

Tema 3. Lenguajes básicos de programación de PLCs (LD, FBD e IL)

AUTOMATIZACIÓN. CURSO 2022-2023

Fernando R. Pardo Seco – fernando.pardo@usc.es

Introducción a la programación de PLCs

ESPECIFICACIONES



MODELO SISTEMA
DE CONTROL

ANÁLISIS



ASIGNACIÓN
E/S



INDUSTRIAL
AUTOMATION

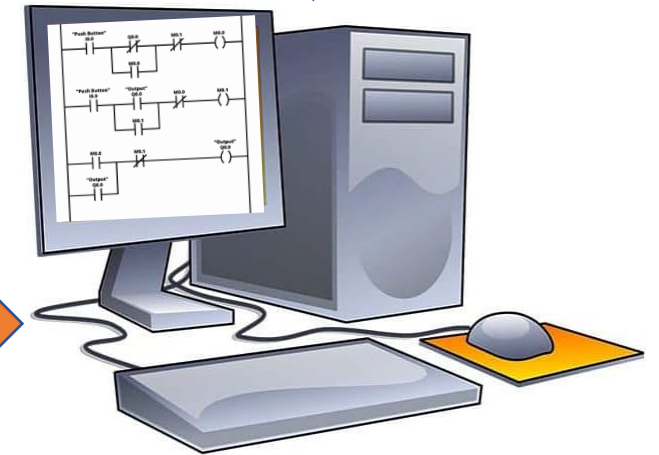
AUTÓMATA PROGRAMABLE - PLC

SISTEMA OPERATIVO - FIRMWARE

CPU

INTERFACES E/S

MEMORIA



UNIDAD DE PROGRAMACIÓN:

- Lenguaje
- Editor



SEÑALES DE EMERGENCIA

Introducción a la programación de PLCs

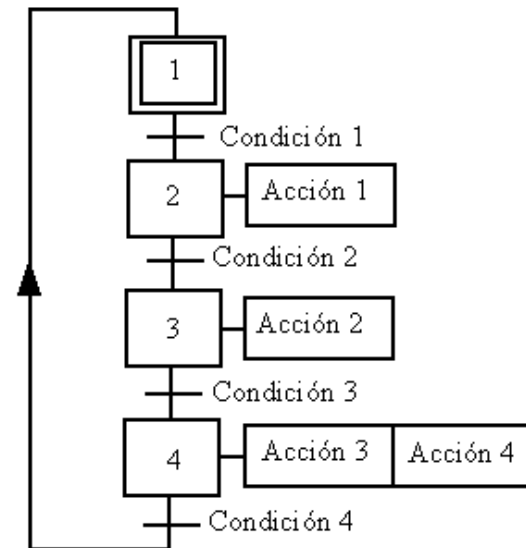
- Se puede dividir la programación del PLC en etapas:
 1. Definir el sistema de control: ¿Qué debe hacer? ¿En qué orden?.....: Diagrama de flujo, descripción literal GRAFCET (Tema 4).
 2. Identificar señales de Entrada y de salida al PLC
 3. Modelar el sistema de control: Relaciones entre variables, secuencias. Algebraica o gráfica.
 4. Asignar señales de E/S o señales internas del PLC a las variables
 5. Codificar la representación del modelo: Lenguaje de programación
 6. Simular el programa (Recomendable) y depurar
 7. Carga el programa en el PLC
 8. Depurar el programa. Backup.

Introducción a la programación de PLCs

- Definición sistema de control:
 - Sistemas sencillos: Descripción en lenguaje vulgar
 - Representación basada en símbolos:
 - Proposicional: descripción literal
 - Algebraica: funciones booleanas y aritméticas
 - Gráfica: esquema contacto, diagramas lógicos, diagramas de flujo , GRAFCET
 - Descripción literal:
 - Descripción exhaustiva del proceso de control enumerando literalmente las acciones a realizar e incluyendo las condiciones de transición y/o habilitación.
 - Puede ser de difícil comprensión

Introducción a la programación de PLCs

- Definición sistema de control:
 - Funciones algebraicas:
 - Cada salida se obtiene a partir de:
 - Directamente de la descripción del proceso a controlar
 - Utilizando tabla de verdad, minimización Karnaugh,...
 - Dificultad para sistemas secuenciales
 - Diagramas de flujo: Tema 2
 - GRAFCET: Tema 4



Introducción. Tipos de datos IEC61131-3

Denominación	Bits	Ejemplo	Descripción
BOOL	1	FALSE or TRUE	Variable binaria o lógica
INT	16	-32768 a 32767	Entero con signo
UINT	16	0 a 65535	Entero sin signo
REAL	32	0.3456	Real
BYTE	8	0 a 255	Conjunto 8 bits (byte)
WORD	16	0 a 65535	Conjunto 16 bits (word)
DWORD	32	0 a $2^{32}-1$	Conjunto 32 bits (double Word)
TIME	32	T#3d6h3m45s2.8ms	Duración
DATE	16	D#2022-02-02	Fecha
TIME_OF_DAY	32	TOD#17:55:23.34	Hora del día
DATE_AND_TIME	64	DT#2012-01-12-16:45:06.12	Fecha y Hora
STRING		'Hello World'	Cadena caracteres

Introducción. Funciones IEC61131-3

Numéricas de una variable		Aritméticas de dos o más operandos	
ABS	Valor absoluto	ADD	Suma
SQRT	Raíz cuadrada	MUL	Multiplicación
LN	Logaritmo natural	Aritméticas de dos operandos	
LOG	Logaritmo base 10	SUB	Resta
EXP	Exponencial natural	DIV	División
SIN	SENO (radianes)	MOD	Módulo
COS	COSENO (radianes)	EXPT	Elevación a exponente
TAN	TANGENTE (radianes)	MOVE	Asignación
ASIN	ARCOSENO		
ACOS	ARCOCOSENO		
ATAN	ARCOTANGENTE		

Introducción. Funciones IEC61131-3

Desplazamiento o decalaje		Lógicas (Booleanas)	
SHL	Desplazamiento a izquierda	AND	Operación lógica Y
SHR	Desplazamiento a derecha	OR	Operación lógica O
ROR	Rotación a la derecha	XOR	Operación lógica O exclusiva
ROL	Rotación a la izquierda	NOT	Negación
Selección		Comparación	
SEL	Selección	GT	Mayor
MAX	Máximo	GE	Mayor o igual
MIN	Mínimo	EQ	Igual
LIMIT	Limitador	LE	Menor o igual
MUX	Multiplexor	LT	Menor
		NE	Distinto (No igual)

Introducción. Funciones IEC61131-3

Cadenas de caracteres			
LEN	Longitud	INSERT	Inserción
LEFT	Caracteres a la izquierda	DELETE	Borrado
RIGHT	Caracteres a la derecha	REPLACE	Sustitución
MID	Caracteres intermedios	FIND	Búsqueda
CONCAT	Concatenación		
Conversión de tipo			
XX TO XXX	Conversión de un tipo a otro		

Introducción. Bloque Funcional IEC61131-3

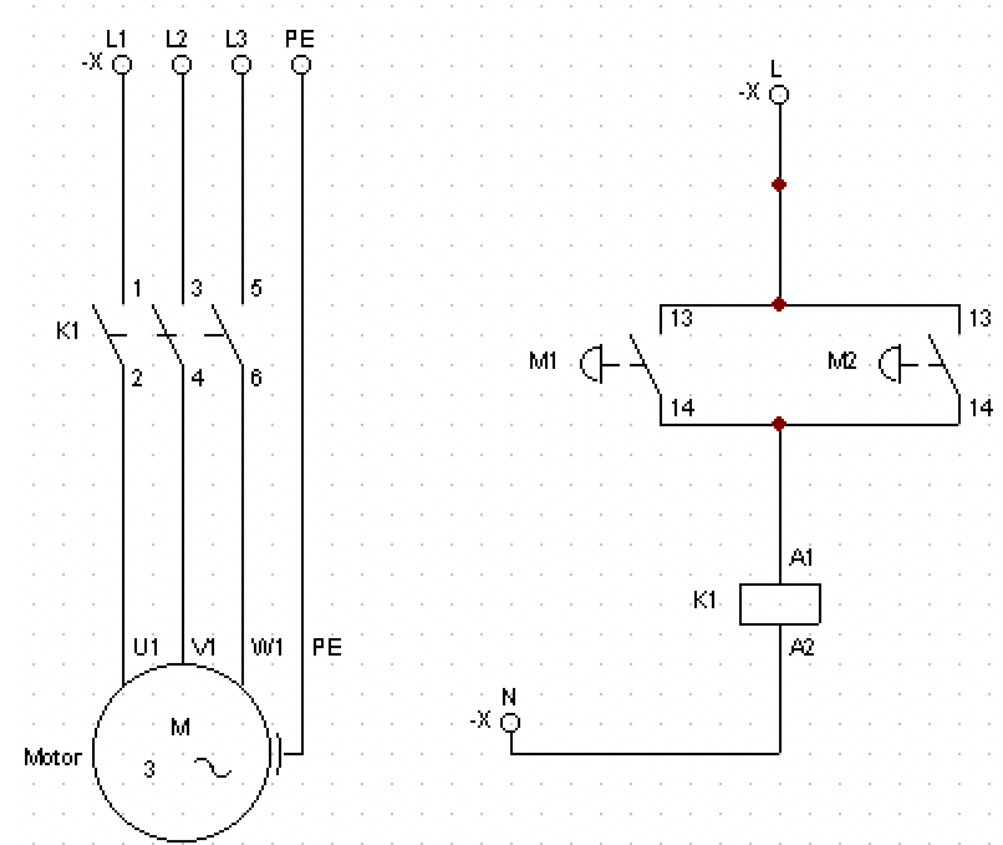
Bloque Funcional	Operadores	Descripción
SR	S1, R	Biestable RS de activación prioritaria
RS	S, R1	Biestable RS de desactivación prioritaria
R_TRIG	CLK	Convertidor de flancos ascendentes en impulso
F_TRIG	CLK	Convertidor de flanco descendente en impulso
CTU	CU, R, PV	Contador ascendente
CTD	CD, LD, PV	Contador descendente
CTUD	CU, CD, R, LD, PV	Contador reversible
TP	IN, PT	Temporizador de impulso
TON	IN, PT	Temporizador de retardo a la conexión
TOFF	IN, PT	Temporizador de retardo a la desconexión

Introducción. Variables IEC61131-3

Keyword	Utilización de la variable
VAR	Variable local interna de la unidad de organización en la que se declara
VAR_INPUT	Suministrada externamente. No modificable desde la unidad de organización
VAR_OUTPUT	Suministrada por la unidad de organización
VAR_IN_OUT	Suministrada externamente. Modificable desde la unidad de organización
VAR_EXTERNAL	Variable global suministrada externamente. Modificable desde la unidad de organización
VAR_GLOBAL	Variable global
VAR_ACCESS	Variable accesible a través de una vía de comunicación
RETAIN	Variable no volátil (Se mantiene aunque se deje de alimentar el sistema)
CONSTANT	Constante (Ejemplo binario 2#0000_0111 (7 decimal))
AT	Asignación local

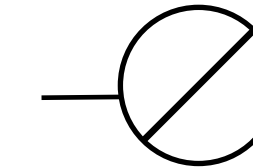
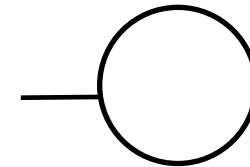
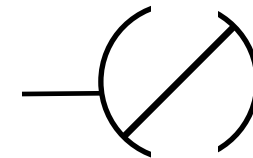
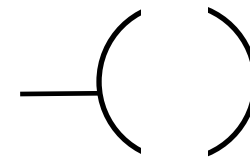
Lenguaje Ladder (LD)

- Antes de la aparición del PLC los automatismos eran realizados por electricistas.
- El lenguaje LADDER (escalera) surgió como lenguaje fácilmente entendible por electricistas
- Facilitó la transición de sistemas de control por relés al uso de PLCs
- Siemens: **KOP** (Kontaktplan)



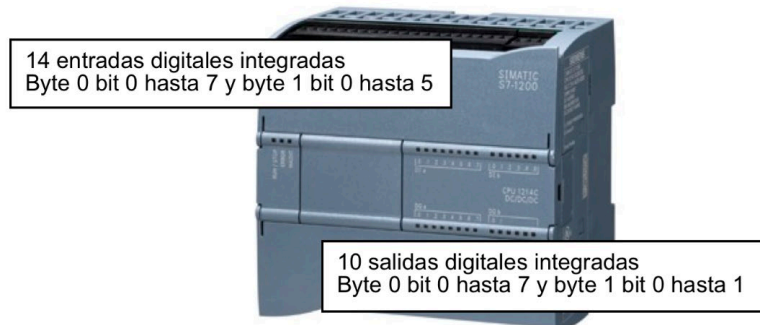
Lenguaje Ladder (LD)

- Un programa ladder es similar a un esquema con ramas de relés: Lenguaje gráfico.
- Cada rama contiene una serie de contactos (serie o paralelo) que proporcionan una salida (activación o desactivación de bobina).
- Las variables lógicas se representan mediante contactos: NO ó NA y NC.
- Símbolos estandarizados IEC-61131



Lenguaje Ladder (LD)

- Pequeñas diferencias entre fabricantes de PLCs
- 2 barras verticales (izquierda Vcc (24 V) y derecha GND)
- El flujo de la señal va de izquierda a derecha
- Cada símbolo va asociado a una posición de memoria (entradas o “relés internos”)
- Los contactos se representan con una letra y dos número que indican a el módulo y el borne



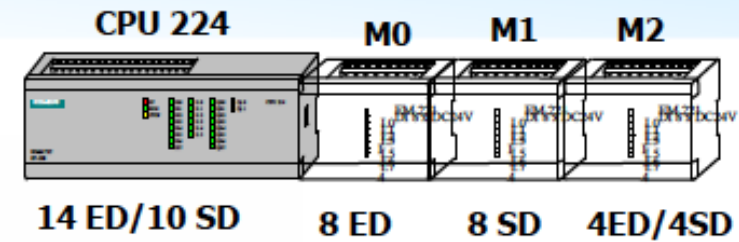
Entrada: %I0.3: Dirección de entrada 0 (byte) y bit 3 → Cuarta entrada

Salida: %Q1.1: Dirección de salida 1 (byte) y bit 1 → Décima salida

Lenguaje Ladder (LD)

MATRIZ I

	7	6	5	4	3	2	1	0
Fila 7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Fila 6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Fila 5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Fila 4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Fila 3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Fila 2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Fila 1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Fila 0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>



Modulo 2 de I3.0 a I3.3

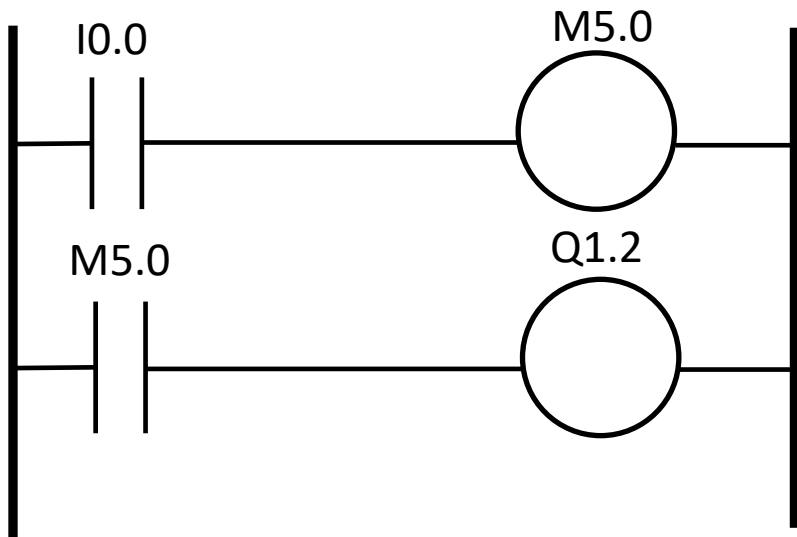
Modulo 0 de I2.0 a I2.7

CPU-224 de I1.0 a I1.5

CPU-224 de I0.0 a I1.7

Lenguaje Ladder (LD)

- Entradas: **Ix.x** o Ex.x (depende del fabricante) (I: Input – E: Eingang)
- Salidas: Sx.x, **Qx.x** o Ax.x (depende del fabricante) (A:Ausgang)
- En el PLC podemos usar “relés internos”, denominados **Marcas**, que son posiciones de memoria que indican el estado de un “*relé*”. Se suelen identificar con la letra M.

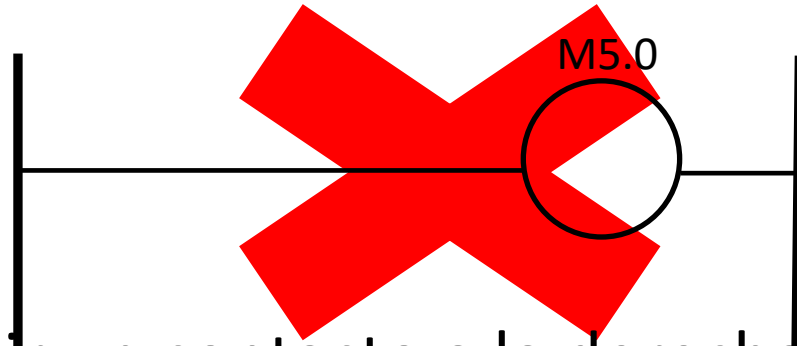


La bobina M5.0 depende de la entrada I0.0, pero se trata de una marca interna (no conectada a un borne de salida)

La salida Q1.2 (conectada a borne de salida) se activa a través de la activación del contacto M5.0 (marca)

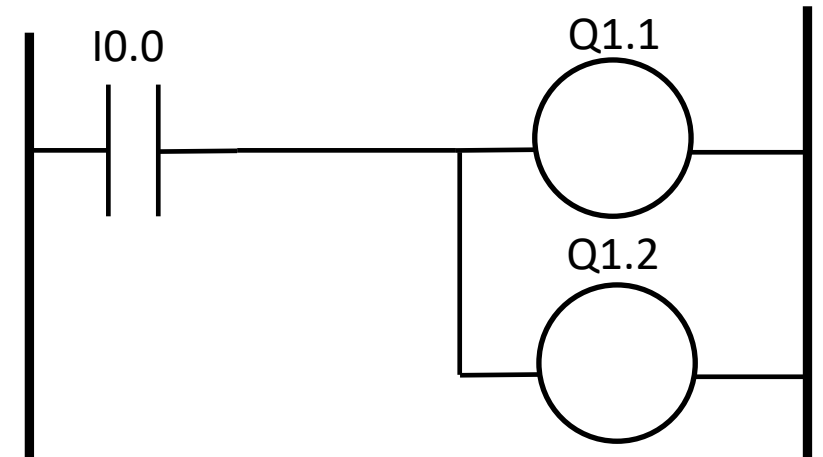
Lenguaje Ladder (LD)

- Una bobina no puede estar conectada directamente a la barra de inicio (Vcc)

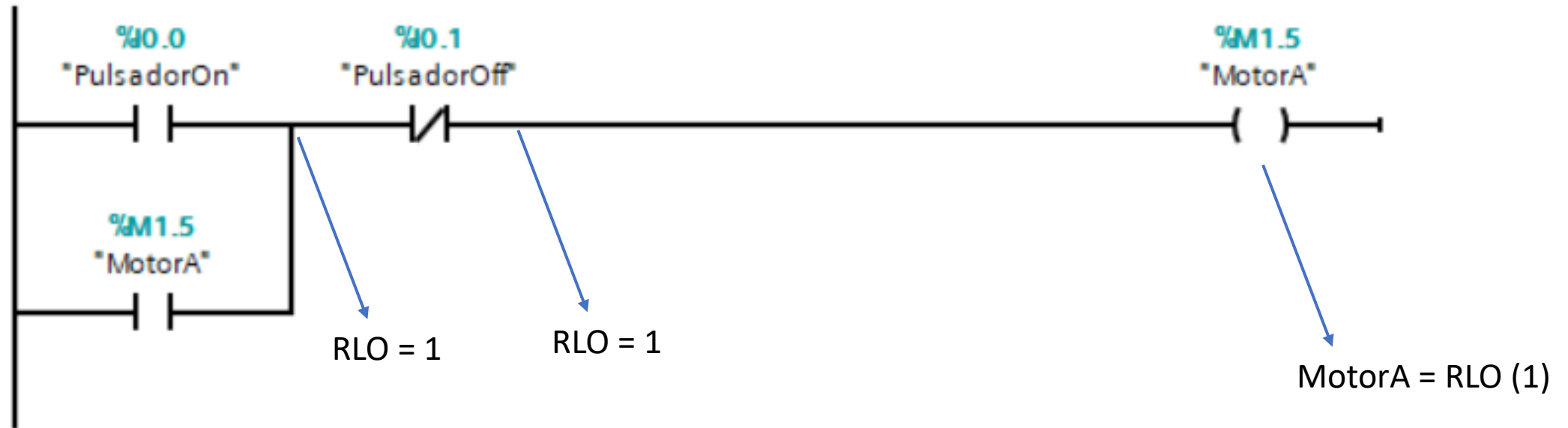


- No se puede introducir un contacto a la derecha de la bobina (todas las líneas acaban en una bobina).
- Contactos ilimitados en serie/paralelo
- Se puede usar una salida como entrada auxiliar
- Se pueden colocar en paralelo dos o más bobinas
- El resultado de la operación se guarda en **RLO**

RLO: Result of Logic Operation



Lenguaje Ladder (LD)



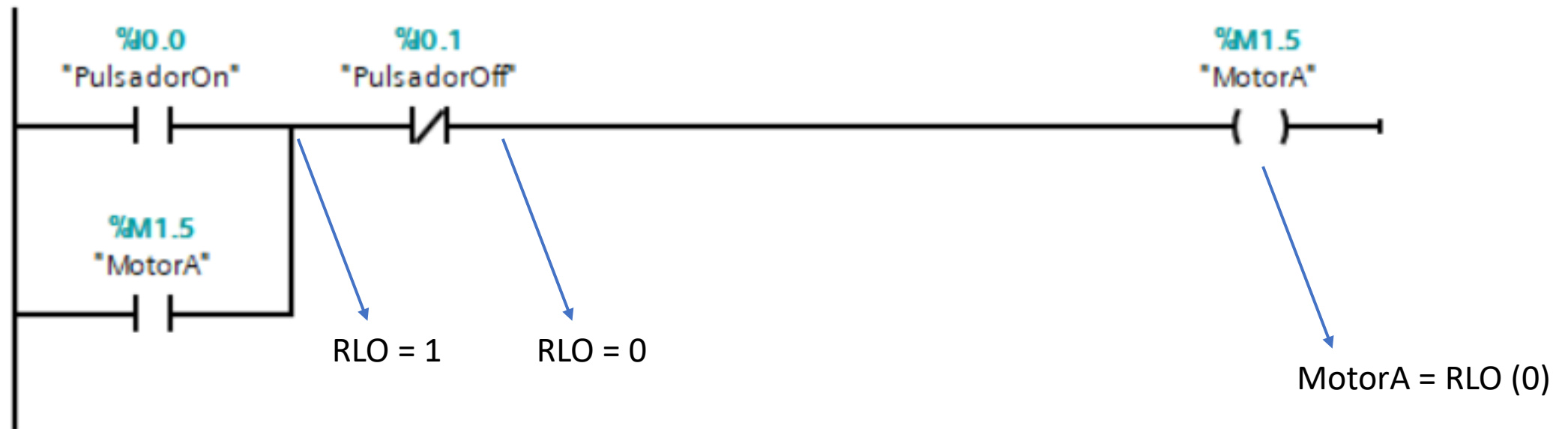
CICLO 1

PulsadorOn = 1

PulsadorOff = 0

MotorA = 1

Lenguaje Ladder (LD)



CICLO 2

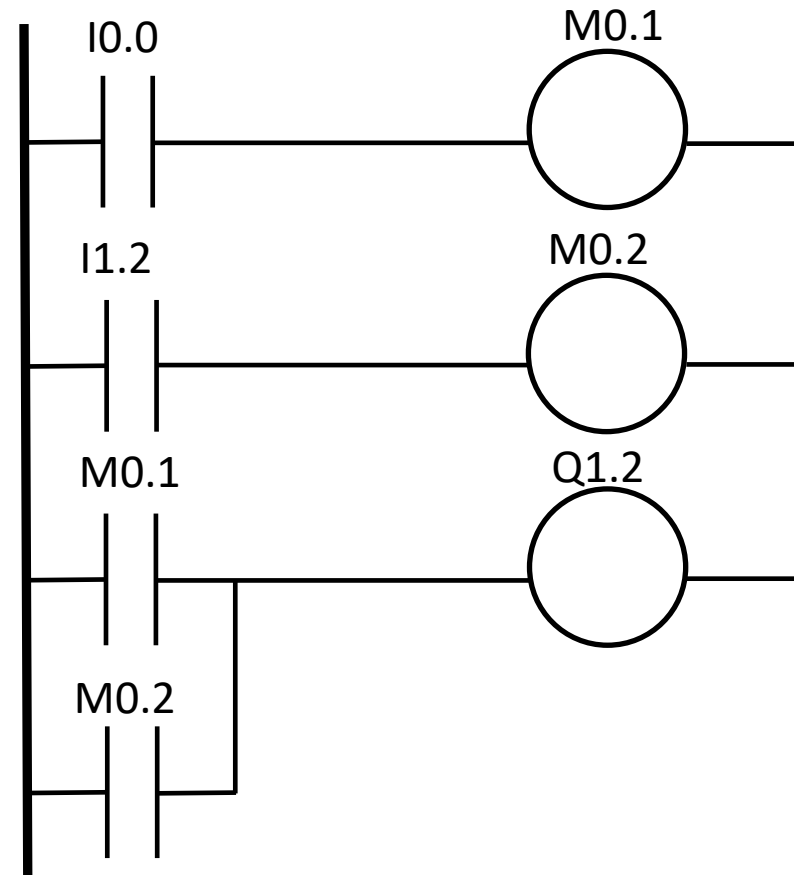
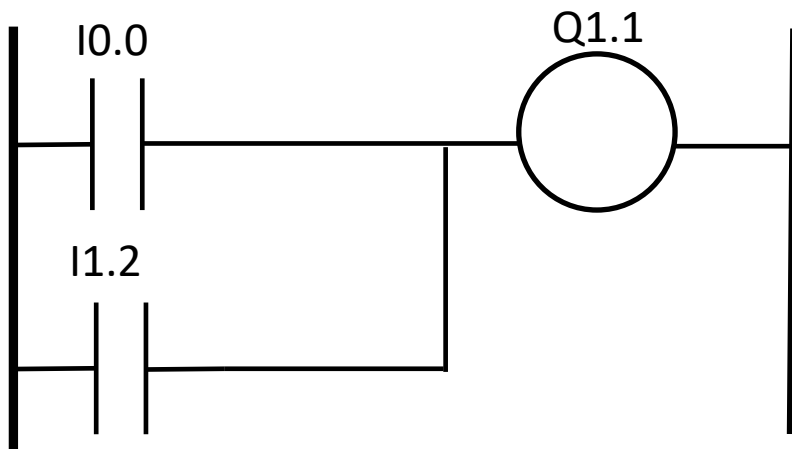
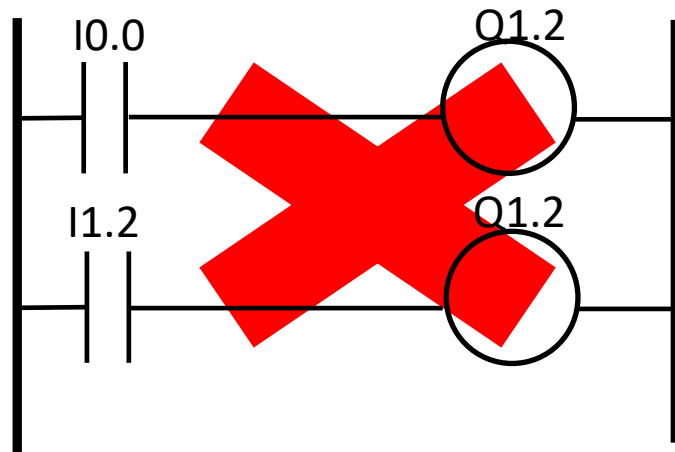
PulsadorOn = 1

