



SIMATIC S-1200 NIVEL BÁSICO

EUSEBIO GÓMEZ GARCÍA

1

¿Qué es la automática?

La automática pretende facilitar y mejorar el desarrollo de diferentes actividades a las personas, colaborando con ellas o sustituyéndolas en la toma de decisiones y en su puesta en práctica.

De manera formal “la automática es la disciplina que trata de sustituir al operador humano en sus tareas físicas o mentales por dispositivos artificiales”

¿Qué es la automatización industrial?

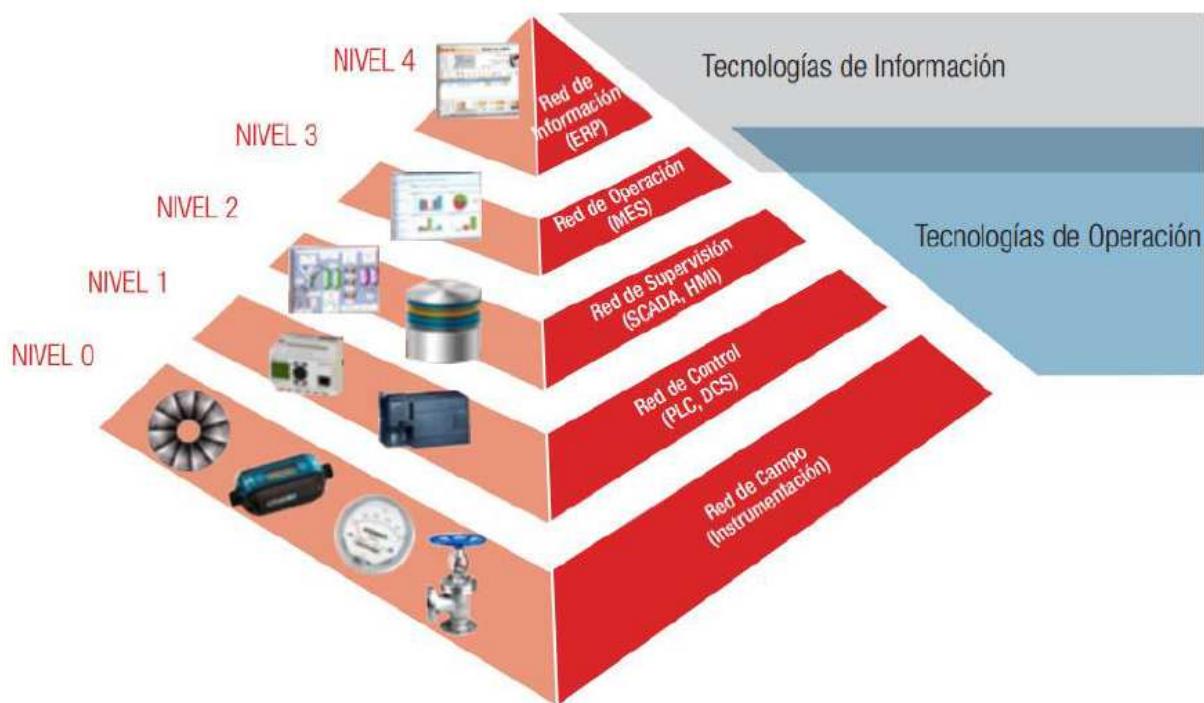
La automatización industrial consiste en el uso de tecnologías para la regulación, el control y la monitorización de dispositivos, máquinas y procesos industriales, reduciendo al máximo la intervención humana.

Se trata de tener el control en tareas asociadas a procesos cuya complejidad hace imposible que sean realizados por personas sin ayuda de máquinas (microcirugías, ensayos en biomedicina, diseño de chips y muchos otros) o tareas que se desarrollan en ambientes peligrosos o resultan fatigosas y molestas. Se necesita que sean las máquinas y robots quienes ayuden a las personas o directamente los realicen ellos.

La automatización también busca mejorar los tiempos de producción permitiendo producir más, con menos errores y garantizando la calidad del producto final.

2

Pirámide de Automatización Industrial



3

Dispositivos de campo Captadores



ultrasonido



codificador angular



final de carrera



fotoeléctrico



inductivo



fibra óptica

4

Preactuadores



relé



electroválvula
neumática



contactor



electroválvula
hidráulica



variador de frecuencia

 sistema para el control de la velocidad rotacional de un motor de corriente alterna

Actuadores



motor AC



pistón hidráulico



pistón neumático



pinzas neumáticas



pistón neumático

Controladores



Typ FEC (Festo)



CX1010 (Beckhoff)



CPM1A (Omron)



Logo (Siemens)



FP2 (Panasonic)

7

Familia Simatic Siemens



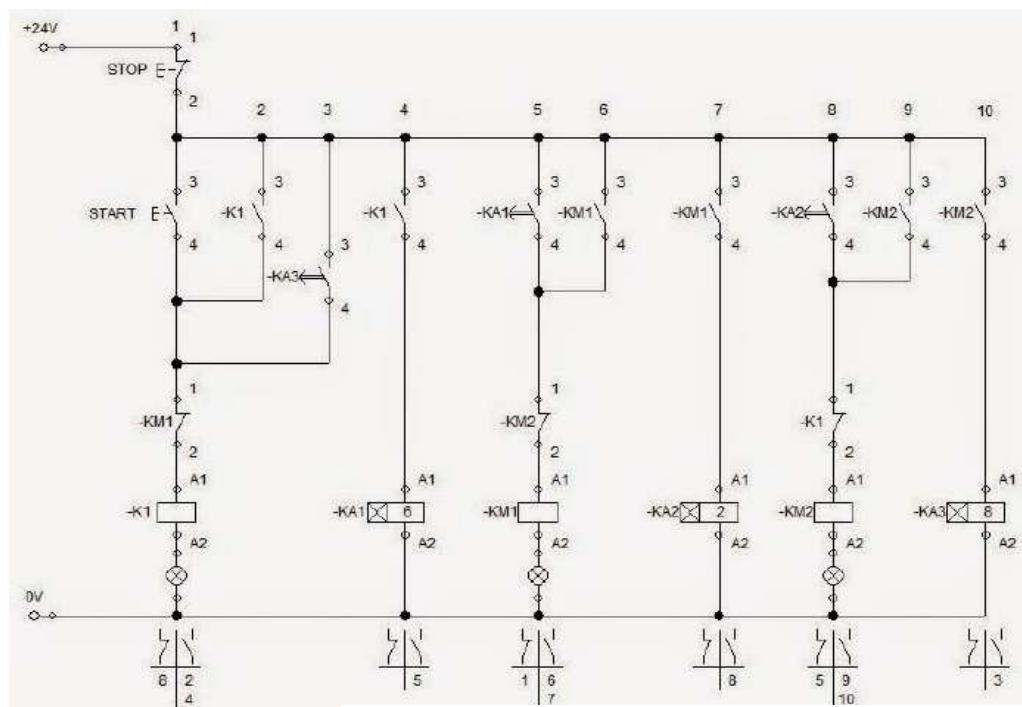
8

HMI (Human Machine Interface)



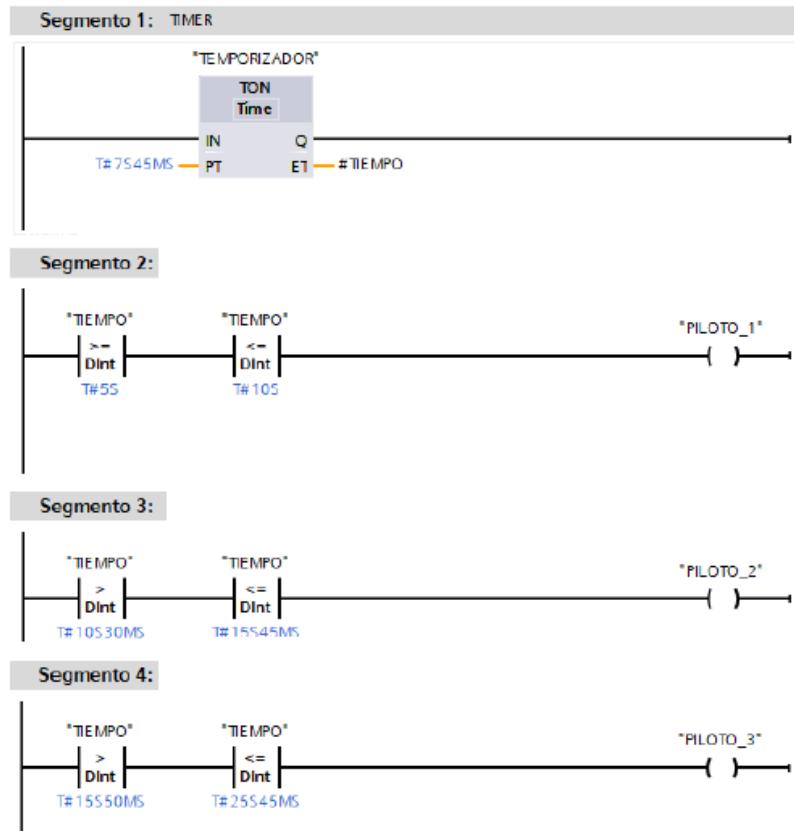
9

Implementación de circuitos de control Lógica cableada



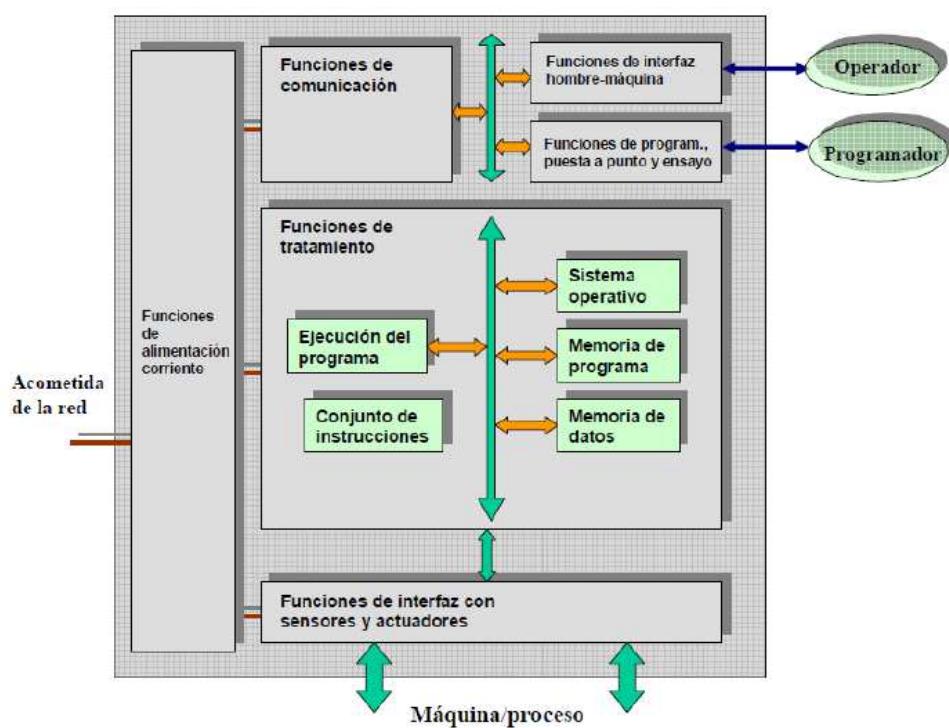
10

Lógica programada



11

Estructura de un controlador lógico programable (PLC)



12

PLC, Programmable logic controller.

Un autómata programable (Programmable Logic Controller, PLC) es una máquina electrónica, diseñada para ser utilizada en un entorno industrial, generalmente hostil. Utiliza una memoria programable para el almacenamiento interno de instrucciones, orientadas al usuario, para implantar soluciones específicas tales como funciones lógicas, secuencias, temporizaciones, recuentos y funciones aritméticas, entre otras, con el fin de controlar mediante entradas y salidas, digitales y analógicas, diversos tipos de máquinas o procesos.



13

Memorias

Toda la información del PLC está contenida en memorias de semiconductor (dispositivo capaz de almacenar datos binarios).

Memorias RAM de lectura/Escritura, (Random-Access Memory). Pueden ser leídas y modificadas cuantas veces sea necesario a través de buses internos y de forma rápida.

Sus inconvenientes son su *baja densidad de integración* y sobre todo su *carácter volátil*.

Se utilizan generalmente como memorias de datos y únicamente como memoria de programa en caso de que puedan ser respaldadas por batería o condensador de alta potencia.

Memorias ROM, (read-only memory) sólo lectura, no reprogramables.

No pueden ser modificadas de ninguna forma, se utilizan dentro del PLC para almacenar el PROGRAMA MONITOR (rutas de INIT, interfaces, error, etc.), además este programa puede contener el INTÉRPRETE del programa de usuario.

El contenido de esta memoria no es accesible desde el exterior.

Memorias EPROM, reprogramables.

Se pueden programar con un circuito especial después de borrarlas con rayos ultravioleta, se suelen utilizar para guardar programas de usuario una vez depurado convenientemente.

14

Memorias

Memorias EEPROM. Se pueden reprogramar por medios eléctricos, pero sus tiempos de acceso para lectura y para escritura son largos comparados con los de las memorias RAM/EPROM.

El número de operaciones de borrado/escritura está limitado a unos cientos de miles, por lo que situadas dentro del área de trabajo se destruirían rápidamente bajo la acción del micro. En la actualidad se usa de forma más frecuente la combinación RAM+EEPROM, para los casos de interrupción del suministro eléctrico, solución que está sustituyendo a la clásica RAM+batería.

Memoria flash. Tiene una tecnología de almacenamiento, derivada de la memoria EEPROM, que permite la lectura-escritura de múltiples posiciones de memoria en la misma operación. Gracias a ello, la tecnología *flash*, permite velocidades de funcionamiento muy superiores frente a la tecnología EEPROM tradicional que sólo permite actuar sobre una única celda de memoria en cada operación de programación. Se trata de la tecnología empleada en los dispositivos pendrive.



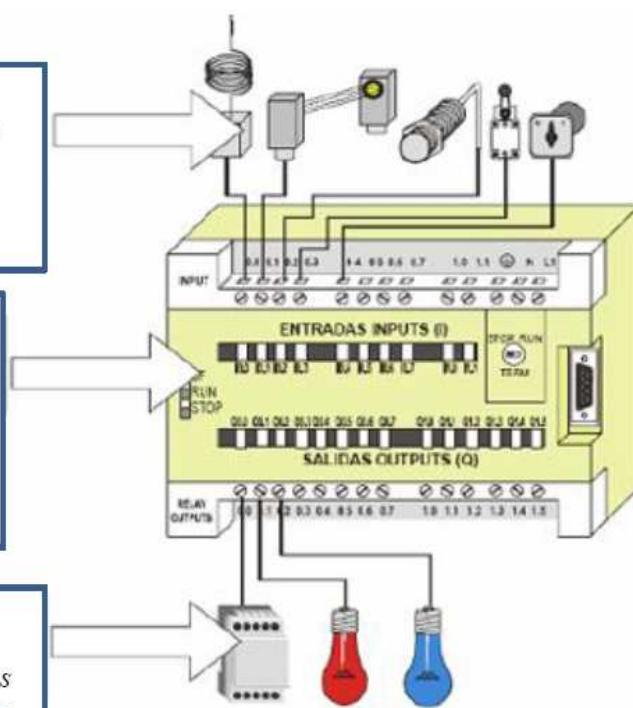
15

Ciclo de funcionamiento

Consulta el estado de las entradas y guarda los datos en la memoria “Imagen de entradas”

Lee y ejecuta las instrucciones del programa con los datos de las entradas. El resultado lo guarda en la memoria “Imagen de salidas”

Ejecuta las órdenes de activación/desactivación de salidas con los datos guardados en la memoria “Imagen de salidas”



16



PRINCIPIOS BÁSICOS DE PROGRAMACIÓN

Visión Práctica S1200

17

Introducción a la programación

- Podríamos dividir la programación del autómata en **varios pasos** :
 - **Definir el sistema de control** (que debe hacer, en que orden, etc.): diagrama de flujo, la descripción literal o un grafo GRAFCET.
 - **Identificar las señales de entrada y salida del autómata.**
 - Representar el sistema de control mediante un modelo, indicando todas las funciones que intervienen, las relaciones entre ellas, y la secuencia que deben seguir. Algebraica (instrucciones literales) o gráfica (símbolos gráficos).
 - **Asignar las direcciones de entrada/salida o internas del autómata** a las correspondientes del modelo.
 - Codificar la representación del modelo. **Lenguaje de programación,**
 - **Cargar el programa** en la memoria del autómata desde la unidad de programación.
 - **Depurar el programa** y obtener una copia de seguridad.

18

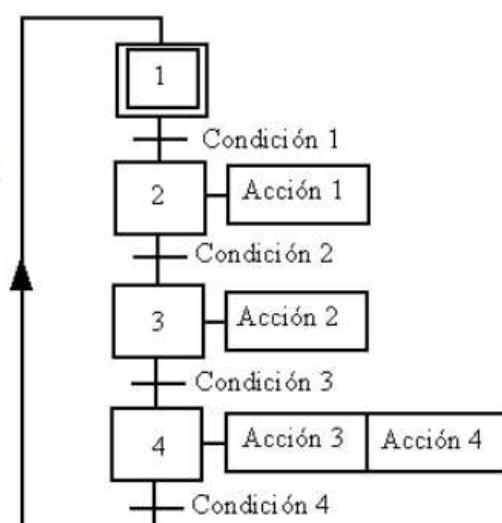
Definición del sistema de control

- Diagramas de flujo
 - Organigrama y flujograma, es un sistema de representación que se basa en una serie de símbolos que según un convenio establecido tienen un determinado significado.
- INICIO FIN
- ACCIONES
- TOMA DE DECISIONES
 - NO
 - SI
- 2 Niveles:
 - Nivel 1:
 - secuencia de acciones a realizar
 - representar el funcionamiento general del sistema
 - Nivel 2:
 - especificadas las acciones en forma de instrucción: instrucciones entendibles directamente por el autómata o en forma de funciones lógicas.

19

Definición del sistema de control

- Grafcet(Graphe de Comande Etape Transition,cuya traducción literal es Gráfico de Orden Etapa Transición)
 - “es una secuencia de etapas que tienen asociadas unas determinadas acciones a realizar sobre el proceso junto con las condiciones o transiciones que provocan que se produzca el paso de una etapa a otra”
 - Normalizado: International Electrotechnical Commission IEC 848
 - Una de las mejores herramientas para representar automatismos secuenciales



20

Definir las variables que intervienen

- Definir las variables que intervienen y asignarles direcciones de memoria

- Ejemplo:

- "Control de una puerta corredera accionada por medio de un motor. El contactor S1 produce la apertura de la puerta, el contactor S2 controla el cierre de la puerta. El interruptor E3 de final de carrera se activa cuando la puerta esta abierta, y el interruptor E2 de fin de carrera se activa cuando la puerta esta cerrada. La puerta se abre al aplicar una determinada presión sobre un sensor de paso de vehículos E1 situado enfrente de la puerta. Si el sensor E1 no se activa, la puerta se cierra—después de transcurridos 10 segundos. Si se activa E1, se cierra el contactor S2 y se mantiene cerrado hasta que el interruptor E3 de final de carrera desactive el contacator S2."

Cuando se esta abriendo la puerta, o bien cuando una vez abierta haya detectado un vehículo con el sensor E1, el temporizador T1 no se activa. Si no se dan estas circunstancias y la puerta esta abierta E3 activado, se activa el temporizador T1, y transcurridos 10 segundos, la puerta se cierra mediante el contactor S1 por el temporizador T1. La acción de cerrar se produce hasta que o bien se detecta fin de carrera E2 o bien se detecta otro vehículo mediante la activación de E1 en cuyo caso se abre la puerta activando para ello el contactor S2. Las lámparas LED1 y LED2 indican cuando se está cerrando o abriendo la puerta respectivamente."

21

Definir las variables que intervienen

- Tabla de asignación de variables

Tipo	Termino	Símbolo	Descripción
Entradas	E1	VEHICULO	Sensor presencia vehículo
	E2	CERRADA	Límite puerta cerrada
	E3	ABIERTA	Límite puerta abierta
Salidas	S1	CERRAR	Contactor cerrar
	S2	ABRIR	Contactor abrir
	LED1	LEDCER	Lámpara puerta cerrándose
	LED2	LEDABI	Lámpara puerta abriendose
Temporizador	T1	TIEMPO	Temporizador 10 seg

- Posteriormente se asignan las direcciones físicas a estas variables

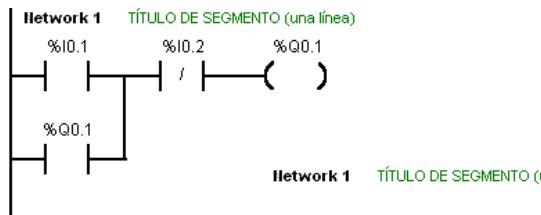
Tipo	Termino	Símbolo	Dirección
Entradas	E1	VEHICULO	I0.0
	E2	CERRADA	I0.1
	E3	ABIERTA	I0.2
Salidas	S1	CERRAR	Q0.0
	S2	ABRIR	Q0.1
	LED1	LEDCER	Q0.2
	LED2	LEDABI	Q0.3
Temporizador	T1	TIEMPO	T5

22

Lenguajes de programación

• Lenguajes de programación- para el S7-1200

- KOP (esquema de contactos) es un lenguaje de programación gráfico. Su representación se basa en esquemas de circuitos.
- FUP (diagrama de funciones) es un lenguaje de programación que se basa en los símbolos lógicos gráficos empleados en el álgebra booleana.
- SCL (structured control language) es un lenguaje de programación de alto nivel basado en texto.



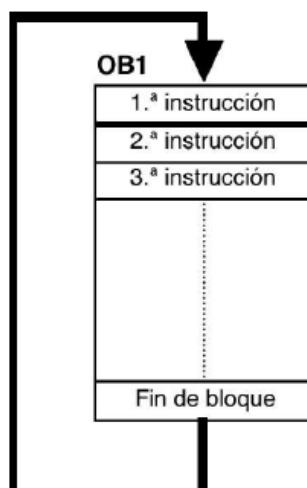
```
9 //Ejemplos SCL (C) REEA 2013
10 //Activar una salida (Set)
11 IF "Entrada1"=true THEN
12   "Salida1":=true;
13 END_IF;
14 //Desactivar una salida (Reset)
15 IF "Entrada2"=true THEN
16   "Salida1":=false;
17 END_IF;
```

23

Organización de programas

• Programación lineal

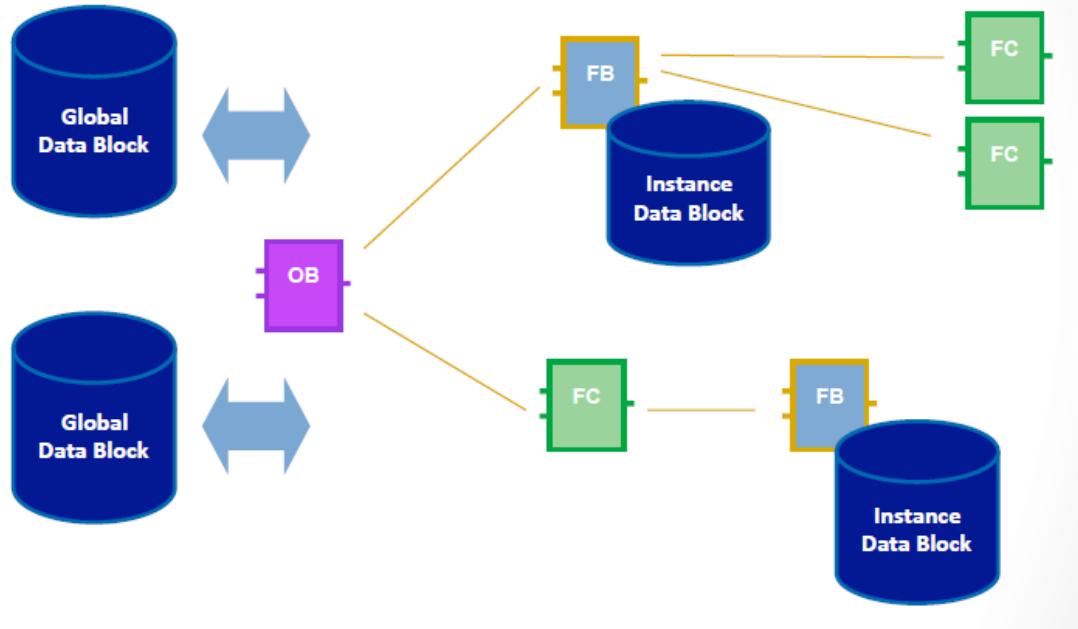
- Un programa lineal ejecuta todas las instrucciones de la tarea de automatización de forma secuencial, es decir, una tras otra. Generalmente, el programa lineal deposita todas las instrucciones del programa en un OB de ciclo (como OB 1), encargado de la ejecución cíclica del programa.



