Ejemplos

Tabla de Símbolos

- Para hacer mas legibles los diagramas de contactos se pueden utilizar símbolos descriptivos para las variables del proceso (entradas, salidas, marcas, variables...)
- La tabla de Símbolos contiene la asignación de los símbolos a direcciones de memoria
- La 'compilación' del diagrama convierte los símbolos en las referencias de memoria que son enviadas al autómata.

		Símbolo	Dirección	Comentario
1		S1	I0.0	Sensor de presencia antes barrera de entrada
2		S2	I0.1	Sensor de presencia detrás barrera de entrada
3		S3	I0.2	Sensor de presencia antes barrera de salida
4		S4	I0.3	Sensor de presencia detrás barrera de salida
5		S5	I0.4	Sensor ficha salida (24V OK)
6		Marcha	I0.5	Botón de marcha (24V activado)
7		Paro	I0.6	Botón de Paro (24V activado)
8		Reset	I0.7	Botón de Reset contador (24V activado)
9		Motor1	Q0.0	Motor barrera entrada
10		Motor2	Q0.2	Motor barrera salida
11		LV	Q0.3	Luz verde
12		LR	Q0.4	Luz Roja

ISA-UMH Lenguajes de Programación STEP7

ÍNDICE

- Introducción a SETP 7: KOP
 - Distribución de la memoria
 - Lenguaje KOP: Diagrama de contactos
 - Operaciones básicas: contactos y salidas
 - Operaciones con temporizadores
 - Operaciones con contadores
 - Operaciones de comparación
 - Operaciones de transferencia
 - Operaciones aritméticas
 - Tabla de Símbolos
 - Ejemplos

ISA-UMH Lenguajes de Programación STEP7

58

Ejercicio: Puerta Garaje

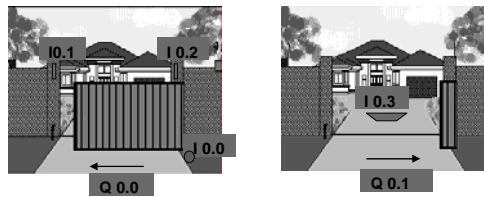
- Cuando se accione el pulsador de apertura de puerta, la puerta se abre (si no estaba abierta) y cuando el vehículo se encuentra en el interior del recinto y presiona el sensor de paso la puerta se cierra.

- ENTRADAS :

- I 0.0 : Pulsador de apertura de puerta.
- I 0.1 : Sensor de fin de carrera (puerta cerrada)
- I 0.2 : Sensor de fin de carrera (puerta abierta)
- I 0.3 : Sensor de paso de vehículo.

- SALIDAS :

- Q 0.0 : Cerrar puerta
- Q 0.1 : Abrir puerta



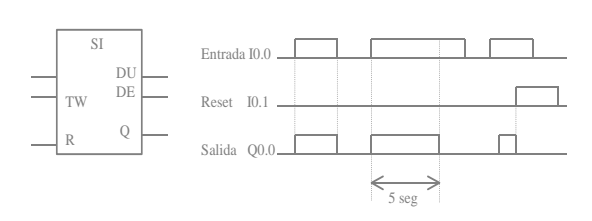
	🔍	📄	Símbolo	Dirección	Comentario
1			Pulsador	I0.0	Pulsador de entrada
2			FC_Close	I0.1	Final de Carrera puerta Cerrada
3			FC_Open	I0.2	Final de Carrera puerta Abierta
4			S1	I0.3	Sensor de presión Vehículo dentro
5			MotorClose	Q0.0	Motor puerta [Cerrar]
6			MotorOpen	Q0.1	Motor puerta [Abrir]

59

Ejercicios

- Temporizadores:

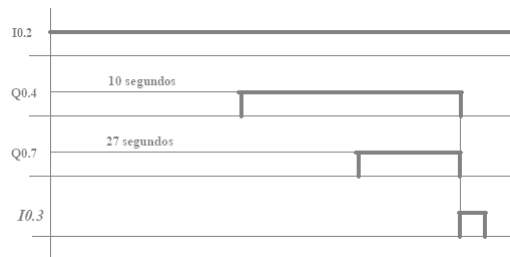
- Utilizando temporizadores emular el funcionamiento del temporizador de impulso (SI)



Ejercicios

■ Temporizador:

- Mediante el uso de un solo temporizador (T37) active las salidas Q0.4 y Q0.7 una vez hayan transcurridos 10 y 27 segundos respectivamente desde que se produjo la activación de la entrada I0.2.
- Mediante la entrada I0.3 se vuelve a las condiciones iniciales Q0.4 y Q0.7 desactivadas.