PulsadorOff = 1

MotorA = 0

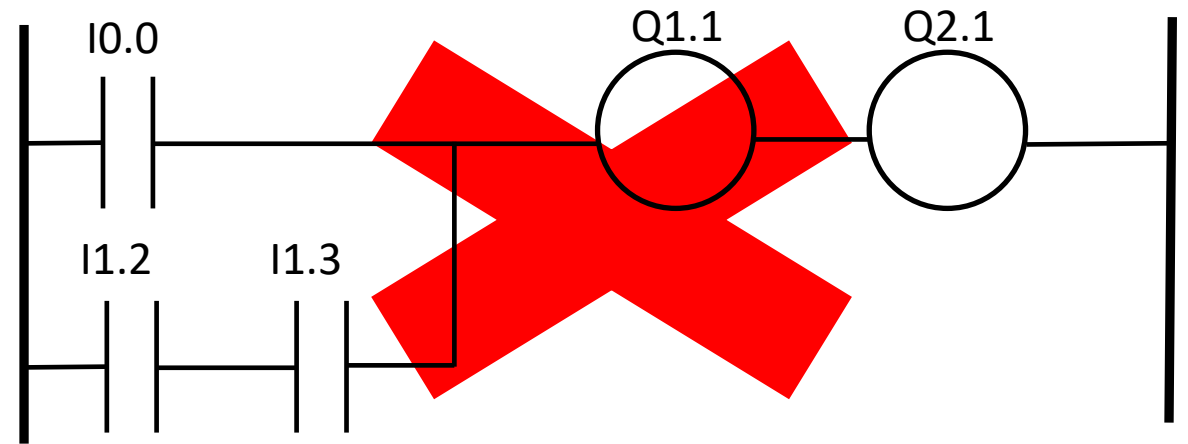
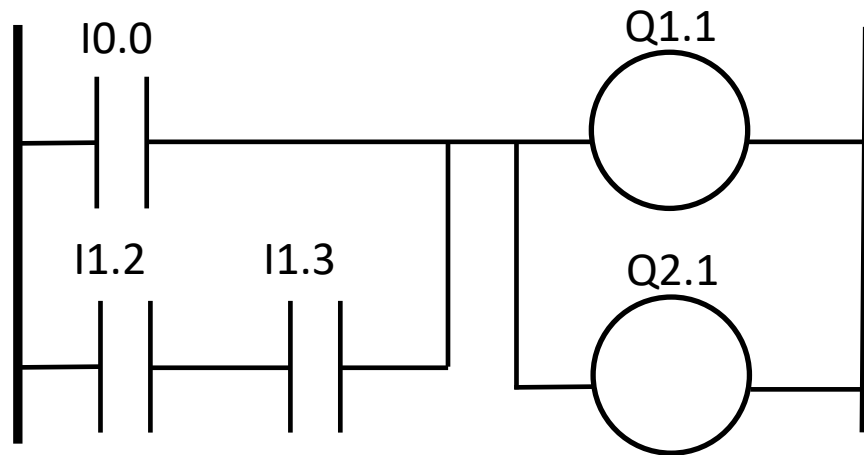
Lenguaje Ladder (LD)

- “No recomendable” usar la misma bobina como final de distintas líneas



Lenguaje Ladder (LD)

- No puede haber dos bobinas en serie

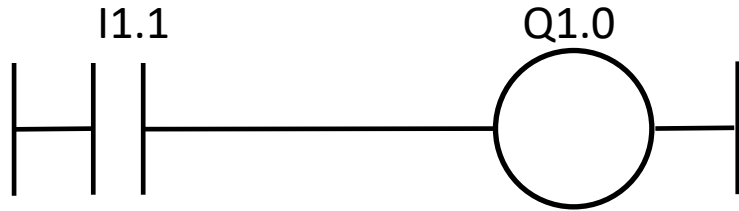


$$Q1.1 = I0.0 + I1.2 \cdot I1.3$$

$$Q2.1 = I0.0 + I1.2 \cdot I1.3$$

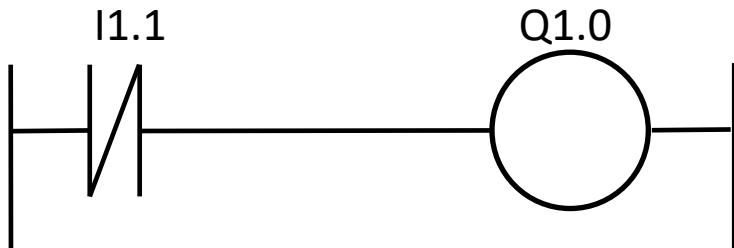
LENGUAJE FOB - KOP

- Selección de una variable directa



A Q1.0 se le asigna el valor de I1.1

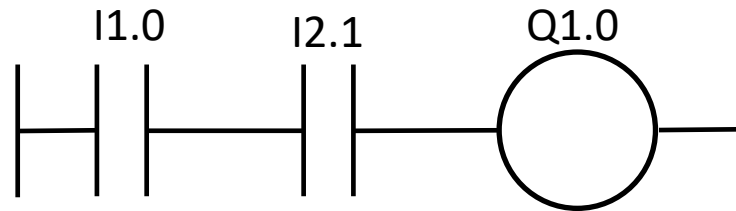
- Selección de una variable invertida



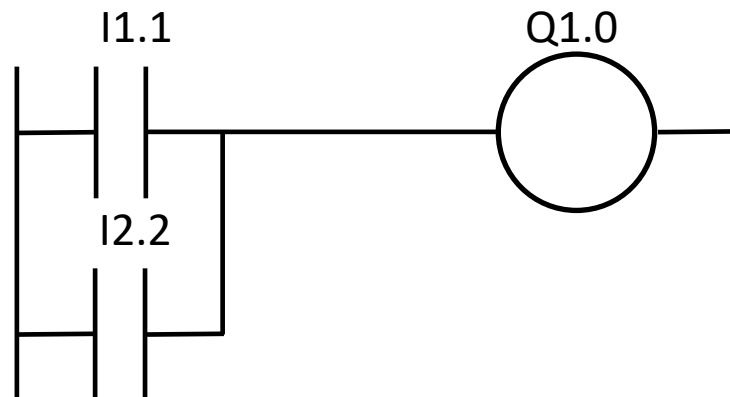
A Q1.0 se le asigna el valor **contrario** de I1.1

LENGUAJE FOB - KOP

- AND (conexión serie):

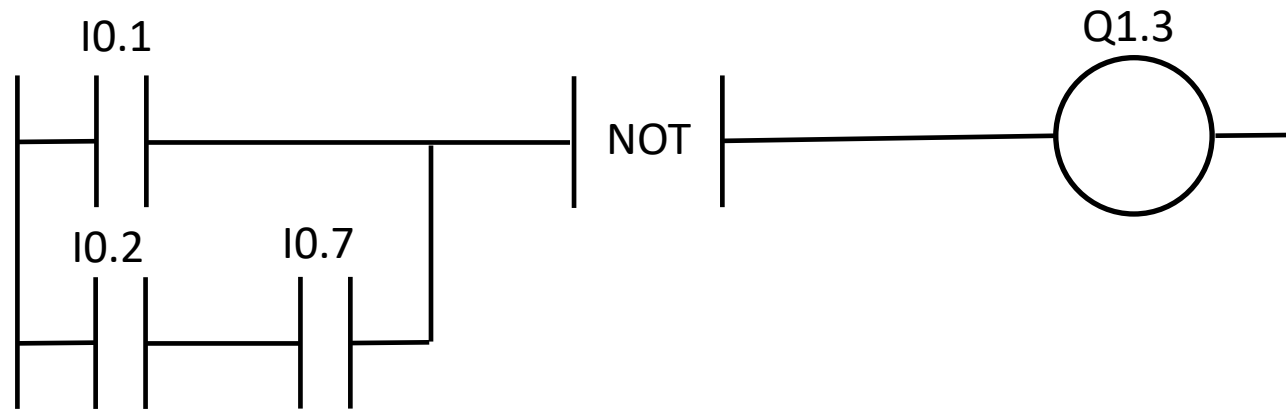


- OR (conexión paralelo):



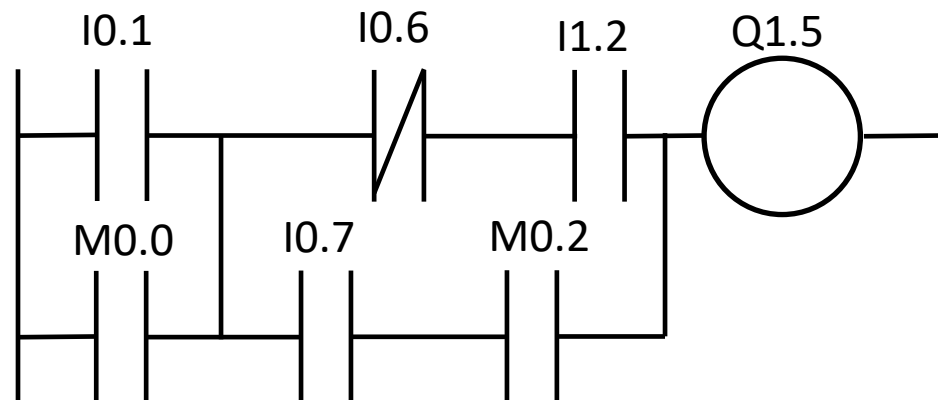
Lenguaje Ladder (LD)

- NOT



$$Q1.3 = \overline{I0.1 + (I0.2 \cdot I0.7)}$$

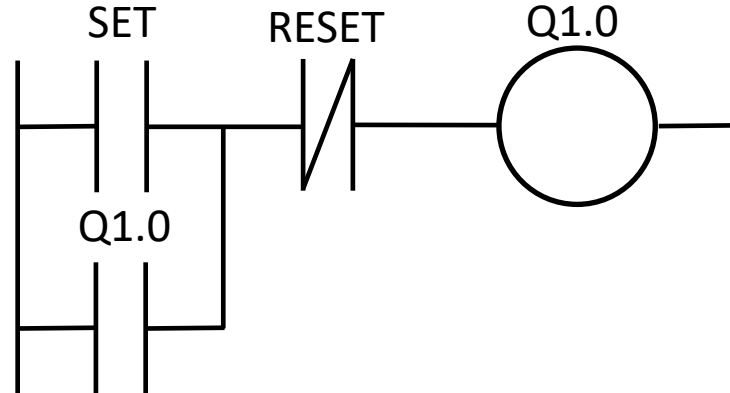
- ???



Lenguaje Ladder (LD). MEMORIZACIÓN

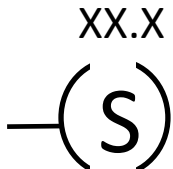
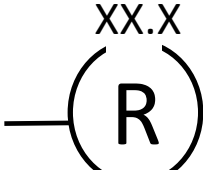
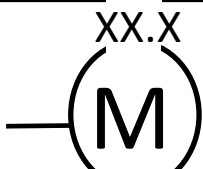
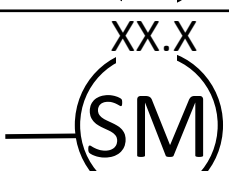
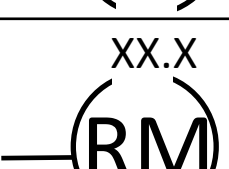
SET: Enclavar, activar, meter, entrar, ON

RESET: desenclavar, desactivar, tirar, caer, OFF



ENCLAVAMIENTO

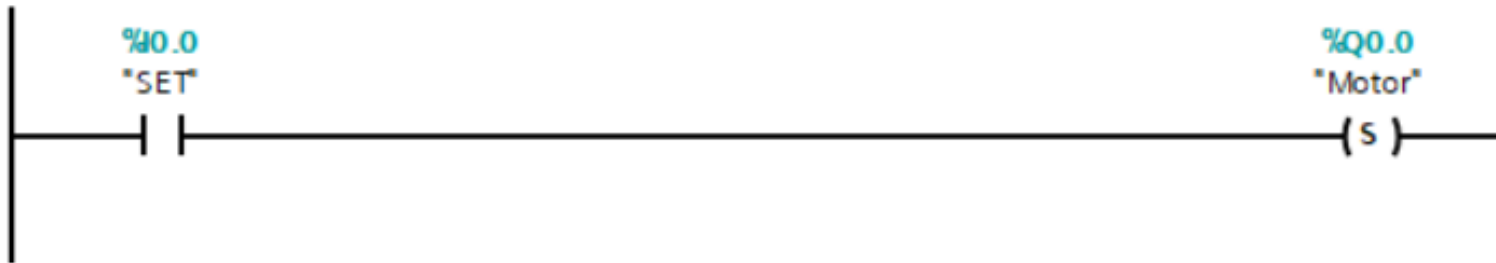
Lenguaje Ladder (LD). MEMORIZACIÓN

	La variable asociada con la bobina se activa cuando se cierra es esquema de contactos asociados de su rama y permanece activa aunque el circuito se abra.
	La variable asociada con la bobina se desactiva cuando se cierra es esquema de contactos asociados de su rama y permanece desactiva aunque el circuito se abra
	La variable asociada con la bobina se desactiva o desactiva de acuerdo al esquema de contactos de su rama y mantiene su valor aunque se corte la alimentación
	Comportamiento similar al de la bobina SET, pero mantiene su valor aunque la alimentación se corte.
	Comportamiento similar al de la bobina RESET, pero mantiene su valor aunque la alimentación se corte

Lenguaje Ladder (LD). MEMORIZACIÓN

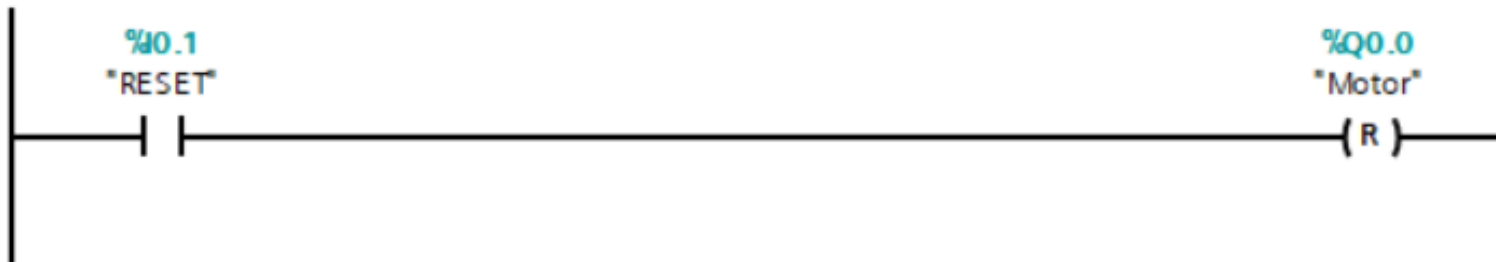
▼ Segmento 1: Set de la salida que controla el motor

Comentario



▼ Segmento 2: Reset de la salida que controla el motor

Comentario



IMPORTANTE:

Si usamos SET y RESET debe haber siempre una pareja de SET/RESET para la variable de la bobina

Lenguaje Ladder (LD). MEMORIZACIÓN

XX.X -(SET_BF) "n"	Pone a 1 la variable asociada (XX.X) y los siguientes n bits
XX.X -(RESET_BF) "n"	Pone a 0 la variable asociada (XX.X) y los siguientes n bits



Si PulsadorOn = 1:

M1.2, M1.3 y M1.4 se ponen a 1 (SET)

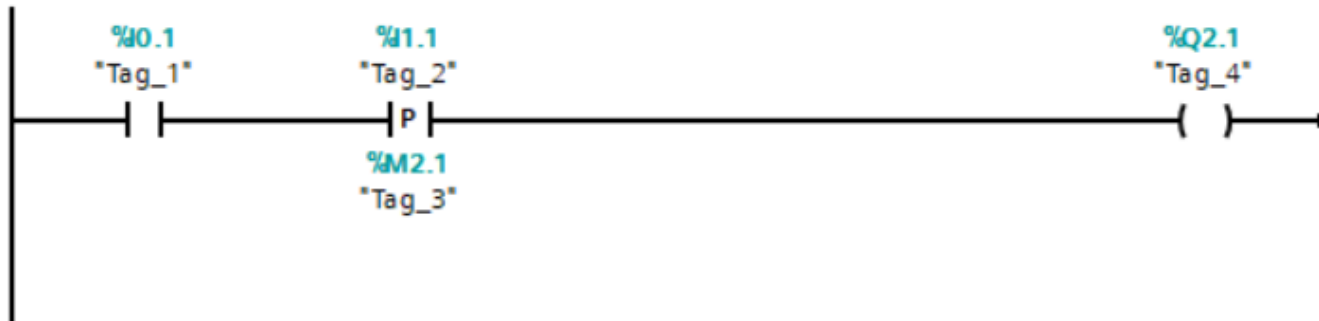
➡ M20.2, M20.3 ,M20.4, M20.5, M20.6 se
ponen a 0 (RESET)

Las variables permanecen

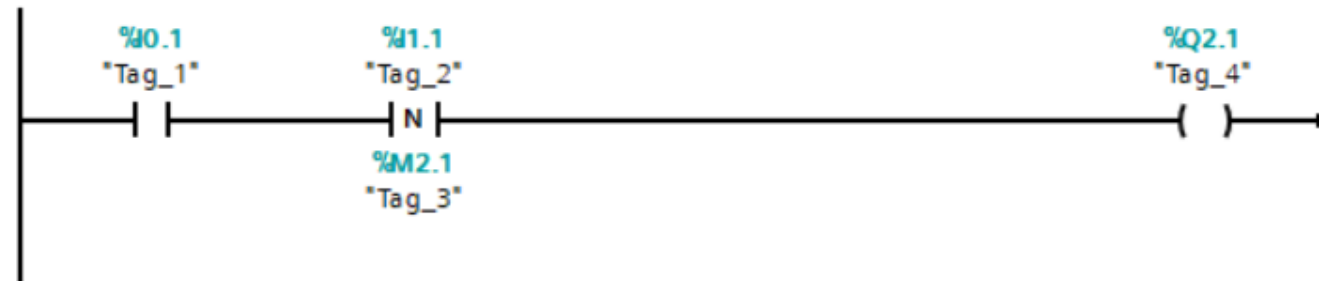
activas/inactivas aunque PulsadorOn = 0

Lenguaje Ladder (LD). Operaciones con flancos

- La detección de flancos en una variable se implementa a través de bloques: POS (flanco positivo) y NEG (flanco negativo)



La bobina Q2.1 se activa **durante un ciclo de scan** cuando I1.1 pasa de 0 a 1 y cuando el contacto I1.0 está cerrado. La variable M2.1 guarda el valor de I1.1 en el ciclo anterior

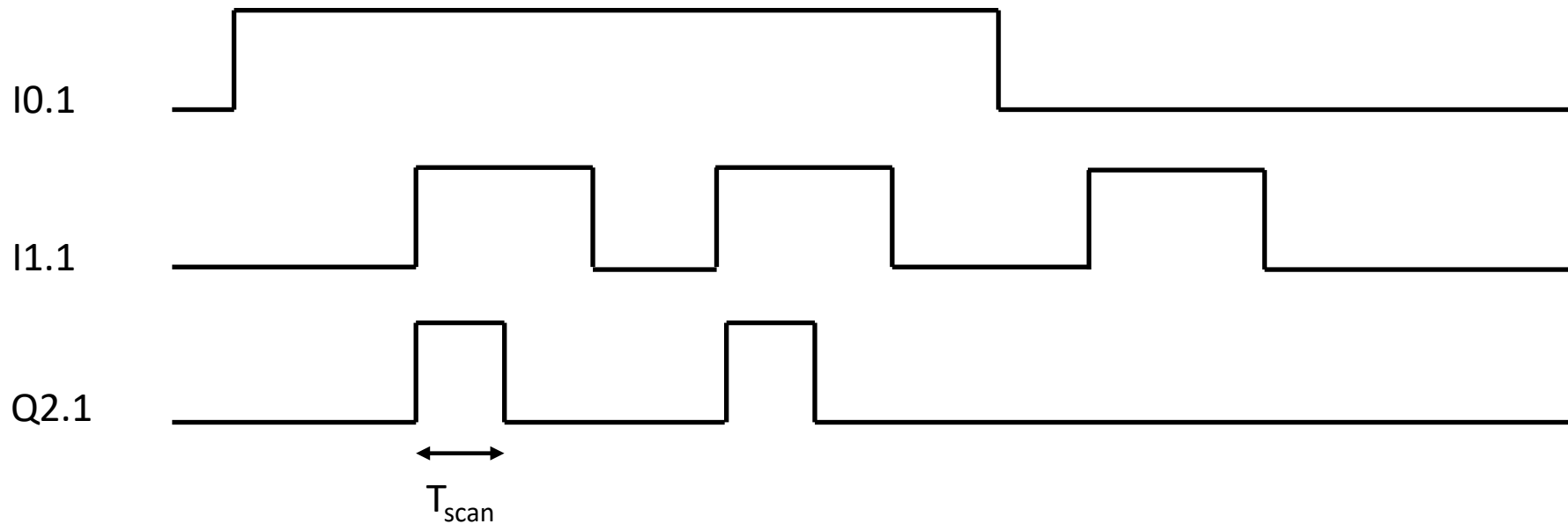


La bobina Q2.1 se activa **durante un ciclo de scan** cuando I1.1 pasa de 1 a 0 y cuando el contacto I0.1 está cerrado. La variable M2.1 guarda el valor de I1.1 en el ciclo anterior

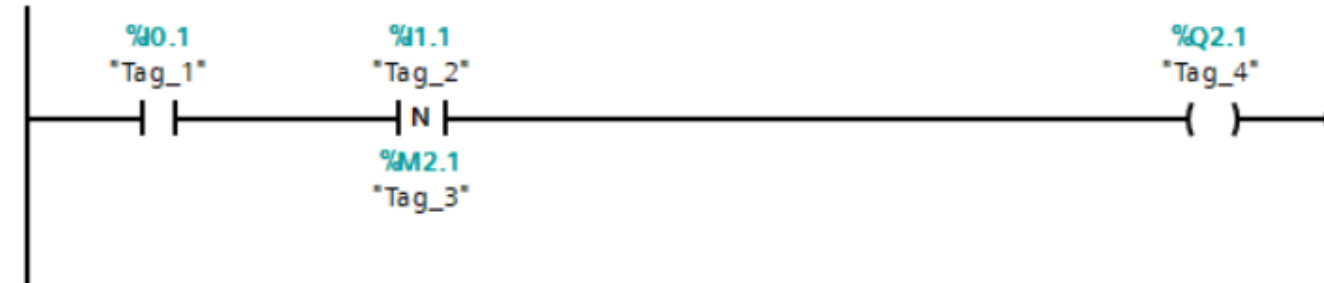
Lenguaje Ladder (LD). Operaciones con flancos



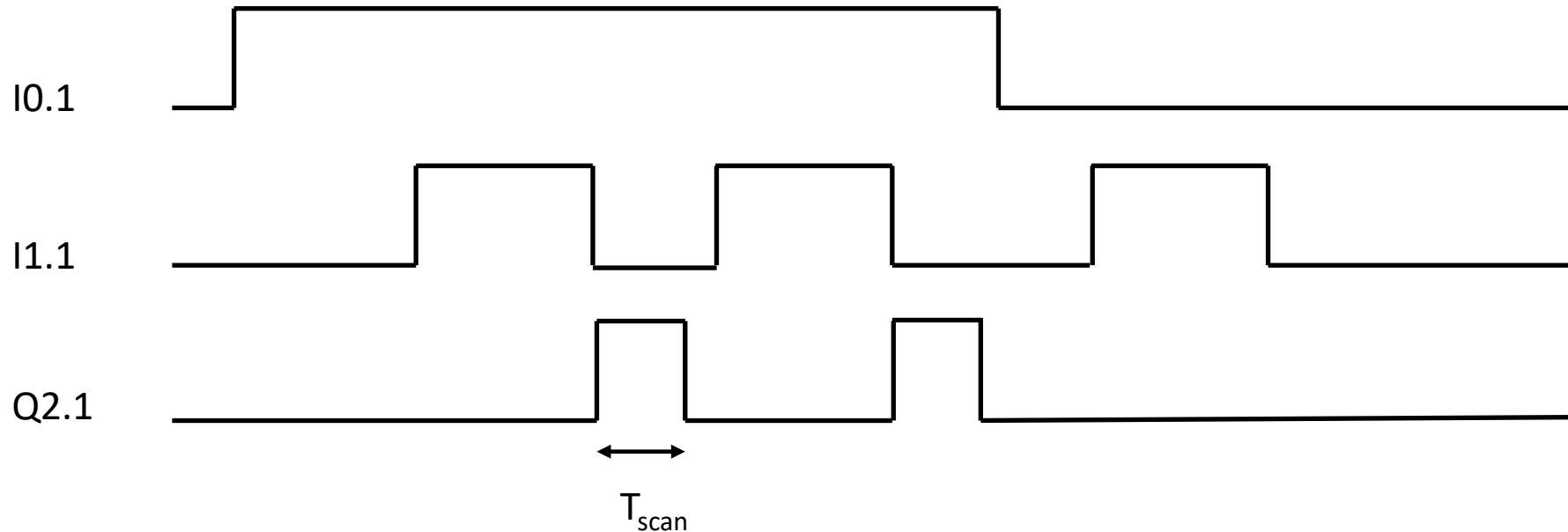
La bobina Q2.1 se activa **durante un ciclo de scan** cuando I1.1 pasa de 0 a 1 e I0.1 está activada. La marca M2.1 guarda el valor de I1.1 en el ciclo anterior (**No usar en otra parte del programa**)



Lenguaje Ladder (LD). Operaciones con flancos

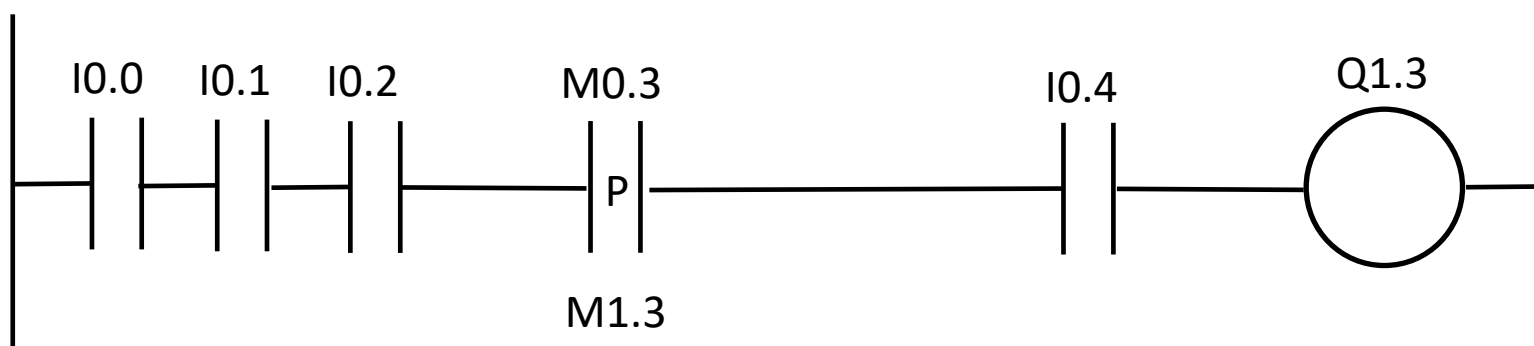


La bobina Q2.1 se activa **durante un ciclo de scan** cuando I1.1 pasa de 1 a 0 y cuando el contacto I0.1 está cerrado. La variable M2.1 guarda el valor de I1.1 en el ciclo anterior.