24

Organización de programas

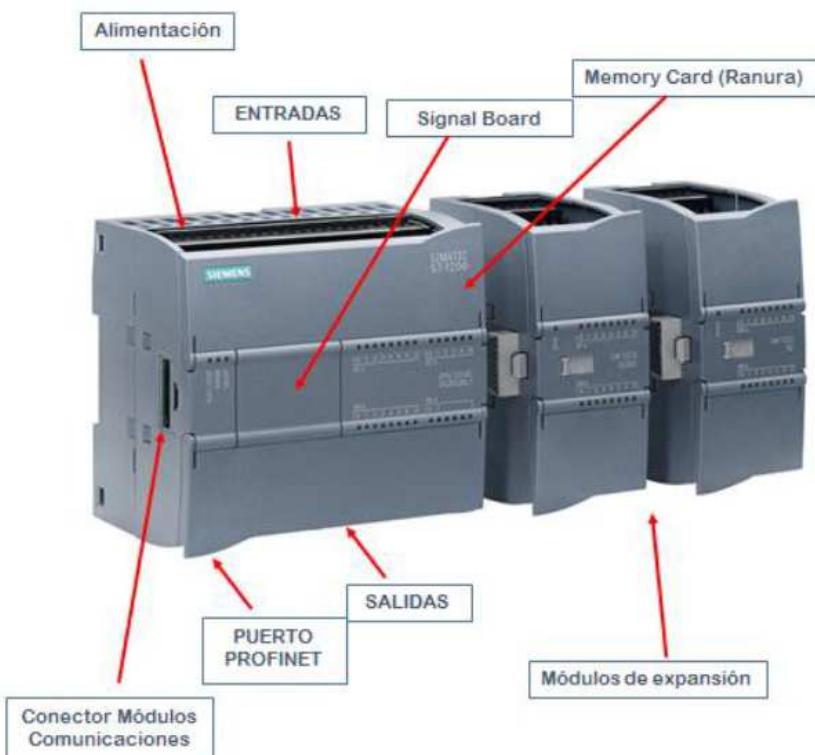
- Programación estructurada



La profundidad máxima de anidamiento es de 16

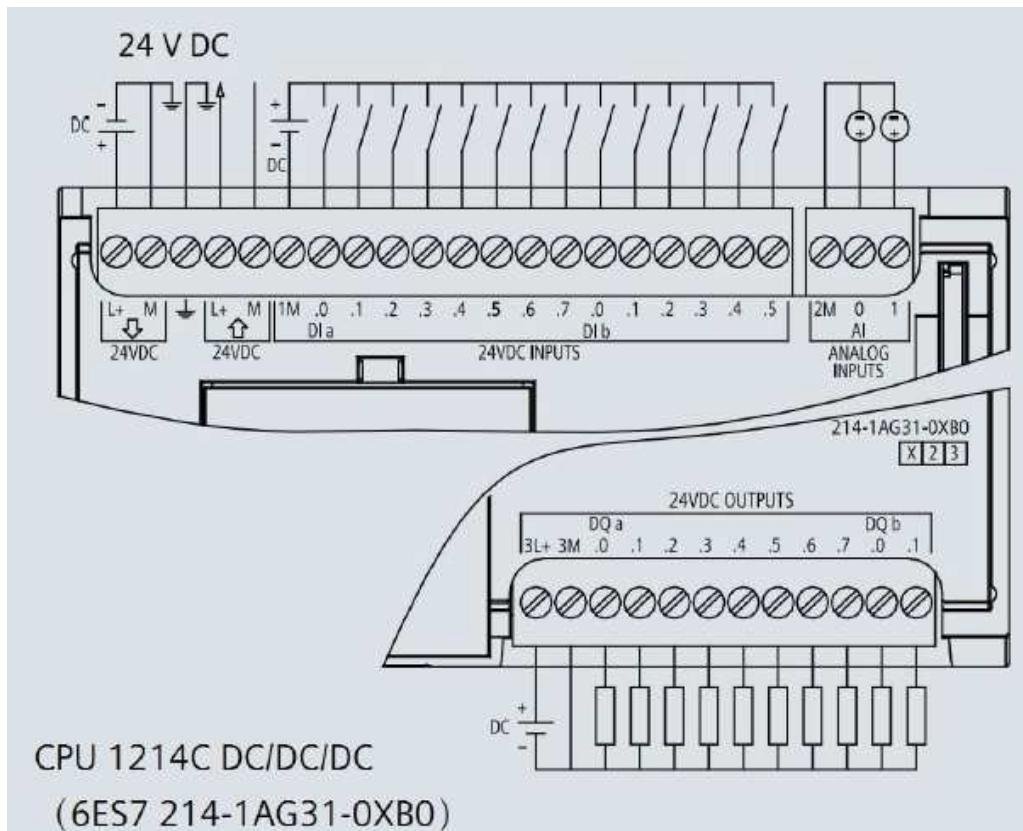
25

CPU serie 1200. SIEMENS



26

CPU 1214C. Esquema de conexiones



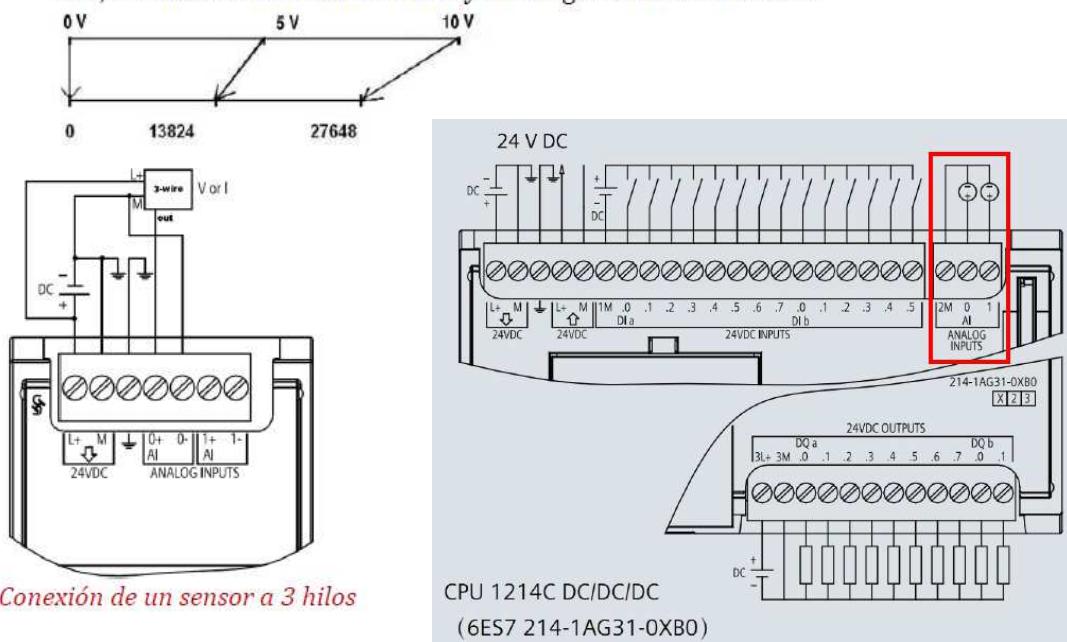
27

Entradas/Salidas analógicas

A diferencia de una señal digital (binaria) para el PLC, que solo puede adoptar los estados de señal +24V o 0V, las señales analógicas pueden adoptar cualquier valor dentro de un rango determinado.

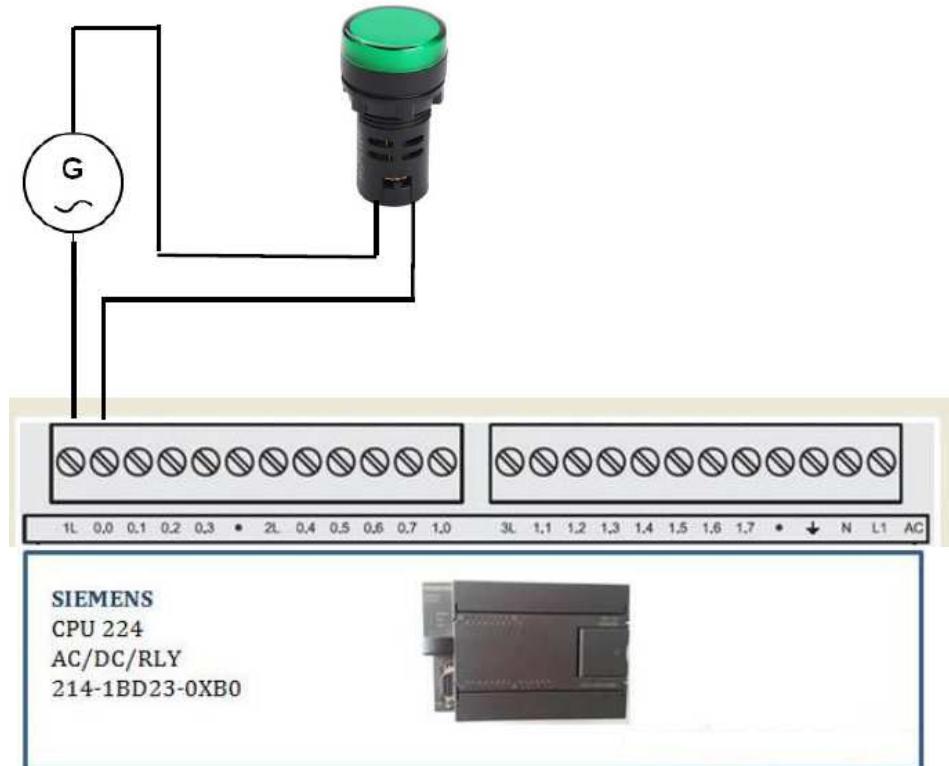
Así las magnitudes, temperatura, caudal o velocidad, entre otras se transforman con un transductor de medida en tensiones y/o intensidades eléctricas.

El PLC S7-1200 posee dos entradas analógicas integradas de voltaje con un rango de 0 a 10V, con una resolución de 10 bits y un rango total de 0 a 27648.



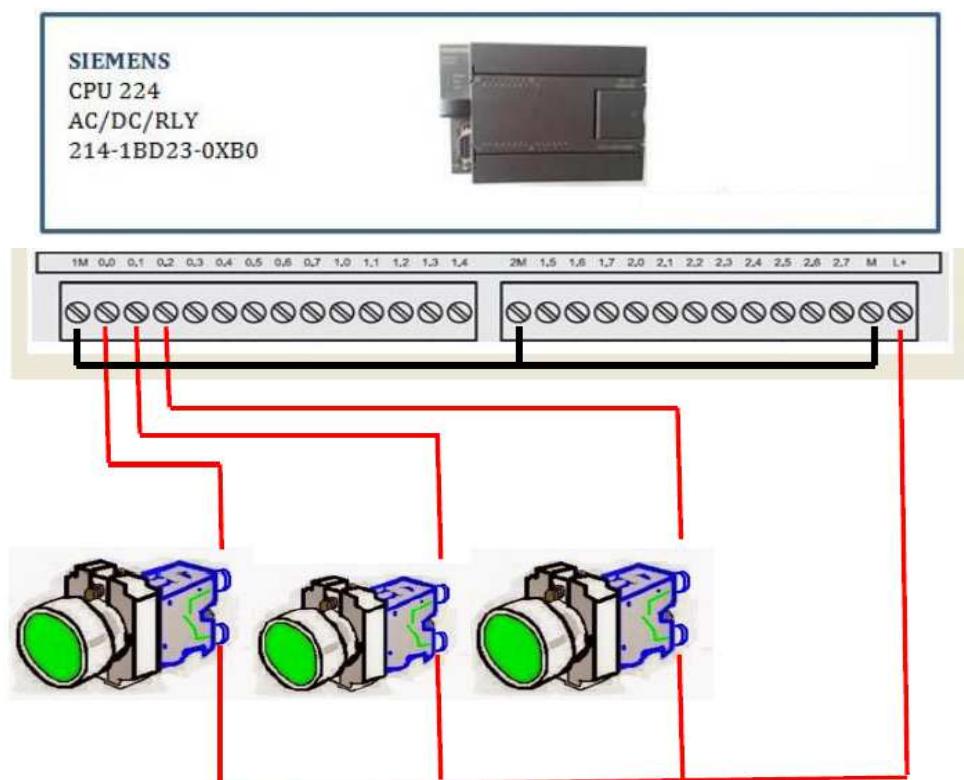
28

Ejemplo de conexionado de entradas salidas digitales



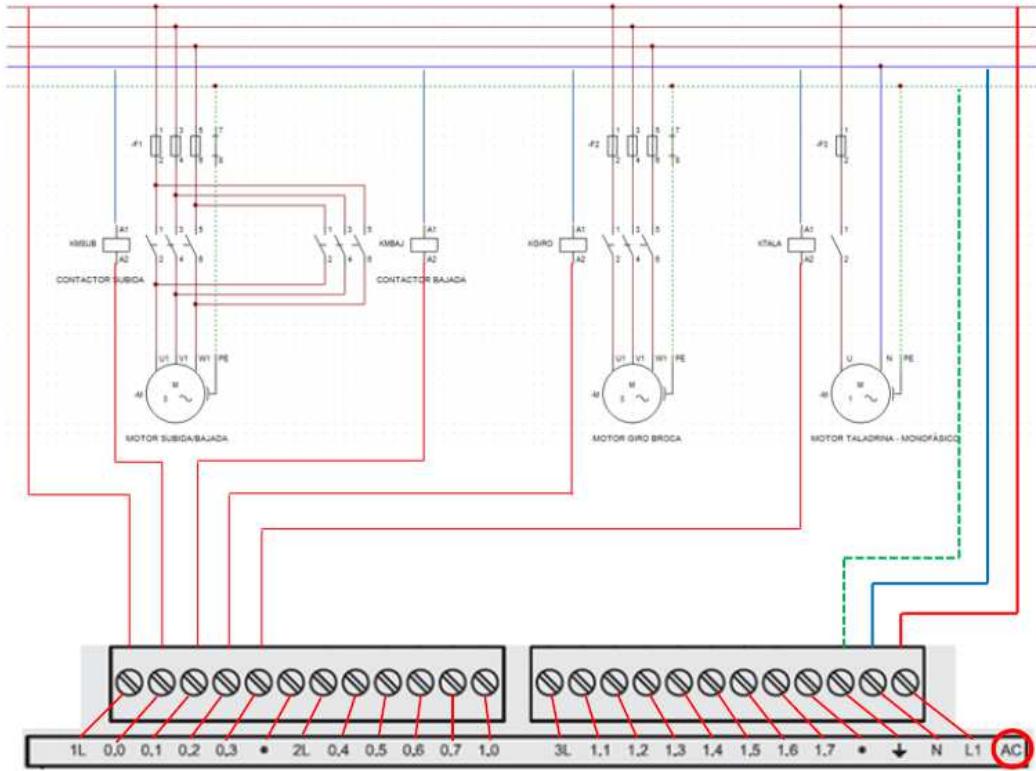
29

Ejemplo de conexionado de entradas salidas digitales



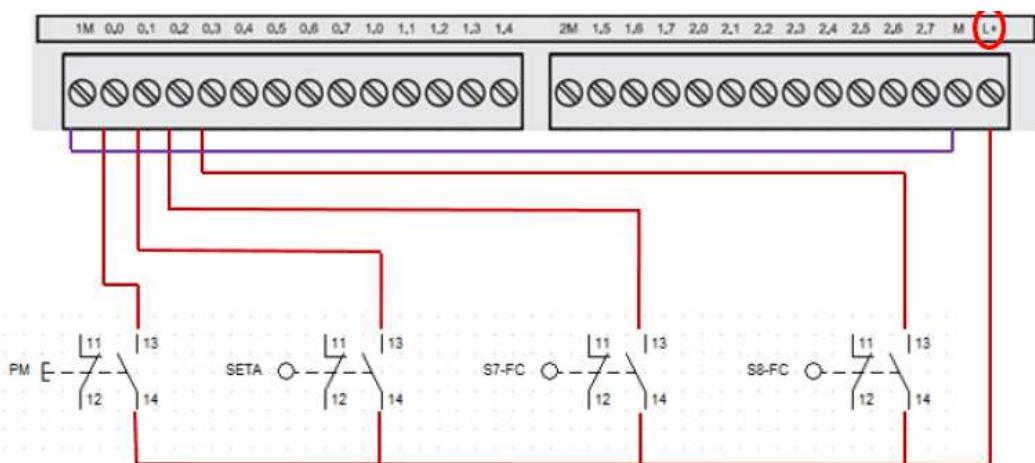
30

Ejemplo de conexionado de entradas salidas digitales



31

Ejemplo de conexionado de entradas salidas digitales



32

Configuración del mapa de la “Memoria Imagen de ENTRADAS/SALIDAS”

ENTRADA/INPUT/EINGABEN: I/E

BYTES	BITS
Byte 0	7 6 5 4 3 2 1 0
Byte 1	7 6 5 4 3 2 1 0
Byte 2	7 6 5 4 3 2 1 0
Byte 3	7 6 5 4 3 2 1 0
Byte 4	7 6 5 4 3 2 1 0
Byte 5	7 6 5 4 3 2 1 0
Byte 6	7 6 5 4 3 2 1 0
Byte 7	7 6 5 4 3 2 1 0

I 4.2
E 4.2

I 7.1
E 7.1

En la CPU-1200, disponemos de un mapa de memoria para IMAGEN DE ENTRADAS de 8 bytes numerados del 0 al 7.

Cada byte dispone de 8 celdas (bits). Cada una de las ENTRADAS está asociada a uno de estos bits, por lo que se pueden direccionar (conectar) 64 ENTRADAS.

Se denomina con el nombre ENTRADA: (INPUT en inglés, EINGABEN en alemán).

Ejemplo:

I 4.2 o E 4.2

Configuración del mapa de la “Memoria Imagen de ENTRADAS/SALIDAS”

SALIDA/OUTPUT/AUSGABEN: Q/A

BYTES	BITS
Byte 0	7 6 5 4 3 2 1 0
Byte 1	7 6 5 4 3 2 1 0
Byte 2	7 6 5 4 3 2 1 0
Byte 3	7 6 5 4 3 2 1 0
Byte 4	7 6 5 4 3 2 1 0
Byte 5	7 6 5 4 3 2 1 0
Byte 6	7 6 5 4 3 2 1 0
Byte 7	7 6 5 4 3 2 1 0

Q
4.5

Q
6.3

En las CPUs-1200, disponemos de un mapa de memoria para IMAGEN DE SALIDAS de 8 bytes numerados del 0 al 7.

Cada byte dispone de 8 celdas (bits). Cada una de las SALIDAS está asociada a uno de estos bits, por lo que se pueden direccionar (conectar) 64 SALIDAS.

Se denomina con el nombre SALIDA: (QUIT en inglés, AUSGABEN en alemán).

Ejemplo:

Q 6.3 o A 6.3

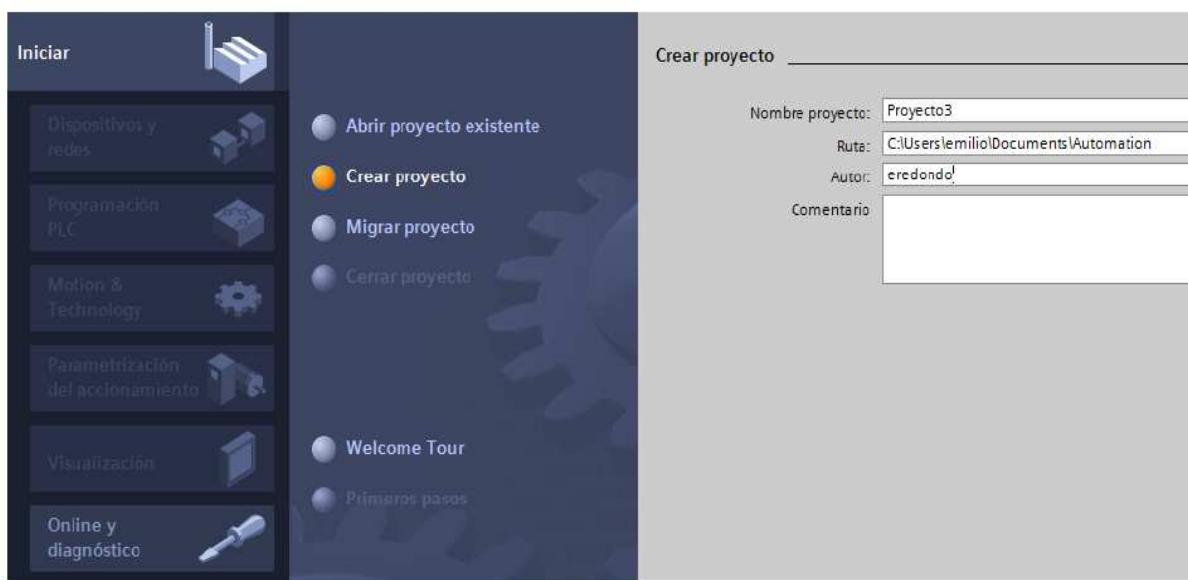


ENTORNO DE PROGRAMACIÓN TIA PORTAL

Visión Práctica S1200

35

TIA_PORTAL (TOTALLY INTEGRATED AUTOMATION PORTAL)



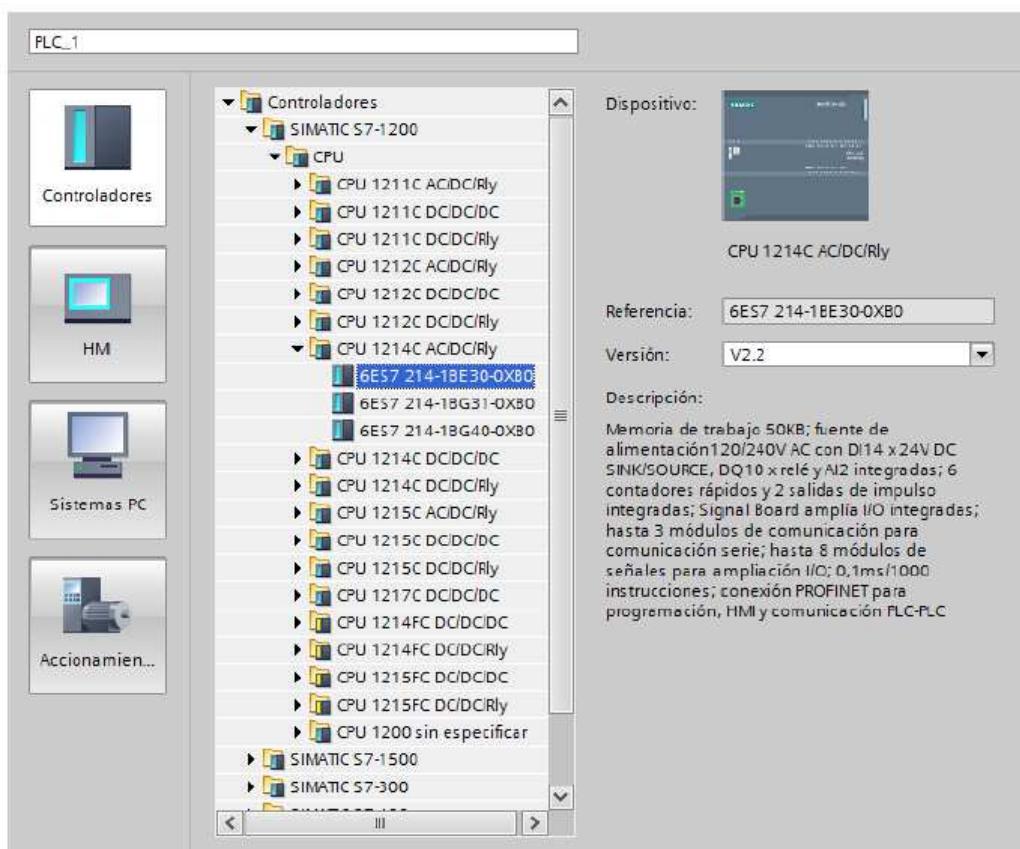
36

Seguidamente se debe configurar un dispositivo (PLC)

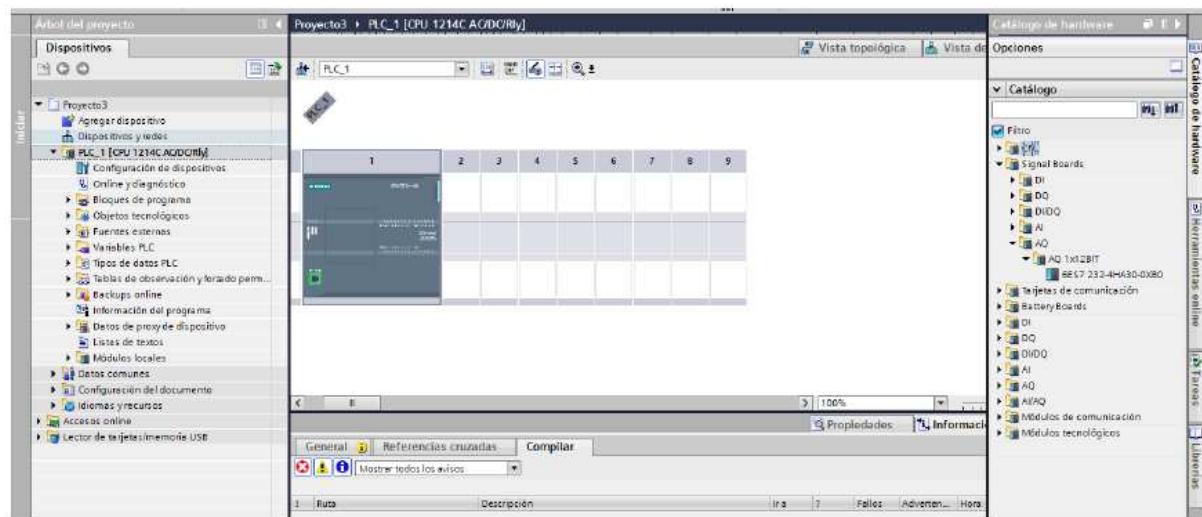
En la pantalla de elección de dispositivos podemos elegir Controladores (PLCs), HMI (Paneles táctiles), Sistemas de control con PCs industriales y Accionamientos (Variadores de frecuencia)



Seguidamente se debe configurar un dispositivo (PLC)

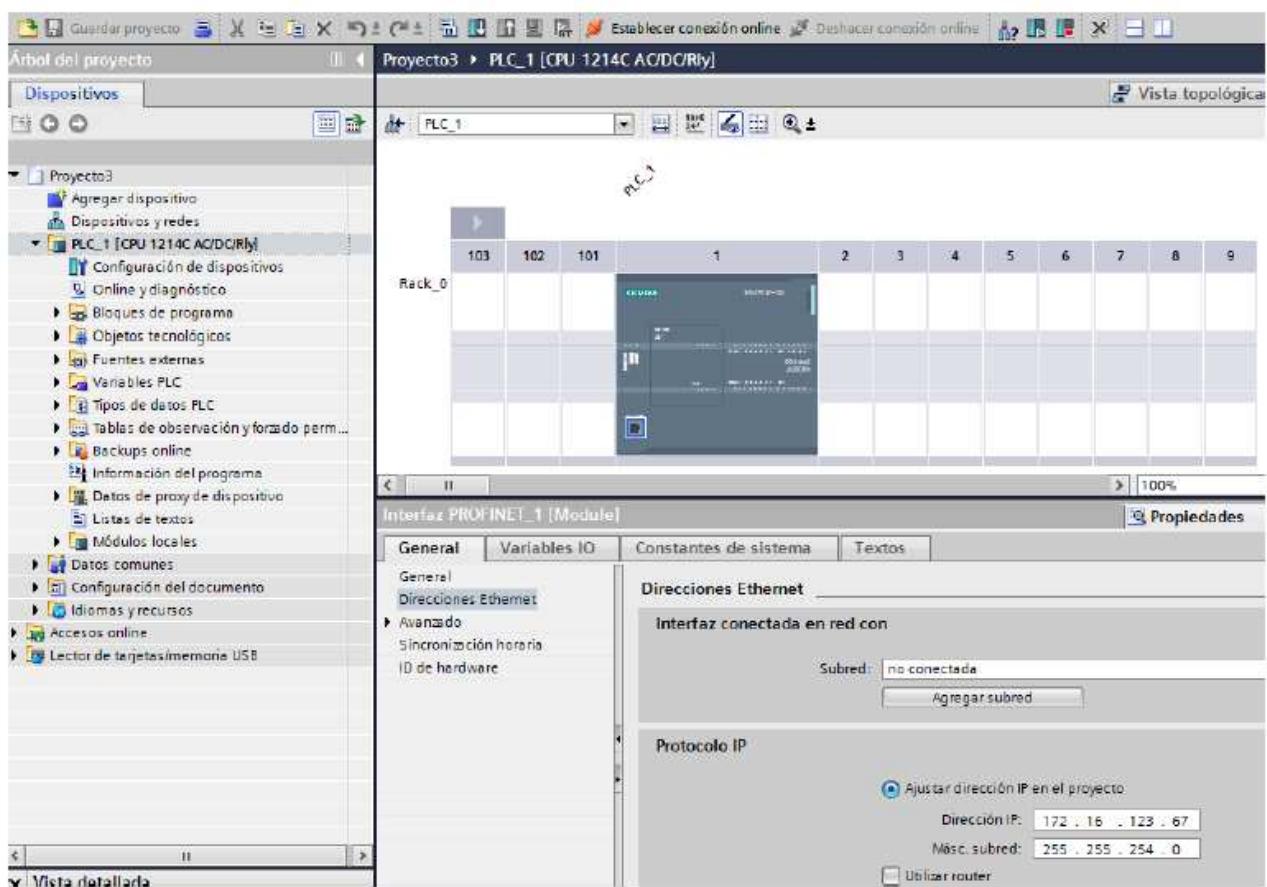


Una vez elegido el PLC debemos terminar de configurarlo añadiendo la Signal Board (Tarjeta de señales) de salidas analógicas (AQ)



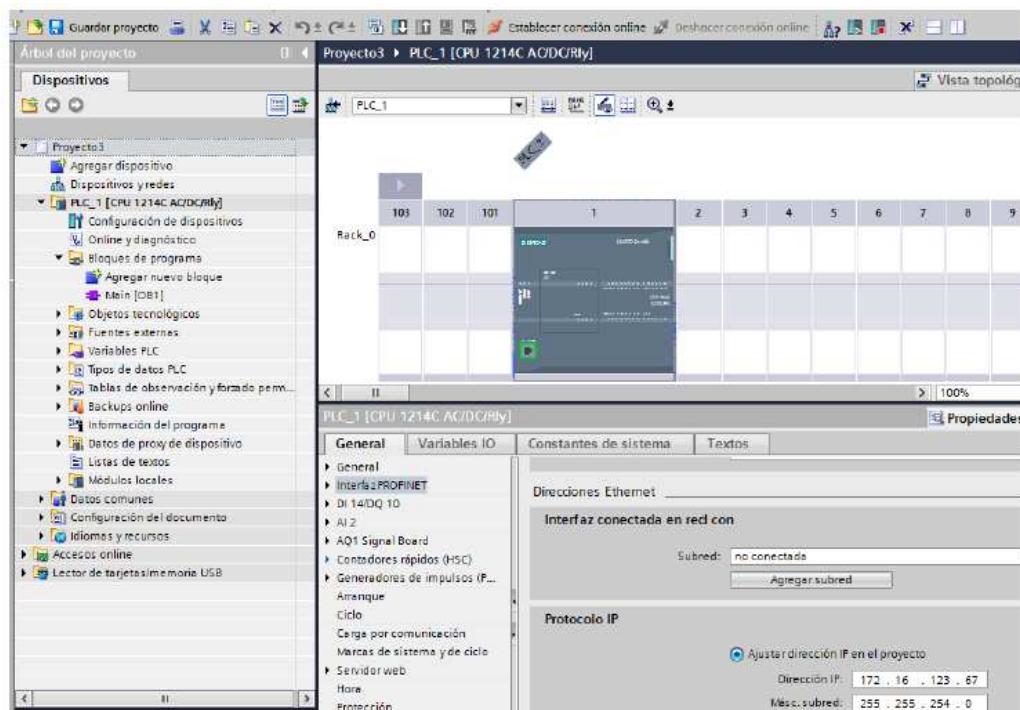
Para finalizar debemos configurar la dirección IP, clic sobre el Puerto Profinet y en la pestaña PROPIEDADES vamos a DIRECCIONES ETHERNET y en Ajustar dirección IP indicamos la dirección IP de nuestro PLC

39



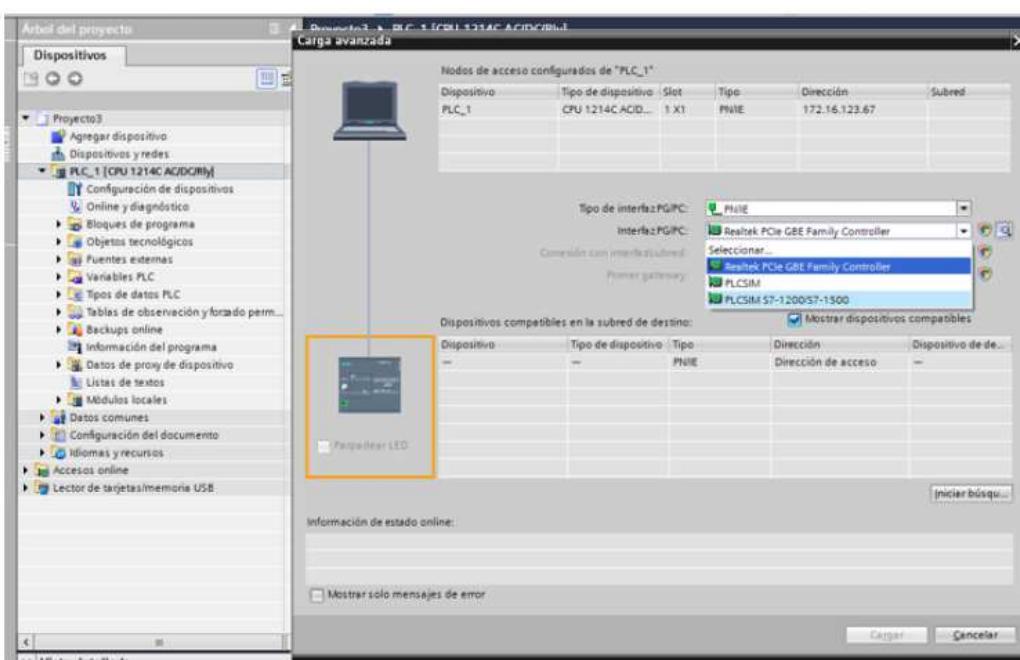
40

Para comprobar la correcta conexión con nuestro PLC, transferimos todo el Hardware configurado al PLC.