Ejercicio: Parking 1

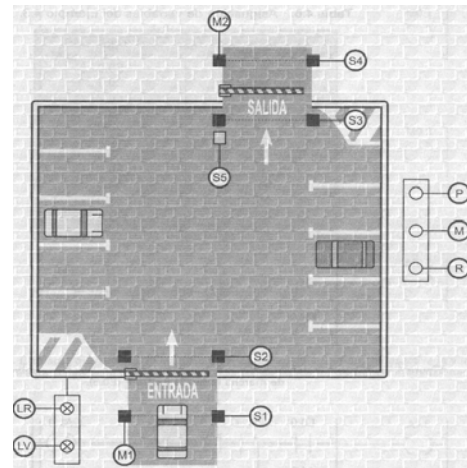
- Cuando llega un coche (fotocélula I0.0) y el parking esté libre, queremos que se abra la barrera (Q0.0). A la salida no tenemos barrera. Cuando sale un coche simplemente sabemos que ha salido (fotocélula I0.1).
- En el parking caben 10 coches. Cuando el parking tenga menos de 10 coches queremos que esté encendida la luz de libre (Q0.1). Cuando en el parking haya 10 coches queremos que esté encendida la luz de ocupado (Q0.2).
- Además queremos que si el parking está ocupado y llega un coche que no se le abra la barrera.
- Mediante la entrada I0.2 resetearemos el contador.



	↶	↷	Símbolo	Dirección	Comentario
1			Capacidad	+10	Número máximo de vehículos en el Parking
2			CicloInicial	SM0.1	Marca de Ciclo inicial (Reinicio)
3			Contador	C0	Contador Vehículos
4			S1	I0.0	Sensor Entrada vehículos
5			S2	I0.1	Sensor Salida vehículos
6			Reset	I0.2	Reset Contador
7			MotorBE	Q0.0	Motor Barrera de entrada (1-> abierta, 0-> cerrada)
8			LV	Q0.1	Luz Verde (Libre)
9			LR	Q0.2	Luz Roja (Ocupado)

Ejercicio: Parking 2

- El garaje tiene capacidad para 10 vehículos.
- Tiene dos barreras: entrada y salida , con dos sensores de presencia cada una (antes y después de la barrera)
- Las barreras se abrirán si detecta vehículo en S1 (entrada) o S3+S5 (salida)
- Las barreras se cerrarán cuando todo el vehículo haya pasado ya por S2/S4
- El sensor S5 se activa con la ficha de control de la salida
- Luz Verde indica menos de 10 vehículos y luz roja (completo)
- Pulsadores (no guardan el estado):
 - Inicio (M)
 - Paro (P) (cierra la entrada y activa LR)
 - Reset (R) contador
- El Inicio/Paro no resetea el estado de ocupación del parking



ISA-UMH Lenguajes de Programación STEP7

Ejercicio: Parking 2

■ Tabla de Símbolos

			Símbolo	Dirección	Comentario
1			CicloInicia	SM0.1	Marca de Primer Ciclo/Reinicio
2			Capacidad	+10	Número máximo de vehículos
3			Contador	C2	Contador de vehículos
4			Estado	M0.0	Estado Marcha/Paro
5			SBE1	I0.0	Sensor de presencia antes barrera de entrada
6			SBE2	I0.1	Sensor de presencia detrás barrera de entrada
7			SBS3	I0.2	Sensor de presencia antes barrera de salida
8			SBS4	I0.3	Sensor de presencia detrás barrera de salida
9			STicket5	I0.4	Sensor ficha salida (24V DK)
10			Marcha	I0.5	Pulsador de marcha (24V activado)
11			Paro	I0.6	Pulsador de Paro (24V activado)
12			Reset	I0.7	Pulsador de Reset contador (24V activado)
13			MotorE1	Q0.0	Motor barrera entrada
14			MotorS2	Q0.1	Motor barrera salida
15			LV	Q0.2	Luz verde
16			LR	Q0.3	Luz Roja

ISA-UMH Lenguajes de Programación STEP7