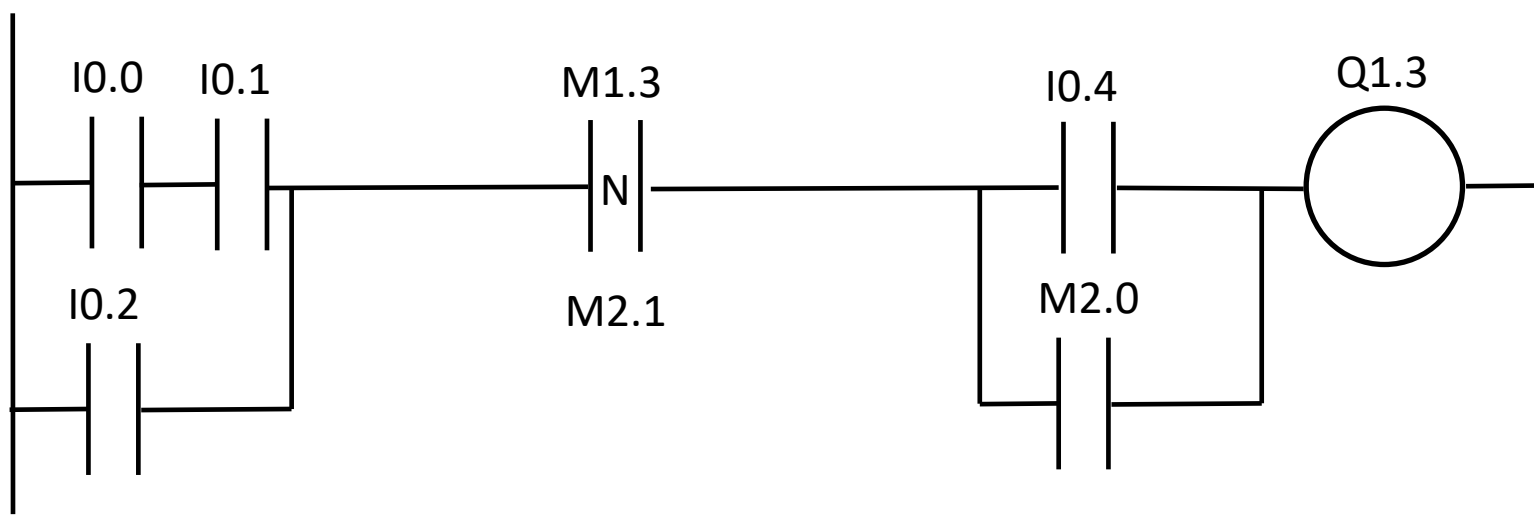
Lenguaje Ladder (LD). Operaciones con flancos

$$Q1.3 = I0.0 \cdot I0.1 \cdot I0.2 \cdot M0.3 \uparrow \cdot I0.4$$



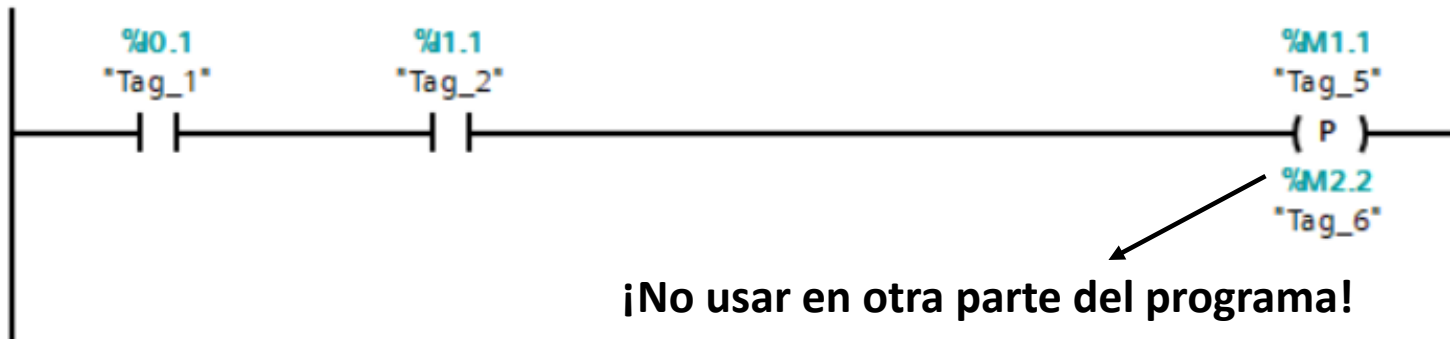
Lenguaje Ladder (LD). Operaciones con flancos

Q1.3 = ????

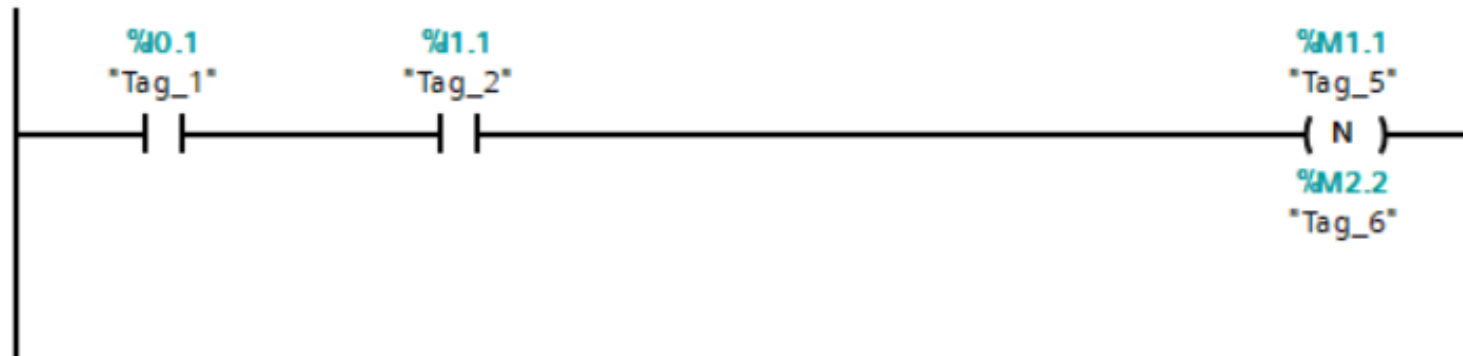


Lenguaje Ladder (LD). Operaciones con flancos

ACTIVAR SALIDA CON FLANCO DE SEÑAL ASCENDENTE/DESCENDENTE



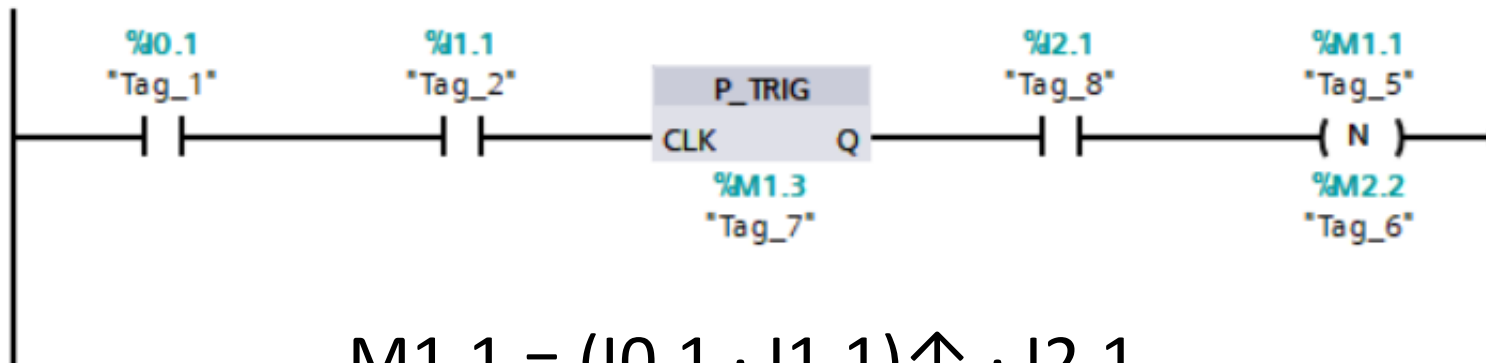
M1.1 se activa cuando el RLO tiene un flanco ascendente: I0.1·I1.1 pasa de 0 a 1 (1 ciclo de scan)



M1.1 se activa cuando el RLO tiene un flanco descendente: I0.1·I1.1 pasa de 1 a 0 (1 ciclo de scan)

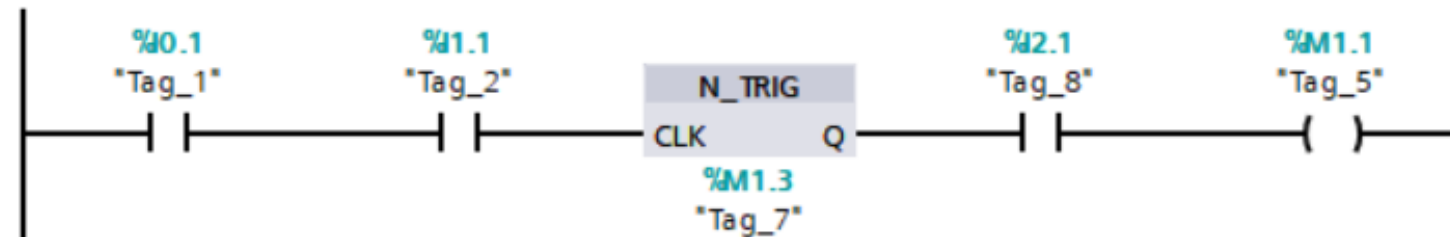
Lenguaje Ladder (LD). Operaciones con flancos

DETECTAR FLANCO DE SEÑAL ASCENDENTE/DESCENDENTE



$$M1.1 = (I0.1 \cdot I1.1) \uparrow \cdot I2.1$$

M1.1 se activa cuando I0.1·I1.1 pasa de 0 a 1 (1 ciclo de scan) e I2.1 está activada. M1.3 no participa en la expresión lógica

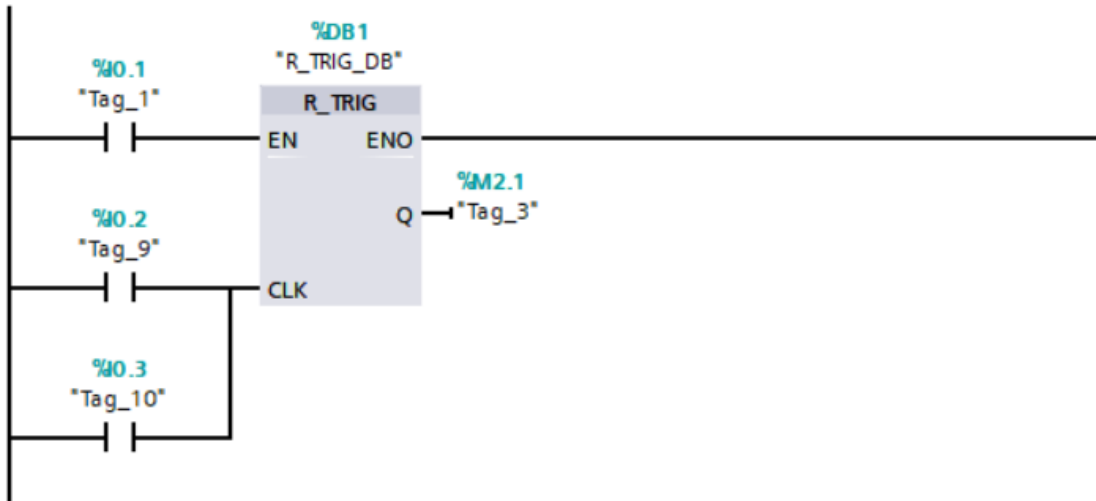


$$M1.1 = (I0.1 \cdot I1.1) \downarrow \cdot I2.1$$

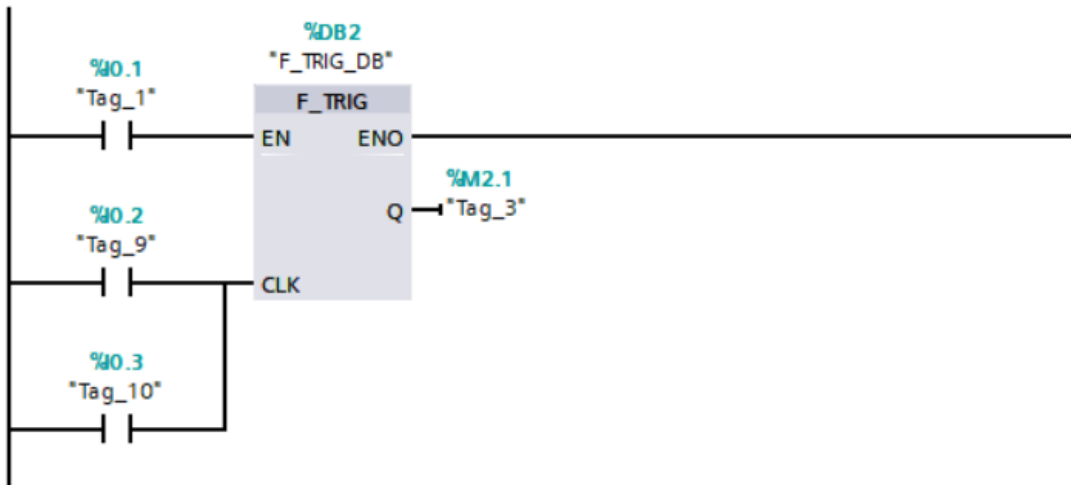
M1.1 se activa cuando I0.1·I1.1 pasa de 1 a 0 (1 ciclo de scan) e I2.1 está activada. M1.3 no participa en la expresión lógica

Lenguaje Ladder (LD). Operaciones con flancos

DETECTAR FLANCO DE SEÑAL ASCENDENTE/DESCENDENTE



M2.1 se activa cuando I0.2 + I1.3 pasa de 0 a 1 (1 ciclo de scan) e I0.1 está activada



M2.1 se activa cuando I0.2 + I1.3 pasa de 1 a 0 (1 ciclo de scan) e I0.1 está activada

Lenguaje Ladder (LD)

Tenemos un depósito de agua que incluye los siguientes elementos de control:

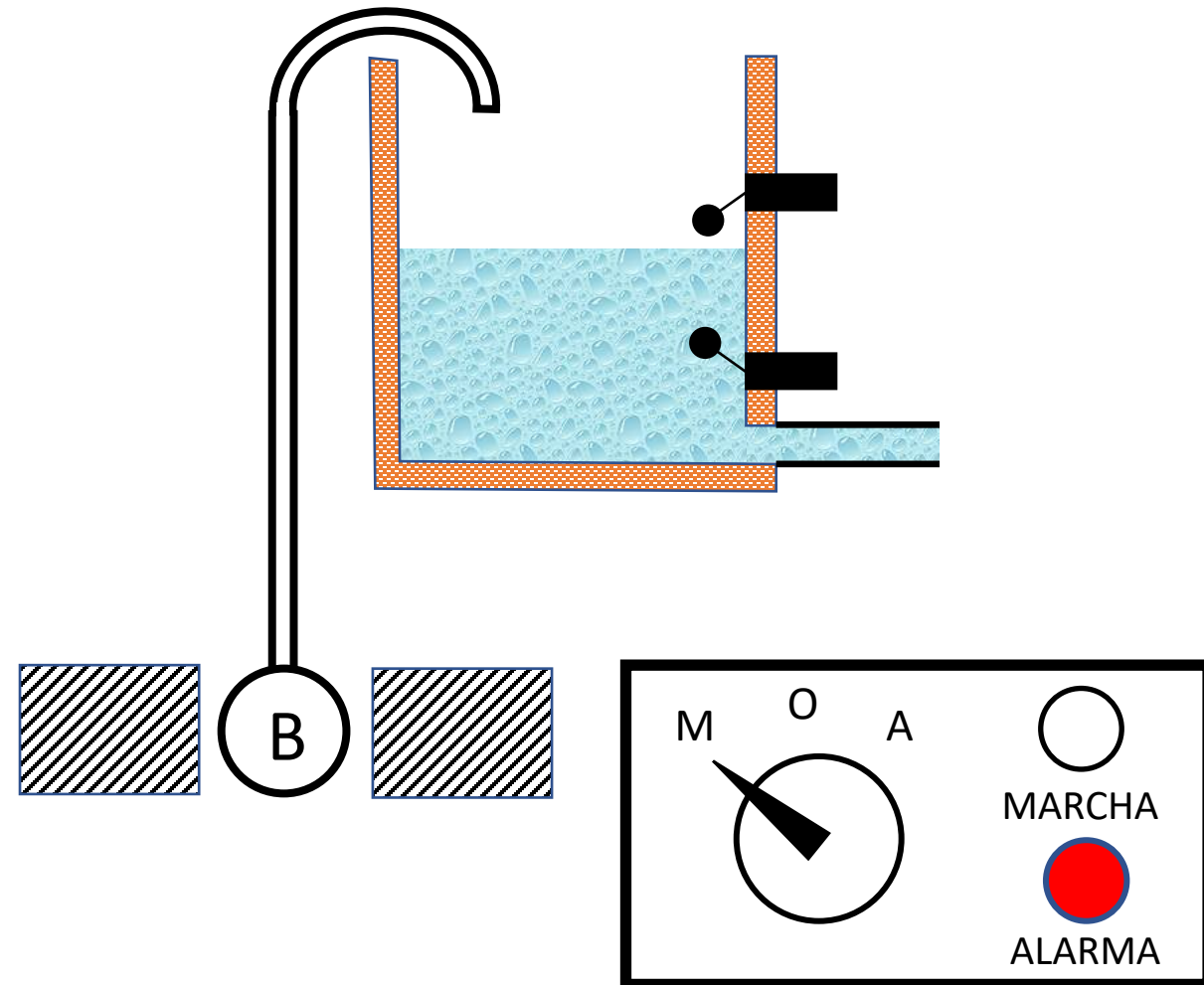
- Dos detectores de nivel que detectan el nivel máximo (I1) y mínimo (I2) del depósito
- Una bomba que suministra agua al depósito
- Un panel de control con tres posiciones: M (manual), A (automático) y O (fuera de servicio)
- Un relé de protección térmica

Requisitos:

- Si está activada la posición M la bomba funciona de forma continua con independencia del nivel del depósito.
- Si está activada la posición A el depósito debe estar entre el máximo y el mínimo, de tal forma que cuando el depósito alcanza el mínimo la bomba funciona hasta llegar al máximo.
- Si está activada a posición O la bomba está fuera de servicio.
- El relé térmico para la bomba cuando su temperatura alcanza un valor fijado y se debe iluminar la lámpara de Alarma. El relé térmico es un contacto NC (Funcionamiento normal → Térmico = 1).
- Cuando la bomba está en marcha se debe activar la lámpara de marcha.

Lenguaje Ladder (LD)

Entradas	Descripción
I0.0	Interruptor modo manual
I0.1	Interruptor modo automático
I0.2	Boya nivel inferior I2
I0.3	Boya nivel superior I1
I0.4	Contacto auxiliar NC relé térmico
Salida	Descripción
Q0.0	Contactor de la bomba
Q0.1	Señalización de marcha
Q0.2	Alarma de protección térmica

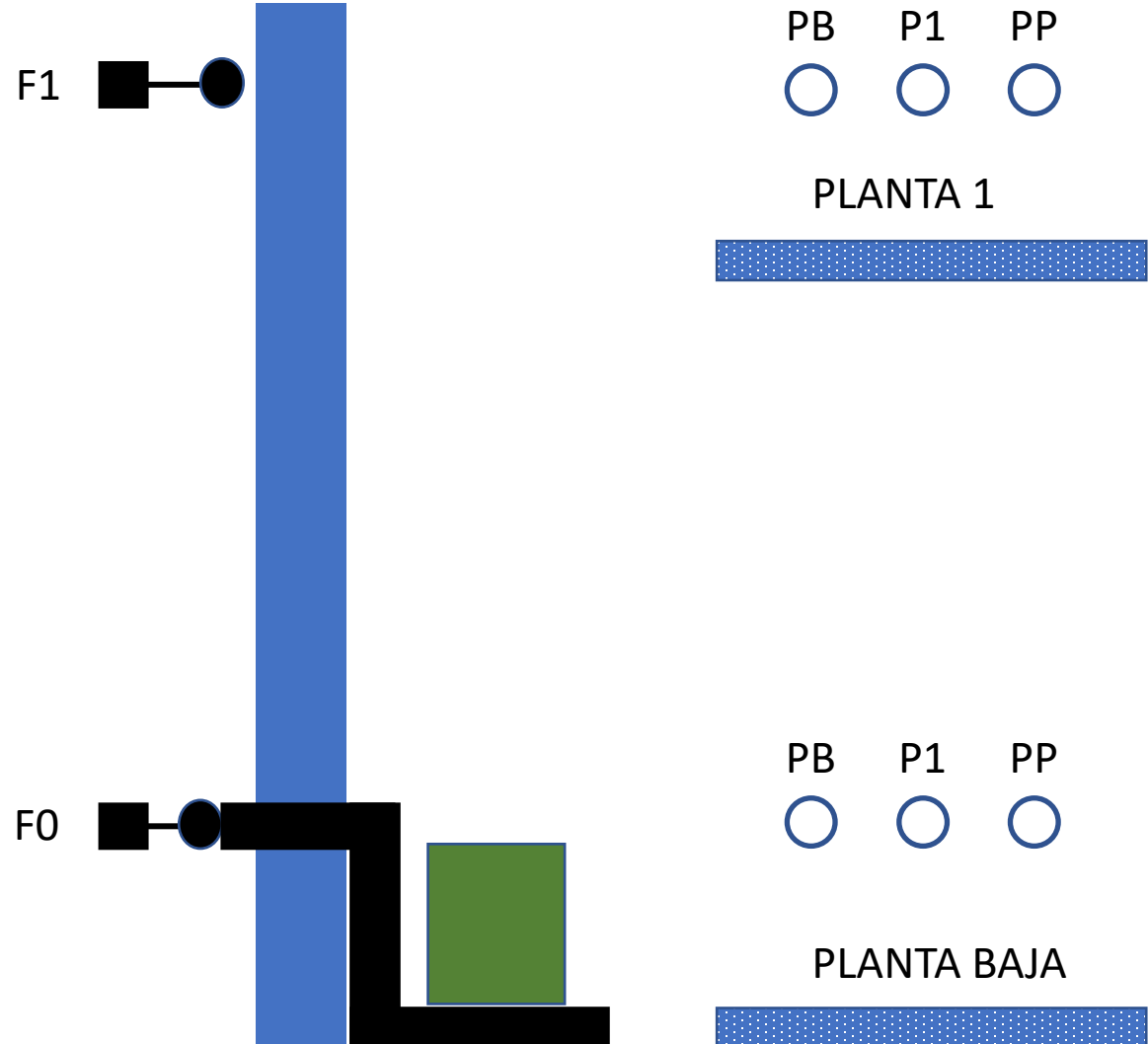


Plataforma elevadora

Entradas: Llamada planta baja (PB), llamada planta 1 (P1), pulsador de paro (PP), F0 y F1

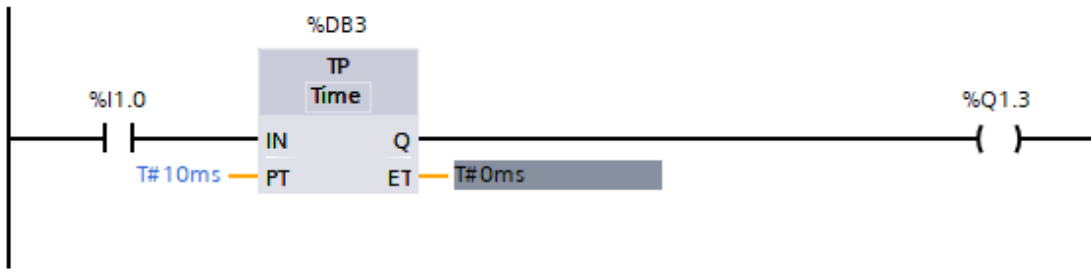
Salidas: Motor subir (Sub), motor bajar (Baj)

Pulsadores de cada planta conectados en paralelo.

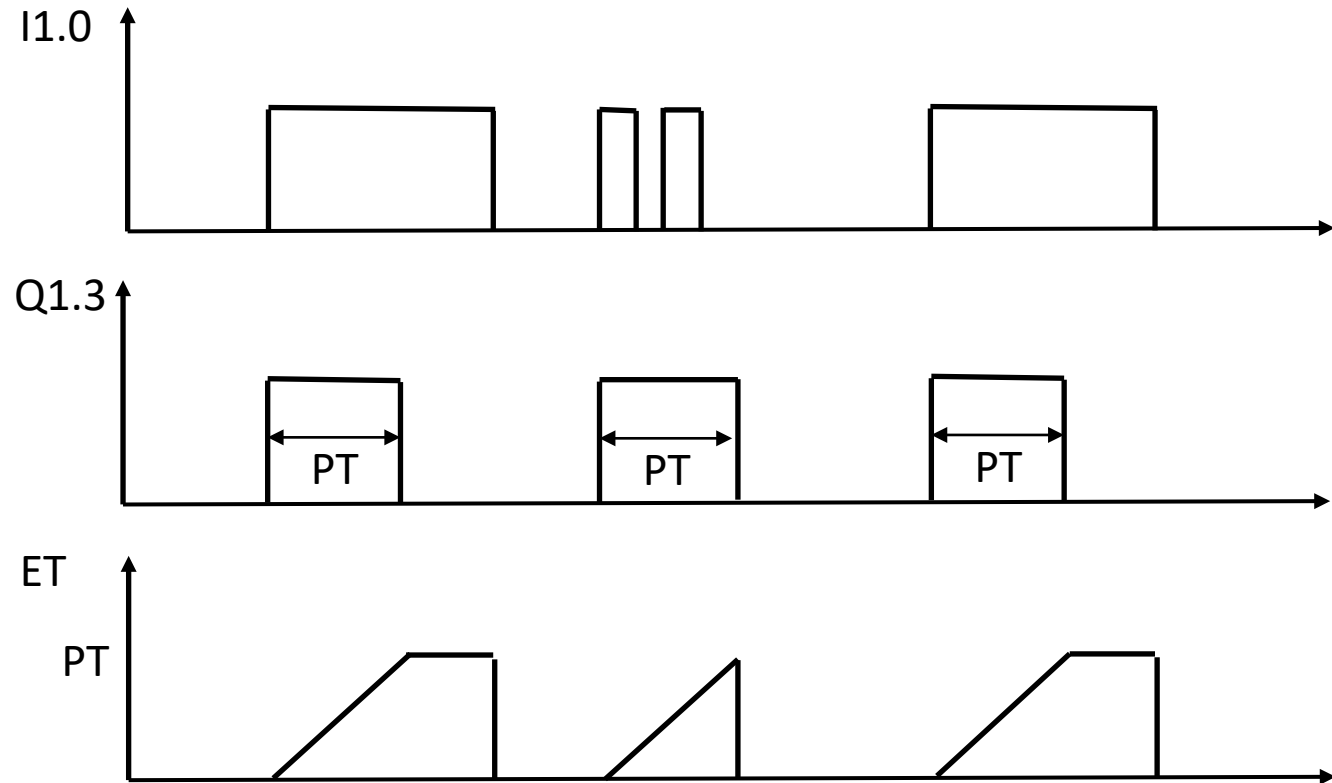


Lenguaje Ladder (LD). Operaciones de temporización

TP: Temporizador de impulsos



Cuando el estado lógico de I1.0 (IN) cambia de “0” a “1” el tiempo programado en “Tiempo Preseleccionado” (PT), empieza a contar y Q1.3 (Q) se pone a 1 (*flanco positivo*). El valor actual del tiempo se guarda en “Tiempo transcurrido” (ET). Una vez transcurrido el tiempo indicado en PT, Q1.3 (Q) pasa a valer 0. ET pasa a valer 0 cuando se desactiva I0.1



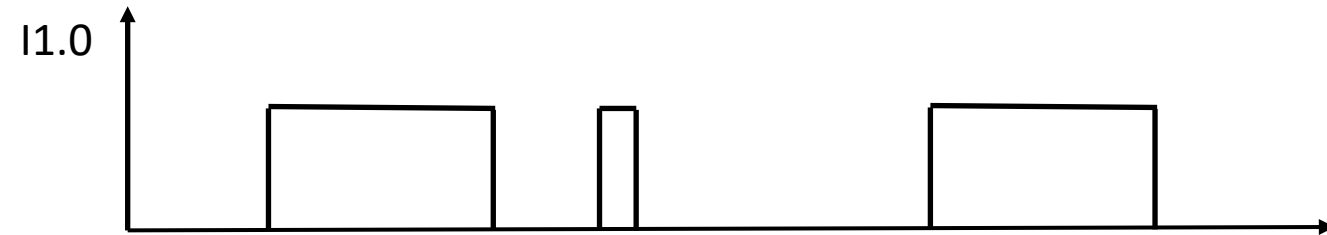
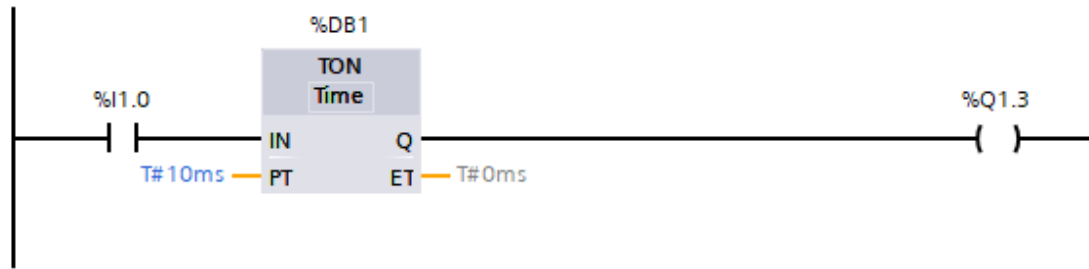
TP: Timer Pulse

PT: Predetermined Time

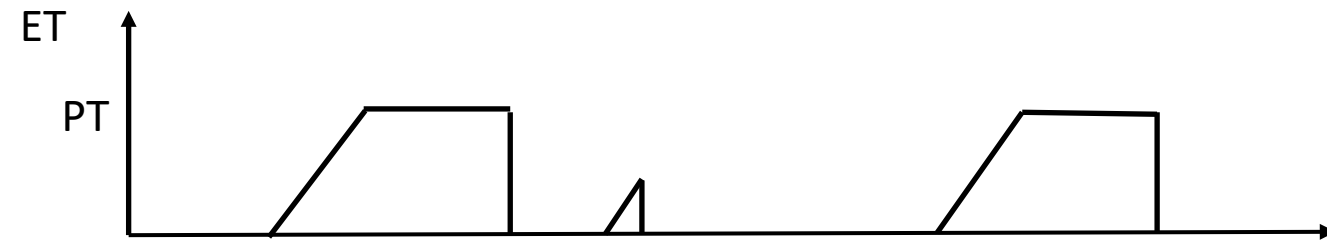
ET: Elapsed Time

Lenguaje Ladder (LD). Operaciones de temporización

TON: Temporizador de retardo a la conexión



Cuando el estado lógico de I1.9 (IN) cambia de "0" a "1" el tiempo programado en "Tiempo Preseleccionado" (PT) empieza a contar. Una vez transcurrido el tiempo PT, Q1.3 (Q) cambia su valor a "1" y mantiene ese valor mientras I0.1 mantenga el valor "1". El valor del tiempo actual se guarda en ET. Si I1.0 cambia su valor de "1" a "0" antes de que transcurra el tiempo PT la salida no se activa.