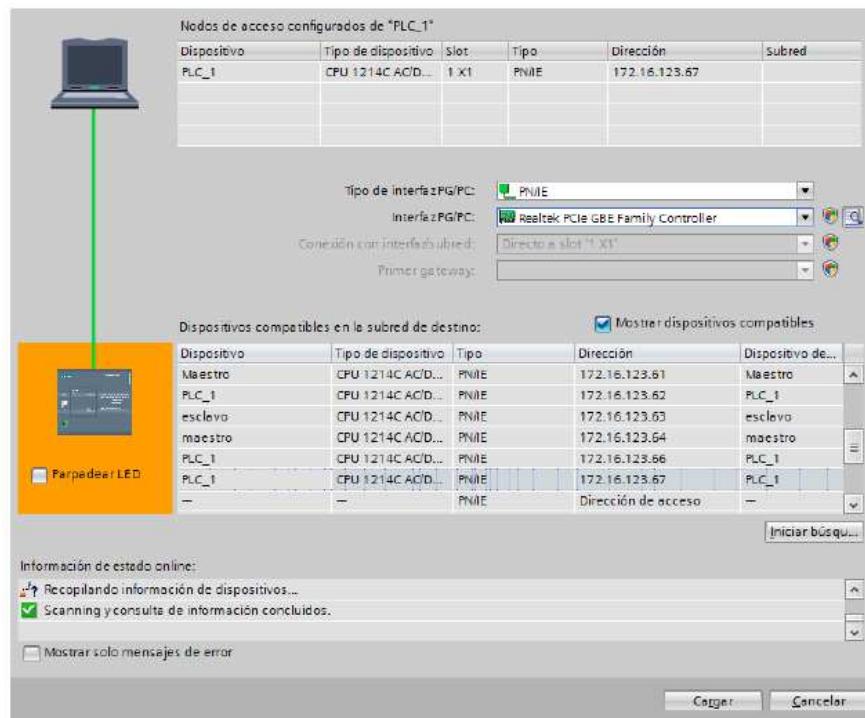
41

Seleccionamos la tarjeta con la que vamos a conectar nuestro PC y el PLC (Elegimos la tarjeta de red de nuestro PC – Realtek PCIe GBE Family Control).



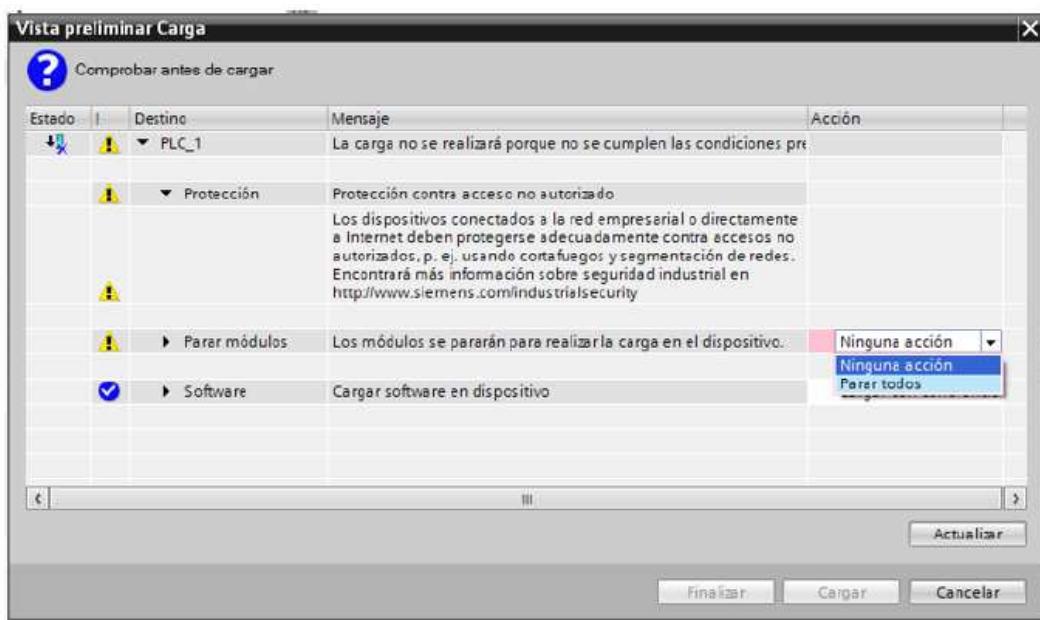
42

En Iniciar búsqueda, el PC comienza a buscar todos los PLCs que están en la red y los muestra en la pantalla los dispositivos compatibles encontrados en la red, haciendo clic elegimos nuestro dispositivo – el que tenga nuestra dirección IP- y en la pestaña CARGAR transferimos el hardware diseñado a nuestro PLC.



43

Si nos pide parar el PLC para poder transferir el hardware elegimos la opción Parar todos y posteriormente Cargar.

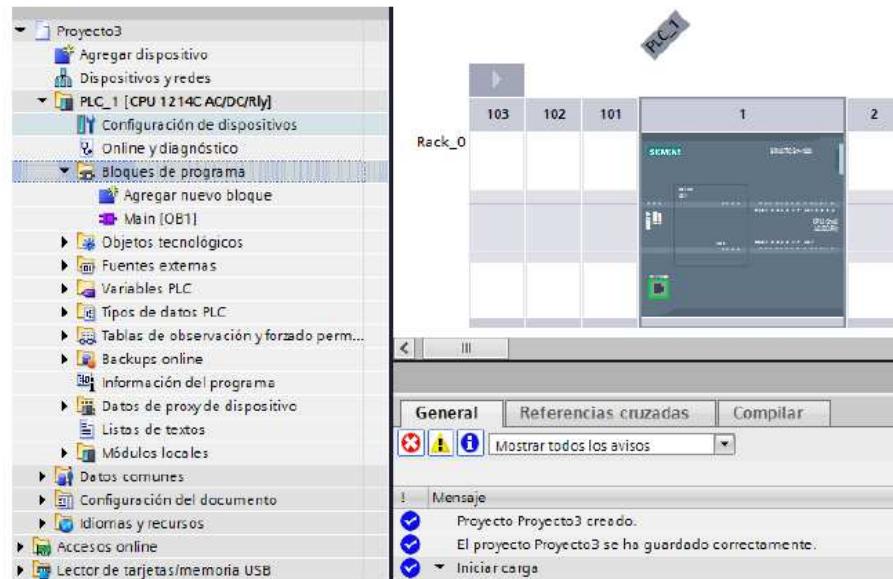


44

Una vez que hemos transferido el hardware y comprobado que estamos ON LINE con nuestro PLC, nos disponemos a realizar la programación (Se podía haber programado también antes de comprobar que estamos ON LINE, pero conviene comprobar que estamos en línea y luego ya comenzar a programar)

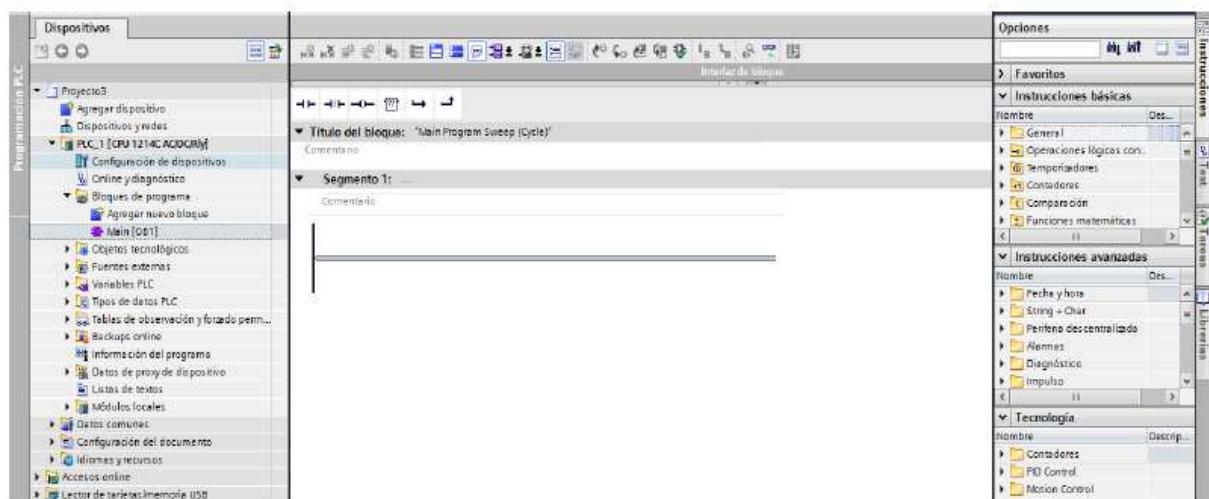
PROGRAMACIÓN

En bloques de programa aparece el Bloque de Organización Principal (Main) OB1. En este bloque comenzaremos a programar.



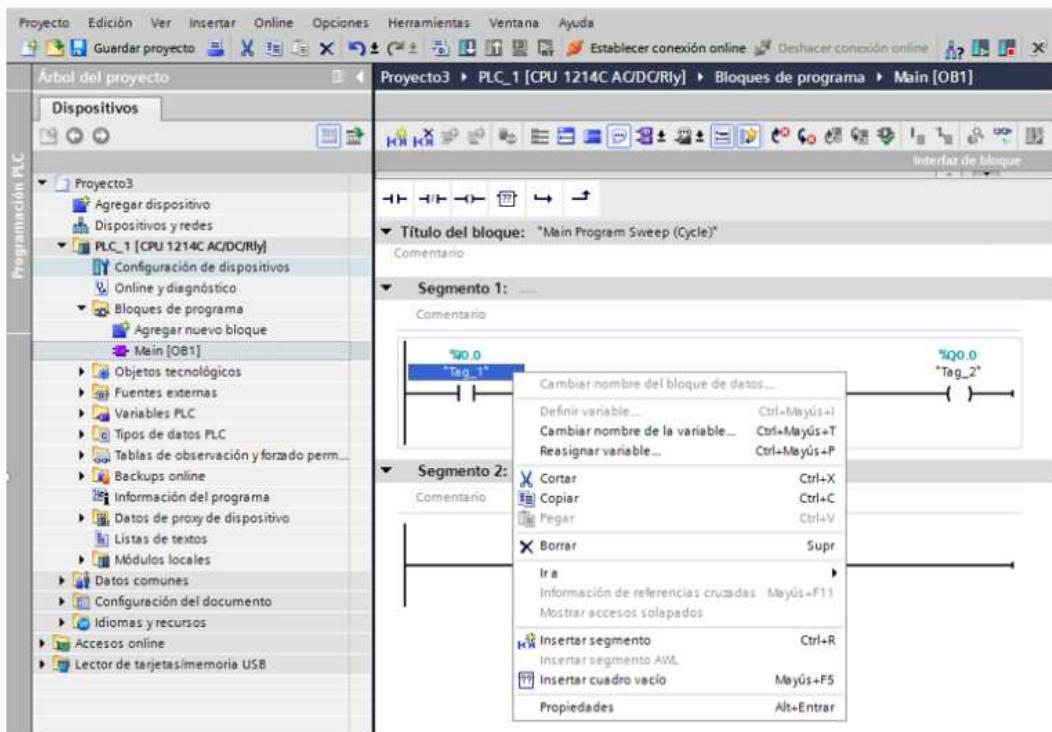
45

Con clic sobre Main (OB1) aparece la pantalla de programación y el primer segmento; en la izquierda de la pantalla tenemos el árbol de dispositivos y en la derecha las instrucciones para programar.



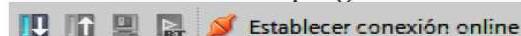
46

Insertamos las instrucciones, por defecto las etiqueta con "Tag_n", con botón derecha "cambiar nombre de la variable", podemos indicarle el nombre que deseamos para esa variable



47

Una vez terminada la programación transferimos el programa al PLC



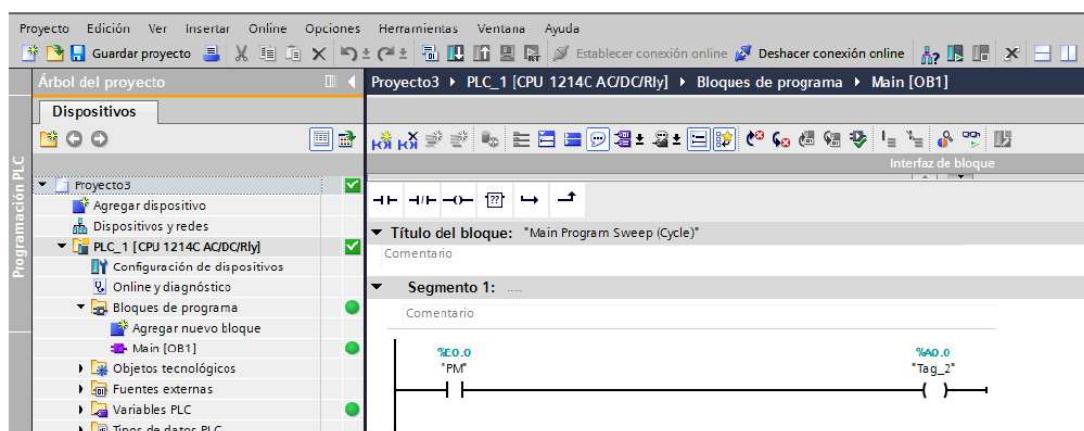
Ponemos el PLC en modo RUN.



También podemos visualizar ON LINE el estado de las variables

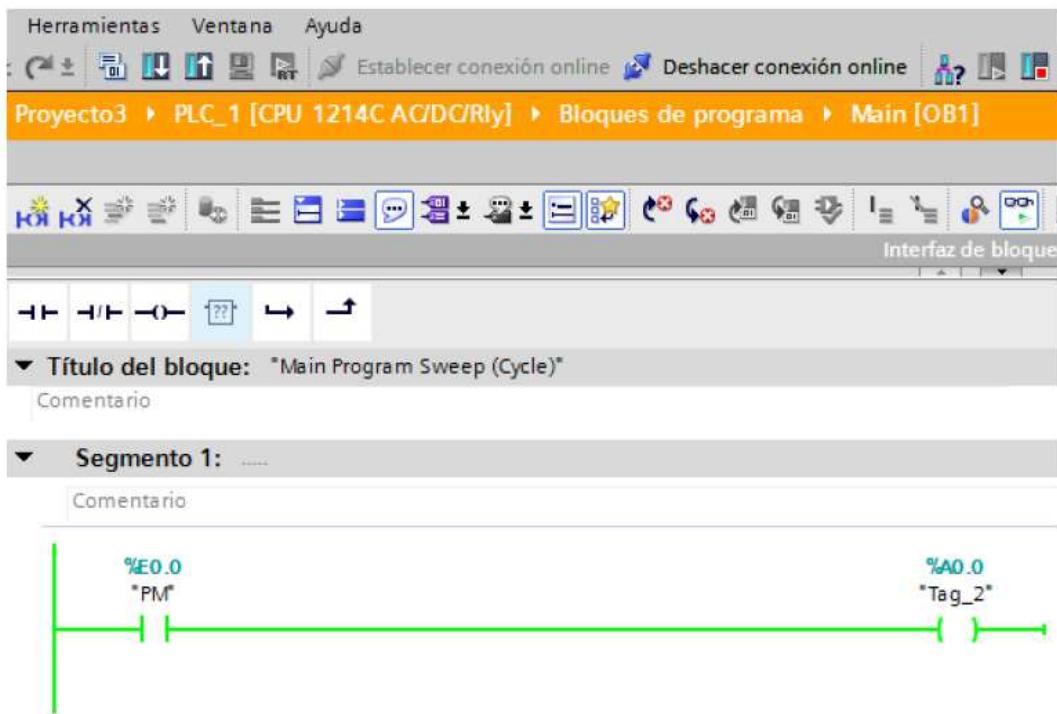


Si todo – hardware y software- está en orden, aparecerán indicadores VERDES en el árbol del proyecto, en caso de algún tipo de error estos indicadores serán de color NARANJA.



48

Se visualiza el estado de funcionamiento del dispositivo quedando:



49



PLCSIM TABLAS Y SECUENCIAS DE SIMULACIÓN

Visión Práctica S1200

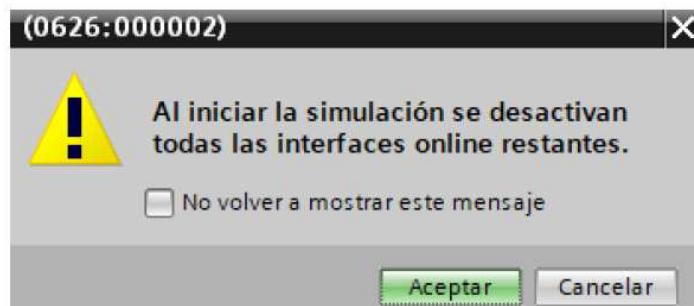
50

PLCSIM

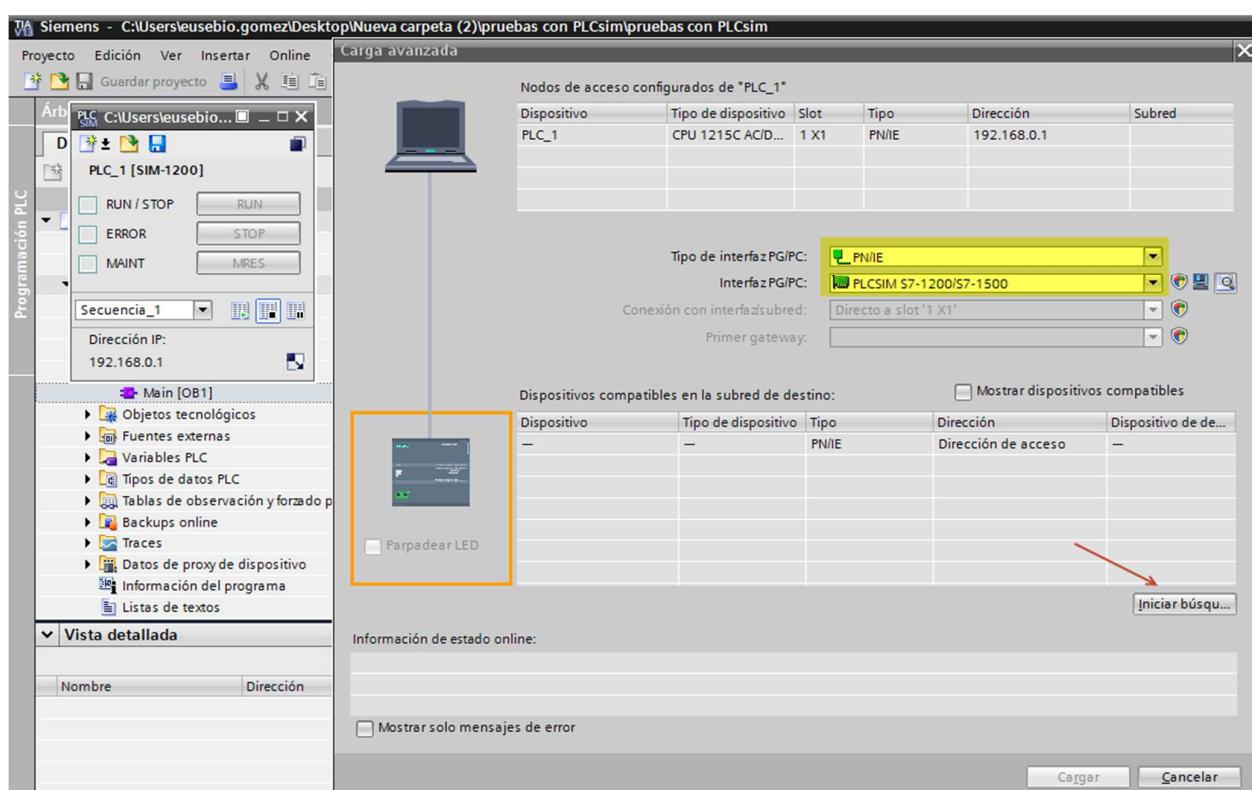
Para comprobar el funcionamiento del programa sin necesidad de transferirlo a un PLC real, podemos hacerlo utilizando la herramienta PLCSIM, programa de simulación de Siemens, para ello activamos el icono correspondiente



Un aviso nos informará que mientras esté activa la simulación se cierran todas las comunicaciones, lo que implica que no podemos conectarnos con el PLC mientras estamos simulando, si queremos comunicarnos con el PLC real antes debemos cerrar el simulador.

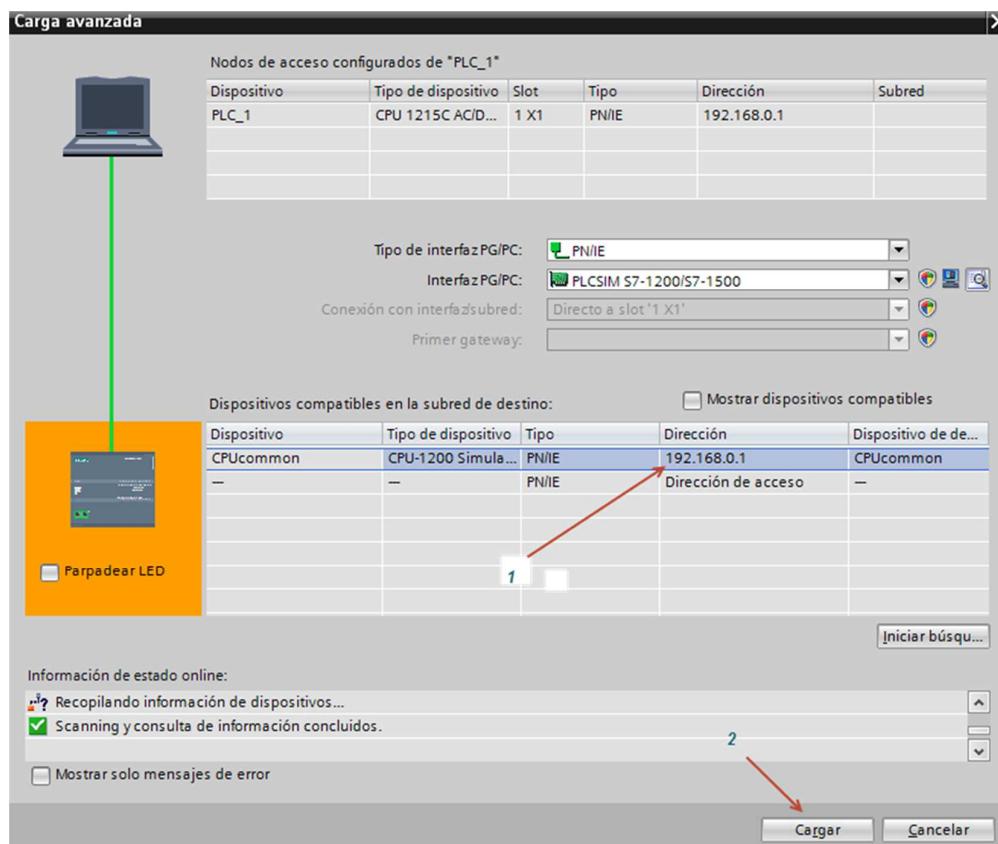


51



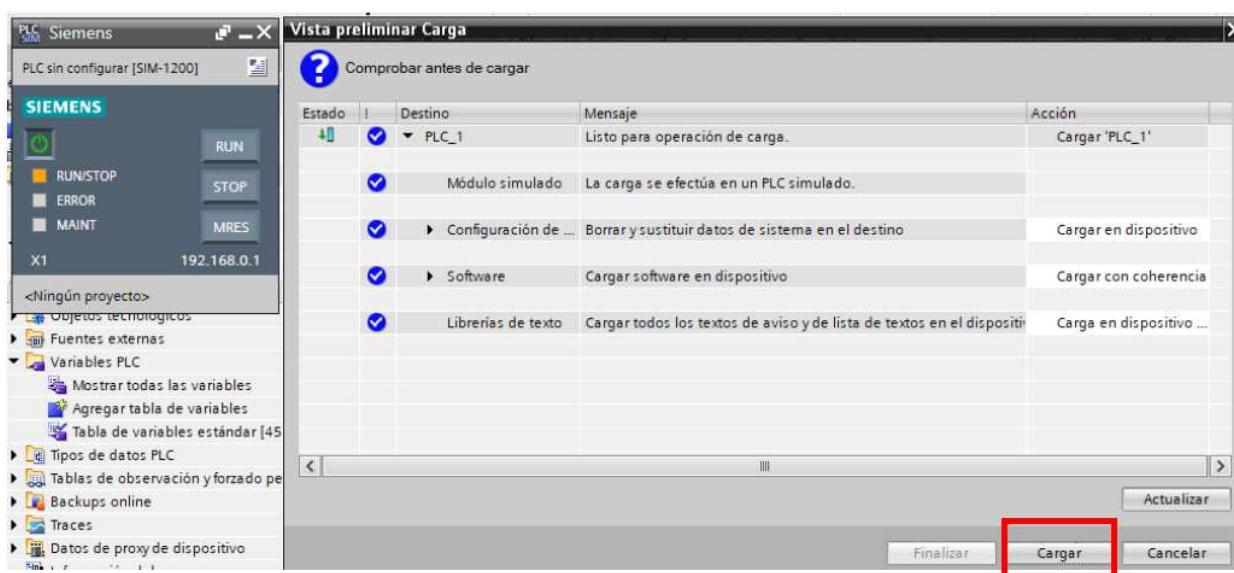
Es importante no asignar IP a la tarjeta de red, dejar que sea el PLCSIM quien le asigne su IP.