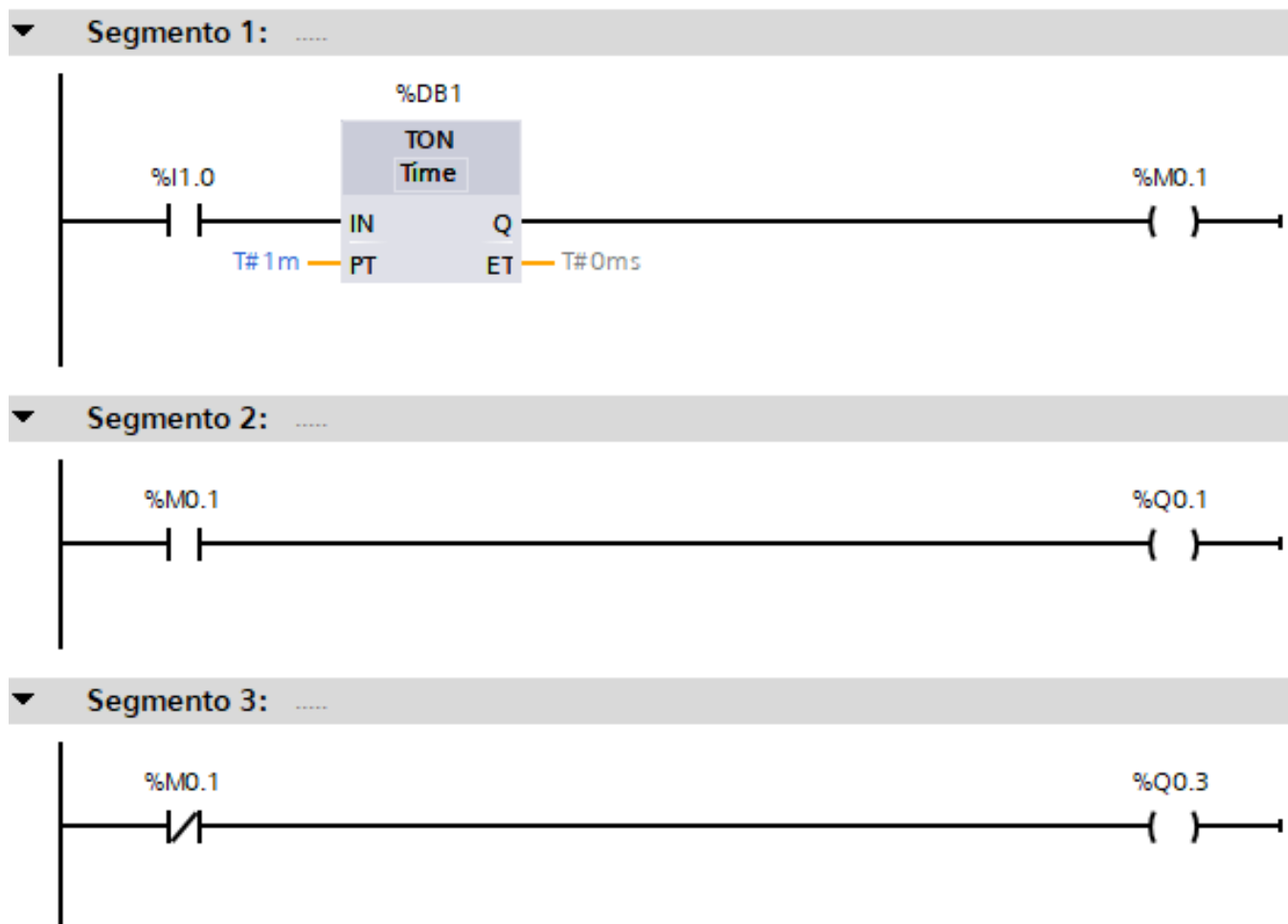


PT: Preselection Time

ET: Elapsed Time

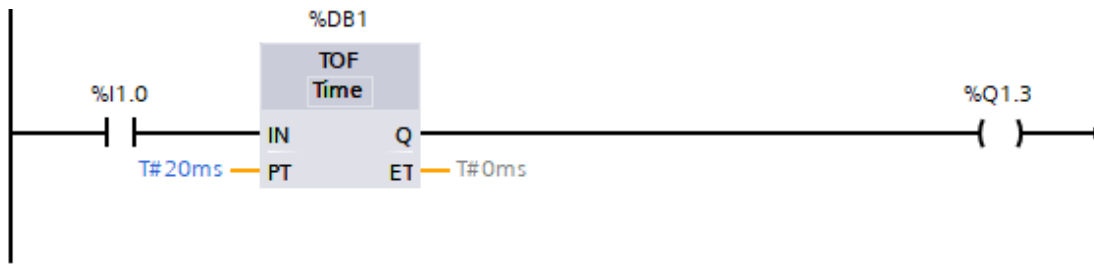
Lenguaje Ladder (LD). Operaciones de temporización

Ejemplo de uso: Activación de una salida transcurrido un cierto tiempo después de activar un **interruptor (I0.1)**

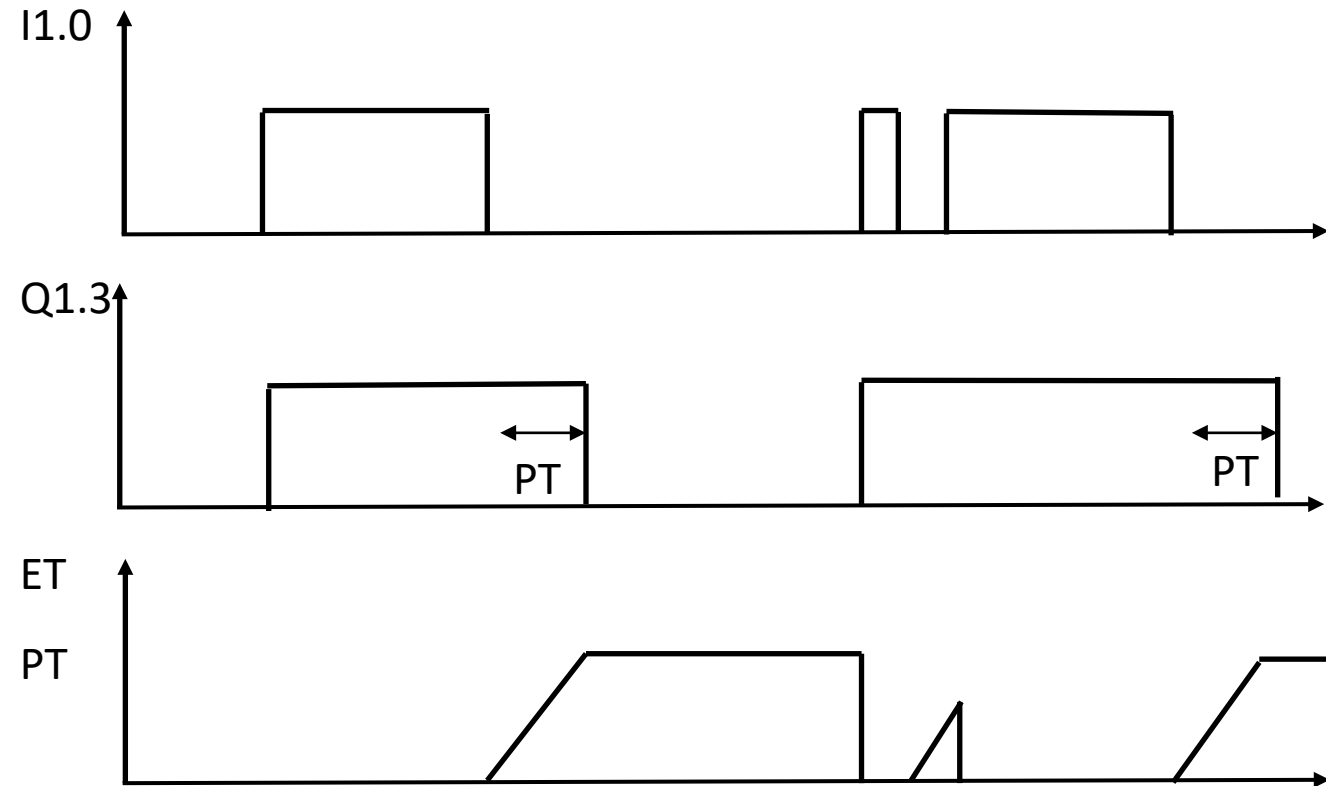


Lenguaje Ladder (LD). Operaciones de temporización

TOF: Temporizador de retardo a la desconexión



La salida (Q) se activa al hacerlo la entrada I1.0 (IN) y se desactiva transcurrido un tiempo PT desde que la entrada (IN) se desactiva (flanco negativo). El tiempo ET comienza a contar cuando la entrada pasa de "1" a "0" hasta alcanzar el valor PT. **En caso de que I0.1 se desactive y active antes de que transcurra el tiempo PT la salida se mantiene activada.**



PT: Preselection Time

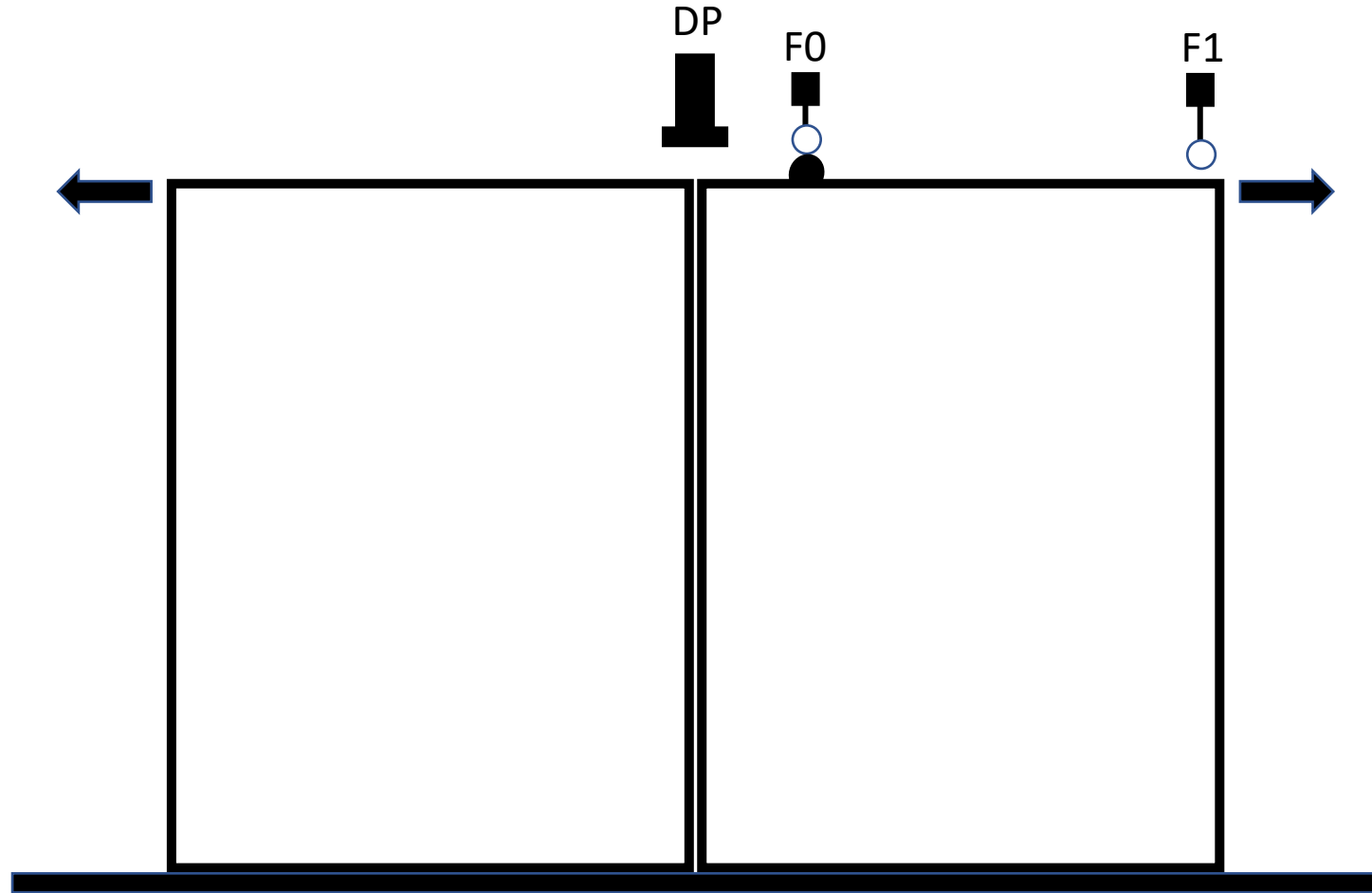
ET: Elapsed Time

Puerta automática

Entradas: Detector de presencia (DP), F0 y F1

Salidas: Abrir (Ab) y Cerrar (Ce)

Al detectar una persona (DP) se abre la puerta, una vez abierta temporiza 10 segundos y se cierra. En caso de que vuelva a detectar otra persona se vuelve a abrir.

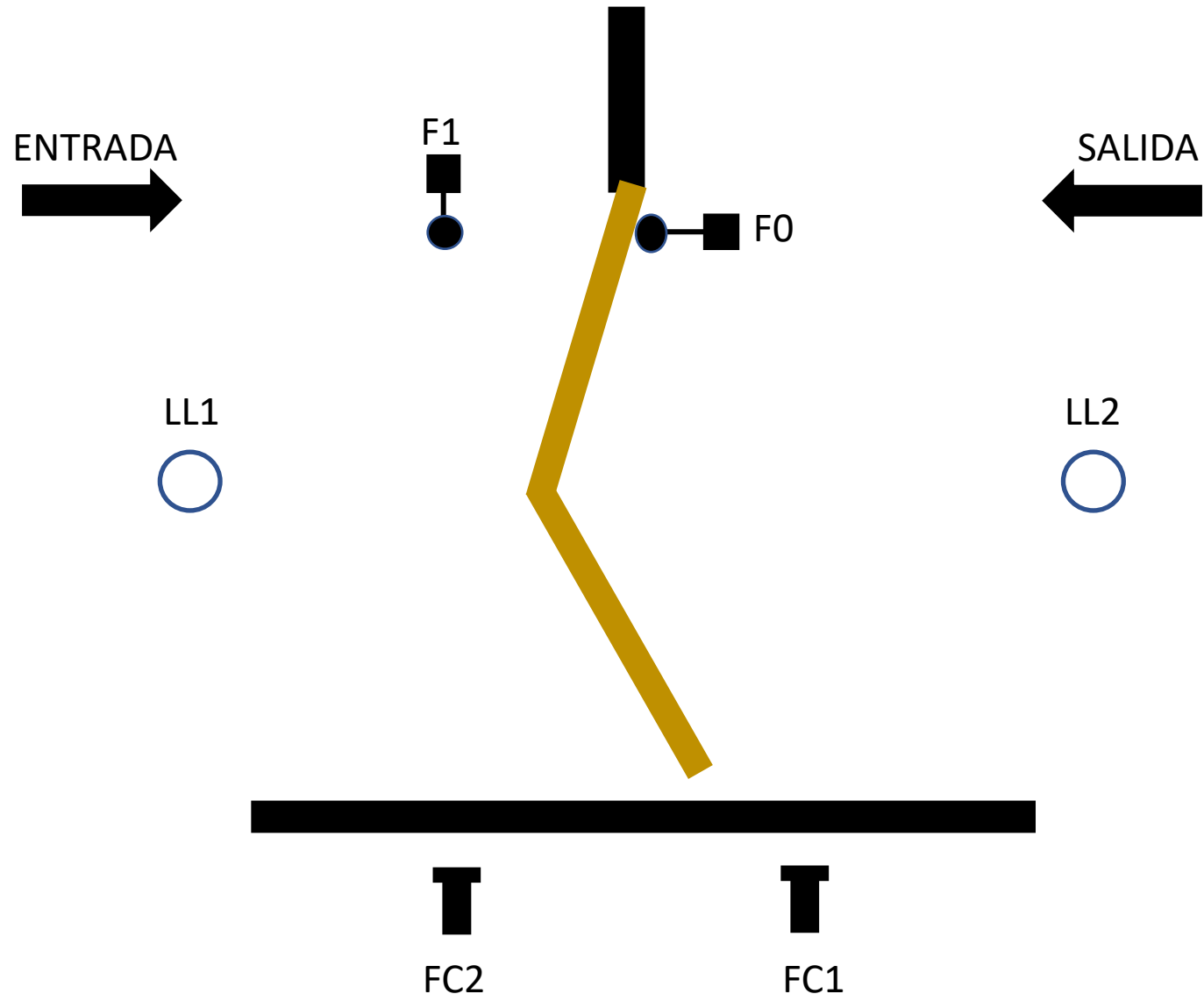


Puerta automática

Entradas: Mando a distancia (MD), Llave 1 (LL1), Llave 2 (LL2), F0, F1, FC1, FC2.

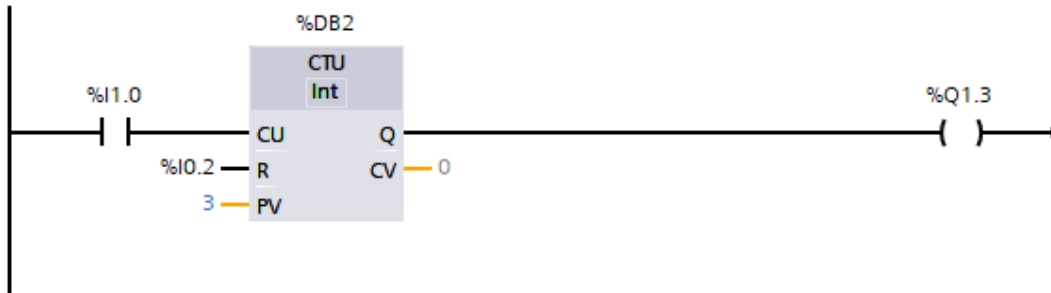
Salidas: Subir (SU), Bajar (BA)

Al activar MD, LL1 o LL2 la puerta sube y después de 10 segundos baja. Si en la bajada alguna de las fotocélulas (FC) se activa la puerta se detiene.



Lenguaje Ladder (LD). Operaciones de temporización

CTU (Counting Up): Contador ascendente



CU: Count Up

R: Reset

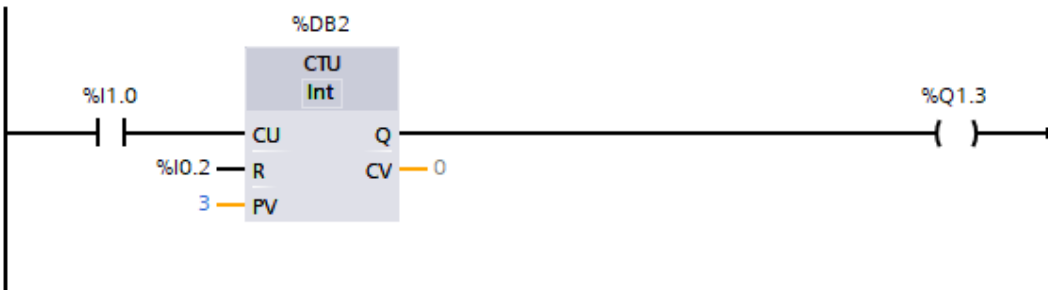
PV: Preselected Value

CV: Current Value

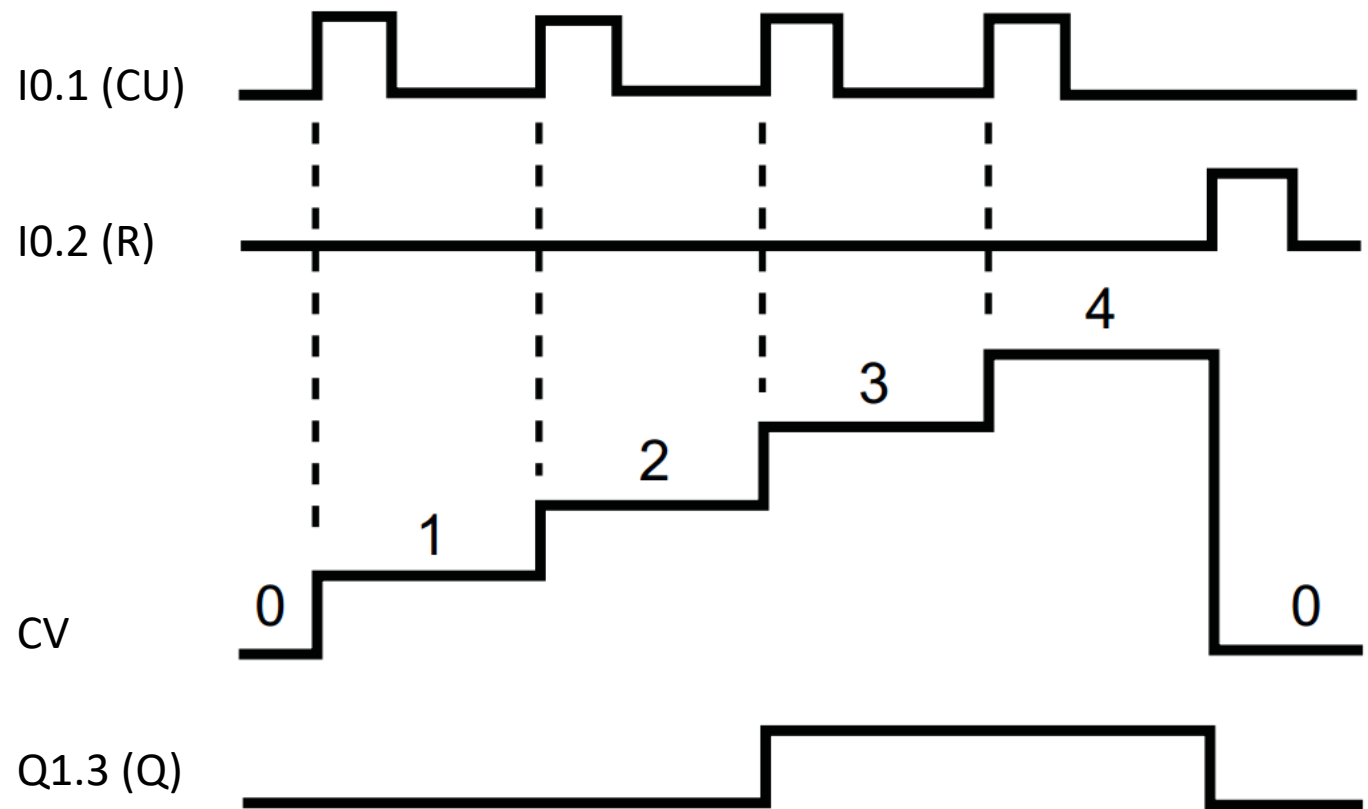
Cuando el estado lógico de I1.0 (CU) cambia de “0” a “1” (flanco positivo) el valor del contador (“Contaje Actual”: CV) se incrementa en una unidad. El valor introducido en PV se utiliza para determinar la activación de Q1.3 (Q). La salida Q1.3 (Q) se activa cuando el valor de CV es mayor o igual que PV. En otro caso Q1.3 (Q) vale cero. I0.2 actúa como señal de “Reset” para poner a 0 el valor de CV.