52



53

Una vez aceptado se inicia el proceso de apertura del simulador



Aparece en pantalla un PLC “Simulado” con dirección IP 192.168.0.1
El programa comienza a compilarse y después aparece la ventana de carga en la que pide “cargar” el programa en el PLC simulado.

54

Una vez cargado el programa en el PLC simulado se puede comprobar su funcionamiento de la misma forma que en un PLC real, utilizando los botones de:

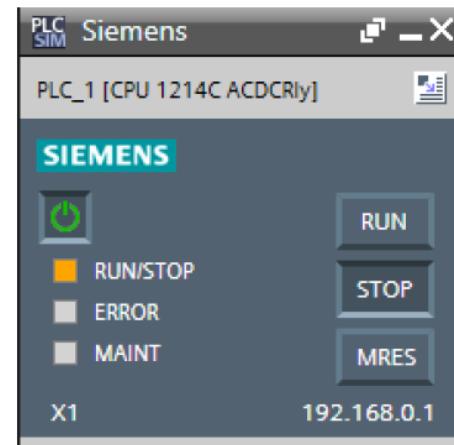
RUN: Ejecución del programa

STOP: Sistema detenido

MRES: Borrado de Memoria

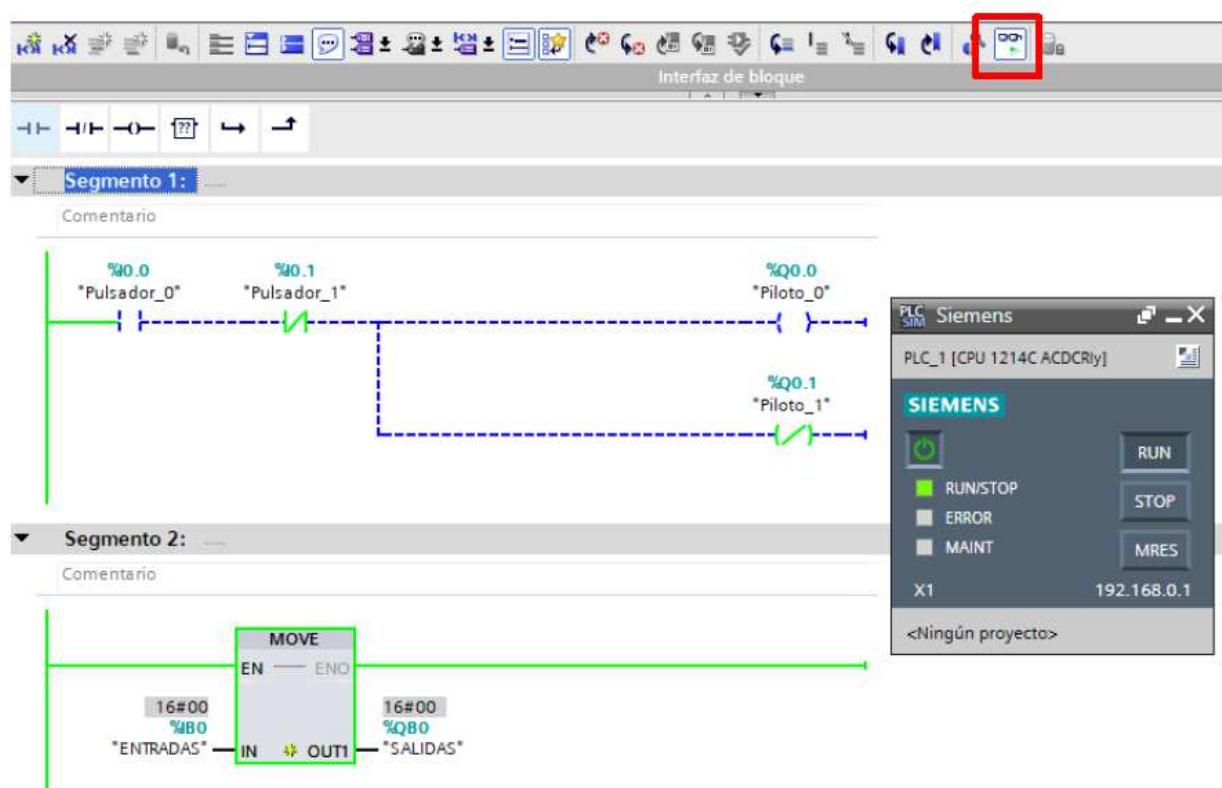
Y los leds que indican si el PLC está en RUN (led ámbar), STOP (led verde), ERROR (led rojo) o en modo mantenimiento MAINT.

Pulsando sobre cada uno de los botones se pone en PLC en el modo requerido.

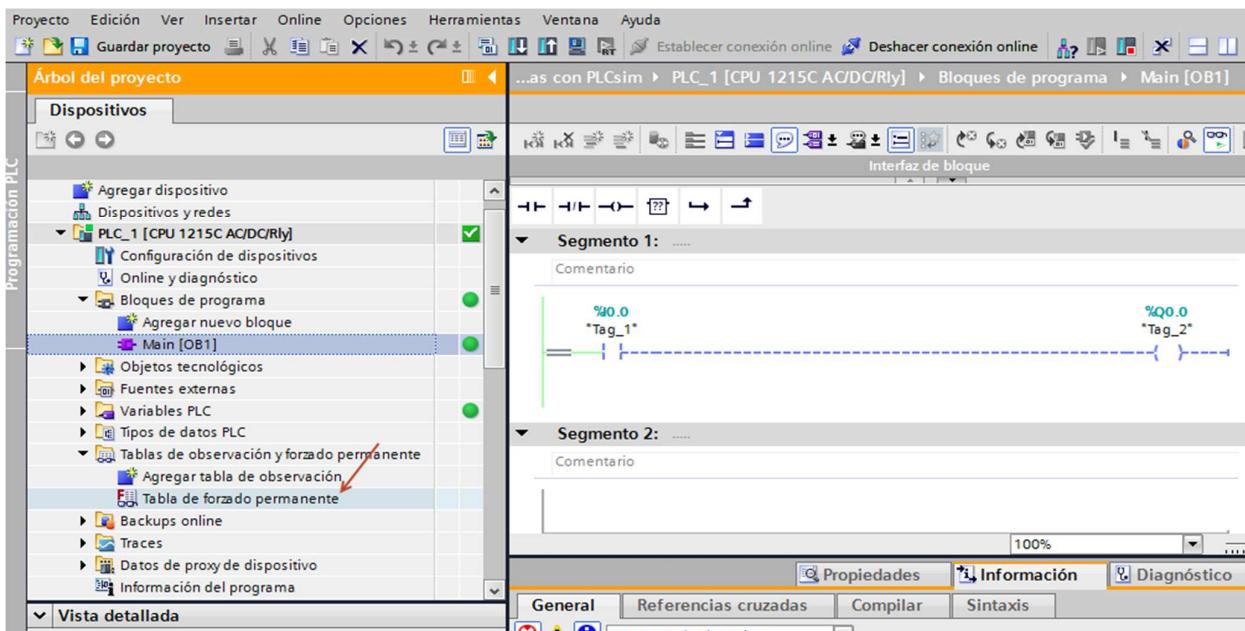


55

Una vez en modo RUN, se puede verificar el estado del programa con el icono "visualizar".



56



57

This screenshot shows the 'Tablas de observación y forzado permanente' (Observation and forced permanent tables) under 'Programación PLC'. The left pane shows the project tree with 'Main [OB1]' selected. The right pane displays a table titled 'Tabla de forzado permanente' with two rows:

	Nombre	Dirección	Formato visualiza...	Valor de observaci...	Valor de forzado ...	F
1	*Tag_1*:P	%IO.0:P	BOOL	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	*Tag_2*:P	%Q0.0:P	BOOL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3		<Agregar>				

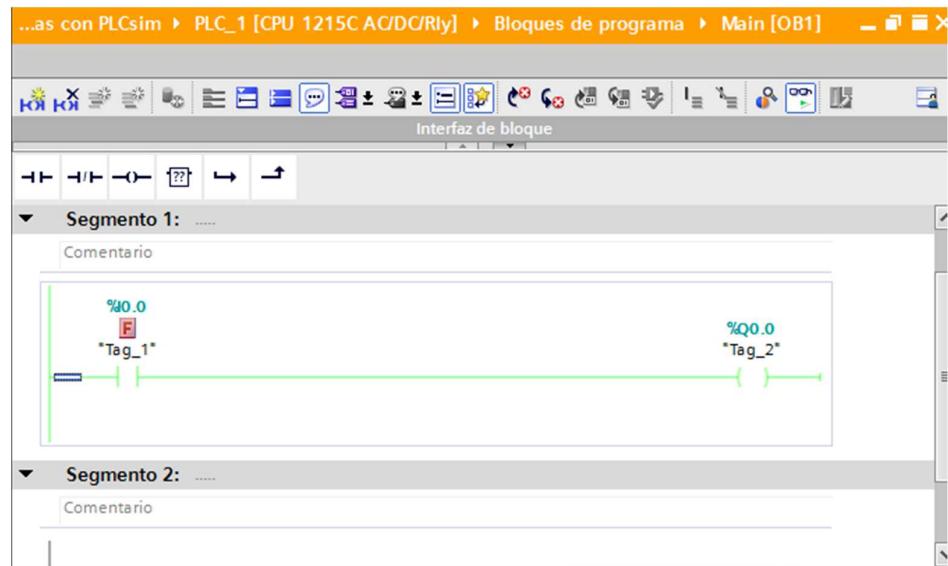
A context menu is open over the second row (Tag_2). The menu items include:

- Actualizar operandos forzados permanentemente
- Forzar permanentemente a 0
- Forzar permanentemente a 1
- Forzar todos permanentemente
- Finalizar forzado permanente
- Forzar permanentemente
- Observar todos
- Observar inmediatamente
- Insertar fila
- Agregar fila
- Cortar
- Copiar
- Pegar
- Borrar
- Cambiar nombre
- Información de referencias cruzadas
- Modo avanzado

58

...JRly] > Tablas de observación y forzado permanente > Tabla de forzado permanente

	Nombre	Dirección	Formato visualiza...	Valor de observac...	Valor de forzado ...	F
1	*Tag_1*:P	%IO0:P	BOOL	0	TRUE	<input checked="" type="checkbox"/>
2	*Tag_2*:P	%Q0.0:P	BOOL	0	0	<input type="checkbox"/>
3	<Agregar>					



59



TIPOS DE DATOS EN PLC

Visión Práctica S1200

60

SINT (ENTEROS DE 8 BITS, CON SIGNO). Un operando del tipo de datos SINT (short INT) tiene una longitud de 8 bits y consta de dos componentes: Un signo y un valor numérico en complemento a dos. Los estados lógicos de los bits 0 a 6 representan el valor del número. El estado lógico del bit 7 representa el signo. El signo puede adoptar el estado lógico "0" para positivo o "1" para negativo. Un operando del tipo de datos SINT ocupa un BYTE en la memoria.

Longitud (bits)	Formato	Rango de valores	Ejemplos de entrada de valores
8	Enteros con signo	-128 hasta 127	+44, SINT#+44 Si se utiliza la tipificación SINT#, el rango de valores va hasta SINT#255 como máximo. Este valor se interpreta como entero con -1.
	Números binarios (sólo positivos)	2#0 hasta 2#01111111	2#00101100, SINT#2#00101100
	Números octales (sólo positivos)	8#0 hasta 8#177	8#54, SINT#8#54
	Números hexadecimales (sólo positivos)	16#0 hasta 16#7F	16#2C, SINT#16#2C Si se utiliza la tipificación SINT#, el rango de valores va hasta SINT#16#FF como máximo. Este valor se interpreta como entero con -1.

61

USINT (ENTEROS DE 8 BITS SIN SIGNO). Un operando del tipo de datos USINT (Unsigned Short INT) tiene una longitud de 8 bits y contiene valores numéricos sin signo. Un operando del tipo de datos USINT ocupa un BYTE en la memoria.

Longitud (bits)	Formato	Rango de valores	Ejemplos de entrada de valores
8	Enteros sin signo	de 0 a 255	78, USINT#78
	Números binarios	de 2#0 a 2#11111111	2#01001110, USINT#2#01001110
	Números octales	de 8#0 a 8#377	8#116, USINT#8#116
	Números hexadecimales	16#0 hasta 16#FF	16#4E, USINT#16#4E

INT (ENTEROS DE 16 BITS, CON SIGNO). Un operando del tipo de datos INT tiene una longitud de 16 bits y consta de dos componentes: Un signo y un valor numérico en complemento a dos. Los estados lógicos de los bits 0 a 14 representan el valor del número. El estado lógico del bit 15 representa el signo. El signo puede adoptar el estado lógico "0" para positivo o "1" para negativo. Un operando del tipo de datos INT ocupa dos BYTE en la memoria.

Longitud (bits)	Formato	Rango de valores	Ejemplos de entrada de valores
16	Enteros con signo	-32768 hasta 32767	+3785, INT#+3785
	Números binarios (sólo positivos)	2#0 hasta 2#0111111111111111	2#0000111011001001, INT#2#0000111011001001
	Números octales	8#0 hasta 8#77777	8#7311, INT#8#7311
	Números hexadecimales (sólo positivos)	16#0 hasta 16#7FFF	16#0EC9, INT#16#0EC9

62

UINT (ENTEROS DE 16 BITS, SIN SIGNO). Un operando del tipo de datos UINT (Unsigned INT) tiene una longitud de 16 bits y contiene valores numéricos sin signo. Un operando del tipo de datos UINT ocupa dos BYTE en la memoria.

Longitud (bits)	Formato	Rango de valores	Ejemplos de entrada de valores
16	Enteros sin signo	de 0 a 65535	65295, <code>UINT#65295</code>
	Números binarios	de <code>2#0</code> a <code>2#1111111111111111</code>	<code>2#1111111000011111</code> , <code>UINT#2#1111111000011111</code>
	Números octales	de <code>8#0</code> a <code>8#177777</code>	<code>8#177417</code> , <code>UINT#8#177417</code>
	Números hexadecimales	16#0 hasta 16#FFFF	<code>16#FF0F</code> , <code>UINT#16#FF0F</code>

DINT (ENTEROS DE 32 BITS, CON SIGNO). Un operando del tipo de datos DINT (Double INT) tiene una longitud de 32 bits y consta de dos componentes: Un signo y un valor numérico en complemento a dos. Los estados lógicos de los bits 0 a 30 representan el valor del número. El estado lógico del bit 31 representa el signo. El signo puede adoptar el estado lógico "0" para positivo o "1" para negativo. Un operando del tipo de datos DINT ocupa cuatro BYTE en la memoria.

Longitud (bits)	Formato	Rango de valores	Ejemplos de entrada de valores
32	Enteros con signo	-2147483648 hasta +2147483647	125790, DINT#125790, L#275
	Números binarios (sólo positivos)	2#0 hasta 2#01111111111111111111111111111111	2#00000000000000000111010110101110, DINT#2#0000000000000000111010110101110
	Números octales (sólo positivos)	8#0 hasta 8#1777777777	8#365536, DINT#8#365536
	Números hexadecimales	16#00000000 hasta 16#7FFFFFFF	16#0001EB5E, DINT#16#0001EB5E

63

UDINT (ENTEROS DE 32 BITS, SIN SIGNO). Un operando del tipo de datos UDINT (Unsigned Double INT) tiene una longitud de 32 bits y contiene valores numéricos sin signo. Un operando del tipo de datos UDINT ocupa cuatro BYTE en la memoria.

Longitud (bits)	Formato	Rango de valores	Ejemplos de entrada de valores
32	Enteros sin signo	de 0 a 4294967295	4042322160, UDINT#4042322160
	Números binarios	de 2#0 a 2#11110000111100001111000011110000 2#11	2#11110000111100001111000011110000, UDINT#2#11110000111100001111000011110000
	Números octales	8#0 hasta 8#3777777777	8#36074170360, UDINT#8#36074170360
	Números hexadecimales	de 16#00000000 a 16#FFFFFF	16#F0F0F0F0, UDINT#16#F0F0F0F0

LINT (ENTERO DE 64 BITS, CON SIGNO). Un operando del tipo de datos LINT (Long INT) tiene una longitud de 64 bits y consta de dos componentes: Un signo y un valor numérico en complemento a dos. Los estados lógicos de los bits 0 a 62 representan el valor del número. El estado lógico del bit 63 representa el signo. El signo puede adoptar el estado lógico "0" para positivo o "1" para negativo. Un operando del tipo de datos LINT ocupa ocho BYTE en la memoria.

64

ULINT (ENTERO DE 64 BITS SIN SIGNO). Un operando del tipo de datos ULINT (Unsigned Long INT) tiene una longitud de 64 bits y contiene valores numéricos sin signo. Un operando del tipo de datos ULINT ocupa ocho BYTE en la memoria.

REAL. Los operandos del tipo de datos REAL tienen una longitud de 32 bits y se utilizan para representar números en coma flotante. Un operando del tipo de datos REAL consta de los tres componentes siguientes:

Signo: el estado lógico del bit 31 determina el signo. El bit 31 puede adoptar los valores "0" (positivo) o "1" (negativo).

Longitud (bits)	Formato	Rango de valores	Ejemplos de entrada de valores
32	Números en coma flotante según IEEE754	-3.402823e+38 a -1.175495e-38 ±0.0	1.0e-5, REAL#1.0e-5
	Números en coma flotante	+1.175495e-38 a +3.402823e+38	1.0, REAL#1.0

65

LREAL. Los operandos del tipo de datos LREAL tienen una longitud de 64 bits y se utilizan para representar valores en coma flotante. Un operando del tipo de datos LREAL consta de los tres componentes siguientes:

Signo: el estado lógico del bit 63 determina el signo. El bit 63 puede adoptar los valores "0" (positivo) o "1" (negativo).

Longitud (bits)	Formato	Rango de valores	Ejemplos de entrada de valores
64	Números en coma flotante según IEEE754	de -1.7976931348623158e+308 a -2.2250738585072014e-308 ±0,0	1.0e-5, REAL#1.0e-5
	Números en coma flotante	de +2.2250738585072014e-308 a +1.7976931348623158e+308	1,0, LREAL#1.0

DATOS de TIEMPO

TIME (IEC)

El contenido de un operando del tipo TIME se interpreta como milisegundos. La representación contiene especificaciones de días (d), horas (h), minutos (m), segundos (s) y milisegundos (ms). La siguiente tabla muestra las propiedades del tipo de datos TIME:

Longitud (bits)	Formato	Rango de valores	Ejemplos de entrada de valores
32	Tiempo con signo	T#-24d20h31m23s648ms bis T#+24d20h31m23s647ms	T#10d20h30m20s630ms, TIME#10d20h30m20s630ms
	Números hexadecimales	16#00000000 hasta 16#7FFFFFFF	16#0001EB5E

No es necesario indicar todas las unidades de tiempo. Por ejemplo, T#5h10s es válido. Si se indica solo una unidad, el valor absoluto de días, horas y minutos no podrá exceder los límites superiores ni inferiores. Si se indica más de una unidad de tiempo, el valor correspondiente no podrá exceder 24 días, 23 horas, 59 minutos, 59 segundos o 999 milisegundos.

LTIME (TEMPORIZADOR IEC)

El contenido de un operando del tipo LTIME se interpreta como nanosegundos. La representación contiene especificaciones de días (d), horas (h), minutos (m), segundos (s), milisegundos (ms), microsegundos (us) y nanosegundos (ns).

Longitud (bits)	Formato	Rango de valores	Ejemplos de entrada de valores
64	Tiempo con signo	de LT#-106751d23h47m16s854ms775us808ns a LT#+106751d23h47m16s854ms775us807ns	LT#11350d20h25m14s830ms652us315ns, LTIME#11350d20h25m14s830ms652us315ns
	Números hexadecimales	16#0 a 16#8000000000000000	16#2

DATOS de FECHA Y HORA

DATE

El tipo de datos DATE guarda una fecha como número entero sin signo. La representación contiene el año, el mes y el día.

El contenido de un operando del tipo de datos DATE corresponde al número de días desde 01/01/1990, expresado en formato hexadecimal (16#0000).

La tabla siguiente muestra las propiedades del tipo de datos DATE:

Longitud (bytes)	Formato	Rango de valores	Ejemplos de entrada de valores
2	Fecha IEC (año-mes-día)	de D#1990-01-01 a D#2168-12-31	D#2009-12-31, DATE#2009-12-31
	Números hexadecimales	de 16#0000 a 16#FF62	16#00F2

DT (DATE_AND_TIME)

El tipo de datos DT (DATE_AND_TIME) guarda información de fecha y hora en formato BCD.

La tabla siguiente muestra las propiedades del tipo de datos DT:

Longitud (bytes)	Formato	Rango de valores	Ejemplo de entrada de valores
8	Fecha y hora (año-mes-día-hora:minuto:segundo:milisegundo ³⁾)	Mín.: DT#1990-01-01- 00:00:00.000 Máx.: DT#2089-12-31- 23:59:59.999	DT#2008-10-25-8:12:34.567, DATE_AND_TIME#2008-10-25-8:12:34.567

LDT (DATE_AND_LTIME)

El tipo de datos LDT (DATE_AND_LTIME) guarda información de fecha y hora en nanosegundos desde el 01.01.1970 0:0.

La tabla siguiente muestra las propiedades del tipo de datos LDT:

Longitud (bytes)	Formato	Rango de valores	Ejemplo de entrada de valores
8	Fecha y hora (año-mes-día-hora:minuto:segundo.nanosegundo)	Mín.: LDT#1970-01-01-0:0:0.000000000 Máx.: LDT#2263-04-11-23:47:16.854775807	LDT#2008-10-25-8:12:34.567
	Números hexadecimales	de 16#0 a 16#7FFF_FFFF_FFFF_FFFF	16#7FFF

DTL

Un operando del tipo de datos DTL tiene una longitud de 12 bytes y guarda datos de fecha y hora en una estructura predefinida.

La tabla siguiente muestra las propiedades del tipo de datos DTL:

Longitud (bytes)	Formato	Rango de valores	Ejemplo de entrada de valores
12	Fecha y hora (año-mes-día-hora:minuto:segundo.nanosegundo)	Mín.: DTL#1970-01-01-00:00:00.0 Máx.: DTL#2262-04-11-23:47:16.854775807	DTL#2008-12-16-20:30:20.250

69

La estructura del tipo de datos DTL consta de varios componentes, cada uno de los cuales puede tener un tipo de datos y un rango de valores distinto. El tipo de datos de un valor indicado debe coincidir con el tipo de datos del componente en cuestión.

La tabla siguiente muestra los componentes de la estructura del tipo de datos DTL y sus propiedades:

Byte	Componente	Tipo de datos	Rango de valores
0	Año	UINT	de 1970 a 2262
1			
2	Mes	USINT	de 1 a 12
3	Día	USINT	1 hasta 31
4	Día de la semana	USINT	de 1 (domingo) a 7 (sábado) El día de la semana no se tiene en cuenta al introducir los valores.
5	Hora	USINT	0 hasta 23
6	Minuto	USINT	0 hasta 59
7	Segundo	USINT	0 hasta 59
8	Nanosegundo	UDINT	de 0 a 999999999
9			
10			
11			

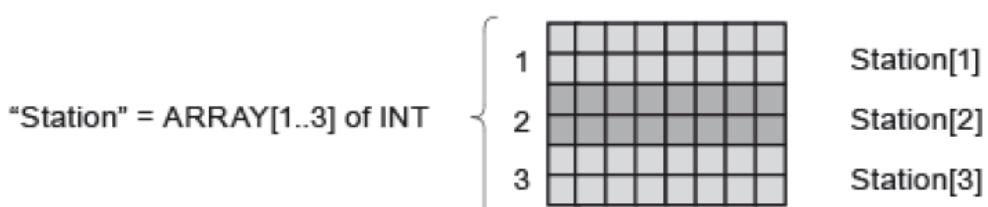
70

ARRAY

Una variable del tipo de datos ARRAY representa una estructura de datos compuesta por un número fijo de componentes del mismo tipo de datos. Para los componentes se permiten todos los tipos de datos salvo ARRAY.

Al crear una variable ARRAY, los límites de indexación se definen entre corchetes, y el tipo de datos se define después de la palabra clave "of". Los límites del ARRAY no solo pueden definirse de forma fija con números enteros o constantes globales o locales o como parámetros formales de un bloque, sino también de forma variable mediante ARRAY[*]. El límite inferior debe ser menor o igual que el límite superior. Un ARRAY puede contener hasta seis dimensiones, cuyos límites se especifican separados entre sí por comas.

La estructura de una variable unidimensional del tipo de datos ARRAY tiene, por ejemplo, el siguiente aspecto



71

CADENA DE CARACTERES

CHAR (CARÁCTER)

Un operando del tipo de datos CHAR tiene una longitud de 8 bits y ocupa un BYTE en la memoria.

El tipo de datos CHAR almacena un solo carácter en formato ASCII. Encontrará información sobre la codificación de caracteres especiales en "Consulte también > STRING". La tabla siguiente muestra el rango de valores del tipo de datos CHAR

Longitud (bits)	Formato	Rango de valores	Ejemplo de entradas de valores
8	Caracteres ASCII	Juego de caracteres ASCII	'A', CHAR#'A'

STRING (CADENA DE CARACTERES)

Un operando del tipo de datos STRING guarda varios caracteres en una cadena que puede estar formada por un máximo de 254 caracteres. En las cadenas de caracteres se admiten todos los caracteres del código ASCII. Los caracteres se introducen entre comillas sencillas.

La tabla siguiente muestra las propiedades de una variable STRING:

Longitud (bytes)	Formato	Rango de valores	Ejemplo de entrada de valores
n + 2 *	Cadena de caracteres ASCII, incluidos los caracteres especiales	0 a 254 caracteres	'Nombre', STRING#NAME'

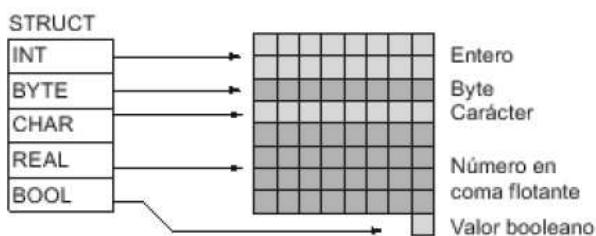
* Un operando del tipo de datos STRING ocupa en la memoria dos bytes más que la longitud máxima especificada.