Lenguaje Ladder (LD). Operaciones de temporización

CTU (Counting Up): Contador ascendente **PV = 3**

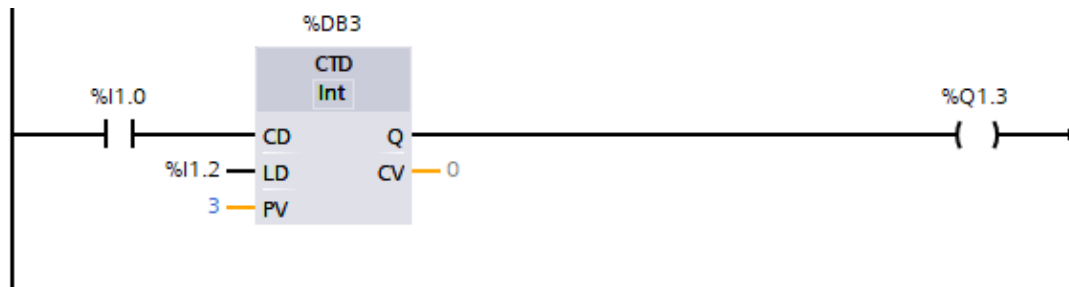


CU: Count Up
R: Reset
PV: Preselected Value
CV: Current Value



Lenguaje Ladder (LD). Operaciones de temporización

CTD (Counting Down): Contador descendente



CD: Count Down

LD: Load

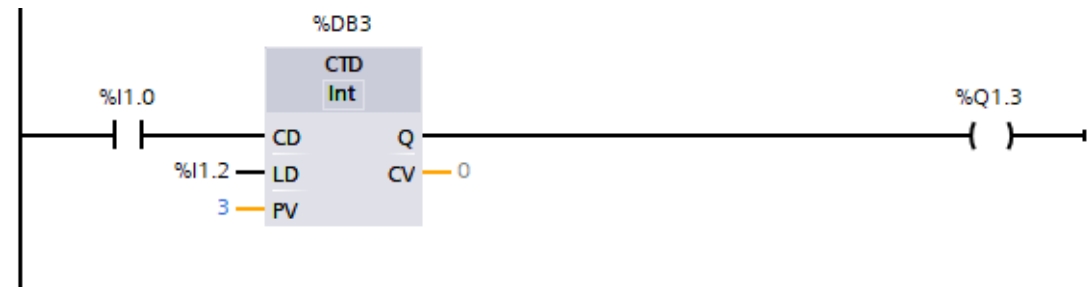
PV: Preselected Value

CV: Current Value

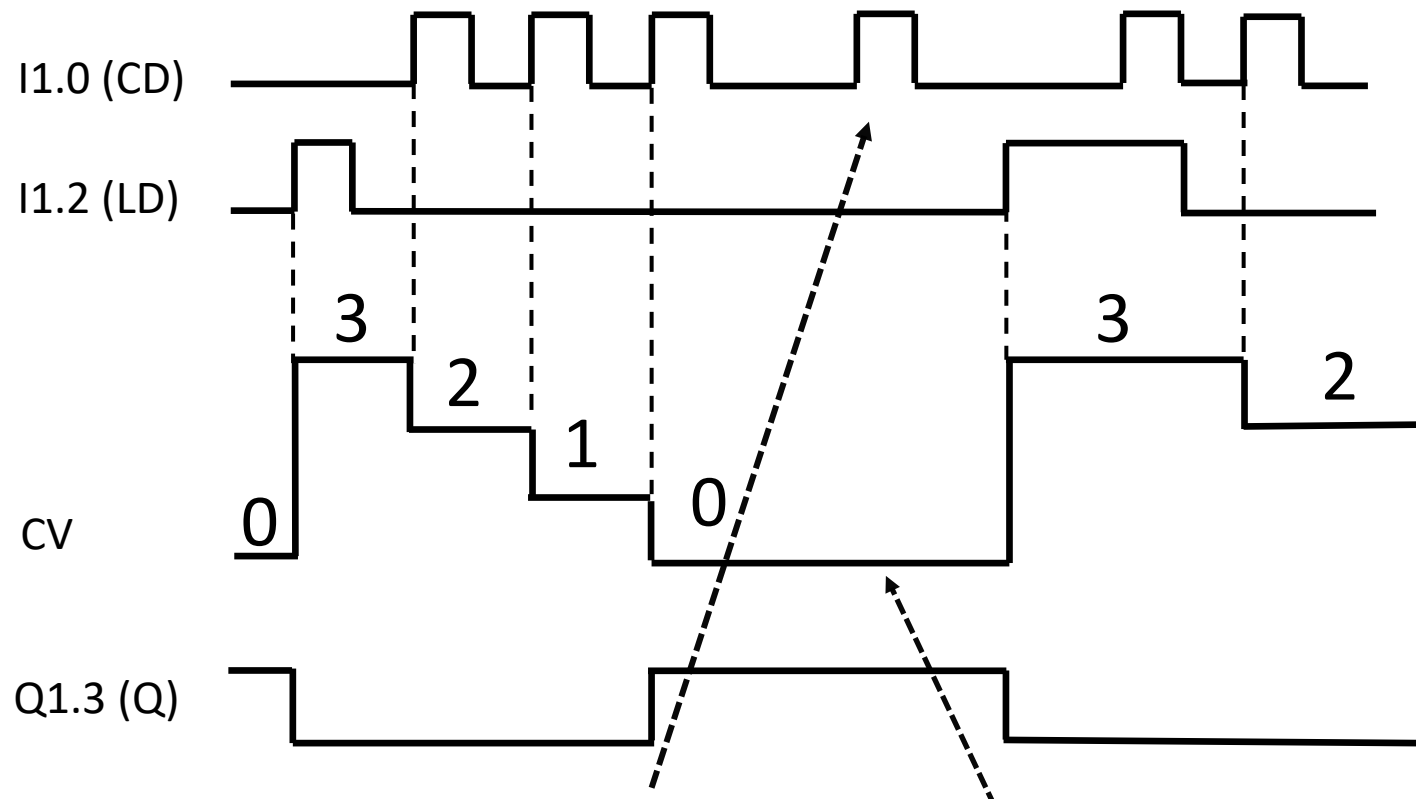
Cuando el estado lógico de I0.1 (CD) cambia de “0” a “1” (flanco positivo) el valor del contador (“Contaje Actual”) se decrementa en una unidad. El valor introducido en PV se utiliza como valor inicial del contador. La salida Q1.3 (Q) se activa cuando el valor de CV es 0, en otro caso Q1.3 vale cero. I1.2 actúa como señal de “Load”, cuando se activa se establece en CV el valor de PV.

Lenguaje Ladder (LD). Operaciones de temporización

CTD (Counting Down): Contador descendente



CD: Count Down
LD: Load
PV: Preselected Value
CV: Current Value

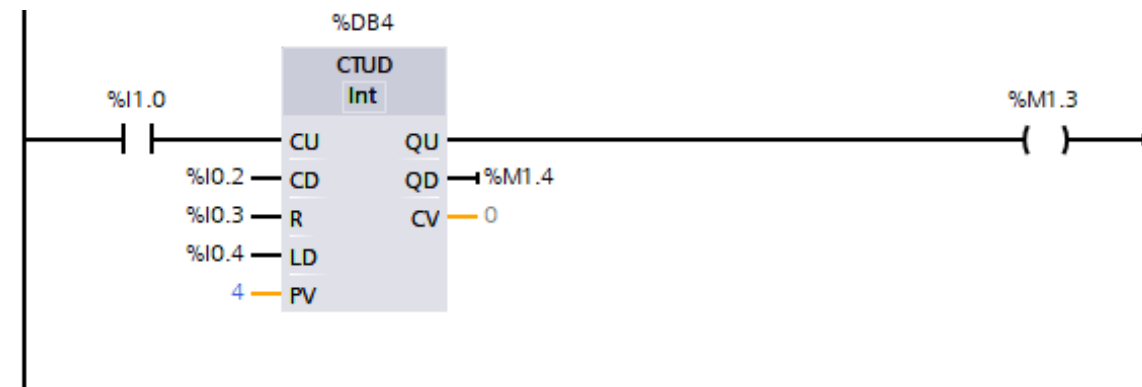


Si el contador se define como UINT → CV tiene como límite el "0"

Si el contador se define como INT → CV puede tomar valores negativos

Lenguaje Ladder (LD). Operaciones de temporización

CTUD (Counting Up Down): Contador ascendente/descendente

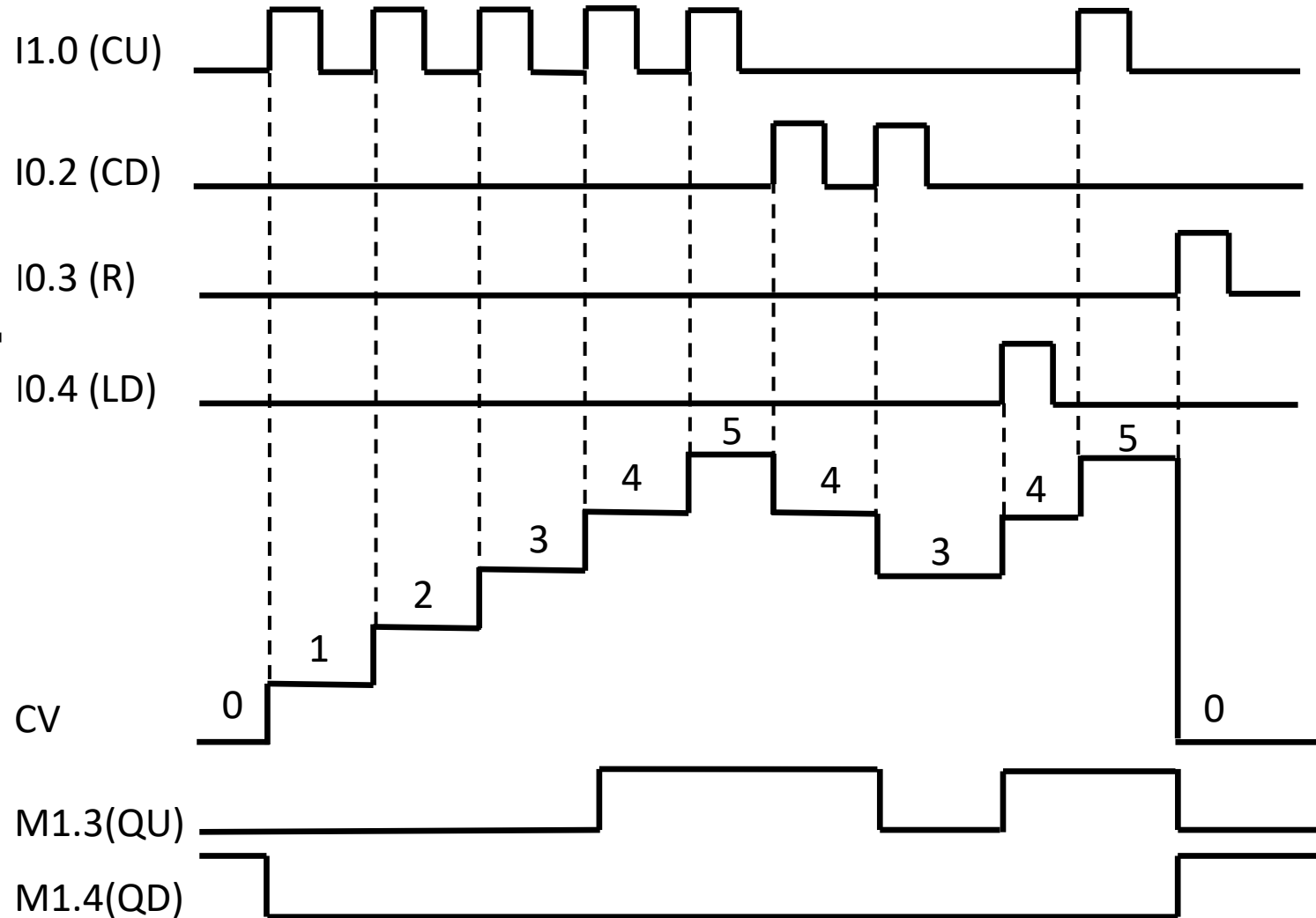
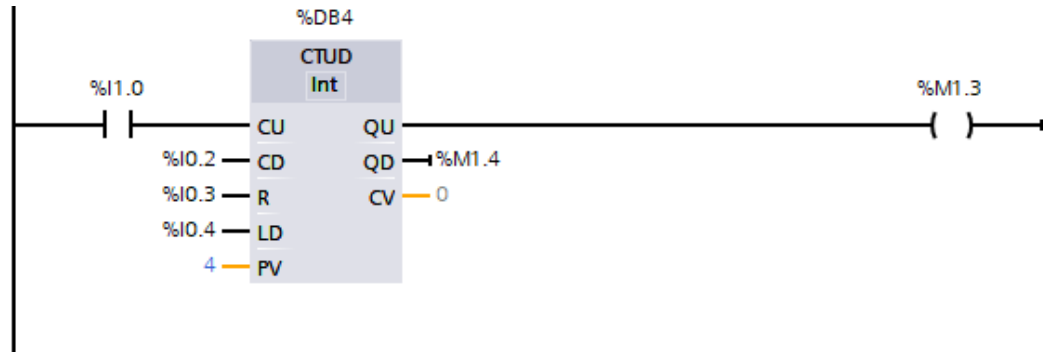


Combina el conteo ascendente y descendente:

- CU: entrada de conteo ascendente.
- CD: entrada de conteo descendente.
- Entrada LD en la que se carga el valor de PV en el contador (CV).
- Salida QU que está activa cuando el valor del contador CV es mayor o igual que PV.
- Salida QD que está activa cuando el valor del contador CV es menor o igual a 0.

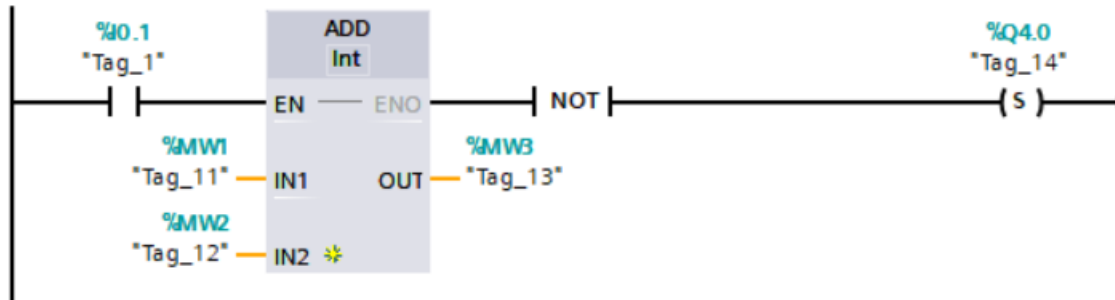
Lenguaje Ladder (LD). Operaciones de temporización

CTUD (Counting Up Down): Contador ascendente/descendente

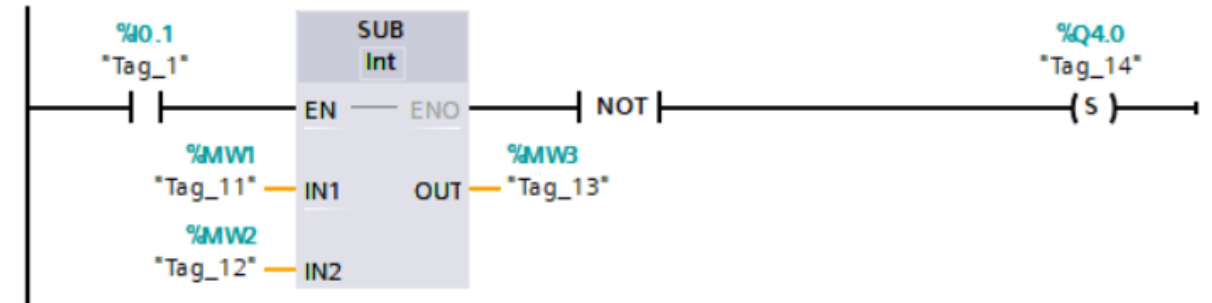


Lenguaje Ladder (LD). Operaciones aritméticas

Operaciones aritméticas (Enteros – Integer, Flotantes – Real)

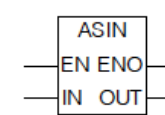
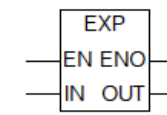
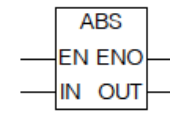
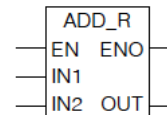


ENO se activa si I0.0 = 1. El resultado de la suma MW1 + MW2 se guarda en MW3. Si el resultado es un valor fuera de rango o si I0.1 = 0, ENO = 0 (Q4.0 = 1).





























ENO se activa si I0.0 = 1. El resultado de la resta MW1 - MW2 se guarda en MW3. Si el resultado es un valor fuera de rango o si I0.0 = 0, ENO = 0 (Q4.0 = 1).

Otras operaciones MUL, DIV, MOD, SQRT, SIN, COS,



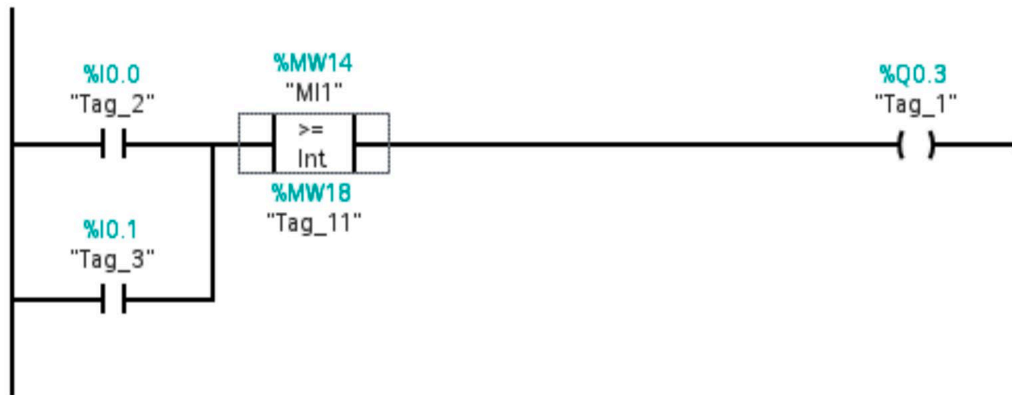
Si hay un overflow al realizar una operación se activan los bits OV y OS del PLC.

Lenguaje Ladder (LD). Operaciones aritméticas

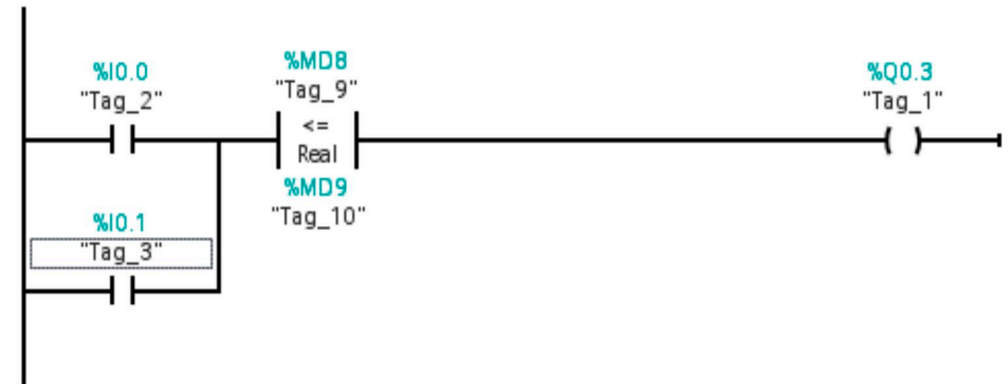
▼  Funciones matemáticas		
	CALCULATE	Calcular
	ADD	Sumar
	SUB	Restar
	MUL	Multiplicar
	DIV	Dividir
	MOD	Obtener resto de división
	NEG	Generar complemento a dos
	INC	Incrementar
	DEC	Decrementar
	ABS	Calcular valor absoluto
	MIN	Determinar mínimo
	MAX	Determinar máximo
	LIMIT	Ajustar valor límite
	SQR	Calcular cuadrado
	SQRT	Calcular raíz cuadrada
	LN	Calcular logaritmo natural
	EXP	Calcular valor exponencial
	SIN	Calcular valor de seno
	COS	Calcular valor de coseno
	TAN	Calcular valor de tangente
	ASIN	Calcular valor de arcoseno
	ACOS	Calcular valor de arcocoseno
	ATAN	Calcular valor de arcotangente
	FRAC	Determinar decimales
	EXPT	Elevar a potencia

Lenguaje Ladder (LD). Operaciones aritméticas

Operaciones de comparación: ==, <>, >, <, >=, <=













Q0.3 se activa si I0.0 ó I0.1 están activas y si MW14 >= MW18



Q0.3 se activa si I0.0 ó I0.1 están activas y si MD8 <= MD9

Lenguaje Ladder (LD). Operaciones aritméticas

▼ < Comparación	
 CMP ==	Igual
 CMP <>	Diferente
 CMP >=	Mayor o igual
 CMP <=	Menor o igual
 CMP >	Mayor
 CMP <	Menor
 IN_Range	Valor dentro del rango
 OUT_Range	Valor fuera del rango
 - OK -	Comprobar validez
 - NOT_OK -	Comprobar invalidez

Lenguaje Ladder (LD). Instrucciones de control

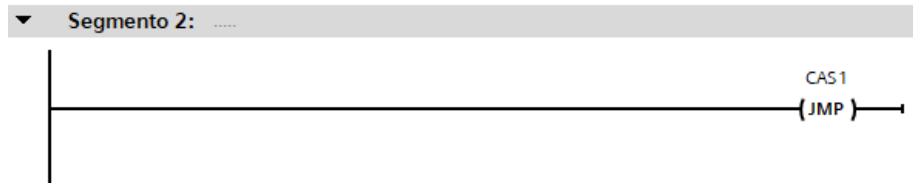
- Las instrucciones de control posibilitan la toma de decisiones que influyen en la ejecución de las restantes instrucciones.
- Las instrucciones de control pueden ser de dos tipos:
 - Instrucciones de salto
 - Instrucciones de llamada y retorno de módulo

Lenguaje Ladder (LD). Instrucciones de control. Salto

- Las instrucciones de salto está ligadas a etiquetas de identificación (LABELS).
- Pueden ser:
 - Incondicionales: El salto a la etiqueta se produce se produce siempre.
 - Condicionales: El salto a la etiqueta se produce si se cumple una determinada condición.

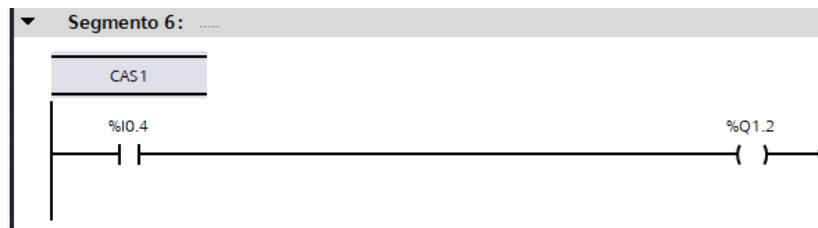
Lenguaje Ladder (LD). Instrucciones de control. Salto

- JMP: salto incondicional. Hace que no se ejecuten las instrucciones situadas entre la línea que activa el salto y la que tenga la etiqueta.



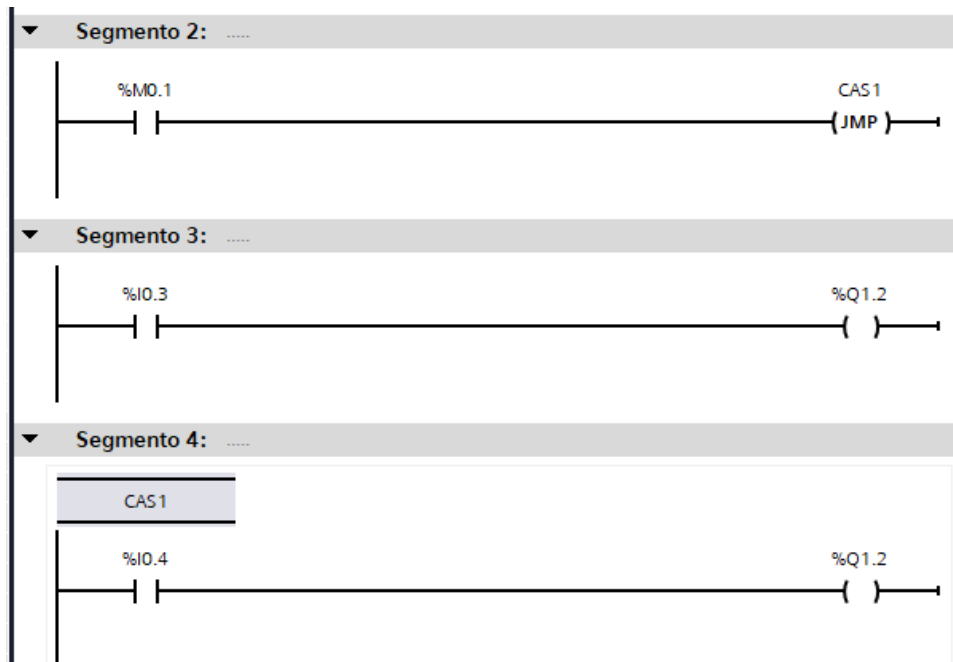
.
. .
.

El salto se ejecuta siempre y se omiten las instrucciones entre la línea de salto y la línea 6 (meta)



Lenguaje Ladder (LD). Instrucciones de control. Salto

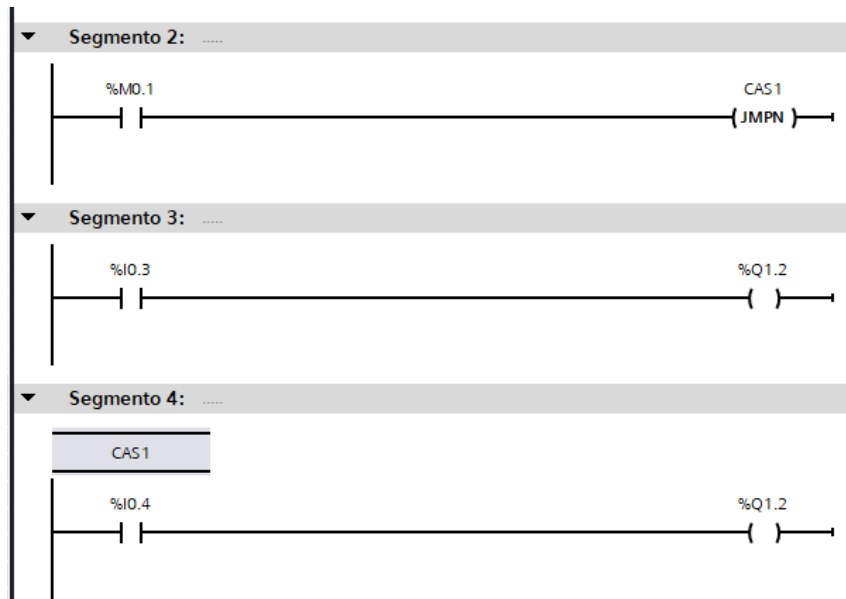
- JMP: salto condicional. Funciona como un salto incondicional, pero se ejecuta el salto si el RLO es 1.



Si `M0.1 = 1` el segmento 3 no se ejecuta, `Q1.1` no se pondría a 0 (1) si `I0.3 = 0` (1).

Lenguaje Ladder (LD). Instrucciones de control. Salto

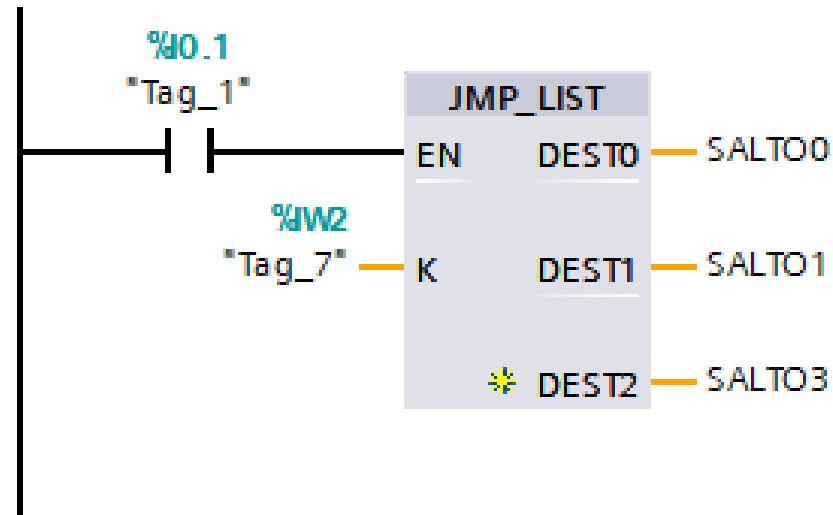
- JMPN: salto condicional negado. Funciona como un salto incondicional, pero se ejecuta el salto si el RLO es 0 .



Si `M0.1 = 0` se produce el salto hasta el segmento 4 (Etiqueta `CAS1`) y el segmento 3 no se ejecuta.

Lenguaje Ladder (LD). Instrucciones de control. Salto

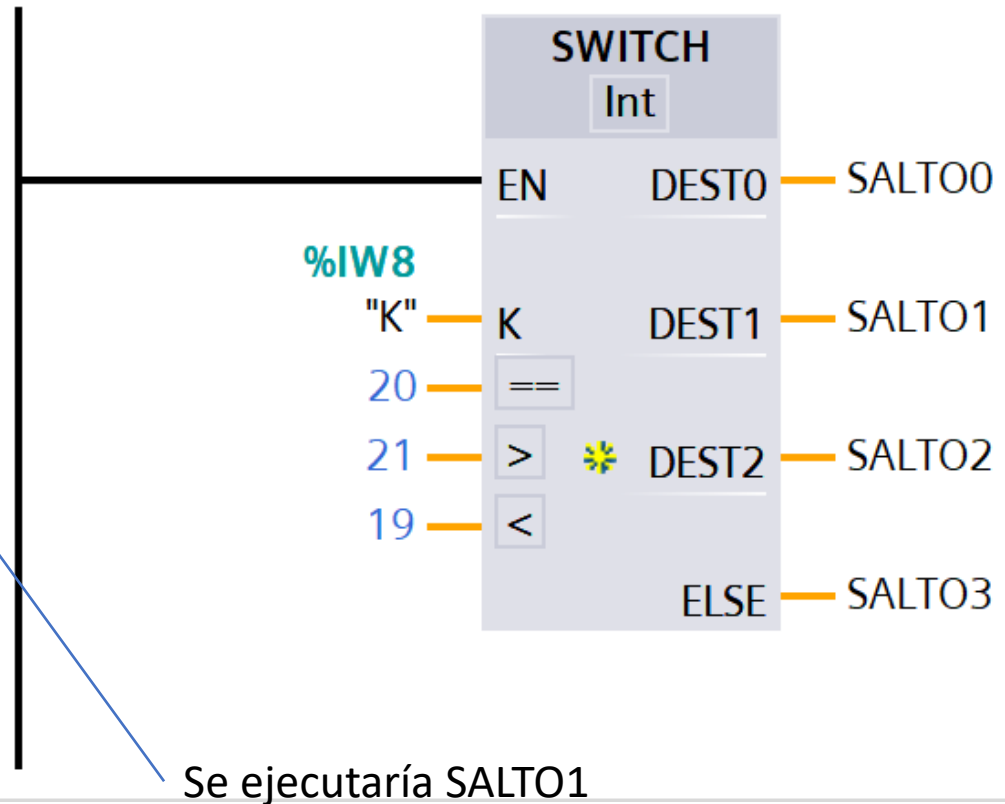
- JMP_LIST permite realizar un salto a diferentes lugares en función de un parámetro (IW2 = 1 → SALTO 1)



Lenguaje Ladder (LD). Instrucciones de control. Salto

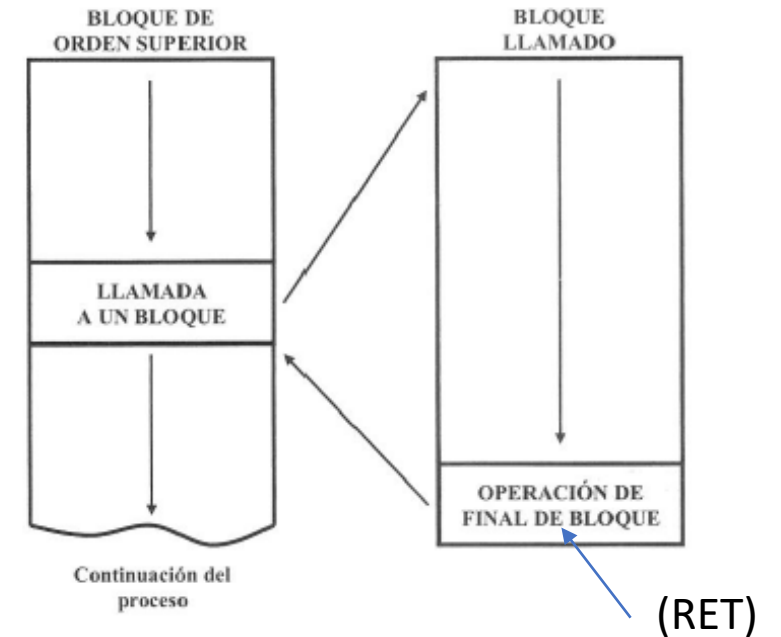
- SWITCH permite realizar un salto a diferentes lugares en función de un parámetro

PARAMETRO	Valor
K	23
==	20
>	21
<	19
DEST0	Salto si K = 20
DEST1	Salto si K > 21
DEST2	Salto si K < 19



Lenguaje Ladder (LD). Instrucciones de control. Llamada y retorno de módulo

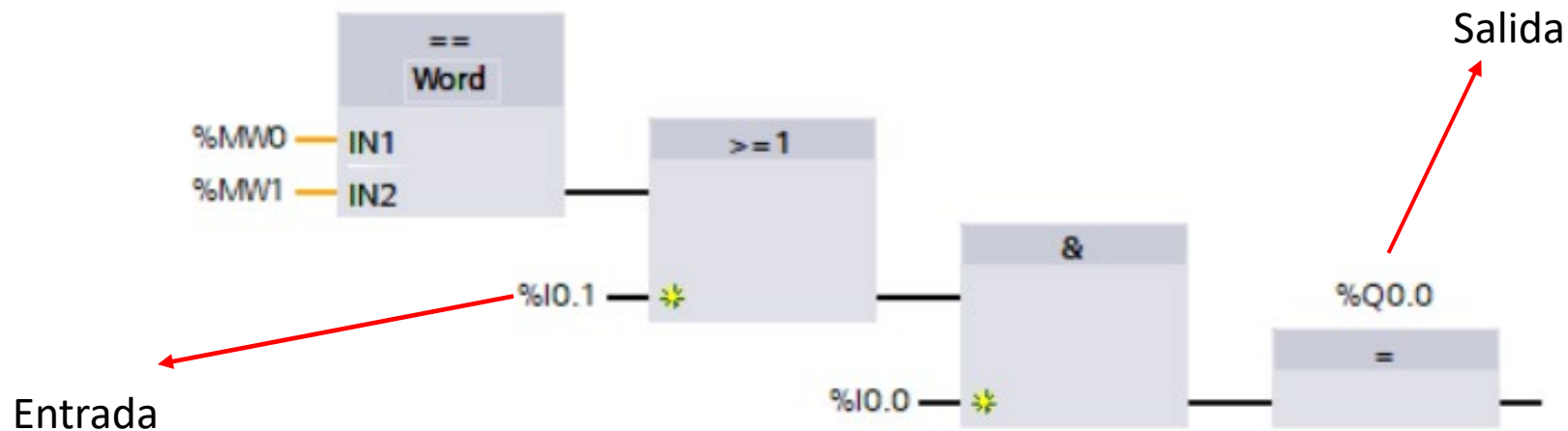
- Algunas tareas en los sistemas de control se tienen que ejecutar varias veces. Se puede definir un bloque funcional que puede ser llamado varias veces para no tener que repetir el código
- Un módulo puede llamar a su vez a otro módulo: Es necesario estructurar estas llamadas



Fuente: Sistemas de automatización y autómatas programables. Enrique Mandado et al.

Lenguaje Function Block Diagram (FBD)

- Utilizado en Europa debido a la mayor formación de ingenieros electrónicos en Alemania que se ocupaban de tareas de control.
- Se utilizan bloques lógicos para representar la interacción entre variables.
- Fácil de utilizar para personas habituadas a implementar Sistemas Digitales.
- Siemens: FUP (Funktionsplan).



Lenguaje Function Block Diagram (FBD)

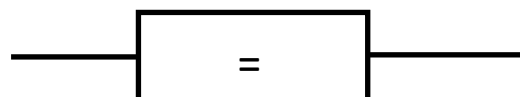
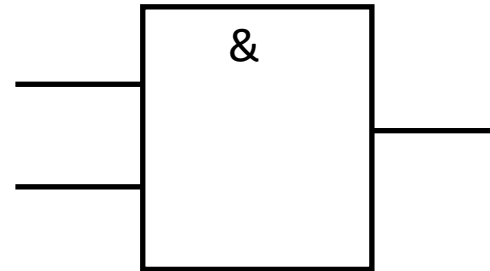
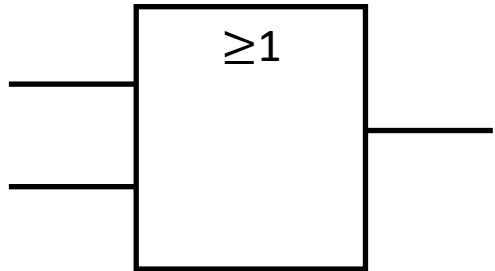
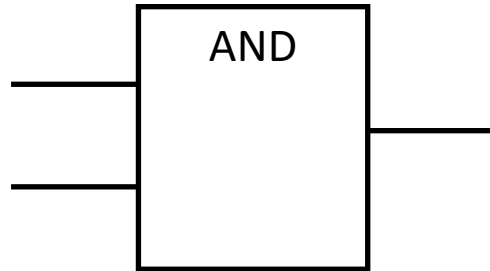
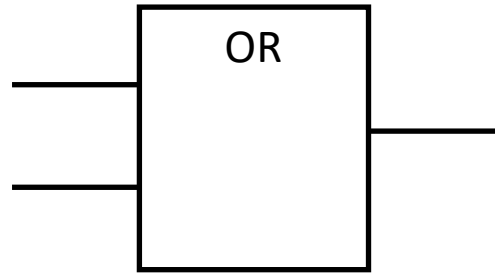
- Un programa FBD consiste en la conexión entre bloques funcionales y datos por líneas de conexión:
 - El programa se ejecuta de arriba a abajo y de izquierda a derecha
 - Se puede conectar la salida de un bloque a la entrada de otro bloque
- La salida de cada bloque es única.
- Distintos tipos de datos (dependiendo del bloque)

Lenguaje Function Block Diagram (FBD)

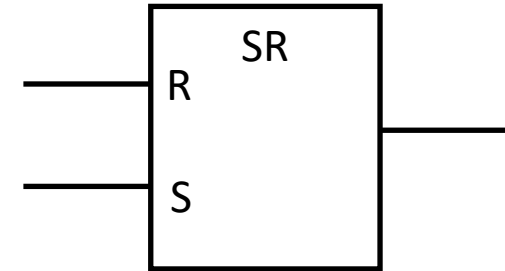
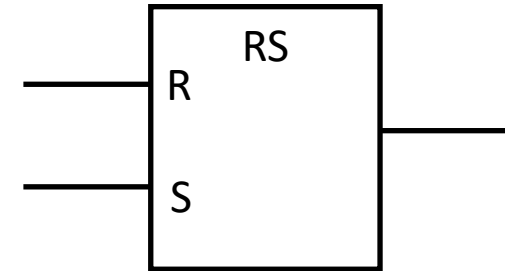
- Grupos de instrucciones:
 - Funciones binarias: Entradas y salidas binarias AND, OR, XOR,... Modificable número de entradas y posibilidad de negación de entradas.
 - Temporizadores y contadores: Bloques vistos en Ladder
 - Funciones de palabras/reales:
 - Comparación: Resultado es un bit
 - Aritméticas: Resultado es una palabra/real (+, -, x, /)
 - Funciones lógicas bit a bit: AND, OR, Rotate, Shift,...
 - Funciones de Control de Programa: JUMP
 - Funciones de Control: PI, PID,...

Lenguaje Function Block Diagram (FBD)

Bloques operaciones lógicas: OR, AND, ...



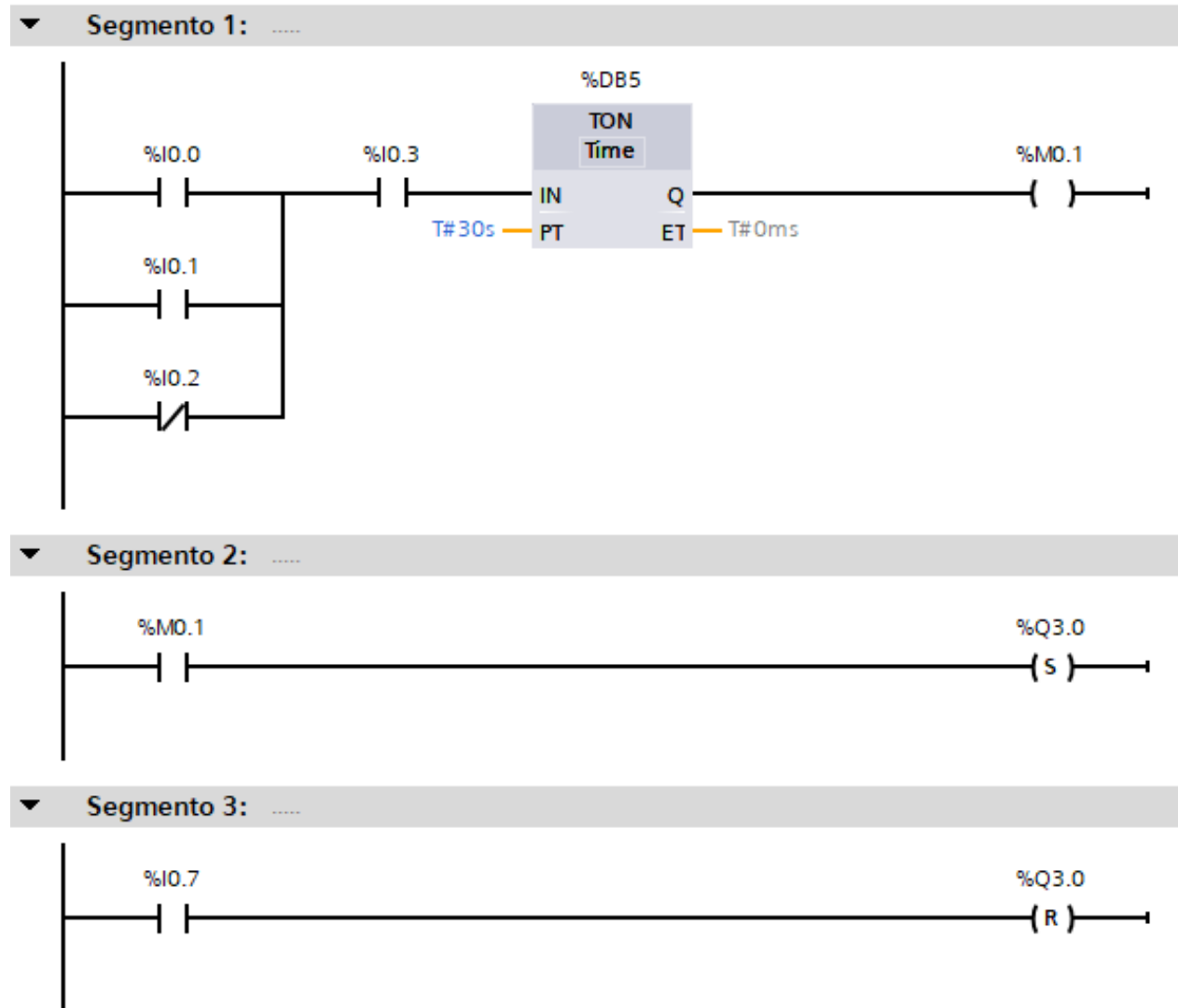
Biestables



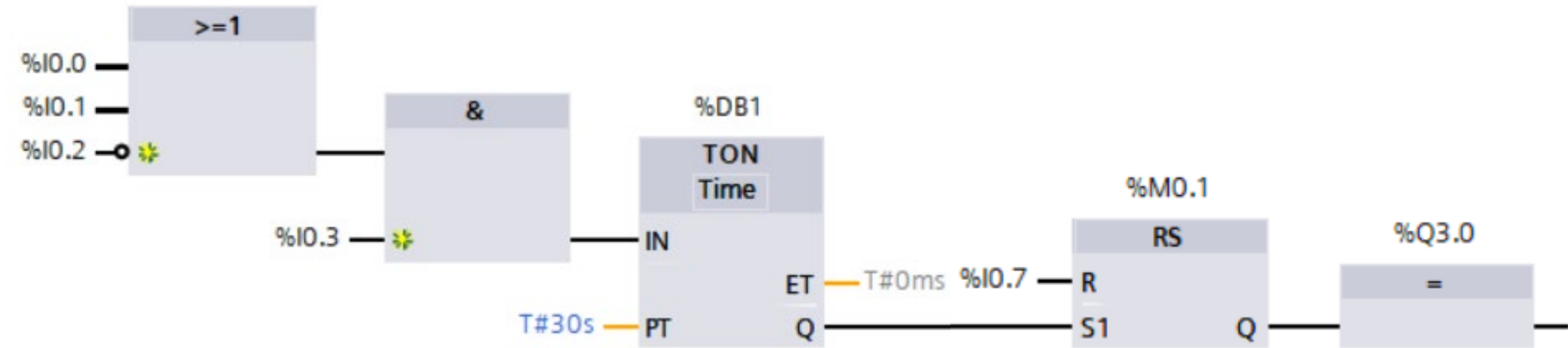
Lenguaje Function Block Diagram (FBD)

- Bloques funcionales:
 - Se definen sobre la base de templates de bloques
 - Template de bloque: programa que define el bloque
 - Para usar el bloque funcional se declara una instancia del bloque
 - Re-uso del código
- Bloque funcional vs Función:
 - Salidas:
 - Bloque funcional: permite más de una salida
 - Función: permite una única salida
 - Variables persistentes (memoria):
 - Función: No
 - Bloque funcional: Sí

Lenguaje Function Block Diagram (FBD)

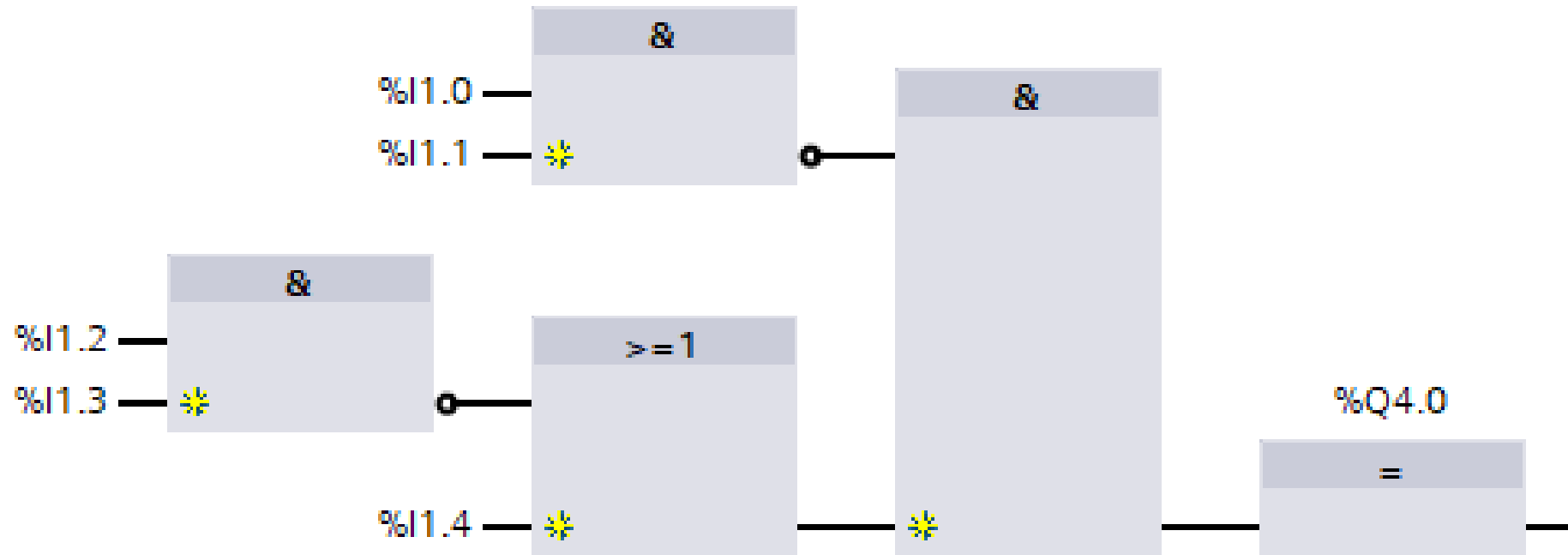


Lenguaje Function Block Diagram (FBD)



Lenguaje Function Block Diagram (FBD)

- ¿Cuándo se activa Q4.0?

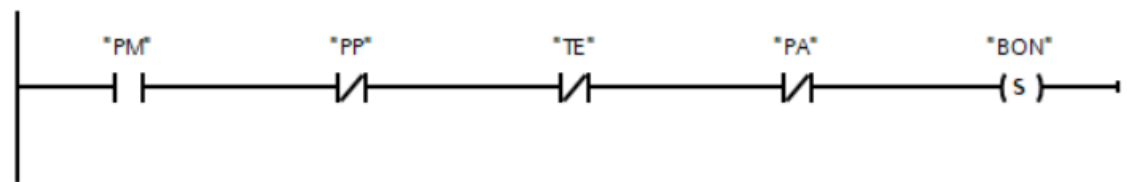


Lenguaje Function Block Diagram (FBD).

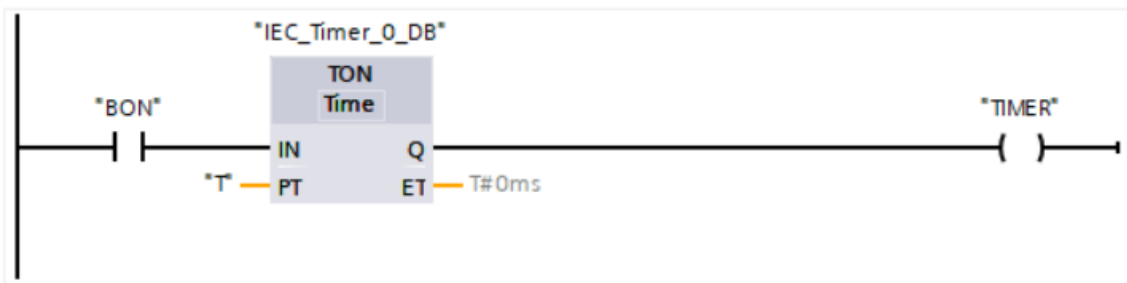
Ejemplo

- Diseña un programa en LD y en FBD que controle el encendido/apagado de una bomba. La bomba se enciende si:
 - Se pulsa el botón de marcha (PM) (NO).
 - La protección térmica (TE) está desactivada (NO)
 - El botón de parada (PP) no está activado (NO)
 - El botón de alarma (PA) no está activado (NO).
- Pasado un tiempo T después del encendido la corriente debe cumplir $I_{\min} < I < I_{\max}$ (I_{\min} e I_{\max} son valores prefijados)
- El motor se para si:
 - Se pulsa el botón de parada (PP) o de alarma (PA).
 - Salta la protección térmica.
 - La corriente no está en los límites fijados.

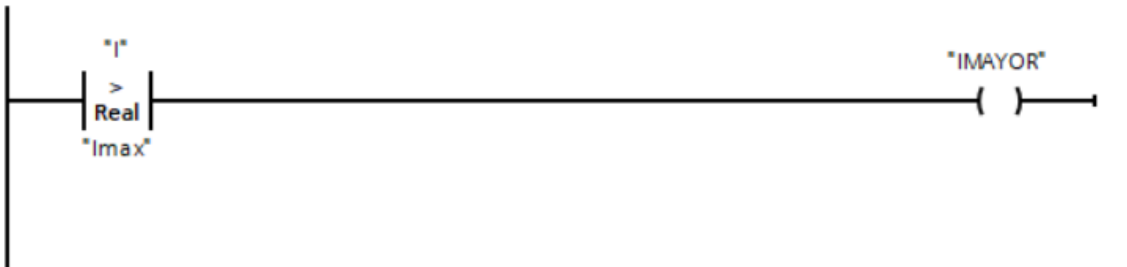
▼ Segmento 1:



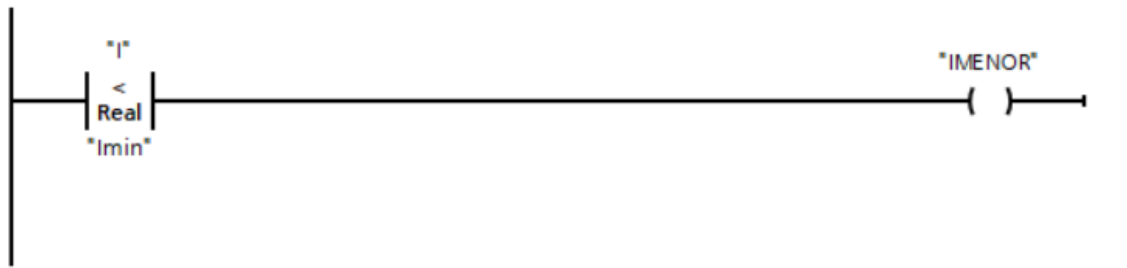
▼ Segmento 2:



▼ Segmento 3:



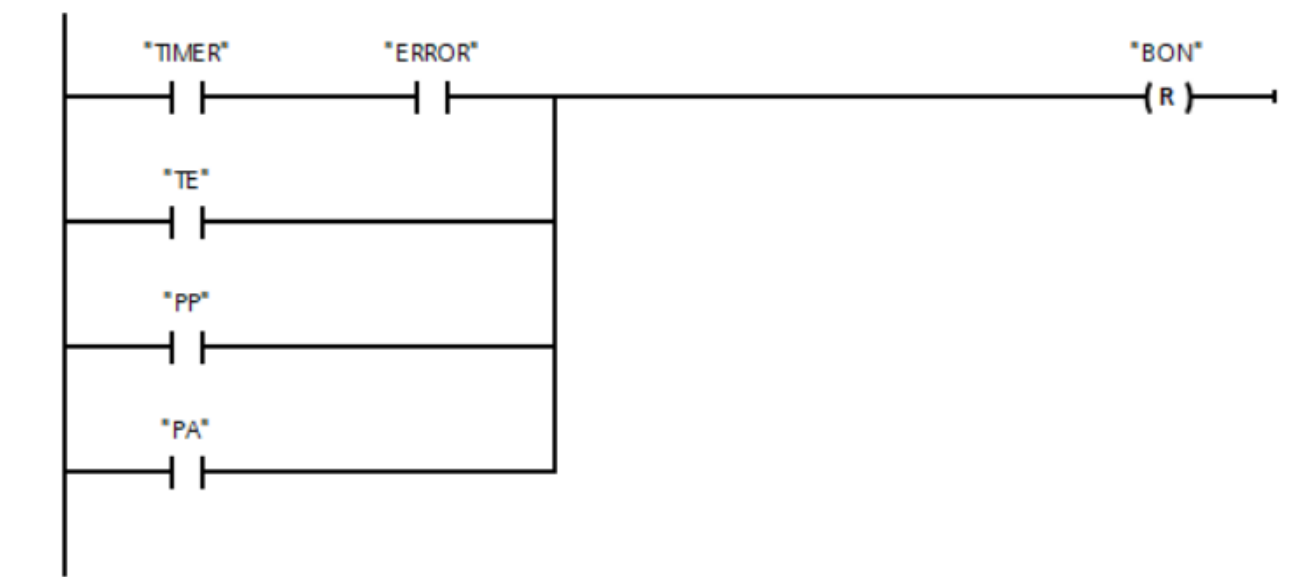
▼ Segmento 4:



▼ Segmento 5:



▼ Segmento 6:



LENGUAJE IL

- El lenguaje IL (Instruction List) consiste en un conjunto de códigos simbólicos que se corresponde a una o más instrucciones en lenguaje máquina del PLC
- Programación muy próxima al lenguaje máquina.
- Variables:
 - Entrada: %In
 - Salida: %Qn
 - Salidas intermedias o marcas: %Mn
- Tipos de datos:
 - Bit: X
 - Byte: B
 - Palabras: W
 - Dobles palabras: DW

Operador	Modificador	Operando	Descripción
LD	N		Selecciona la variable
ST	N		Actúa sobre una variable de salida
S		Variable lógica (Bool)	Pone la variable de 1 bit a 1 (set)
R		Variable lógica (Bool)	Pone la variable de 1 bit a 0 (reset)
AND	N, (Variable lógica (Bool)	Operación and de un bit
OR	N, (Variable lógica (Bool)	Operación or de un bit
XOR	N, (Variable lógica (Bool)	Operación xor de un bit
ADD	(Suma
SUB	(Resta
MUL	(Multiplicación
DIV	(División
MOD	(Resto
GT	(Mayor que
GE	(Mayor o igual que
EQ	(Igual a
NE	(Distinto a

Operador	Modificador	Operando	Descripción
LE			Menor o igual que
LT			Menor que
JMP	C, N	Etiqueta (Label)	Salto a etiqueta
CAL	C, N	Nombre	Saltar a un bloque funcional
RET	C, N		Retorno de bloque funcional
)			Evaluar la operación
AND	N, (Byte, Word, DWord	AND lógico entre elementos
OR	N, (Byte, Word, DWord	OR lógico entre elementos
XOR	N, (Byte, Word, DWord	XOR lógico entre elementos

	<i>Normalizada IEC 1131-3</i>	<i>STEP7 inglesa (Internacional)</i>	<i>STEP7 alemana (SIMATIC)</i>
<i>Variables predefinidas</i>			
Entradas	I, IX, IB, IW	I, IB, IW, ID	E, EB, EW, ED
Salidas	Q, QX, QB, QW, QD	Q, QB, QW, QD	A, AB, AW, AD
Marcas	M, MX, MB, MW, MD	M, MB, MW, MD	M, MB, MW, MD
<i>Operaciones lógicas básicas</i>			
Carga inicial	LD	A ó O	U ó O
Y	AND	A	U
NO-Y	ANDN	AN	UN
O	OR	O	O
NO-O	ORN	ON	ON
O-exclusiva	XOR	X	X
NO-O-exclusiva	XORN	XN	XN
<i>Operaciones con paréntesis</i>			
Y	AND(A(U(
NO-Y	ANDN(AN(UN(
O	OR(O(O(
NO-O	ORN(ON(ON(
O-exclusiva	XOR(X(X(
NO-O-exclusiva	XORN(XN(XN(
Cerrar paréntesis)))
<i>Terminar una cadena lógica</i>			
Asignar	ST	=	=
Desactivar	R	R	R
Activar	S	S	S
<i>Operaciones con flancos</i>			
Flanco negativo	LDF, ORF, ANDF	FN	FN
Flanco positivo	LDR, ORR, ANDR	FP	FP