STRUCT

El tipo de datos STRUCT representa una estructura de datos compuesta por un número fijo de componentes de diferentes tipos de datos. Los componentes de los tipos de datos STRUCT o ARRAY también pueden anidarse en una estructura.

Longitud	Formato	Rango de valores	Ejemplo de entrada de valores
Una variable STRUCT comienza en un byte con dirección par y ocupa la memoria hasta el siguiente límite de palabra.	STRUCT	Son aplicables los rangos de valores de los tipos de datos utilizados.	Rigen las reglas de entrada de valores para los tipos de datos utilizados.

La figura siguiente muestra un ejemplo de la estructura de una variable STRUCT:



73



Operaciones Básicas de Programación

Visión Práctica S1200

74



Operaciones Lógicas con Bits

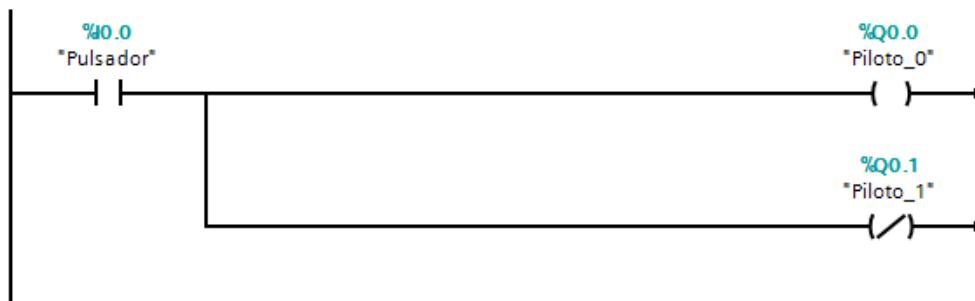
Visión Práctica S1200

- ▶ Segmento 1: SALIDA Y SALIDA NEGADA
- ▶ Segmento 2: BIESTABLE: SET / RESET
- ▶ Segmento 3: BIESTABLE: SET / RESET sobre un CAMPO DE BITS
- ▶ Segmento 4: BIESTABLE FLIP_FLOP: SET / RESET con prioridad RESET
- ▶ Segmento 5: BIESTABLE FLIP_FLOP: RESET / SET con prioridad al SET
- ▶ Segmento 6: FLANCOES DE SUBIDA (FLANCO POSITIVO) Y DE BAJADA (FLANCO NEGATIVO)
- ▶ Segmento 7: ACTIVAR SALIDAS CON FLANCOES DE SUBIDA (FLANCO POSITIVO) Y DE BAJADA (FLANCO NEGATIVO)
- ▶ Segmento 8: CONSULTA (P_TRIGGER Y N_TRIGGER) FLANCOES DE SUBIDA (FLANCO POSITIVO) Y DE BAJADA (FLANCO NEGATIVO)
- ▶ Segmento 9: GENERAR (R_TRIGGER Y F_TRIGGER) FLANCOES DE SUBIDA (FLANCO POSITIVO) Y DE BAJADA (FLANCO NEGATIVO)

75

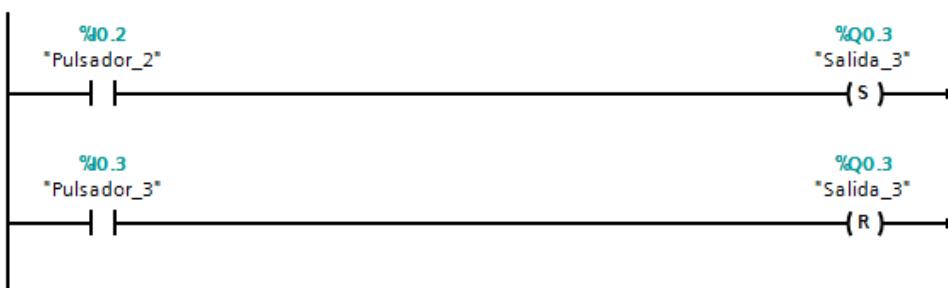
▼ Segmento 1: SALIDA Y SALIDA NEGADA

- ▶ Si la entrada I0.0 = 0 entonces Q0.0 = 0 y Q0.1 = 1
Si la entrada I0.0 = 1 entonces Q0.0 = 1 y Q0.1 = 0
La salida Q0.1 es una salida negada (invertida)



▼ Segmento 2: BIESTABLE: SET / RESET

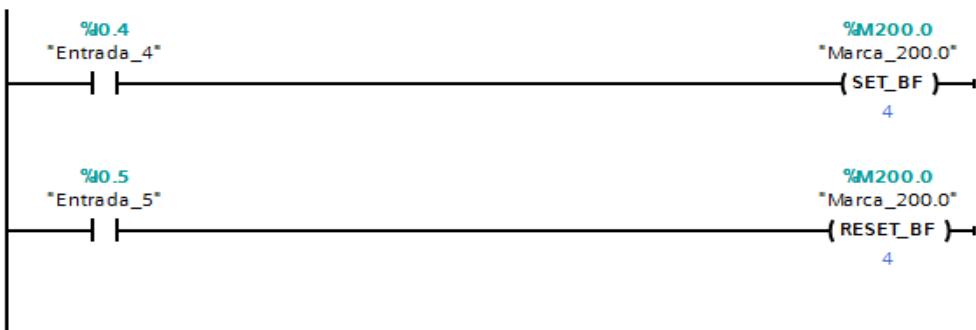
- ▶ Si I0.2 = 1 entonces Set la salida Q0.3
Si I0.3 = 1 entonces Reset la salida Q0.3
Si ambos están activados tiene prioridad lo programado en última posición (I0.3)
Se pueden programar en segmentos diferentes.



76

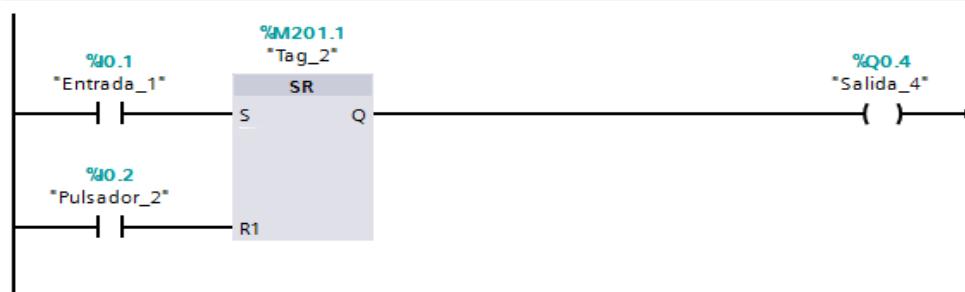
Segmento 3: BIESTABLE: SET / RESET sobre un CAMPO DE BITS

- Si I0.4=1 Set sobre M200.0, M200.1, M200.2 y M200.3 (se activan 4 bits)
- Si I0.5=1 Reset sobre M200.0, M200.1, M200.2 y M200.3 (se resetean 4 bits)
- Si ambos están activados tiene prioridad lo programado en última posición (I0.5)
- Se pueden programar en diferentes segmentos



Segmento 4: BIESTABLE FLIP_FLOP: SET / RESET con prioridad RESET

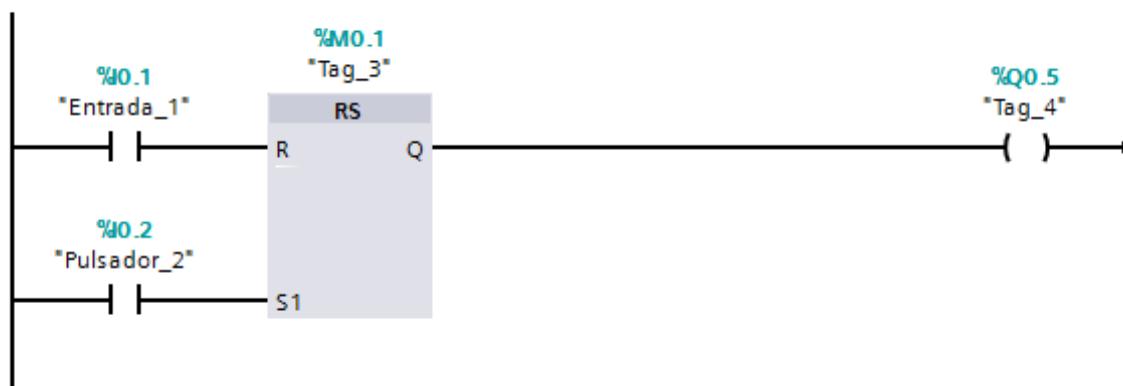
- Si I0.1=1 entonces SET sobre Q0.4
- Si I0.2=1 entonces RESET sobre Q0.4
- Si ambos están activados tiene prioridad lo programado en la parte inferior (R1_RESET)



77

Segmento 5: BIESTABLE FLIP_FLOP: RESET / SET con prioridad al SET

- Si I0.1=1 entonces RESET sobre Q0.5
- Si I0.2=1 entonces SET sobre Q0.5
- Si ambos están activados tiene prioridad lo programado en la parte inferior (S1_SET)
- El estado de la marca M0.1 es, en todo momento igual al de la salida Q0.5



78

▼ Segmento 6: FLANCOS DE SUBIDA (FLANCO POSITIVO) Y DE BAJADA (FLANCO NEGATIVO)

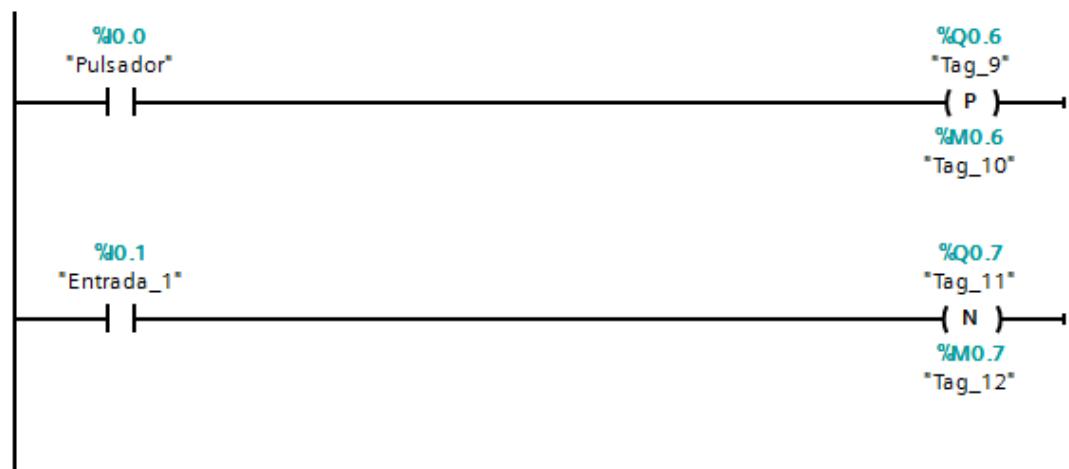
- Con el FP de I0.1 entonces SET la marca M0.3
- Con el FN de I0.2 entonces RESET sobre M0.4
- Las marcha M0.3 y M0.5 es necesario utilizarlas para la correcta ejecución de la instrucción.
- La instrucción compara el estado lógico actual del operando I0.1 con el estado lógico de la consulta anterior que está almacenada en una marca de flancos M0.2.
- Si la instrucción detecta un cambio del resultado lógico de 0 a 1 significa que hay un flanco de señal ascendente.



79

▼ Segmento 7: ACTIVAR SALIDAS CON FALNCOS DE SUBIDA (FLANCO POSITIVO) Y DE BAJADA (FLANCO NEGATIVO)

- Con la activación de I0.0 se genera un flanco sobre Q0.6 con resultado lógico igual a 1 solamente durante el tiempo de ciclo.
- Con desactivación de I0.1 se genera un flanco sobre Q0.7 con resultado lógico igual a 1 solamente durante el tiempo de ciclo.
- La instrucción compara el resultado lógico actual con el resultado lógico de la consulta anterior que está almacenado en una marca de flancos.
- Si la instrucción detecta un cambio del resultado lógico de 0 a 1 significa que hay un flanco de señal ascendente.
- Si la instrucción detecta un cambio del resultado lógico de 1 a 0 significa que hay un flanco de señal descendente.

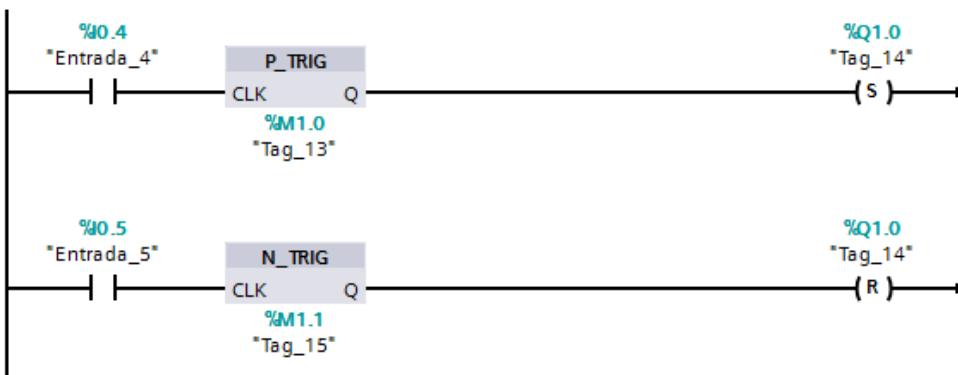


80

▼ Segmento 8: CONSULTA (P_TRIGGER Y N_TRIGGER) FLANcos DE SUBIDA (FLANCO POSITIVO) Y DE BAJADA (FLANCO NEGATIVO)

▼ La instrucción genera flanco de señal ascendente, permite consultar un cambio del estado lógico del resultado lógico de 0 a 1. La instrucción compara es estado lógico actual del resultado lógico con el estado lógico de la consulta anterior, que está guardado en una marca de flancos (M1.0) Si la instrucción detecta un cambio del resultado lógico de 0 a 1 significa que hay un flanco de señal ascendente.

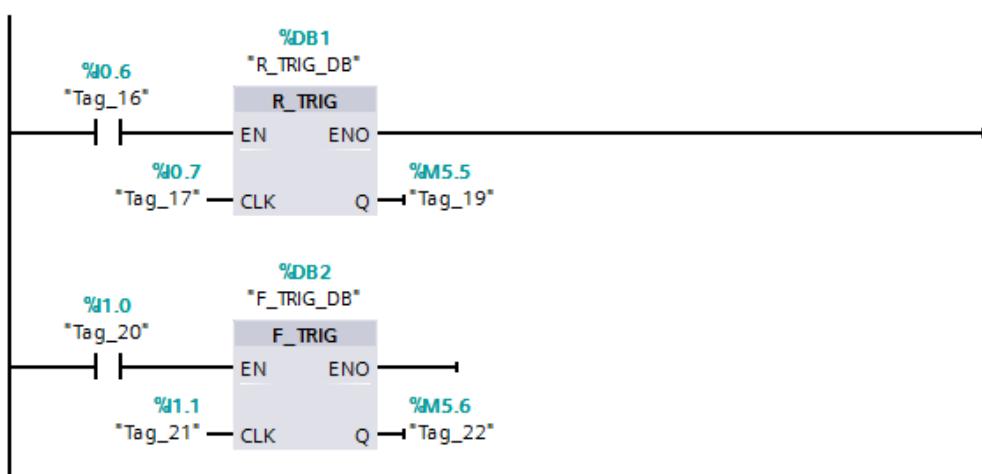
La instrucción generar flanco de señal descendente del resultado lógico permite consultar un cambio del estado lógico del resultado lógico de 1 a 0. La instrucción compara el estado lógico actual del Resultado lógico con el estado lógico de la consulta anterior que está guardado en una marca de flancos (M1.1) Si la instrucción detecta un cambio del Resultado lógico de 1 a 0 significa que hay un flanco de señal descendente.



81

▼ Segmento 9: GENERAR (R_TRIGGER Y F_TRIGGER) FLANcos DE SUBIDA (FLANCO POSITIVO) Y DE BAJADA (FLANCO NEGATIVO)

▼ Las instrucciones detectar flanco de señal ascendente y detectar flanco de señal descendente permite detectar un cambio de estado de 0 a 1 en la entrada CLK. La instrucción compara el valor de entrada actual de la entrada CLK con es estado de consulta anterior (marca de flanco) que esta almacenada en la instancia indicada (BLOQUE DE DATOS). Cuando la instrucción detecta un cambio de estado de 0 a 1 en la entrada CLK en la salida Q se genera un flanco de señal ascendente/descendente es decir que la señal tiene el valor TRUE o 1 exactamente durante un ciclo. En la salida Q no se puede insertar una bobina de asignación, es necesario escribir el dato correspondiente.



82



Circuitos Secuenciales y Realimentación

Visión Práctica S1200

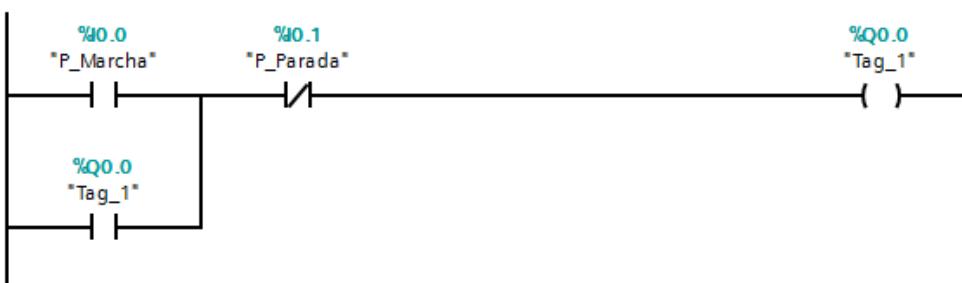
Segmento 1: CIRCUITO AUTOALIMENTADO CON PRIORIDAD A LA PARADA

Segmento 2: CIRCUITO AUTOALIMENTADO CON PRIORIDAD A LA MARCHA

83

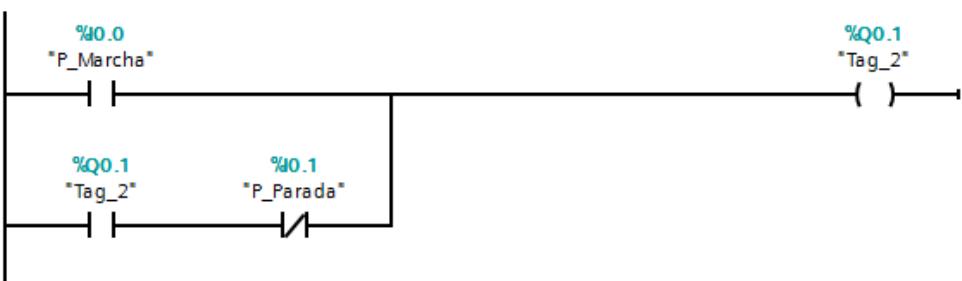
▼ Segmento 1: CIRCUITO AUTOALIMENTADO CON PRIORIDAD A LA PARADA

- En el caso de activar los dos pulsadores a la vez ponemos al sistema en un compromiso ya que le estamos solicitando que se active y que no se active de forma simultánea. En este caso la salida no se activa, decimos que tiene prioridad la parada.



▼ Segmento 2: CIRCUITO AUTOALIMENTADO CON PRIORIDAD A LA MARCHA

- En esta esquema en el caso de activar los dos pulsadores a la vez de forma simultánea ponemos al sistema en un compromiso porque le decimos al PLC que active y desactive la salida Q0.1 a la vez. En este caso la salida se activa y por tanto decimos que prevalece la marcha.



84



Circuitos con Temporizadores

Visión Práctica S1200

Segmento 1: RETARDO A LA CONEXIÓN. TIMER ON DELAY. (TON)

Segmento 2: RETARDO A LA CONEXIÓN CON MEMORIA (TONR)

Segmento 3: RETARDO A LA DESCONEXIÓN (TOF)

Segmento 4: RETARDO TIPO IMPULSO (TP)

Segmento 5: ASOCIACIÓN DE CONTACTOS A UN TEMPORIZADOR

Segmento 6: USO DE LA VARIABLE Q DEL DB A INSTANCIA ASOCIADO AL TEMPORIZADOR

Segmento 7: COMPARAR EL VALOR DE LA SALIDA ET DEL TEMPORIZADOR

Segmento 8: PUESTA A CERO DE UN TEMPORIZADOR

Segmento 9: CARGA DIRECTA DE UN VALOR DE TIEMPO EN TEMPORIZADOR

Segmento 10: MARCAS DEL SISTEMA

85

▼ Segmento 1: RETARDO A LA CONEXIÓN. TIMER ON DELAY. (TON)

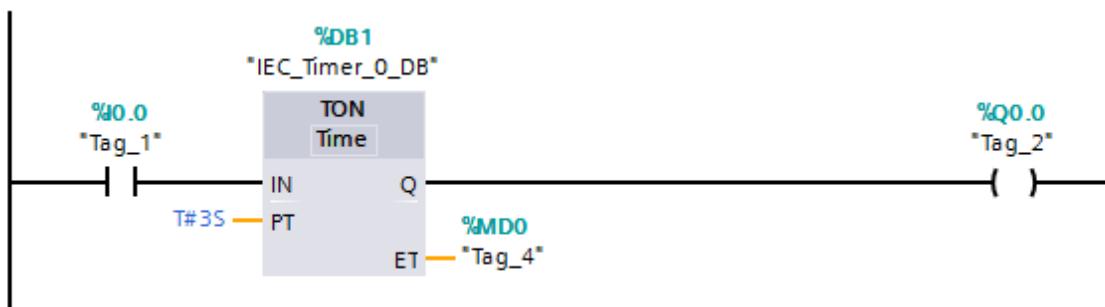
El temporizador cuenta el tiempo que tarda en activarse la salida Q0.0. Una vez activada la entrada de habilitación IN. En este tipo de retardo el valor de conteo actual del temporizador se borra cuando la entrada de habilitación está desactivada, por lo que es necesario mantener siempre la entrada (IN) a 1. El PLC reserva para el temporizador 32 bits para almacenar el valor del conteo.

IN = Entrada de activación

PT = Tiempo de retardo en formato TIME (T#...)

Q = Salida que se activará una vez transcurrido el tiempo de retardo PT

ET = Registro de 32 bits (double word) donde se registra el valor del tiempo transcurrido.



86

▼ Segmento 2: RETARDO A LA CONEXIÓN CON MEMORIA (TONR)

- El temporizador de retardo a la conexión memorizado cuenta también el tiempo que tarda en activarse la salida Q una vez activada la entrada de habilitación, pero a diferencia de lo que ocurre con el temporizador TON en el TONR cuando se desconecta la entrada IN el tiempo no se borra, sino que se recuerda y cuando vuelve a activarse la entrada continúa contando a partir de donde se quedó el conteo. Una vez transcurrido el tiempo programado para el retardo se activará el contacto del temporizador activando en su caso la salida.

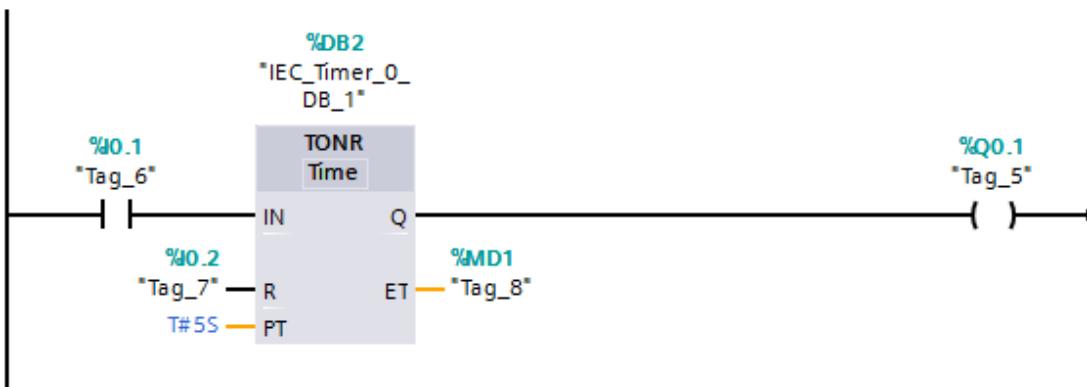
IN = Entrada de activación

R = Entrada de reset

PT = Tiempo de retardo, en formato TIME

Q = Salida que se activará una vez transcurrido el tiempo de retardo PT

ET = Registro de 32 bits (double word) donde se registra el valor del tiempo transcurrido.



87

▼ Segmento 3: RETARDO A LA DESCONEXIÓN (TOF)

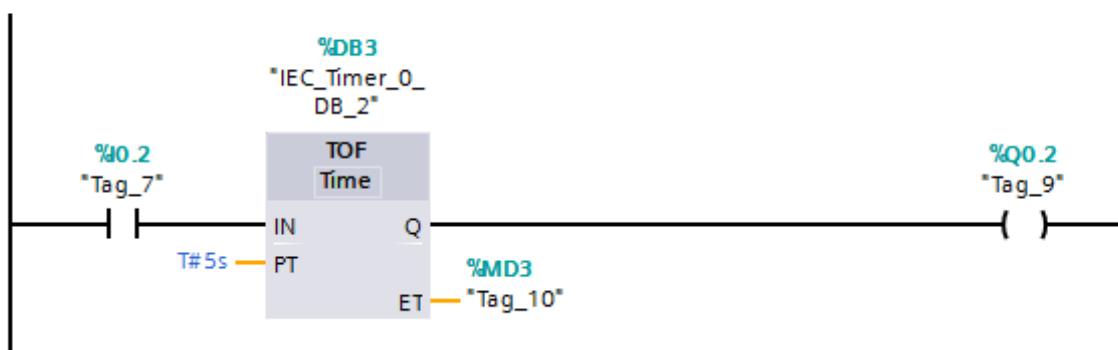
- IN = Entrada de activación

PT = Tiempo de retardo en formato TIME

Q = Salida que se activará una vez transcurrido el tiempo de retardo PT

ET = Registro de 32 bits (double word) donde se registra el valor del tiempo transcurrido.

El temporizador de retardo a la desconexión cuenta el tiempo que tarda en desactivarse la salida Q, una vez desactivada la entrada de habilitación IN. La salida se activa nada más conectar la entrada IN y no se desconecta al desconectar la entrada sino un tiempo después. Una vez transcurrido el tiempo programado para el retardo a la desconexión se desactivará el contacto del temporizador desactivando la salida.

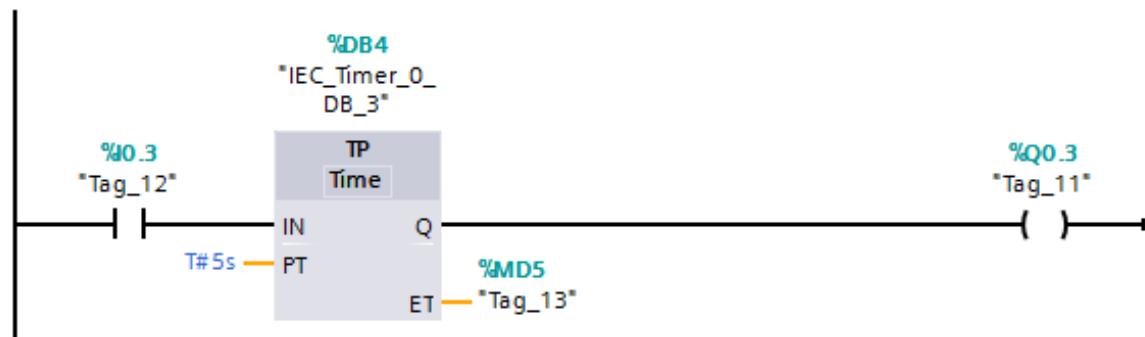


88

Segmento 4: RETARDO TIPO IMPULSO (TP)

- IN = Entrada de activación
- PT = Tiempo de retardo en formato TIME (T#...)
- Q = Salida que se activará una vez transcurrido el tiempo de retardo PT
- ET = Registro de 32 bits (double word) donde se registra el valor del tiempo transcurrido

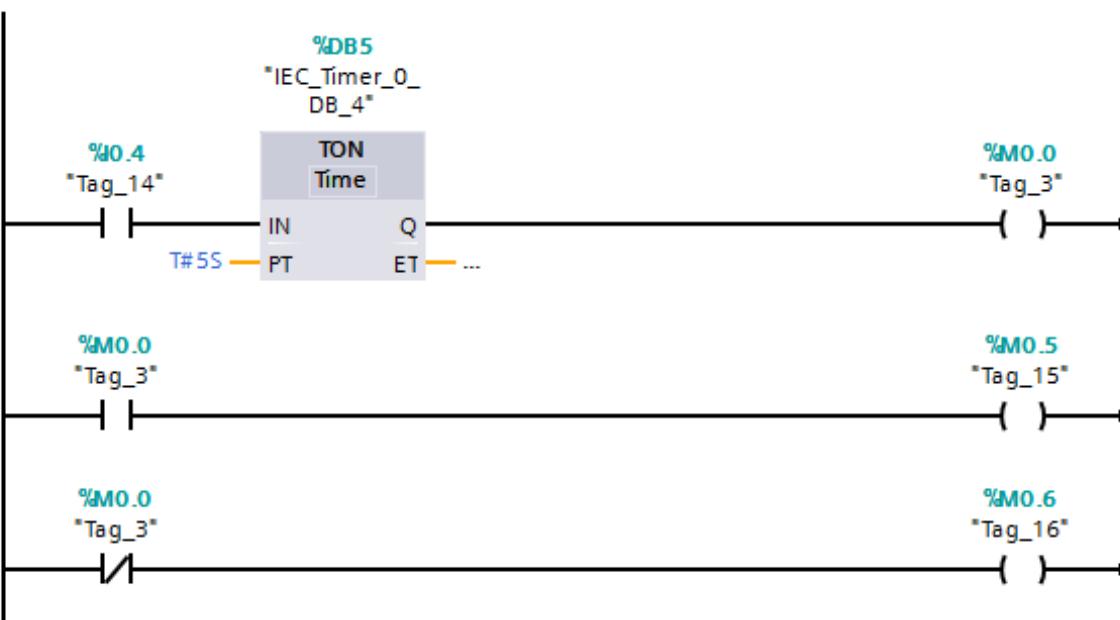
El temporizador de impulso TP cuenta el tiempo que tarda en desactivarse la salida (Q) una vez activada la entrada de habilitación (IN). La salida se activa nada más conectar la entrada IN y se desconecta una vez que haya transcurrido el tiempo t indicado en la entrada (PT). En el caso de desactivar la entrada IN antes de que haya transcurrido el tiempo programado, la salida no se desactivará hasta transcurrido el tiempo programado.



89

Segmento 5: ASOCIACIÓN DE CONTACTOS A UN TEMPORIZADOR

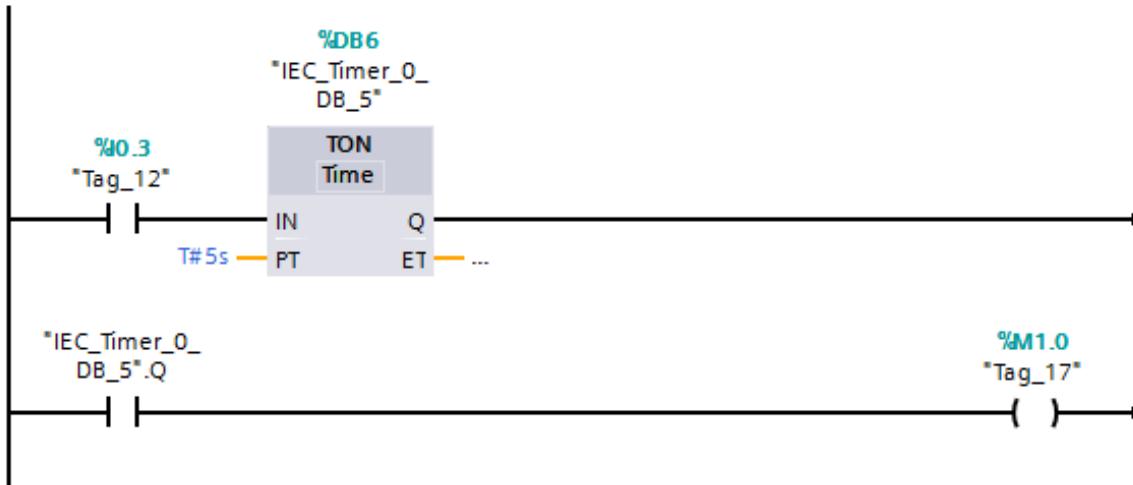
- Cuando transcurrido el tiempo programado se activa la marcha (M0.0) y actúan sus contactos asociados activando o desactivando las salidas o marchas asociadas a los mismos según lo programado.



90

Segmento 6: USO DE LA VARIABLE Q DEL DB A INSTANCIA ASOCIADO AL TEMPORIZADOR

- En este caso no es necesario asignar ninguna bobina a la salida (Q) del temporizador. Sin embargo, los contactos asociados deberán leer el valor de dicha salida directamente del DB asociado. Para ello cada contacto debe tener la siguiente sintaxis: "Nombre_DB".Q, en nuestro caso "IEC_Timer_0_DB_5".Q

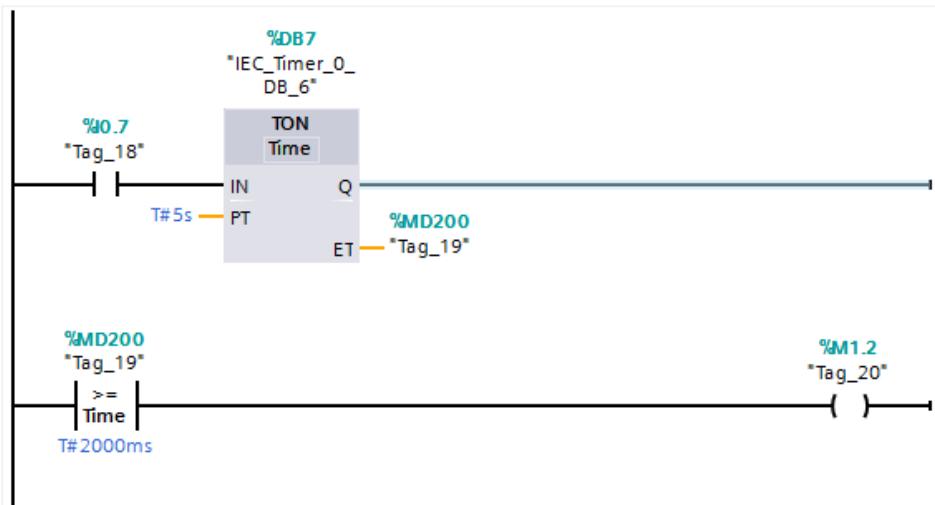


91

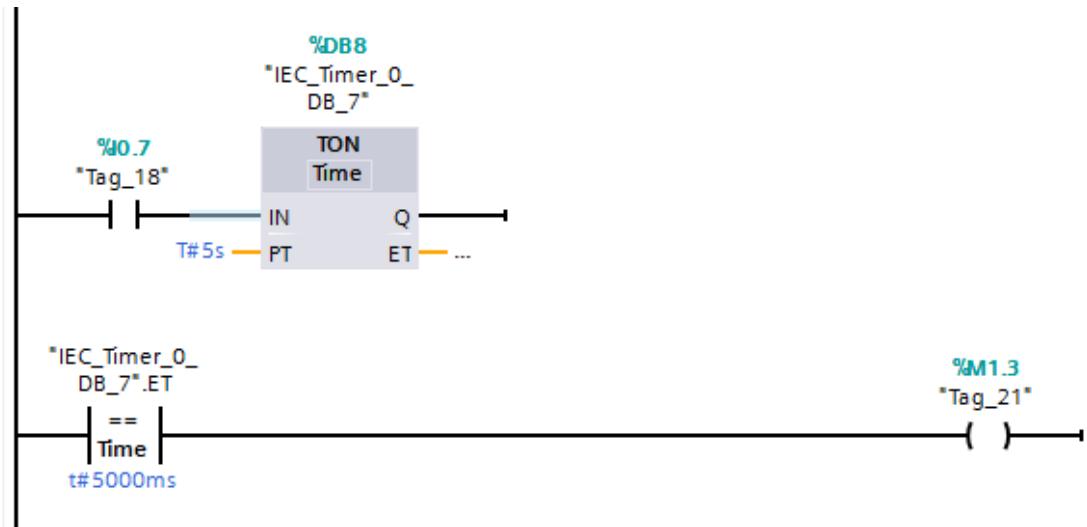
Segmento 7: COMPARAR EL VALOR DE LA SALIDA ET DEL TEMPORIZADOR

- Opción 1: La variable debe tener el formato de doble entero (32 bits) y debe escribirse en la salida (ET) del temporizador. El tiempo se almacena en ella en formato de milisegundos. Por ello en las operaciones de comparación pueden hacerse éstas en milisegundos o en cualquier otra unidad del formato time. En el ejemplo se muestra como la marcha (M1.2) recibe un nivel alto cuando el tiempo es igual o superior a dos segundos. Al estar en formato tiempo se puede escribir indistintamente T#2s o T#2000ms.

Opción 2: De igual forma que para la salida (Q) del DB a instancia, es posible leer la salida ET (que tiene formato doble entero, DInt - 32 bits) y operar con ella en las comparaciones sin necesidad de crear ninguna variable intermedia (MD200).



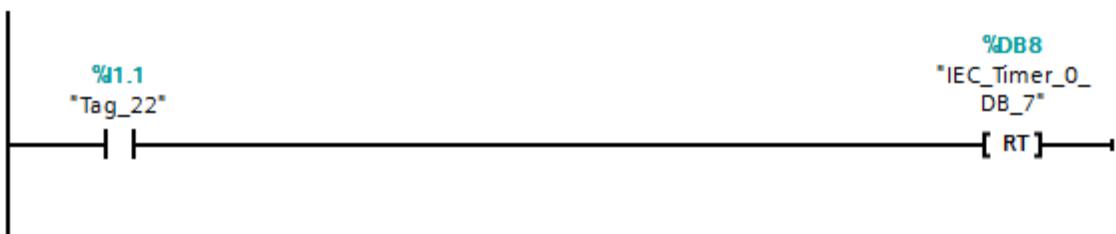
92



93

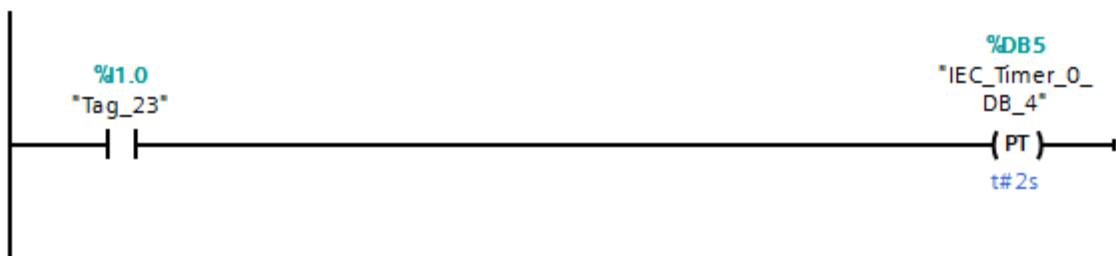
Segmento 8: PUESTA A CERO DE UN TEMPORIZADOR

- Cuando sea necesaria la puesta a cero de algún temporizador, se puede hacer mediante su entrada de Reset si el dispositivo lo tiene, ya que no todos los temporizadores tienen la entrada de Reset (R) o utilizando la instrucción [R].



Segmento 9: CARGA DIRECTA DE UN VALOR DE TIEMPO EN TEMPORIZADOR

- Cuando sea necesario cargar con un determinado valor un temporizador se puede hacer utilizando la instrucción [PT], como se indica en la figura.



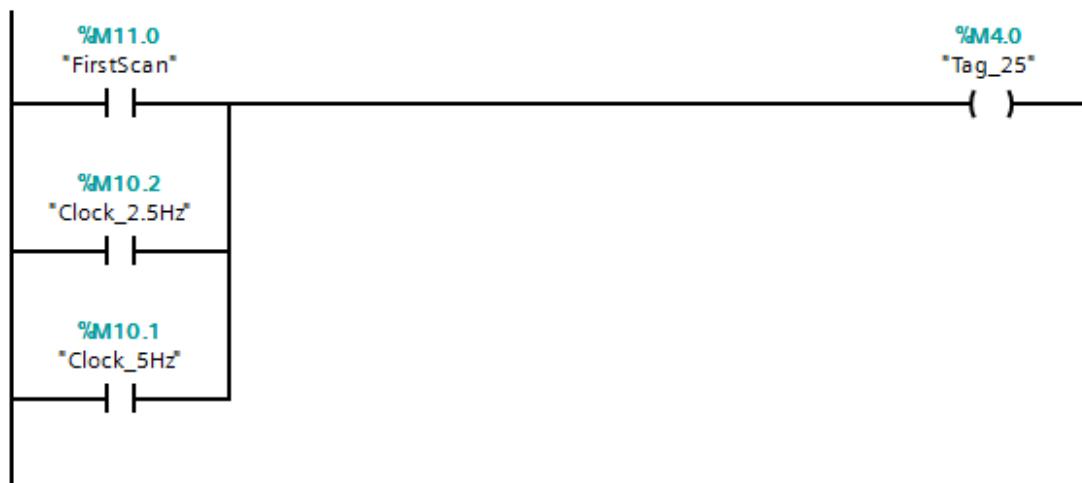
94

▼ Segmento 10: MARCAS DEL SISTEMA

► Son marcas programadas para realizar una función específica, debemos asignar el Byte de Marcas que alojará a estas marcas. Así en el ejemplo se asigna el Byte nº 11 de marcas del sistema y el byte nº 10 para marcas de ciclo de reloj.

La marca (M11.0) se pondrá a nivel lógico solo durante el primer ciclo de scan, después la marca adquiere el valor lógico bajo.

Las marcas (M10.2 y M10.1) son marcas de ciclo de reloj las cuales aportan señales cuadradas de una frecuencia determinada 2,5 Hz y 5 Hz. Hay otras para distintas configuraciones. Estas 8 marcas una vez activadas realizarán de forma continua una intermitencia a la frecuencia que corresponda a cada una de ellas.



95



Circuitos con Contadores

Visión Práctica S1200

Segmento 1: CONTADOR ASCENDENTE / UP CTU

Segmento 2: CONTADOR DESCENDENTE / DOWN CTD

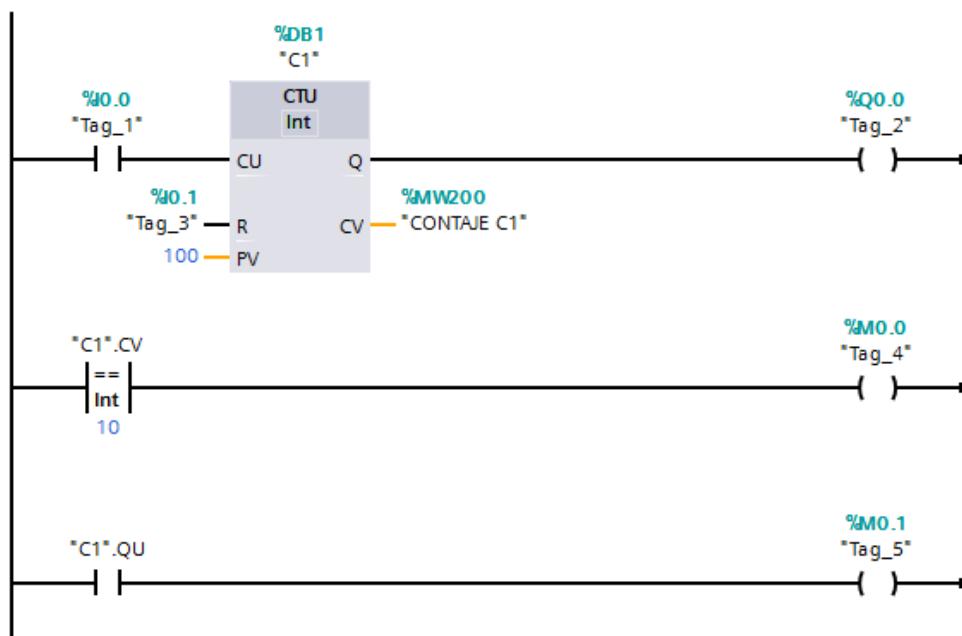
Segmento 3: CONTADOR ASCENDENTE-DESCENDENTE / CTUD

96

▼ Segmento 1: CONTADOR ASCENDENTE / UP CTU

- El contador ascendente CTU empieza a contar adelante a partir del valor actual cuando se produce un flanco positivo en la entrada de contaje adelante (CU). El contador se inicializa y se pone a cero cuando se activa la entrada de desactivación (R). La salida (Q) se activa cuando el contador es igual o superior al valor seleccionado en (PV). En la salida (CV) disponemos de un registro con los datos actuales del contaje.

Se pueden utilizar comparaciones con el valor de contaje CV.
Igualmente se puede utilizar el dato de salida (Q) en otros segmentos.

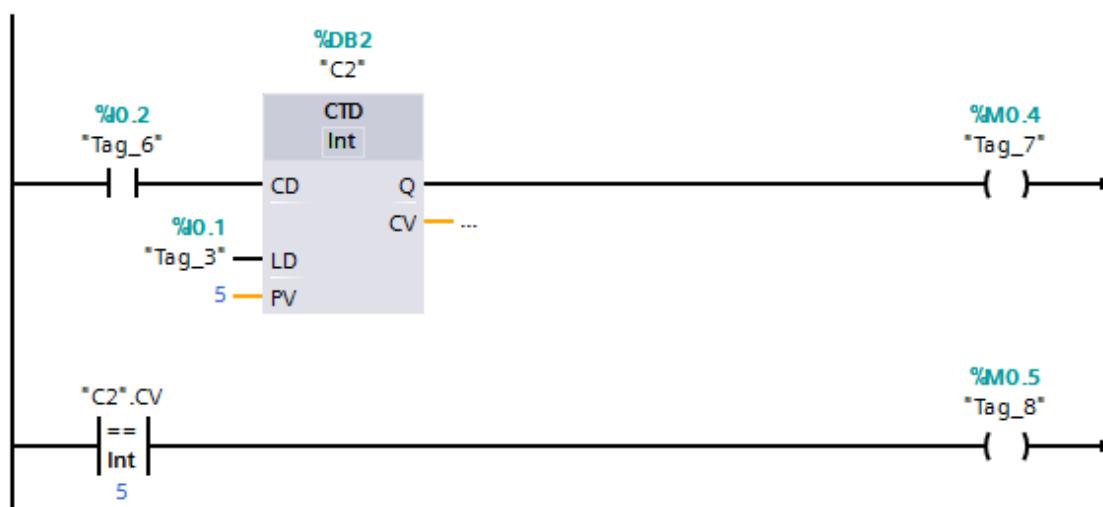


97

▼ Segmento 2: CONTADOR DESCENDENTE / DOWN CTD

- El contador descendente CTD empieza a descontar a partir del valor actual cuando se produce un flanco positivo en la entrada de contaje hacia atrás (CD). El contador se posiciona en el valor prefijado (PV) al activar la entrada (LD) carga. La salida (Q) se activa cuando el valor de contaje es igual o inferior a cero. En la salida CV disponemos de un registro con los datos actuales del contaje. Igualmente se puede utilizar el dato C2. CV para realizar comparaciones.

En el ejemplo siguiente la salida Q se activa cuando el valor del contaje es igual o inferior a CERO poniendo a nivel alto la marca (M0.4) . (Puede contar en negativo). La marca (M0.5) se activa cuando CV= -5

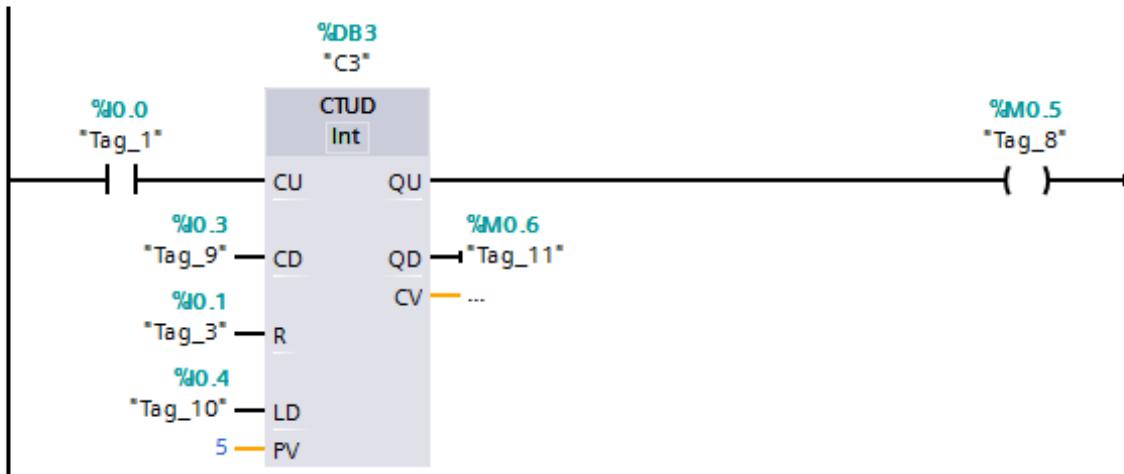


98

▼ Segmento 3: CONTADOR ASCENDENTE-DESCENDENTE / CTUD

- El contador UP/DOWN surge de la integración de los dos contadores anteriores, así dispone de entradas de conteo ascendente CU y de conteo descendente CD. También dispone de entrada de reset y de carga LD para posiciones al contador en el valor indicado en PV.
- La salida QU se activa cuando el contador sea mayor o igual al dato PV.
- La salida QD se activa cuando el contador sea menor o igual a cero.
- En la salida CV disponemos de un registro con los datos actuales del conteo que se puede utilizar para realizar comparaciones.

En el ejemplo siguiente la salida QU se activa cuando el valor de conteo es igual o superior a 5. La salida QD se activa cuando CV=0 o negativo.



99



Operaciones de comparación Visión Práctica S1200

Segmento 1:	IGUAL
Segmento 2:	DIFERENTE
Segmento 3:	MAYOR O IGUAL
Segmento 4:	MENOR O IGUAL
Segmento 5:	MAYOR
Segmento 6:	MENOR
Segmento 7:	DENTRO DE RANGO (IN_RANGE)
Segmento 8:	FUERA DE RANGO (OUT_RANGE)

100

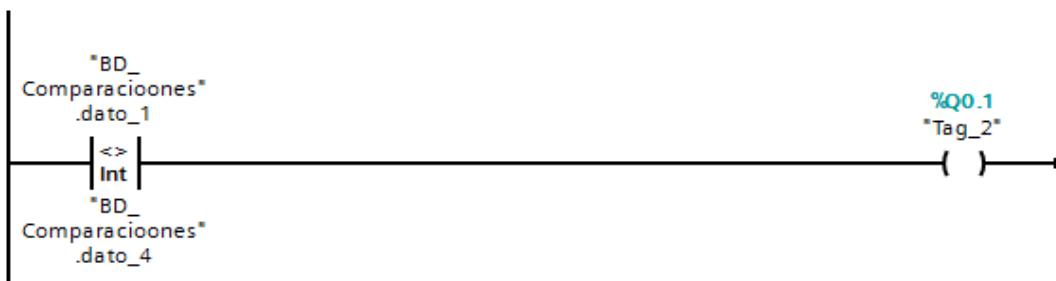
▼ Segmento 1: IGUAL

- La instrucción "Igual" permite consultar si son iguales el primer (<Operando1>) y segundo (<Operando2>) valor de comparación. Si se cumple la condición de la comparación, la instrucción devuelve el resultado lógico (RLO) "1". Si la condición de la comparación no se cumple, la instrucción devuelve el RLO "0".



▼ Segmento 2: DIFERENTE

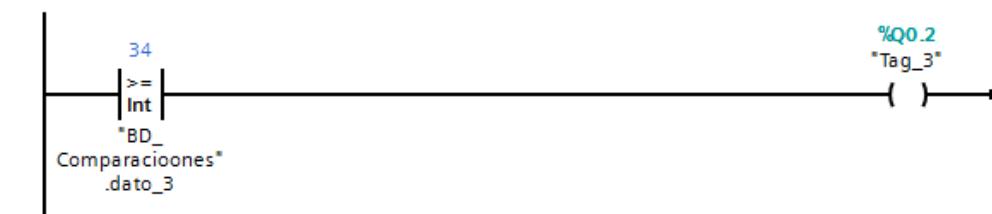
- La instrucción "Diferente" permite consultar si son diferentes el primer (<Operando1>) y segundo (<Operando2>) valor de comparación. Si se cumple la condición de la comparación, la instrucción devuelve el resultado lógico (RLO) "1". Si la condición de la comparación no se cumple, la instrucción devuelve el RLO "0".



101

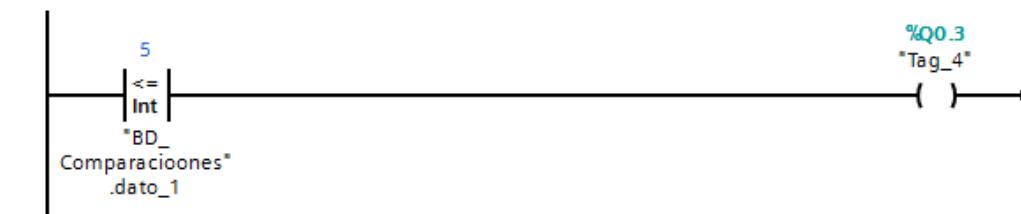
▼ Segmento 3: MAYOR O IGUAL

- La instrucción "Mayor o igual" permite consultar si el primer valor de comparación (<Operando1>) es mayor o igual que el segundo (<Operando2>). Ambos valores de comparación deben ser del mismo tipo de datos. Si se cumple la condición de la comparación, la instrucción devuelve el resultado lógico (RLO) "1". Si la condición de la comparación no se cumple, la instrucción devuelve el RLO "0".



▼ Segmento 4: MENOR O IGUAL

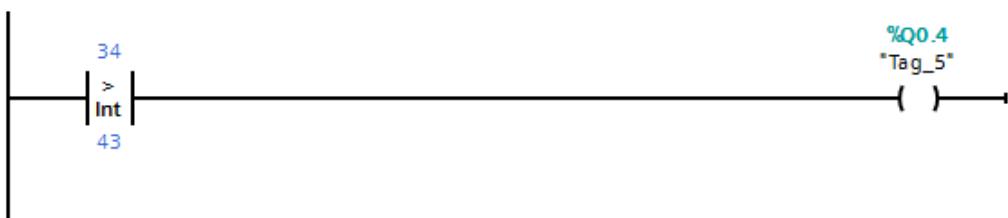
- La instrucción "Menor o igual" permite consultar si el primer valor de comparación (<Operando1>) es menor o igual que el segundo (<Operando2>). Ambos valores de comparación deben ser del mismo tipo de datos. Si se cumple la condición de la comparación, la instrucción devuelve el resultado lógico (RLO) "1". Si la condición de la comparación no se cumple, la instrucción devuelve el RLO "0".