Lenguaje IL. Instrucciones variables lógicas

- Las instrucciones que operan con variables lógicas especifican un solo operando en la instrucción, como la mayoría de operaciones necesitan dos operandos el otro se guarda en un flip-flop o biestable denominado **RLO** (Result of Logic Operation).
- SIEMENS lo denomina AWL.
- Las instrucciones que operan con variables lógicas pueden ser:
 - Selección, entrada y salida o de operación
 - Operación con flancos
 - Memorización

Operador	Modificador	Operando	Descripción
A ó O	N	Variable lógica (Boolean)	Selecciona la primera variable
=			Actúa sobre una variable de salida
S		Variable lógica (Boolean)	Pone la variable de 1 bit a 1 (set)
R		Variable lógica (Boolean)	Pone la variable de 1 bit a 0 (reset)
A	N, (Variable lógica (Boolean)	Operación and de un bit
O	N, (Variable lógica (Boolean)	Operación or de un bit
X	N, (Variable lógica (Boolean)	Operación xor de un bit
)			Evalúa la operación
NOT		RLO	Invierte el contenido de RLO
CLR		RLO	Pone a cero el RLO
SET		RLO	Pone a uno el RLO
FP		Variable lógica (Boolean)	Flanco positivo
FN		Variable lógica (Boolean)	Flanco negativo

Instrucciones que operan sobre variables lógicas en STEP 7 (Nomenclatura internacional)

Lenguaje IL. Instrucciones variables lógicas

- Las instrucciones de selección, entrada y salida o de operación con variables lógicas realizan alguna de estas acciones:
 - Seleccionar una variable para usarla como operando o lectura de una entrada.
 - Activación o desactivación de una salida.
 - Operación con variables lógicas.
 - Inicialización de RLO
- **Instrucciones sin paréntesis:** A ó O, A, AN, O, ON, X, XN
 - A ó O: Carga inicial de variable en el RLO:
A %I0.1 // (RLO = I0.1) ó O %I0.1 // (RLO = I0.1)

Lenguaje IL. Instrucciones variables lógicas

Instrucciones sin paréntesis

- AN ó ON : Carga invertida: Carga la variable invertida en el RLO:

AN %I0.1 // (RLO = NOT I0.1) ON %I0.1 // (RLO = NOT I0.1)

- = : Asigna el valor del RLO a la variable indicada

A %I0.1. // RLO = I0.1

= %Q0.2 // Q0.2 = RLO = I0.1

Lenguaje IL. Instrucciones variables lógicas

Instrucciones sin paréntesis:

- O: OR lógico

$$Q0.0 = Q0.5 + I3.1 + M2.7$$

```
O  %Q0.5    // RLO = Q0.5
O  %I3.1     // RLO = RLO OR I3.1
O  %M2.7     // RLO = RLO OR M2.7
=  %Q0.0     // Q0.0 = RLO
```

- ON: OR entre una variable y la inversa de la variable especificada

$$Q0.0 = Q0.5 + \overline{I3.1} + \overline{M2.7}$$

```
O  %Q0.5     // RLO = Q0.5
ON %I3.1      // RLO = RLO OR  $\overline{I3.1}$ 
ON %M2.7      // RLO = RLO OR  $\overline{M2.7}$ 
=  %Q0.0      // Q0.0 = RLO
```

Lenguaje IL. Instrucciones variables lógicas

Instrucciones sin paréntesis

- A: AND lógico

$$Q0.0 = Q0.5 \cdot I3.1 \cdot M2.7$$

```
A %Q0.5      // RLO = Q0.5
A %I3.1      // RLO = RLO AND I3.1
A %M2.7      // RLO = RLO AND M2.7
= %Q0.0      // Q0.0 = RLO
```

- AN: AND entre una variable y la inversa de la variable especificada

$$Q0.0 = \overline{Q0.5} + \overline{I3.1} + \overline{M2.7}$$

```
AN %Q0.5     // RLO =  $\overline{Q0.5}$ 
AN %I3.1     // RLO = RLO AND  $\overline{I3.1}$ 
AN %M2.7     // RLO = RLO AND  $\overline{M2.7}$ 
= %Q0.0      // Q0.0 = RLO
```

Lenguaje IL. Instrucciones variables lógicas

Instrucciones sin paréntesis

- X: XOR lógico

$$Q0.0 = Q0.5 \oplus I3.1 \oplus M2.7$$

```
A %Q0.5      // RLO = Q0.5
X %I3.1       // RLO = RLO XOR I3.1
X %M2.7       // RLO = RLO XOR M2.7
= %Q0.0       // Q0.0 = RLO
```

- XN: XOR entre una variable y la inversa de la variable especificada

$$Q0.0 = \overline{Q0.5} \oplus I3.1 \oplus \overline{M2.7}$$

```
AN %Q0.5      // RLO =  $\overline{Q0.5}$ 
X %I3.1       // RLO = RLO XOR I3.1
XN %M2.7      // RLO = RLO XOR  $\overline{M2.7}$ 
= %Q0.0       // Q0.0 = RLO
```

Lenguaje IL. Instrucciones variables lógicas

O %I0.4
AN %I1.7
O %M2.5
AN %I1.6
= %Q1.0

?

Lenguaje IL. Instrucciones variables lógicas

- **Instrucciones con paréntesis:** A(, O(, X, AN(, ON(, X(, XN(

O	%I1.7	Carga en RLO la variable I1.7
O	%I2.9	OR entre I1.7 e I2.9
AN	%M0.5	AND con M0.5 negado
O(Nuevo término OR
O	%M2.3	Carga en RLO la variable M2.3
AN	%I1.6	OR con I1.6 negado
ON	%M2.6	Cierre OR de las dos expresiones
)		Cierre OR
=	%Q0.8	Asignar el valor del RLO a Q0.8

$$Q0.8 = \left((I1.7 + I2.9) \cdot \overline{Q0.5} \right) + (M2.3 \cdot I1.6 + \overline{M2.6})$$

Lenguaje IL. Instrucciones variables lógicas

Tenemos un electrocompresor que inyecta aire comprimido en un calderín y que está accionado por un motor eléctrico, alimentado por un contactor que posee los siguientes elementos de control:

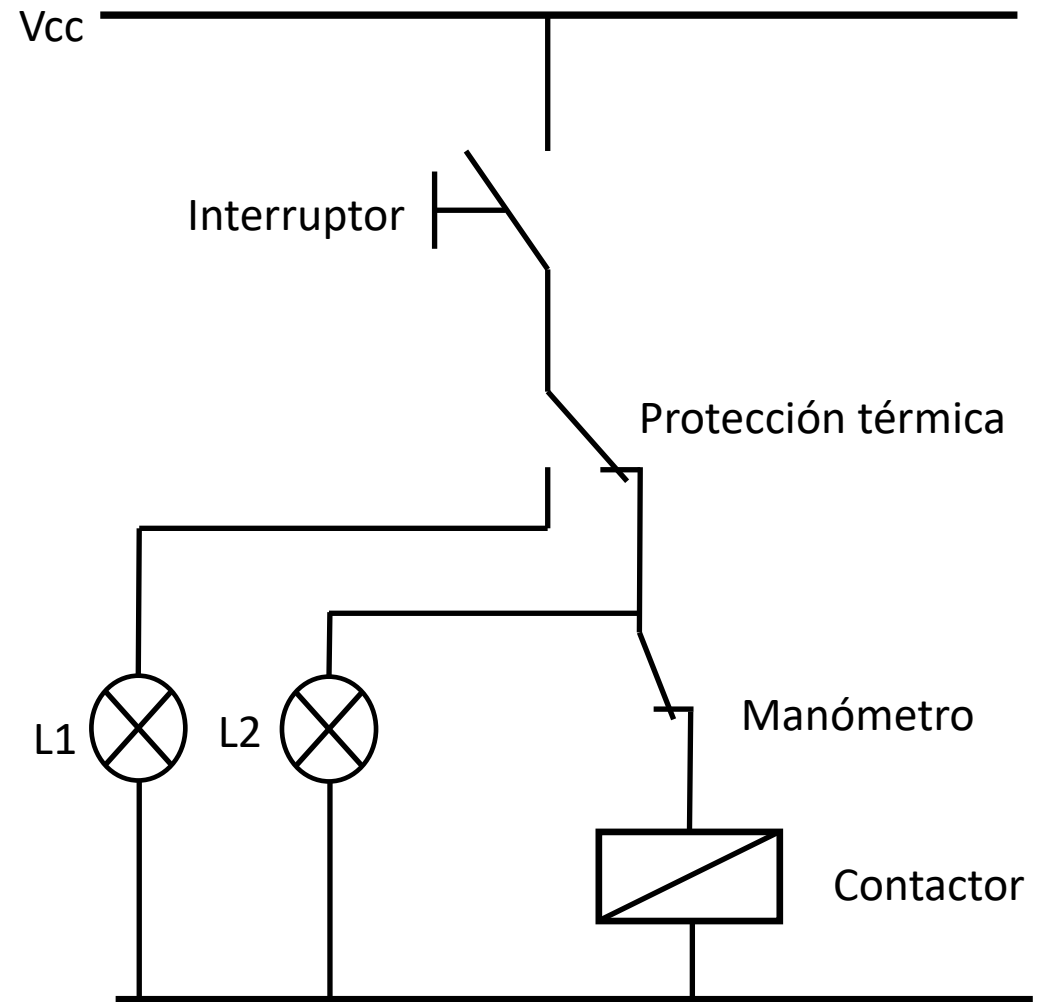
- Un interruptor de marcha que activa el contactor.
- Un relé de protección térmica de sobreintensidad del motor que dispone de un contactor NC que se abre cuando la intensidad que circula por el motor supera un valor umbral.
- Un manómetro en el calderín que posee un contacto NC que se abre cuando se superan los 5 KPa provocando la parada del motor y se cierra cuando la presión desciende de 4 KPa.

Diseña un programa en IL (AWL) que controle el electrocompresor de acuerdo con estas especificaciones:

- El estado del circuito de control de señaliza mediante dos lámparas L1 y L2.
- La lámpara L1 de señalización de alarma se enciende cuando se dispara la protección térmica.
- La lámpara L2 de señalización de servicio se enciende cuando está cerrado el interruptor de marcha y no se ha disparado la protección térmica, independientemente del estado de manómetro.

Lenguaje IL. Instrucciones variables lógicas

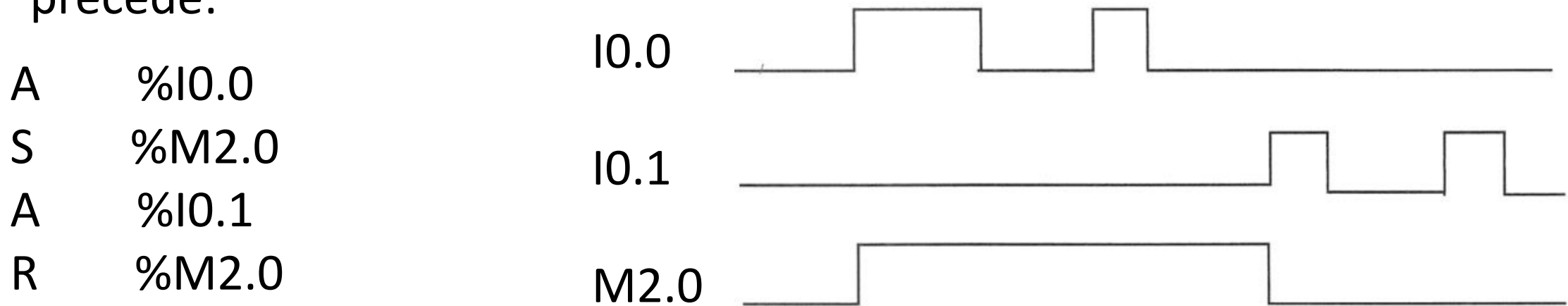
Entradas	Descripción
I0.0	Interruptor de marcha
I0.1	Contacto (NC) de protección térmica
I0.2	Contacto (NC) del manómetro
Salidas	Descripción
Q1.0	Contactor del motor
Q1.1	Alarma de protección térmica L1
Q1.2	Señalización de servicio L2



Lenguaje IL. Instrucciones variables lógicas

Instrucciones de memorización

- Instrucciones que actúan sobre el RLO: SET (puesta a 1), CLR (puesta a 0) y NOT (invierta el valor del RLO)
- Set (S) y Reset (R) actúan sobre el valor de la variable lógica que les precede:



BIESTABLE RS desactivación prioritaria

Lenguaje IL. Instrucciones variables lógicas

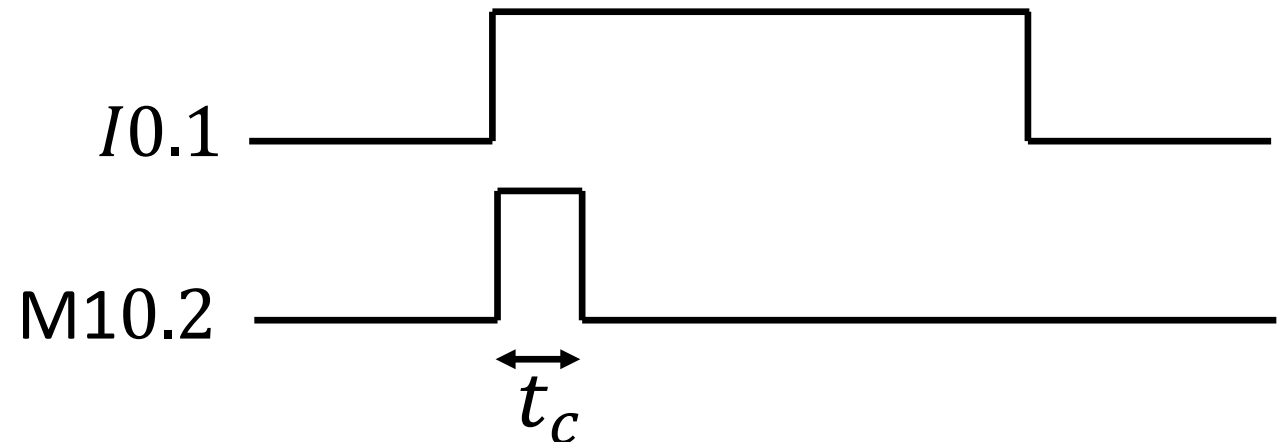
- Instrucciones que operan por flancos: FP (flancos de subida) o FN (flancos de bajada). Se necesita una marca auxiliar para almacenar el resultado de la variable en el ciclo anterior (no se puede utilizar a lo largo del programa)

A I0.1
FP M37.1
= M10.2
A M10.2
A I1.3
= Q2.5



A I0.1
FP M37.1
A I1.3
= Q2.5

$$Q2.5 = (I0.1 \uparrow \cdot I1.3)$$



Lenguaje IL

Tenemos un depósito de agua que incluye los siguientes elementos de control:

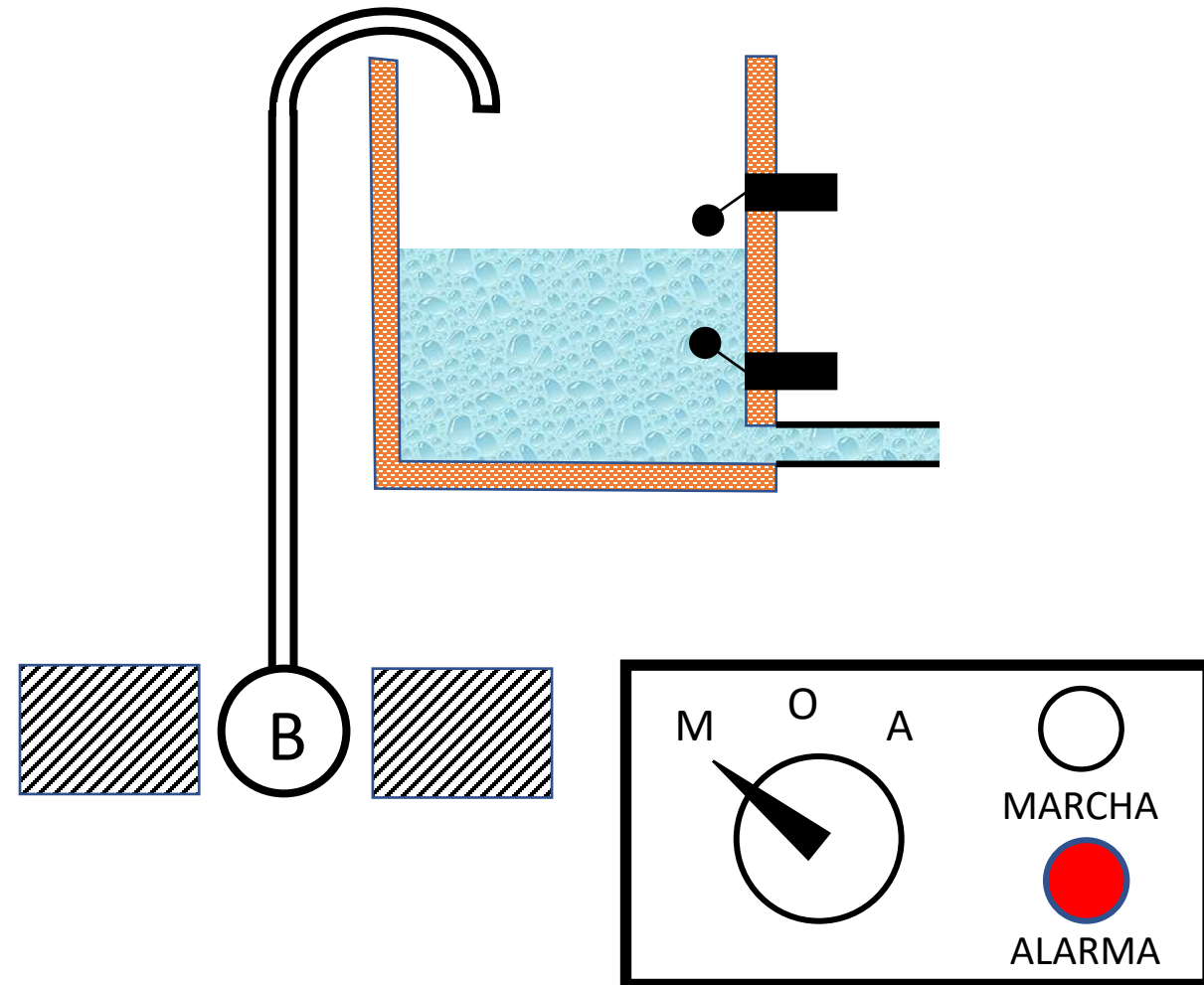
- Dos detectores de nivel que detectan el nivel máximo (I1) y mínimo (I2) del depósito
- Una bomba que suministra agua al depósito
- Un panel de control con tres posiciones: M (manual), A (automático) y O (fuera de servicio)
- Un relé de protección térmica

Requisitos:

- Si está activada la posición M la bomba funciona de forma continua con independencia del nivel del depósito.
- Si está activada la posición A el depósito debe estar entre el máximo y el mínimo, de tal forma que cuando el depósito alcanza el mínimo la bomba funciona hasta llegar al máximo
- Si está activada a posición O la bomba está fuera de servicio.
- El relé térmico para la bomba cuando su temperatura alcanza un valor fijado y se debe iluminar la lámpara de Alarma.
- Cuando la bomba está en marcha se debe activar la lámpara de marcha.

Lenguaje Ladder (IL)

Entradas	Descripción
I0.0	Interruptor modo manual
I0.1	Interruptor modo automático
I0.2	Boya nivel inferior I2
I0.3	Boya nivel superior I1
I0.4	Contacto auxiliar NC relé térmico
Salida	Descripción
Q0.0	Contactor de la bomba
Q0.1	Señalización de marcha
Q0.2	Alarma de protección térmica



Lenguaje IL. Instrucciones combinaciones binarias

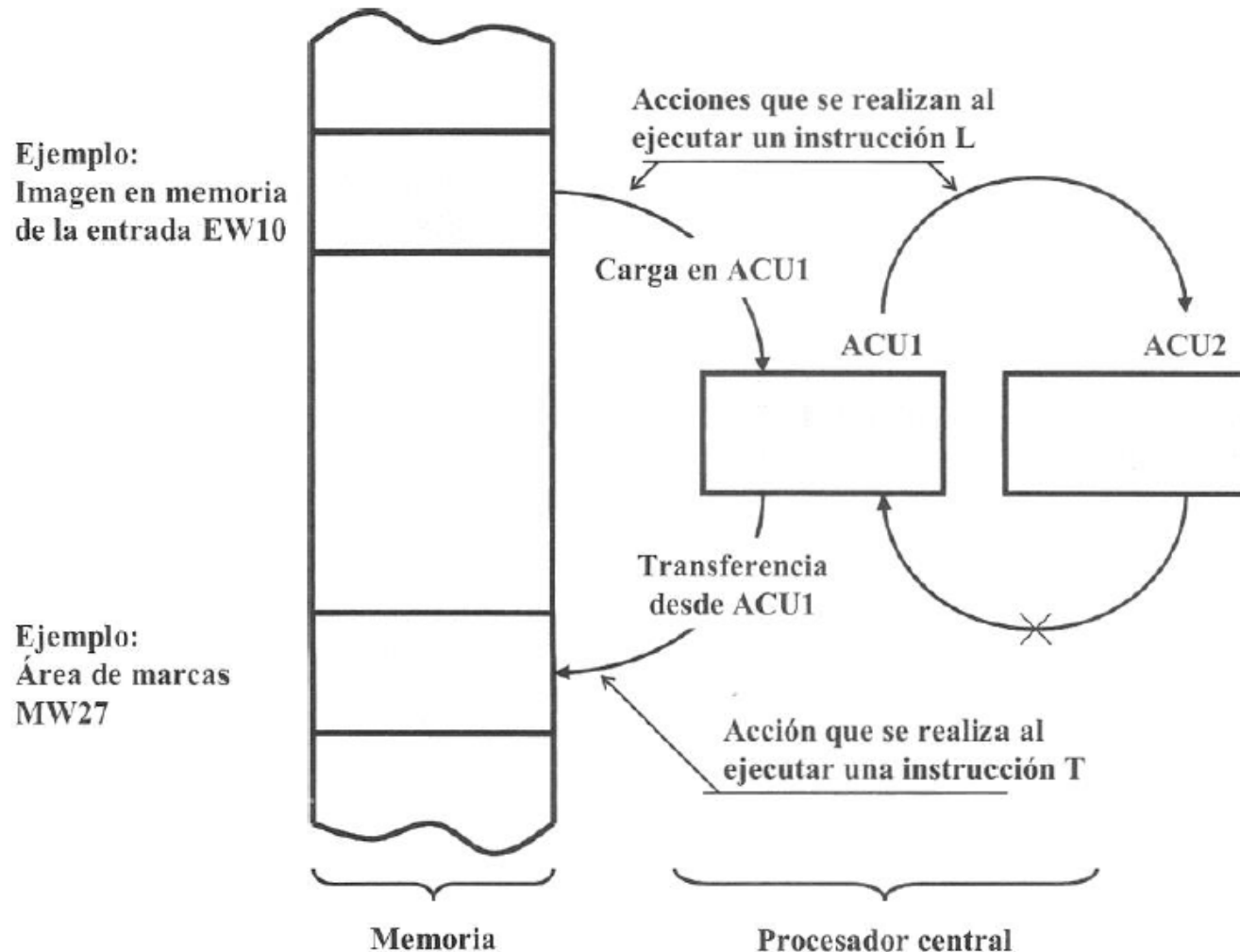
- Operan sobre palabras lógicas (16 o 32 bits)
- El resultado se almacena en un acumulador o en en registro RLO (operaciones lógicas)
- Pueden ser de 4 tipos:
 - Selección: L → L IW10 ; L QB6
 - Transferencia: T → T MW4
 - Aritméticas: ADD, SUB, MUL, DIV
 - Comparación: > (GT), >= (GE), < (LT), <= (LE), == (EQ), <> (NE):

```
1      L      %MD1
2      L      5
3      ==I
4      A(
5      L      %IW1
6      L      7
7      >=I
8      )
9      A      %I1.0
10     =      %Q1.2
```

$$Q1.2 = (IW0=5) \cdot (IW1 \geq 7) \cdot I1.0$$

- Lógicas

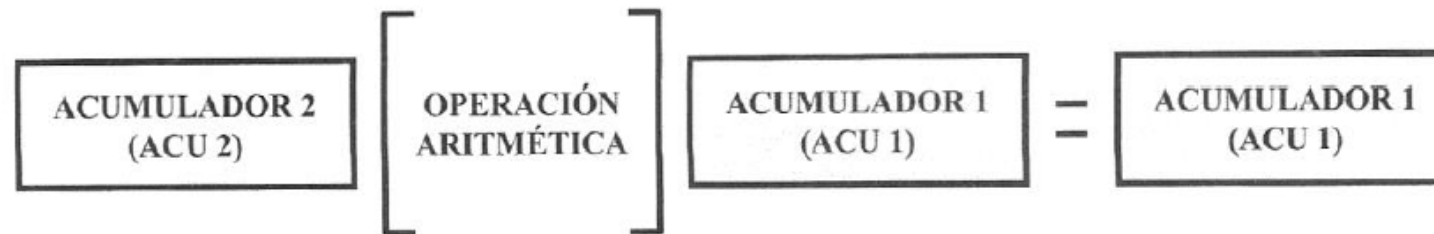
Lenguaje IL. Instrucciones combinaciones binarias



Cuando se carga un dato (Operación L) se almacena su valor en ACU1 y al mismo tiempo el valor que teníamos almacenado en ACU1 se transfiere a ACU2.

ACU1 y ACU 2 son dos registros de 32 bits.