102

▼ Segmento 5: MAYOR

- ▼ La instrucción "Mayor" permite consultar si el primer valor de comparación (<Operando1>) es mayor que el segundo (<Operando2>). Ambos valores de comparación deben ser del mismo tipo de datos. Si se cumple la condición de la comparación, la instrucción devuelve el resultado lógico (RLO) "1". Si la condición de la comparación no se cumple, la instrucción devuelve el RLO "0".



▼ Segmento 6: MENOR

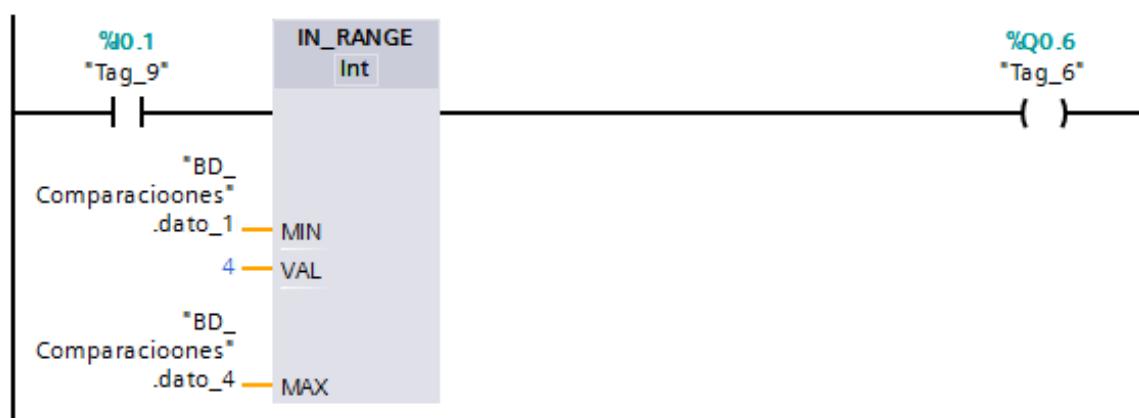
- ▼ La instrucción "Menor" permite consultar si el primer valor de comparación (<Operando1>) es menor que el segundo (<Operando2>). Ambos valores de comparación deben ser del mismo tipo de datos. Si se cumple la condición de la comparación, la instrucción devuelve el resultado lógico (RLO) "1". Si la condición de la comparación no se cumple, la instrucción devuelve el RLO "0".



103

▼ Segmento 7: DENTRO DE RANGO (IN_RANGE)

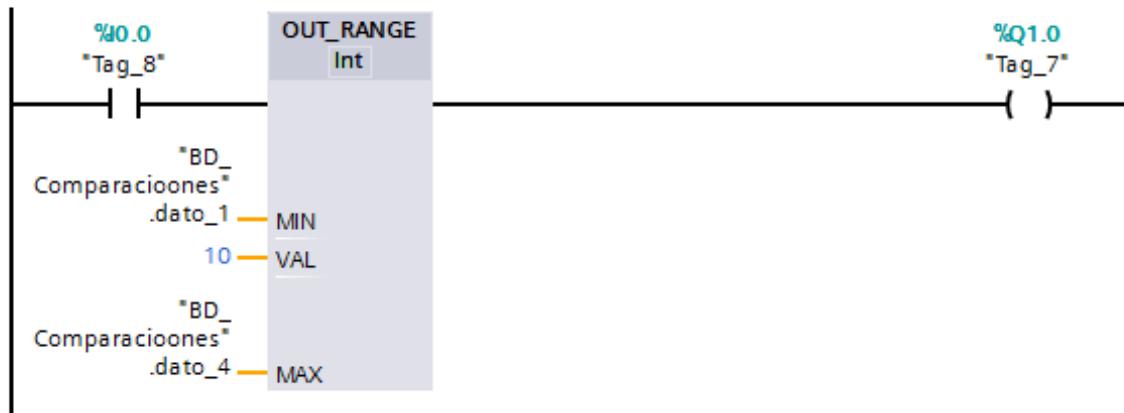
- ▼ La instrucción "Valor dentro del rango" permite consultar si el valor de la entrada VAL está dentro de un determinado rango de valores. Los límites del rango de valores se determinan mediante las entradas MIN y MAX. La instrucción "Valor dentro del rango" compara el valor de la entrada VAL con los valores de las entradas MIN y MAX y transfiere el resultado a la salida del cuadro. Si el valor de la entrada VAL cumple la comparación MIN <= VAL o VAL <= MAX, la salida del cuadro devuelve el estado lógico "1". Si no se cumple la comparación, la salida del cuadro devuelve el estado lógico "0". Si el estado lógico de la entrada del cuadro es "0", no se ejecuta la instrucción "Valor dentro del rango".



104

▼ Segmento 8: FUERA DE RANGO (OUT_RANGE)

- La instrucción "Valor fuera del rango" permite consultar si el valor de la entrada VAL está fuera de un determinado rango de valores. Los límites del rango de valores se determinan mediante las entradas MIN y MAX. La instrucción "Valor fuera del rango" compara el valor de la entrada VAL con los valores de las entradas MIN y MAX y transfiere el resultado a la salida del cuadro. Si el valor de la entrada VAL cumple la comparación MIN > VAL o VAL > MAX, la salida del cuadro devuelve el estado lógico "1". La salida del cuadro también devuelve el estado lógico "1" si un operando indicado del tipo de datos REAL contiene un valor no válido.
- La salida del cuadro devuelve el estado lógico "0" si el valor de la entrada VAL no cumple la condición MIN > VAL o VAL > MAX .
- Si el estado lógico de la entrada del cuadro es "0", no se ejecuta la instrucción "Valor fuera del rango".



105



Funciones Matemáticas Visión Práctica S1200

Segmento 1: SUMA ADICIÓN (ADD)

Segmento 2: RESTA SUSTRACCIÓN (SUB)

Segmento 3: MULTIPLICACIÓN (MUL)

Segmento 4: DIVISIÓN (DIV)

Segmento 5: RESTO DE UNA DIVISIÓN (MOD)

Segmento 6: FUNCIÓN CALCULAR (CALCULATE)

Segmento 7: COMPLEMENTO A DOS NEGATIVO (NEG)

Segmento 8: INCREMENTAR EN 1 (INC)

Segmento 9: DECREMENTAR EN 1 (DEC)

Segmento 10: VALOR ABSOLUTO (ABS)

Segmento 11: DETERMINAR EL VALOR MÍNIMO (MIN)

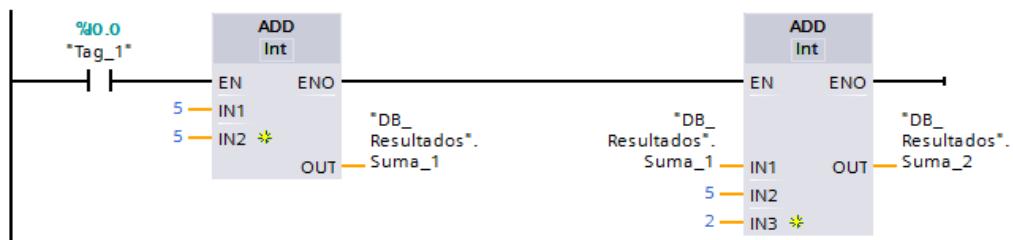
Segmento 12: DETERMINAR EL VALOR MÁXIMO (MAX)

Segmento 13: AJUSTAR EL VALOR LÍMITE (LIMIT)

106

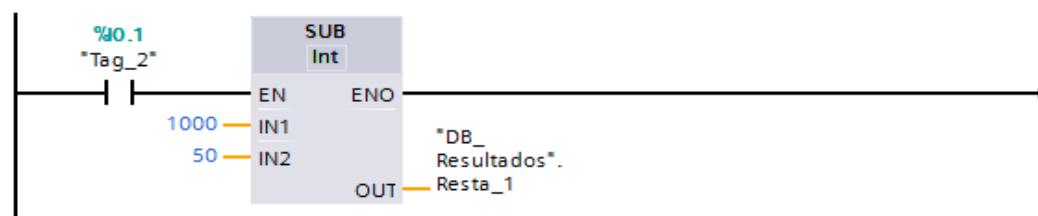
Segmento 1: SUMA ADICIÓN (ADD)

- Cuando la entrada (I0.0) está a nivel alto entonces se efectúa las sumas programadas.
 - 1.- IN1 + IN2 = OUT
 - 2.- IN1 + IN2 + IN3 = OUT
 Es posible insertar entradas nuevas clicando en el punto amarillo de la caja.
 La salida de habilitación ENO devuelve un "0" cuando se cumple una de las siguientes condiciones: 1.- La entrada de habilitación EN está a "0". 2.- El resultado de la instrucción está fuera de rango permitido para el tipo de datos indicado en la salida OUT. 3.- Un número en coma flotante tiene un valor no válido.



Segmento 2: RESTA SUSTRACCIÓN (SUB)

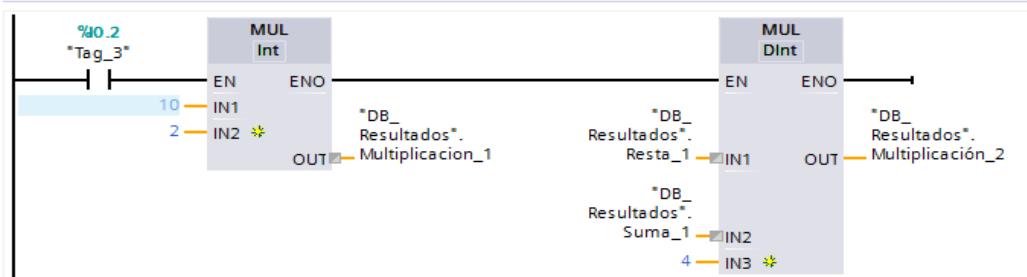
- Cuando (I0.1) es valor lógico alto entonces $IN1 - IN2 = OUT$. El resultado se almacena en el bloque de datos en la variable (Resta_1). Podemos efectuar operaciones de resta aplicando variables de diferentes bloques de datos, variables de contadores, de temporizadores etc...
 La salida de habilitación ENO devuelve un "0" cuando se cumple una de las siguientes condiciones:
 - 1.- La entrada de habilitación EN está a "0". 2.- El resultado de la instrucción está fuera de rango permitido para el tipo de datos indicado en la salida OUT. 3.- Un número en coma flotante tiene un valor no válido.



107

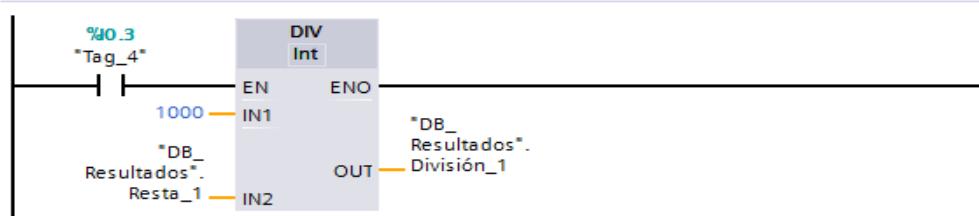
Segmento 3: MULTIPLICACIÓN (MUL)

- La instrucción permite multiplicar el valor de las entradas (IN...) entre si y el resultado lo almacena en la posición que se indica en la salida (OUT). En el ejemplo cuando la entrada está a nivel lógico 1 se efectúan las multiplicaciones quedando los resultados en las posiciones que se indican en las salidas (OUT).
 La salida de habilitación ENO devuelve un "0" cuando se cumple una de las siguientes condiciones: 1.- La entrada de habilitación EN está a "0". 2.- El resultado de la instrucción está fuera de rango permitido para el tipo de datos indicado en la salida OUT. 3.- Un número en coma flotante tiene un valor no válido.



Segmento 4: DIVISIÓN (DIV)

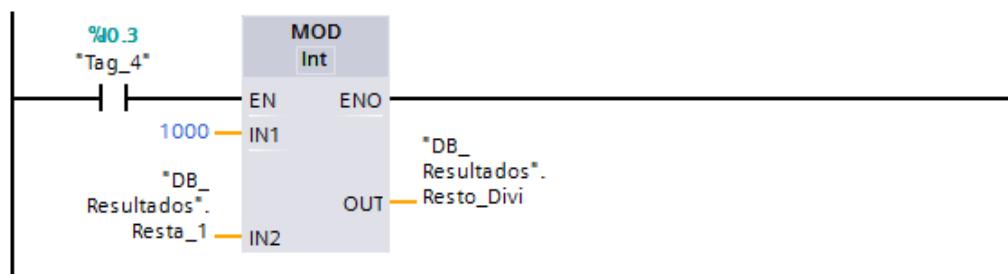
- La instrucción permite dividir el valor de la entrada IN1 por el valor de la entrada IN2 y consultar el cociente en la salida (OUT). En el ejemplo cuando (I0.3)=1 entonces $OUT=IN1/IN2$.
 Importante tener en cuenta el tipo de dato que vamos a trabajar para configurar el dato de la función correctamente. Tendremos que ver el valor de los datos para considerarlos de 8 bits o de 32 bits...
 La salida de habilitación ENO devuelve un "0" cuando se cumple una de las siguientes condiciones:
 - 1.- La entrada de habilitación EN está a "0". 2.- El resultado de la instrucción está fuera de rango permitido para el tipo de datos indicado en la salida OUT. 3.- Un número en coma flotante tiene un valor no válido.



108

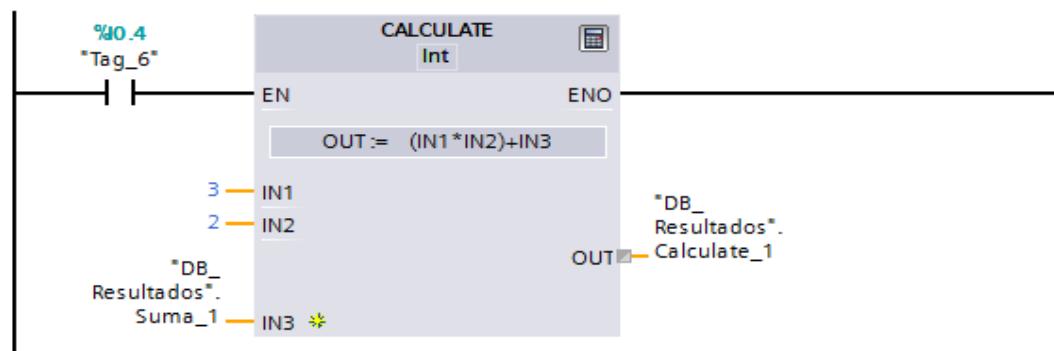
▼ Segmento 5: RESTO DE UNA DIVISIÓN (MOD)

- La instrucción permite dividir el valor de la entrada IN1 por el valor de la entrada IN2 y consultar el resto de la operación en la salida (OUT).



▼ Segmento 6: FUNCIÓN CALCULAR (CALCULATE)

- La instrucción permite definir y ejecutar una expresión para calcular operaciones matemáticas o combinaciones lógicas complejas en función del tipo de datos seleccionado. El cuadro de instrucción contiene el estado básico de dos entradas. El número de entradas es ampliable. Las entradas se enumeran en orden ascendente en el cuadro.



109

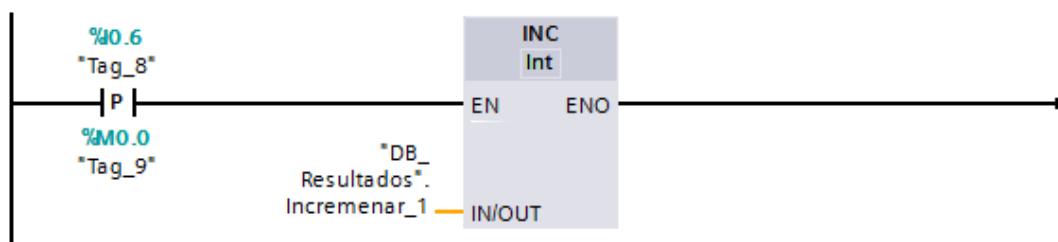
▼ Segmento 7: COMPLEMENTO A DOS NEGATIVO (NEG)

- La instrucción genera el complemento a dos, permite cambiar el signo del valor de la entrada IN y consultar el valor en la salida OUT. Si la entrada IN tiene un valor positivo se deposita el equivalente negativo de este valor en la salida OUT.



▼ Segmento 8: INCREMENTAR EN 1 (INC)

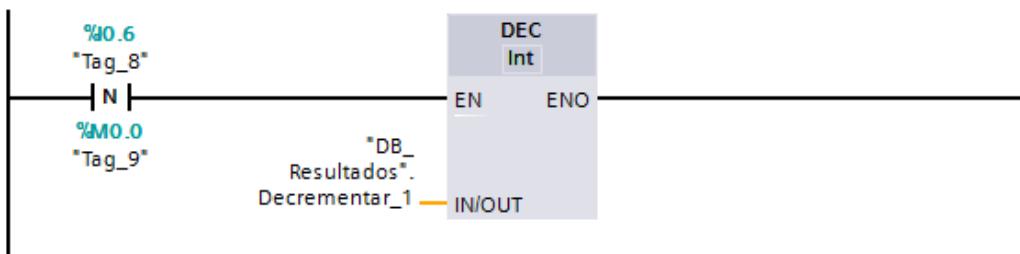
- Aumenta en uno el valor del operando del parámetro IN/OUT. En nuestro ejemplo cada flanco positivo en (I0.6) la variable incrementar_1 contenida en el bloque de datos DB_Resultados se incremente en un. Conviene asignar flanco por impulsos para incrementar, para que el incremento no se haga a la velocidad de ciclo de scand.



110

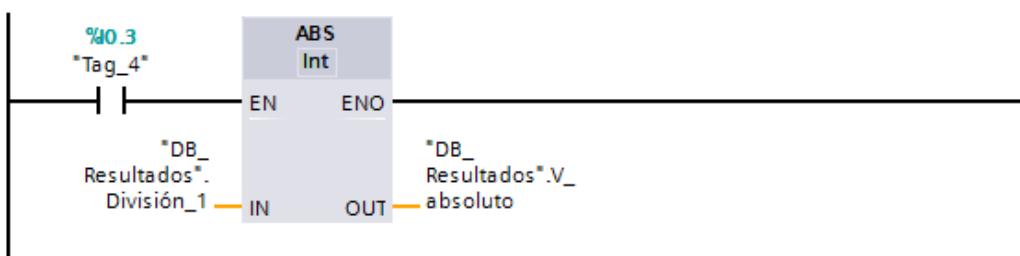
▼ Segmento 9: DECREMENTAR EN 1 (DEC)

- ▼ Decrementa uno el valor del operando del parametro IN/OUT. En nuestro ejemplo cada flanco negativo en (I0.6) la variable decrementar_1 contenida en el bloque de datos DB_Resultados se decremente en uno. Conviene asignar flanco por impulsos para incrementar, para que el incremento no se haga a la velocidad de ciclo de scand.



▼ Segmento 10: VALOR ABSOLUTO (ABS)

- ▼ La instrucción permite calcular el valor absoluto del valor indicado en la entrada IN. En nuestro ejemplo hemos utilizado el resultado de una división.



111

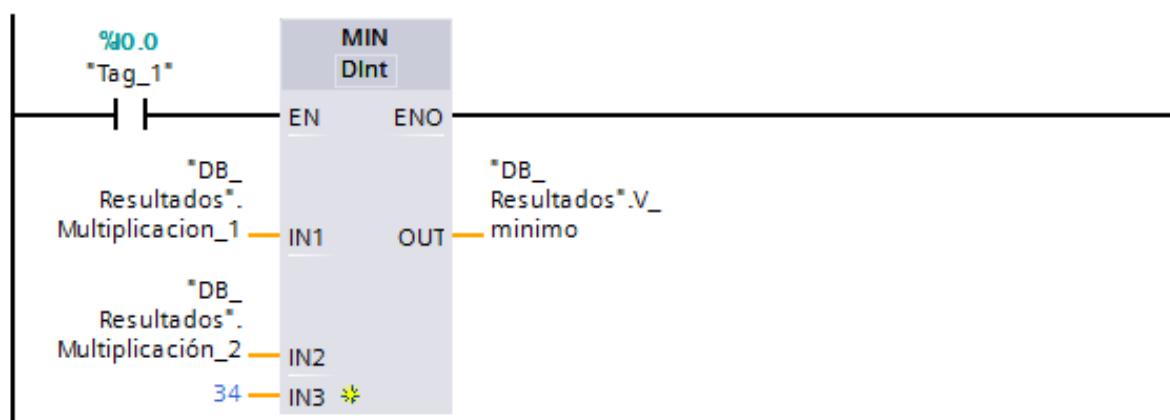
▼ Segmento 11: DETERMINAR EL VALOR MINIMO (MIN)

- ▼ La instrucción "Determinar mínimo" compara los valores de las entradas disponibles y escribe el valor menor en la salida OUT. El número de entradas se puede ampliar en el cuadro de la instrucción mediante entradas adicionales. Las entradas se numeran de forma ascendente en el cuadro. Para la ejecución de la instrucción se deben indicar como mínimo dos valores de entrada y como máximo 100. La salida de habilitación ENO devuelve el estado lógico "0" cuando se cumple una de las condiciones siguientes:

La entrada de habilitación EN devuelve el estado lógico "0".

La conversión implícita de los tipos de datos falla durante la ejecución de la instrucción.

Un número en coma flotante tiene un valor no válido.

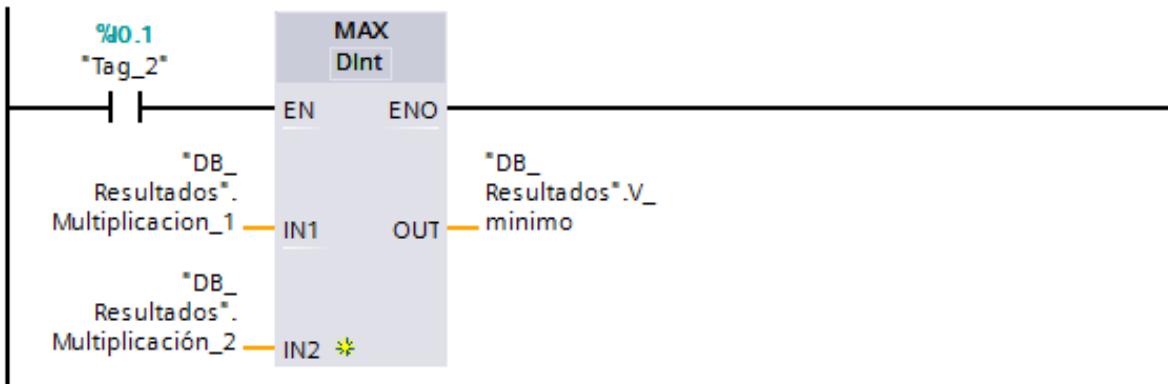


112

▼ Segmento 12: DETERMINAR EL VALOR MÁXIMO (MAX)

- ▼ La instrucción "Determinar máximo" compara los valores de las entradas disponibles y escribe el valor mayor en la salida OUT. El número de entradas se puede ampliar en el cuadro de la instrucción mediante entradas adicionales. Las entradas se numeran de forma ascendente en el cuadro.

Para la ejecución de la instrucción se deben indicar como mínimo dos valores de entrada y como máximo 100. En la lista desplegable "???" del cuadro de la instrucción se puede seleccionar el tipo de datos de la instrucción.

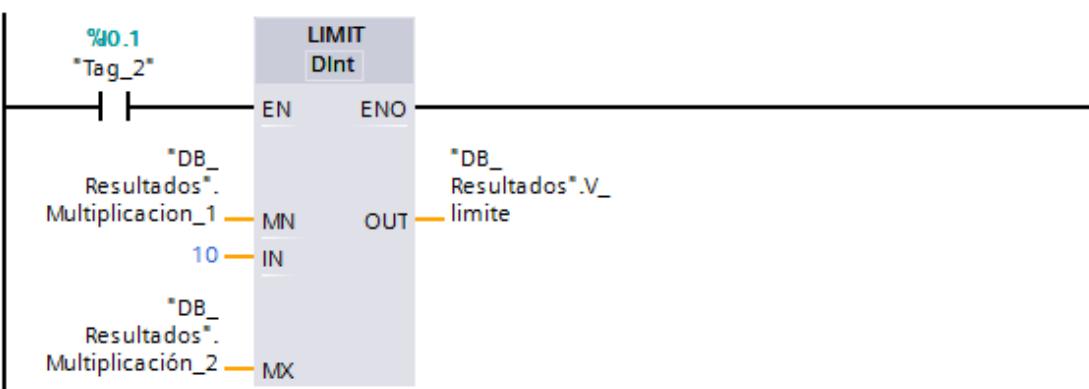


113

▼ Segmento 13: AJUSTAR EL VALOR LÍMITE (LIMIT)

- ▼ La instrucción "Ajustar valor límite" limita el valor en la entrada IN a los valores de las entradas MN y MX. Si el valor de la entrada IN cumple la condición $MN \leq IN \leq MX$, se copia en la salida OUT. Si no se cumple la condición y el valor de entrada IN es menor que el límite inferior MN, la salida OUT adopta el valor de la entrada MN. Si el límite superior MX se rebasa por exceso, la salida OUT adopta el valor de la entrada MX.

Si el valor de la entrada MN es mayor que el de la entrada MX, el resultado no se define y la salida de habilitación ENO es "0". En la lista desplegable "<??>" del cuadro de la instrucción se puede seleccionar el tipo de datos de la instrucción.



114

OTRAS FUNCIONES MATEMÁTICAS

	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

115



Transferencia y Copia de Datos Visión Práctica S1200

Segmento 1: TRANSFERENCIA DE DATOS. COPIA DE DATOS. (MOVE)

Segmento 2: COPIAR ÁREA (MOVE_BLK)

Segmento 3: COPIAR ÁREA SIN INTERRUPCIONES (UMOVE_BLK)

Segmento 4: RELLENAR ÁREA (FILL_BLK)

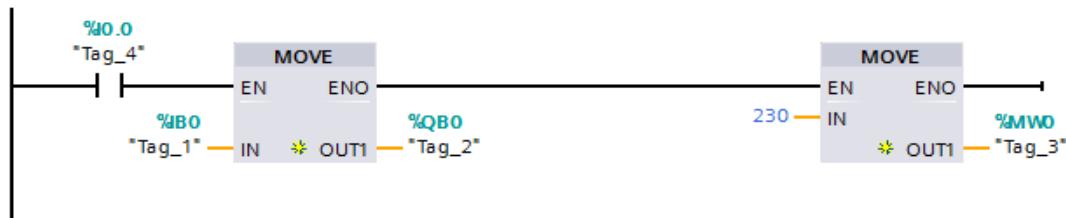
Segmento 5: RELLENAR ÁREA SIN INTERRUPCIONES (UFILL_BLK)

116

Segmento 1: TRANSFERENCIA DE DATOS. COPIA DE DATOS. (MOVE)

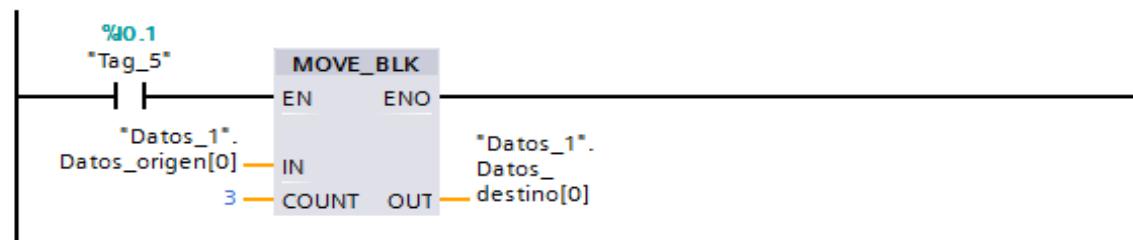
- La instrucción Copiar Valor (MOVE) transfiere el contenido del operando de la entrada (IN) al operando de la salida (OUT1). La transferencia se efectúa siempre por orden ascendente de direcciones. En el ejemplo se transfiere lo que tenemos en el Byte 0 de entradas al Byte 0 de salidas, y el valor 230 a la posición MWO.