Lenguaje IL. Instrucciones aritméticas de dos operandos



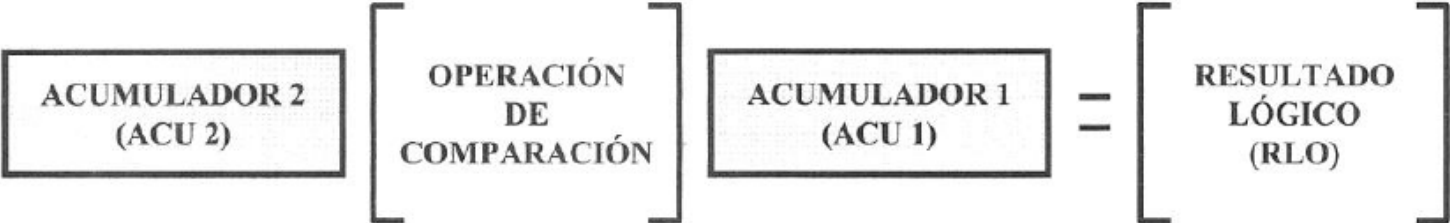
1	L	%MW0
2	L	+43
3	+I	
4	L	%MW2
5	*I	
6	T	%MW3

$$MW3 = (MW0 + 43) * MW2$$

I: enteros; D: enteros dobles; R: reales

Las operaciones aritméticas no modifican el RLO

Lenguaje IL. Instrucciones de comparación



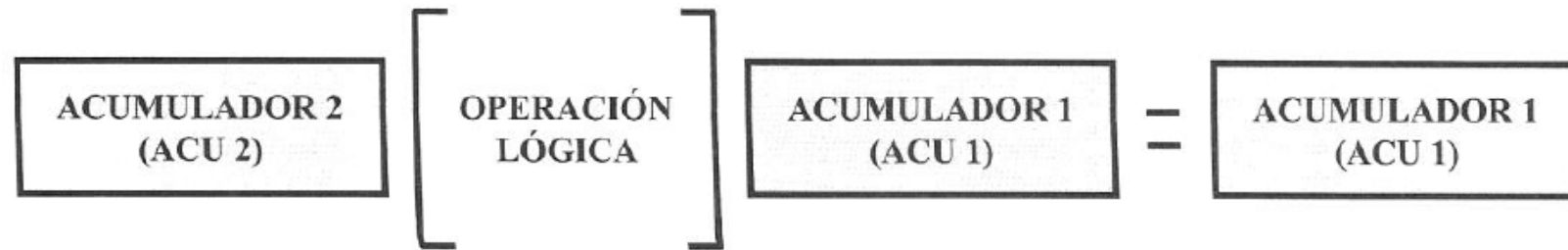
ACU2	Comparación	Operación	ACU1
Número entero Número entero doble Número en coma flotante	igual a	= =I = =D = =R	Número entero Número entero doble Número en coma flotante
	distinto a	<>I <>D <>R	
	mayor que	>I >D >R	
	menor que	<I <D <R	
	mayor o igual	>=I >=D >=R	
	menor o igual	<=I <=D <=R	

- 1
- 2
- 3
- 4
- 5

L %MW0
L +23
>I
A %M2.3
= %M4.6

// ACU1 = MW0
// ACU1 = 23 ACU2 = MW0
// RLO = ACU2 > ACU1 M2.3
// RLO = RLO · M2.3
// M4.6 = RLO

Lenguaje IL. Instrucciones combinaciones binarias



Instrucción	Operación
AW	AND entre los acumuladores (16 bits)
OW	OR entre los acumuladores (16 bits)
XOW	XOR entre los acumuladores (16 bits)
AD	AND entre los acumuladores (32 bits)
OD	OR entre los acumuladores (32 bits)
XOD	XOR entre los acumuladores (32 bits)

1	L	%MW1
2	L	%MW7
3	AW	
4	T	%MW9

Lenguaje IL. Instrucciones de temporización

Instrucción	Temporizador
SP	Impulso
SD	Retardo a la conexión
SF	Retardo a la desconexión



1	A	%I0.1
2	L	S5T#10s
3	SD	%T1
4	A	%T1
5	=	%M0.3

Variable tipo TIMER

Lenguaje IL. Instrucciones de contaje

1	A	%M0.2	// Señal de habilitación
2	FR	%C1	// Habilita el contador
3	A	%M2.1	// Señal de incremento
4	CU	%C1	// Incrementa el valor del contador C1 si I2.1 pasa de 0 a 1
5	A	%M2.2	// Señal de decremento
6	CD	%C1	// Decrementa el valor del contador C1 si I2.2 pasa de 0 a 1
7	A	%M2.3	// Señal de carga de valor inicial
8	L	C#3	// Carga el valor 3 en ACU1
9	S	%C1	// Transfiere el valor de ACU1 a C1 si I2.3 pasa de 0 a 1
10	A	%M2.4	// Señal de borrado
11	R	%C1	// Pone a 0 el contador si I2.4 vale 1
12	A	%C1	// Consulta el estado lógico del contador
13	=	%M4.0	// Activa M4.0 si C1 es mayor que 0

Lenguaje Structured Text (ST)

- ST es un lenguaje de alto nivel similar a Pascal o ADA
- Siemens lo denomina SCL (Structured Control Language)
- Consiste en un conjunto de sentencias de control separadas por “;”
- Estructuras de control:

- := Asignación: Asigna el valor de una expresión a una variable:

```
"T1" := ("T4" AND "T67") OR ("J3" AND NOT "K3");
```

- IF Selección entre alternativas por medio de expresiones booleanas:

```
1 IF "a" < "b" THEN
2     "f" := 5;
3 ELSIF "a" = "b" THEN
4     "f" := 2;
5 ELSE
6     "f" := 13;
7 END_IF;
```

Lenguaje Structured Text (ST)

- Estructuras de control:
 - **CASE** Selección entre alternativas por expresión:

```
1 CASE "b" OF
2     1:
3         "a" := 12;
4     2, 3:
5         "a" := 15;
6     5..10:
7         "a" := 21;
8     ELSE:
9         "a" := 50;
10 END_CASE;
```

**EVITAR
BUCLES
INFINITOS!!**

- **FOR** Bucle con inicialización, condición y progresión

```
VAR
    V: Array [1..5] OF Int := [1, 4, 5, 67, 22];
    l: Int;
    nV: Int := 5;
    Max: Int := 0;
END_VAR;
```

```
FOR l:=1 TO nV BY 1 DO
    IF V[l] > Max THEN
        Max := V[l];
    END_IF;
END_FOR;
```

Lenguaje Structured Text (ST)

- **WHILE** Bucle con condición para continuar:

```
I := 1;  
WHILE I <= Nv DO  
    IF V[I] > Max  
        Max := V[I];  
    END_IF;  
    I := I+1;  
END_WHILE;
```

**EVITAR
BUCLES
INFINITOS!!**

- **REPEAT** Bucle con condición para finalizar:

```
I := 1;  
REPEAT  
    IF V[I] > Max  
        Max := V[I];  
    END_IF;  
    I := I+1;  
UNTIL I > Nv END_REPEAT;
```

*RETURN: Abandona el FB o FC actual
EXIT: Finaliza BUCLE*

Lenguaje Structured Text (ST)

- Operadores:
 - Aritméticos: +, -, *, **, /, MOD ($a^{**}b = a^b$)
 - Comparación: >, <, <=, >=, =, <>
 - Lógicos: NOT, AND ó &, OR, XOR
 - Paréntesis () para modificar la prioridad de los operadores
- Funciones disponibles:
 - y:= ABS (x) Valor absoluto
 - y:= SQRT (x) Raíz cuadrada
 - y:= LN (x) Logaritmo natural
 - y:= SIN (x) Seno
 -

Lenguaje Structured Text (ST)

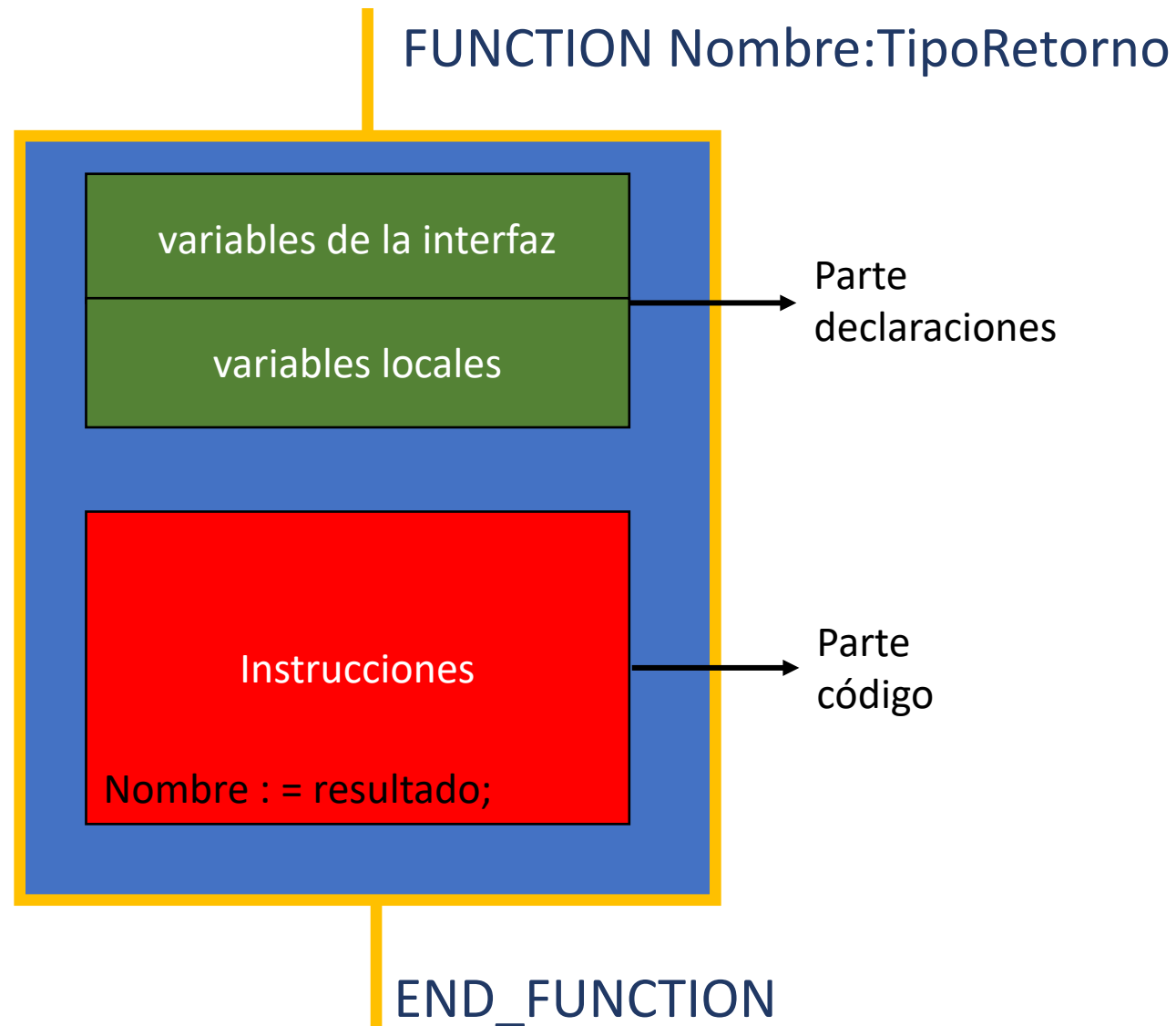
- En un programa vamos a tener los siguientes elementos:
 - Secuencia de instrucciones
 - Llamadas a funciones
 - Llamadas a Bloques Funcionales (FB)
- En lenguaje ST se deben cumplir:
 - Un programa puede llamar a funciones y FB
 - Un FB puede llamar a funciones y a otro FB
 - Una función sólo puede llamar a funciones
 - Un FB no puede llamarse de forma recursiva

Lenguaje Structured Text (ST)

- Función:
 - Secuencia de instrucciones y/o llamadas a otras funciones
 - Operan sobre datos de entrada y producen una salida
 - No tiene memoria o estado interno
 - Se pueden utilizar en cualquier tipo de lenguaje

```
1
2 FUNCTION indicarFueraDeControl : BOOL
3
4 VAR_TEMP                                // Variables temporales
5     alarma: BOOL;
6 END_VAR
7
8                                // Parámetros del bloque
9 VAR_INPUT                            // Parámetros de entrada
10     valorProceso: REAL;
11     valorMaximo, valorMinimo: REAL;
12 END_VAR
13
14                                // Área de instrucciones
15
16 IF valorProceso >= valorMaximo OR valorProceso <= valorMinimo THEN
17     alarma := true;
18 ELSE
19     alarma := false;
20 END_IF;
21
22 indicarFueraDeControl := alarma;
23
24 END FUNCTION
```

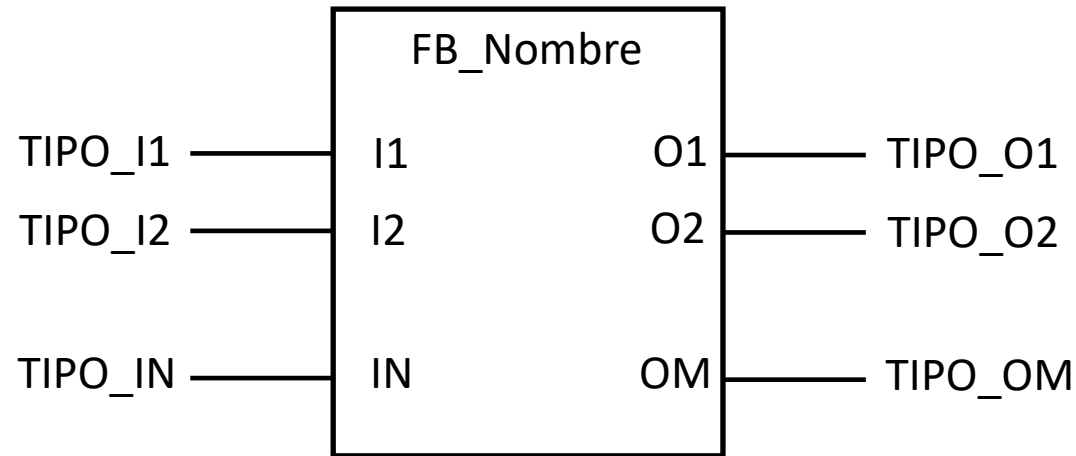
Lenguaje Structured Text (ST)



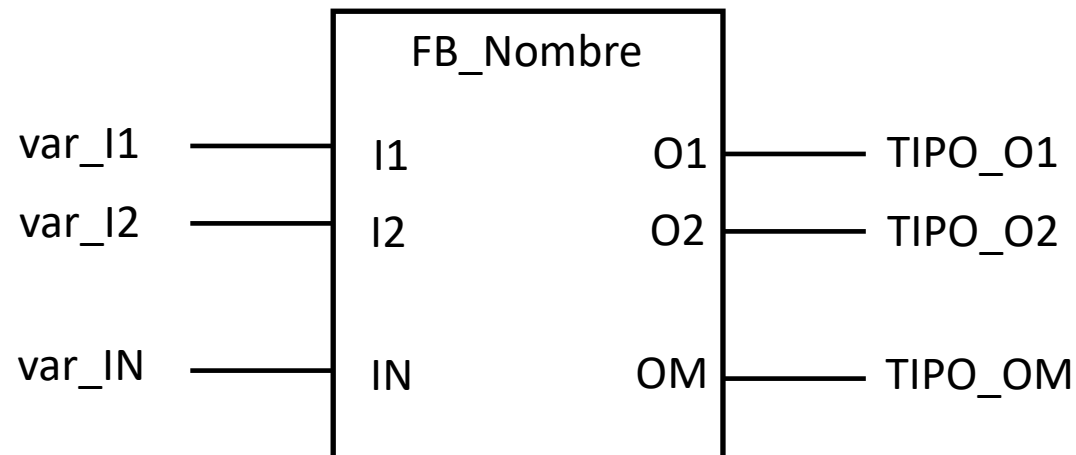
Lenguaje Structured Text (ST)

- FB:

- Es necesario crear una instancia del FB
- Pueden tener estado interno
- Los FB se pueden programar en cualquier tipo de lenguaje



Elemento_FB



VAR

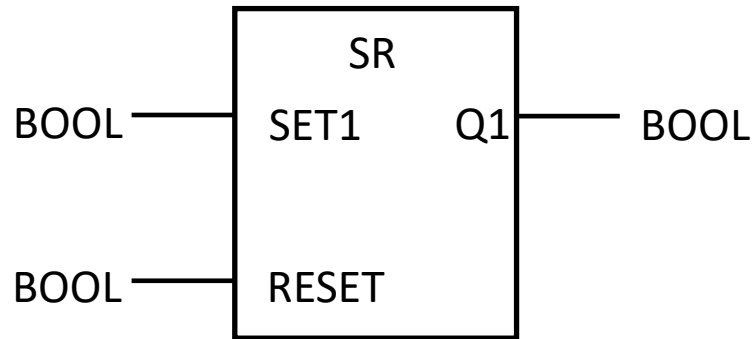
```
Elemento_FB: FB_Nombre;  
var_I1: Tipo_I1;  
...  
var_IN: Tipo_IN;  
var_O1: Tipo_O1;  
...  
var_OM: Tipo_OM;  
END_VAR
```

```
Elemento_FB(I1:=var_I1, ...  
...IN:=var_IN);
```

```
var_O1:= Elemento_FB.O1;  
...  
var_OM:=Elemento_FB.OM;  
...
```

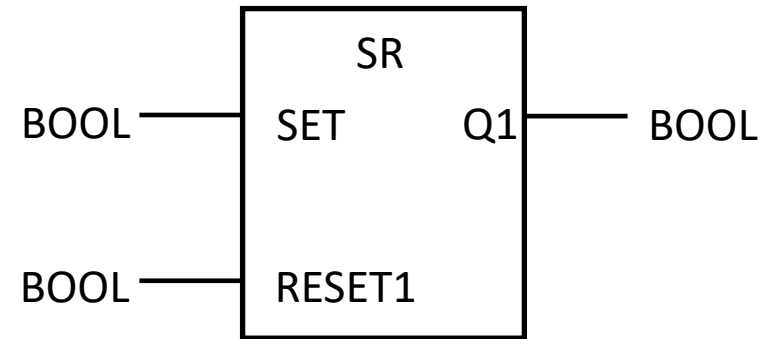
Lenguaje Structured Text (ST)

SR: Set (prioritario) - Reset



```
FUNCTION_BLOCK SR
VAR_INPUT
    SET1: BOOL;
    RESET: BOOL;
END_VAR
VAR_OUTPUT
    Q1: BOOL;
END_VAR
Q1:= SET1 OR (NOT RESET AND Q1);
END_FUNCTION_BLOCK
```

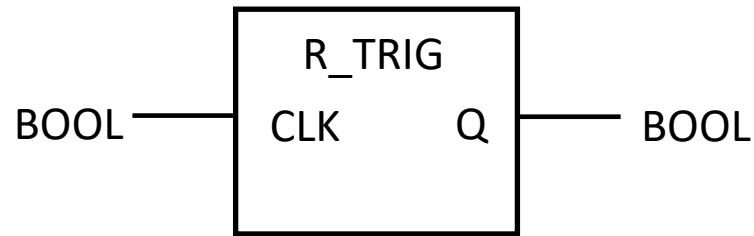
RS: Reset (prioritario) - Set



```
FUNCTION_BLOCK RS
VAR_INPUT
    SET: BOOL;
    RESET1: BOOL;
END_VAR
VAR_OUTPUT
    Q1: BOOL;
END_VAR
Q1:= NOT RESET1 AND (SET OR Q1);
END_FUNCTION_BLOCK
```

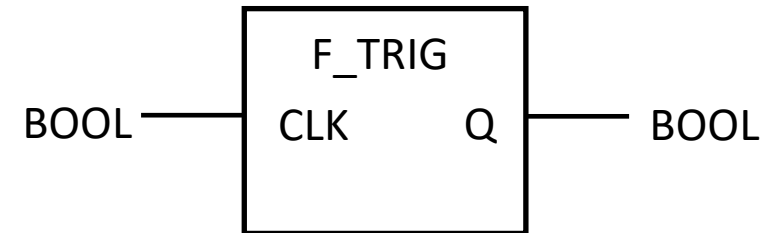
Lenguaje Structured Text (ST)

R_TRIG: flanco de subida



```
FUNCTION_BLOCK R_TRIG
VAR_INPUT
    CLK: BOOL;
END_VAR
VAR_OUTPUT
    Q: BOOL;
END_VAR
VAR
    MEM: BOOL := 0;
END_VAR
Q:= CLK AND (NOT MEM);
MEM := CLK;
END_FUNCTION_BLOCK
```

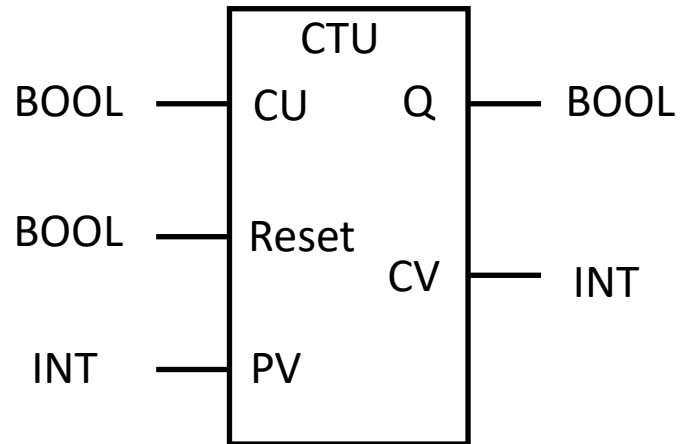
R_TRIG: flanco de bajada



```
FUNCTION_BLOCK R_TRIG
VAR_INPUT
    CLK: BOOL;
END_VAR
VAR_OUTPUT
    Q: BOOL;
END_VAR
VAR
    MEM: BOOL := 0;
END_VAR
Q:= NOT CLK AND MEM;
MEM := CLK;
END_FUNCTION_BLOCK
```

Lenguaje Structured Text (ST)

CONTADOR ASCENDENTE



```
FUNCTION_BLOCK CTU
VAR_INPUT
```

```
CU: BOOL R_EDGE;
Reset: BOOL;
PV: INT;
```

```
END_VAR
```

```
VAR_OUTPUT
```

```
Q: BOOL;
CV: INT;
```

```
END_VAR
```

```
IF Reset THEN
```

```
    CV := 0;
```

```
ELSEIF CU AND (CV < PV)
```

```
    CV := CV + 1;
```

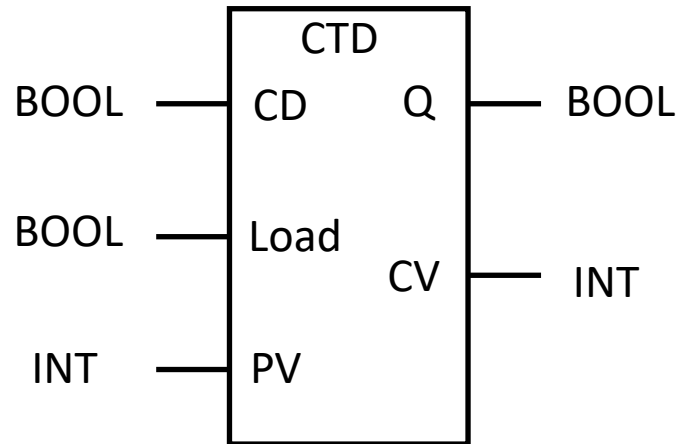
```
Q := (CV >= PV);
```

```
END_FUNCTION_BLOCK
```

RISE EDGE

Lenguaje Structured Text (ST)

CONTADOR DESCENDENTE



```
FUNCTION_BLOCK CTD
VAR_INPUT
```

```
CD: BOOL F_EDGE;
Load: BOOL;
PV: INT;
```

```
END_VAR
```

```
VAR_OUTPUT
```

```
Q: BOOL;
CV: INT;
```

```
END_VAR
```

```
IF Load THEN
```

```
    CV := PV;
```

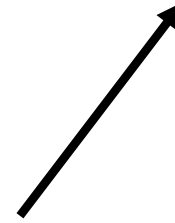
```
ELSEIF CD AND (CV > 0)
```

```
    CV := CV - 1;
```

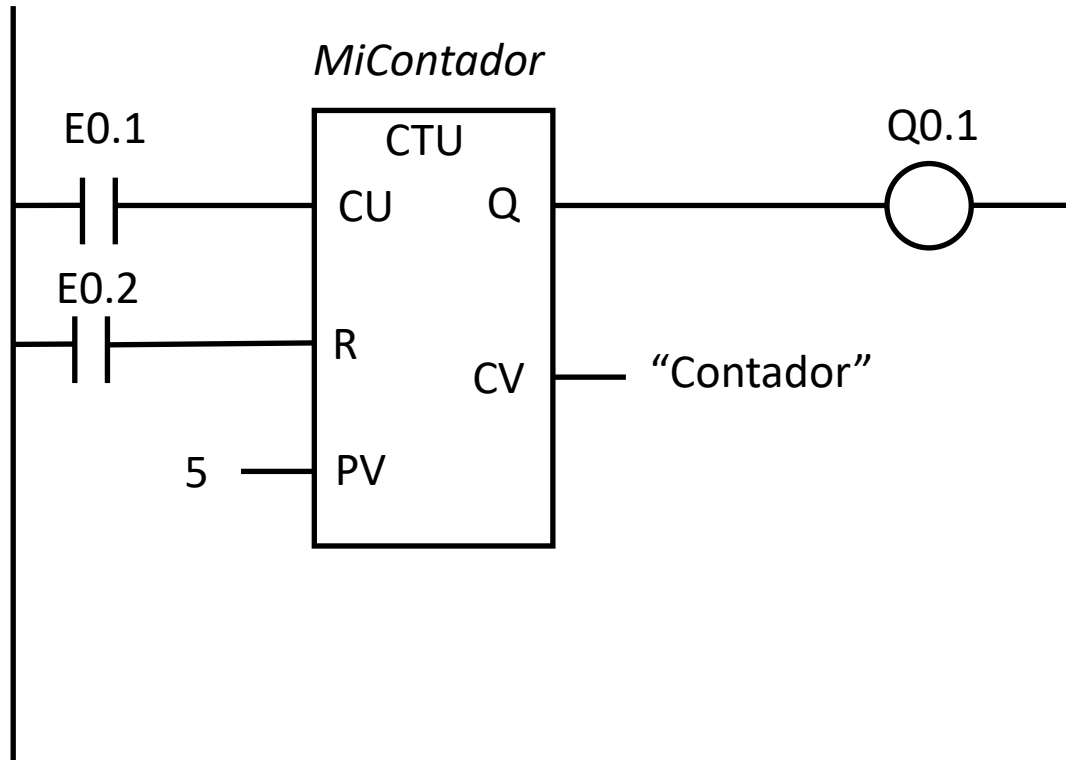
```
Q := (CV <= 0);
```

```
END_FUNCTION_BLOCK
```

FALLING EDGE



Lenguaje Structured Text (ST)

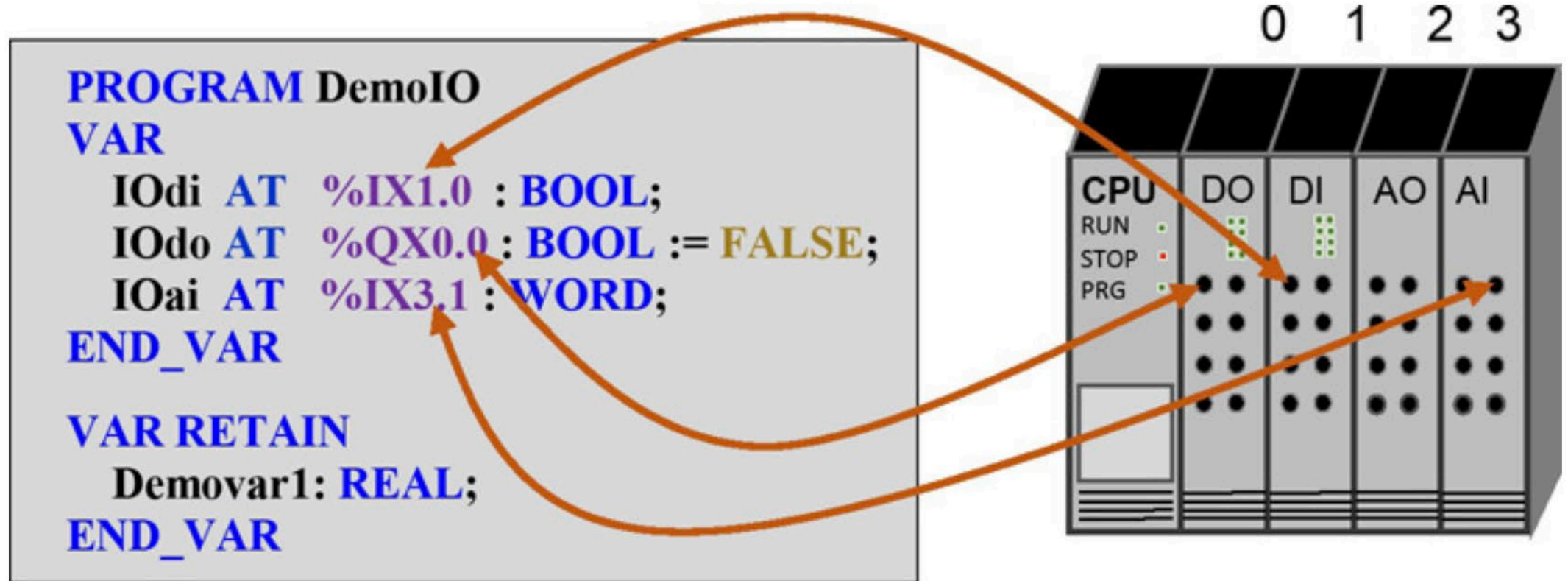


Uso en ST:

```
MiContador(CU:=Entrada1;R:=Entrada2,PV:=5);
```

```
Q0.1 := MiContador.Q;
```


Lenguaje Structured Text (ST)



BIBLIOGRAFÍA

- **Sistemas de automatización y autómatas programables.** Enrique Mandado, Jorge Acevedo, Celso Fernández, Ignacio Armesto, José Luis Rivas, José María Núñez. Ed. Marcombo.
- Automatización: Problemas resueltos con autómatas programables. J. Pedro Romera, J. Antonio Lorite y Sebastián Montoro. Ed. Thomson
- [SIMATIC S7 Controlador programable S7-1200. Manual del Sistema. Siemens.](#)
- Controles PLC con Texto Estructurado (ST). Tom Mejer Antonsen.