Igualmente podemos hacer transferencias de valores de tiempo de temporizadores a una variable y del valor de conteo de un contador a una variable...



Segmento 2: COPIAR ÁREA (MOVE_BLK)

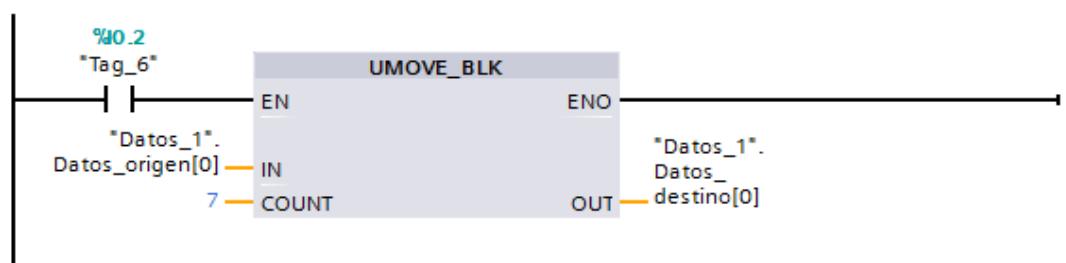
- en un bloque DB creamos dos variables estáticas Array de 10 datos en formato entero, una será Datos_Origen y la otra Datos_destino. En un segmento del mismo insertamos la instrucción trasferir bloque MOVE_BLK. La instrucción copiar área permite copiar los contenidos de un área de memoria (área de origen) en una área de memoria diferente (área destino). El número de elementos que se copian en el área de destino se determina con la entrada COUNT. El ancho del elemento de la entrada IN define el ancho de los elementos que deben copiarse. Al ejecutar la instrucción los 3 primeros elementos del Array Datos_Origen se transfieren al Array destino Datos_destino.



117

Segmento 3: COPIAR ÁREA SIN INTERRUPCIONES (UMOVE_BLK)

- La instrucción Copiar área sinb interrupciones copia sin interrupción los contenidos de un área de memoria (área de origen) en un área de memoria diferente (área destino). El número de elementos que se copian en el área de destino se determina con el parámetro COUNT. El ancho del elemento de la entrada IN define el ancho de los elementos que se deben copiar. Para poder ejecutar la instrucción el área de origen y de destino deben ser del mismo tipo de datos. La operación de copia no debe ser interrumpida por otras actividades del sistema operativo. Por este motivo los tiempos de reacción a alarmas de la CPU podrían aumentar al ejecutar la instrucción copair área sin interrupciones.



Segmento 4: RELLENAR ÁREA (FILL_BLK)

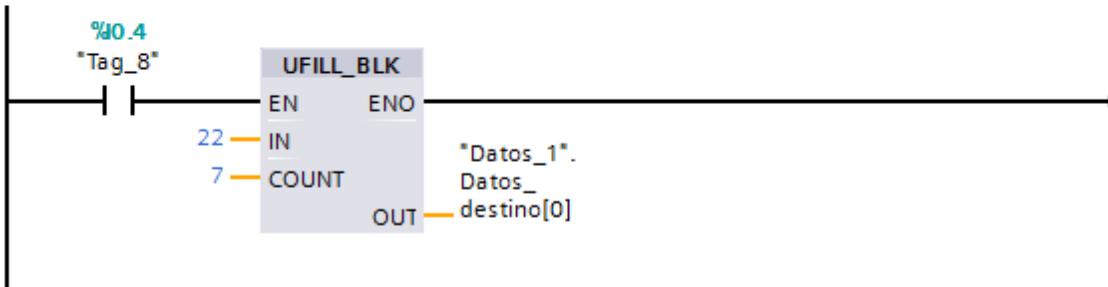
- La instrucción permite llenar un área de memoria con un valor de la entrada IN. El área de destino se llena a partir de la dirección indicada en la salida OUT. El número de repeticiones de copia se determina mediante el parámetro COUNT. Al ejecutar la instrucción el valor de la entrada IN (22) se copia en el área de destino el número de veces especificado en la entrada COUNT.



118

▼ Segmento 5: RELLENAR ÁREA SIN INTERRUPCIONES (UFILL_BLK)

- La instrucción permite llenar un área de memoria con un valor de la entrada IN. El área de destino se rellena a partir de la dirección indicada en la salida OUT. El número de repeticiones de copia se determina mediante el parámetro COUNT. Al ejecutar la instrucción el valor de la entrada IN (22) se copia en el área de destino el número de veces especificado en la entrada COUNT.
La operación de copia no debe ser interrumpida por otras actividades del sistema operativo. Por este motivo los tiempos de reacción de alarmas de la CPU podrán aumentar al ejecutar la instrucción.



119



Rotación y Desplazamiento Visión Práctica S1200

Segmento 1: ROTACIÓN A LA IZQUIERDA (ROL)

Segmento 2: ROTACIÓN A LA DERECHA (ROR)

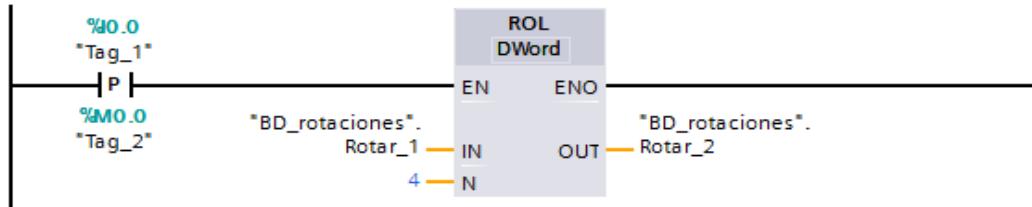
Segmento 3: DESPLAZAMIENTO HACIA LA IZQUIERDA (SHL)

Segmento 4: DESPLAZAMIENTO HACIA LA DERECHA (SHR)

120

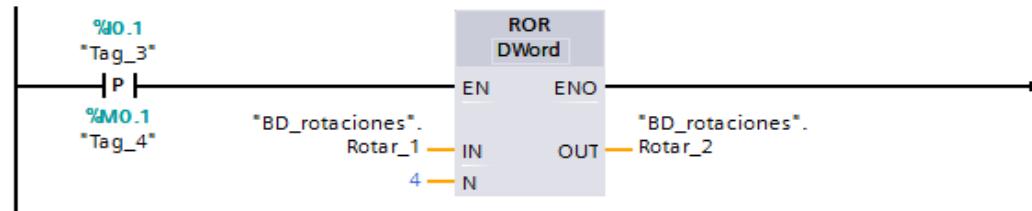
Segmento 1: ROTACIÓN A LA IZQUIERDA (ROL)

- Permite rotar el contenido del operando de la entrada IN de bit en bit hacia la izquierda y consultar el resultado en la salida OUT. El parámetro N determina el número de bits que deben rotarse. Los bits que quedan libres al realizar la rotación se rellenan con los bits desplazados hacia fuera. Es importante que el elemento que proporciona la orden para que se ejecute la rotación o el desplazamiento lo ejecute con su flanco positivo para evitar que la acción se efectúe a la velocidad de ciclo miles de ves por segundo.

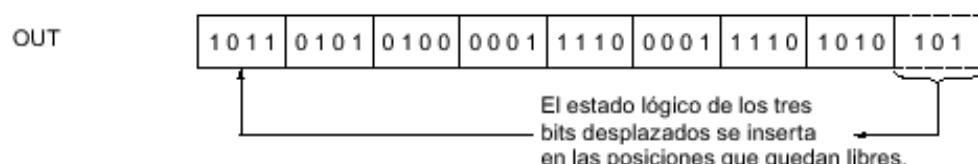
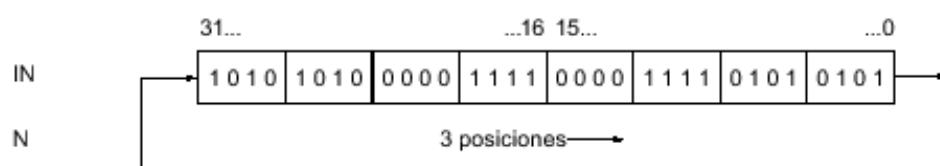


Segmento 2: ROTACIÓN A LA DERECHA (ROR)

- Permite rotar el contenido del operando de la entrada IN de bit en bit hacia la derecha y consultar el resultado en la salida OUT. El parámetro N determina el número de bits que deben rotarse. Los bits que quedan libres al realizar la rotación se rellenan con los bits desplazados hacia fuera. Es importante que el elemento que proporciona la orden para que se ejecute la rotación o el desplazamiento lo ejecute con su flanco positivo para evitar que la acción se efectúe a la velocidad de ciclo miles de ves por segundo.



121

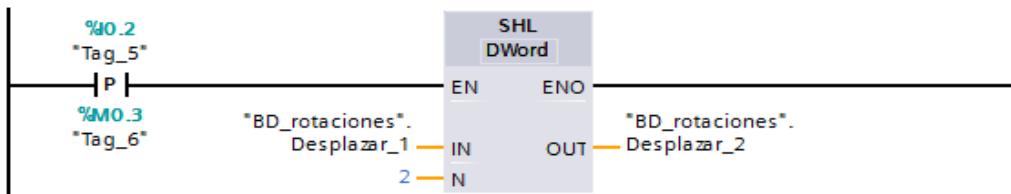


Parámetro	Operando	Valor
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

122

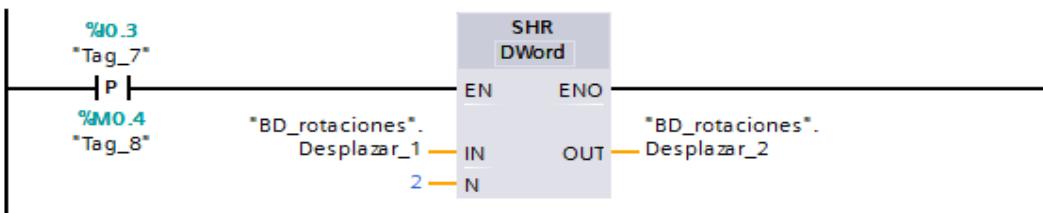
▼ Segmento 3: DESPLAZAMIENTO HACIA LA IZQUIERDA (SHL)

- Permite desplazar el contenido del operando de la entrada IN de bit en bit hacia la izquierda y consultar el resultado en la salida OUT. El parámetro N determina el número de bits que deben desplazarse. Los bits que quedan libres en el área derecha del operando al realizar el desplazamiento se rellenan con ceros.
Es importante que el elemento que proporciona la orden para que se ejecute la rotación o el desplazamiento lo ejecute con su flanco positivo para evitar que la acción se efectúe a la velocidad de ciclo miles de ves por segundo.

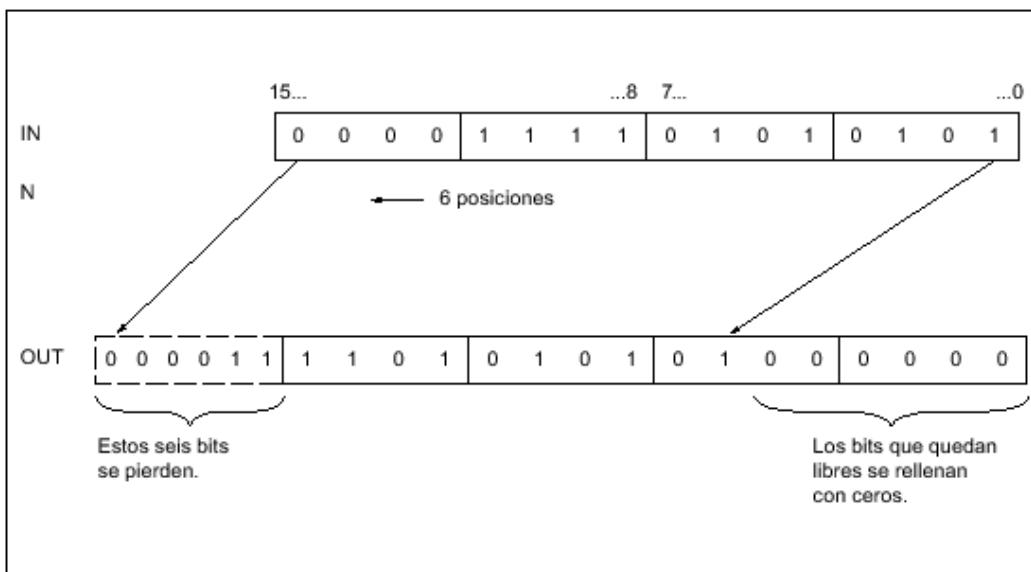


▼ Segmento 4: DESPLAZAMIENTO HACIA LA DERECHA (SHR)

- Permite desplazar el contenido del operando de la entrada IN de bit en bit hacia la derecha y consultar el resultado en la salida OUT. El parámetro N determina el número de bits que deben desplazarse. Los bits que quedan libres en el área izquierda del operando al realizar el desplazamiento se rellenan con ceros.
Es importante que el elemento que proporciona la orden para que se ejecute la rotación o el desplazamiento lo ejecute con su flanco positivo para evitar que la acción se efectúe a la velocidad de ciclo miles de ves por segundo.



123



Parámetro	Operando	Valor
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

124



Operaciones Lógicas con palabras

Visión Práctica S1200

Segmento 1: OPERACION LOGICA Y (AND)

Segmento 2: OPERACIÓN O (OR)

Segmento 3: OPERACIÓN OR EXCLUSIVA (XOR)

Segmento 4: COMPLEMENTO A 1 (INVERT)

Segmento 5: DESCODIFICAR

Segmento 6: CODIFICAR

Segmento 7: SELECCIONAR

Segmento 8: MULTIPLEXAR

Segmento 9: DEMULTIPLEXAR

125

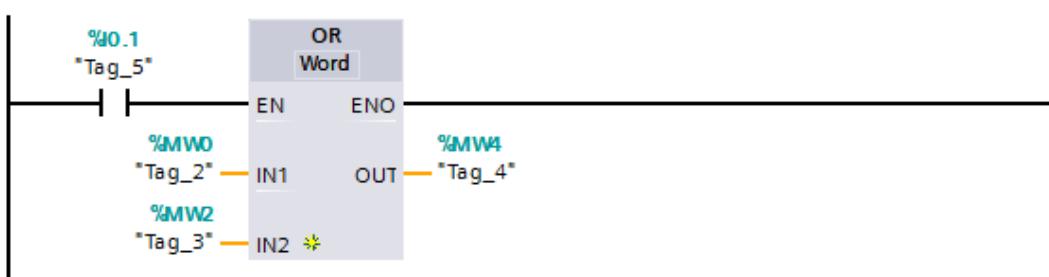
Segmento 1: OPERACION LOGICA Y (AND)

- La instrucción realiza una operación AND entre IN1 y IN2 consultando el resultado en la salida OUT. Cuando la entrada I0.0 está a 1 se realiza una operación AND (multiplicación binaria)



Segmento 2: OPERACIÓN O (OR)

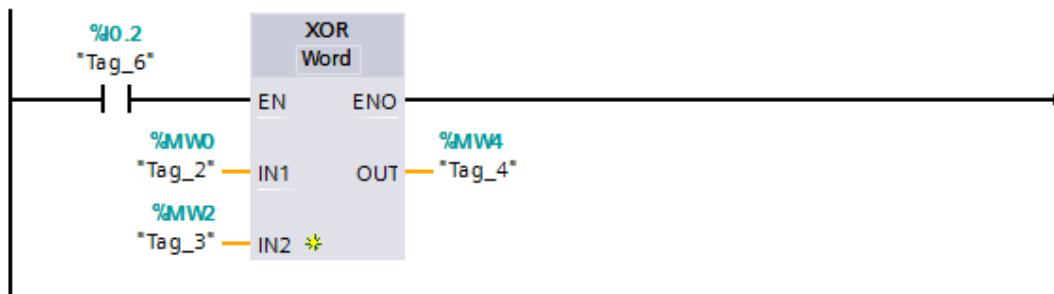
- La instrucción realiza una operación O (OR) equivalente a una suma lógica. Cuando I0.1 está a nivel alta se hace la OR entre IN1 y IN2 consultando el resultado en la salida OUT. Se puede ampliar las entradas pinchando en el punto amarillo del recuadro de la función.



126

Segmento 3: OPERACIÓN OR EXCLUSIVA (XOR)

- Cuando en la entrada I0.2 tenemos nivel alto se permite una OR EXCLUSIVA (XOR) entre IN1 y IN2 consultando el resultado en la salida OUT.



Segmento 4: COMPLEMENTO A 1 (INVERT)

Invierte el estado lógico de los bits de (IN) y se consulta en la salida (OUT)



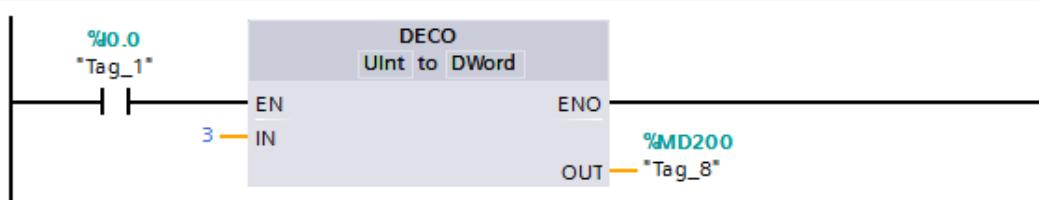
127

Segmento 5: DESCODIFICAR

- La instrucción "Descodificar" activa un bit predeterminado por el valor de entrada en el valor de salida. La instrucción "Descodificar" lee el valor de la entrada IN y activa el bit del valor de salida, cuya posición de bit equivale al valor leído. Los demás bits del valor de salida se rellenan con ceros. Si el valor de la entrada IN es mayor que 31, se ejecuta una instrucción modulo 32. En la lista desplegable "???" del cuadro de la instrucción se puede seleccionar el tipo de datos de la instrucción.

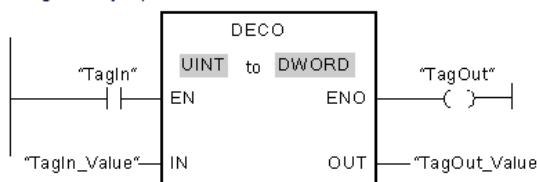
Si el operando "I0.0" devuelve el estado lógico "1", se ejecuta la instrucción. La instrucción lee el número de bit "3" del valor del operando "IN" de la entrada y activa el tercer bit del valor del operando "MD200" de la salida.

Si no se producen errores al ejecutar la instrucción, la salida de habilitación ENO devuelve el estado lógico "1"



Ejemplo

El siguiente ejemplo muestra el funcionamiento de la instrucción:



La figura siguiente muestra el funcionamiento de la instrucción con valores de operandos concretos:

"TagIn_Value"

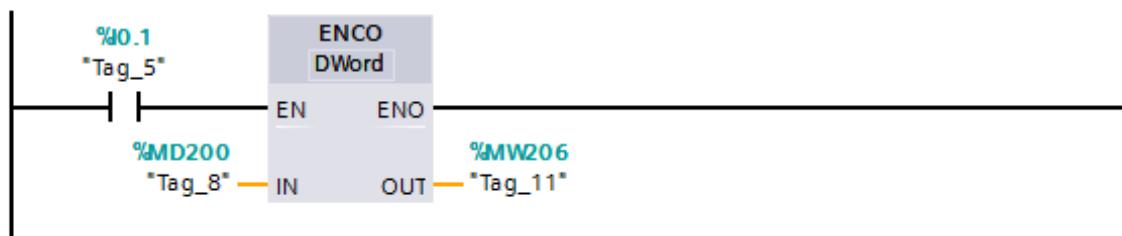
31 16 15 ...	3 ... 0
0000 0000 0000 0000	0000 0000 0000 1000	

"TagOut_Value"

128

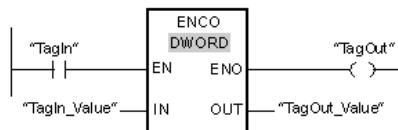
▼ Segmento 6: CODIFICAR

- La instrucción "Codificar" permite leer el número del bit menos significativo que está activado en el valor de entrada y depositarlo en la salida OUT.
- La instrucción "Codificar" selecciona el bit menos significativo del valor de la entrada IN y escribe su número de bit en la variable de la salida OUT.
- Si el operando "I0.1" devuelve el estado lógico "1", se ejecuta la instrucción. La instrucción selecciona el bit menos significativo que está activado en la entrada "MD200" y escribe el bit "3" en la variable de la salida "MW206".
- Si no se producen errores al ejecutar la instrucción, la salida de habilitación ENO devuelve el estado lógico "1".



Ejemplo

El siguiente ejemplo muestra el funcionamiento de la instrucción:



La figura siguiente muestra el funcionamiento de la instrucción con valores de operandos concretos:

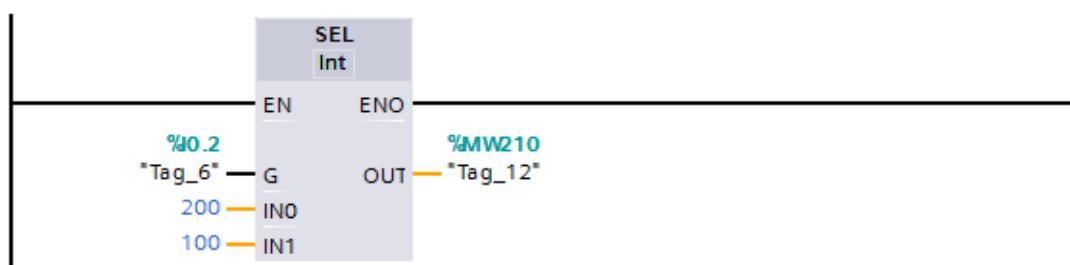
31 16 15 ...	3 ... 0
"TagIn_Value"		
0000 1111 0000 0101 0000 1001 0000 1000		

"Tag Out_Value"

129

▼ Segmento 7: SELECCIONAR

- La instrucción "Seleccionar" permite elegir, en función de un interruptor (entrada G), una de las entradas IN0 o IN1 y copiar su contenido en la salida OUT. Si la entrada G tiene el estado lógico "0", se copia el valor de la entrada IN0. Si la entrada G tiene el estado lógico "1", se copia el valor de la entrada IN1 en la salida OUT.
- Si el operando "EN" tiene el estado lógico "1", se ejecuta la instrucción. Dependiendo del estado lógico de la entrada "I0.2", se selecciona el valor de la entrada "IN0" o "IN1" y se copia en la salida "OUT". Si no se producen errores al ejecutar la instrucción, la salida de habilitación ENO devuelve el estado lógico "1".



Parámetro	Operando	Valor	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

130

▼ Segmento 8: MULTIPLEXAR

- La instrucción "Multiplexar" permite copiar el contenido de una entrada seleccionada en la salida OUT. El número de entradas seleccionables en el cuadro de la instrucción se puede ampliar. Se puede declarar un máximo de 32 entradas. Las entradas se numeran automáticamente en el cuadro. La numeración comienza por IN0 y continúa en orden ascendente con cada nueva entrada. El parámetro K determina la entrada cuyo contenido se copia en la salida OUT. Si el valor del parámetro K es mayor que el número de entradas disponibles, el contenido del parámetro ELSE se copia en la salida OUT y a la salida de habilitación ENO se le asigna el estado lógico "0". Si el operando "I0.3" devuelve el estado lógico "1", se ejecuta la instrucción. De acuerdo con el valor del operando "K", se copia el valor de la entrada "IN...O,1,2..." y se asigna al operando de la salida "OUT".

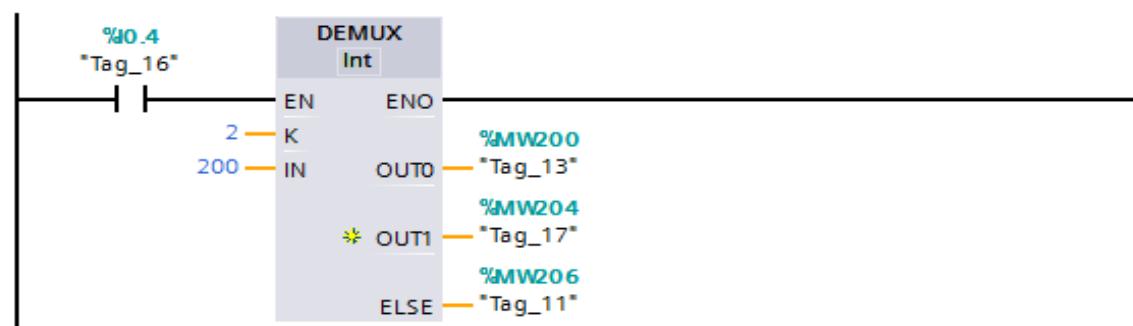


Parámetro	Operando	Valor
K	Tag_Number	1
IN0	Tag_Value_0	DW#16#00000000
IN1	Tag_Value_1	DW#16#003E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#003E4A7D

131

▼ Segmento 9: DEMULTIPLEXAR

- La instrucción "Desmultiplexar" permite copiar el contenido de la entrada IN en una salida seleccionada. El número de salidas seleccionables del cuadro de la instrucción se puede ampliar. Las salidas se numeran automáticamente en el cuadro. La numeración comienza por OUT0 y continúa en orden ascendente con cada nueva salida. El parámetro K permite determinar la salida en la que se copia el contenido de la entrada IN. Las demás salidas no cambian. Si el valor del parámetro K es mayor que el número de salidas disponibles, el contenido de la entrada IN se copia en el parámetro ELSE y a la salida de habilitación ENO se le asigna el estado lógico "0". La instrucción "Desmultiplexar" solo se puede ejecutar si las variables de la entrada IN y las de todas las salidas son del mismo tipo de datos. El parámetro K es una excepción, ya que en el mismo sólo pueden indicarse números enteros. Si la entrada "I0.4" devuelve el estado lógico "1", se ejecuta la instrucción. Según el valor del operando "K", se copia el valor de la entrada IN en la salida correspondiente.



Parámetro	Operando	Valores	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFF	DW#16#003E4A7D

132

Parámetro	Operando	Valores	
OUT0	Tag_Output_0	Sin cambios	Sin cambios
OUT1	Tag_Output_1	DW#16#FFFFFF	Sin cambios
ELSE	Tag_Output_2	Sin cambios	DW#16#003E4A7D

133



Control de programa Visión Práctica S1200

Segmento 1: SALTAR SI EL RESULTADO LÓGICO RLO=1 (JMP)

Segmento 2: LABEL SALTO_1

Segmento 3: SALTAR SI EL RESULTADO LÓGICO RLO=0 (JMPN)

Segmento 4: LABEL SALTO_2

Segmento 5: DEFINIR LISTA DE SALTOS (JMP_LIST)

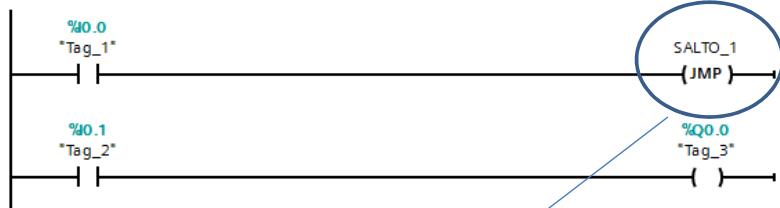
Segmento 6: DISTRIBUIDOR DE SALTOS (SWITCH)

Segmento 7: LABEL SALTO_3

134

▼ Segmento 1: SALTAR SI EL RESULTADO LÓGICO RLO=1 (JMP)

- La instrucción "Saltar si RLO = 1" permite interrumpir la ejecución lineal del programa y continuarla en un segmento diferente. El segmento de destino tiene que marcarse con una etiqueta (LABEL). El nombre de la etiqueta se indica en el comodín situado encima de la instrucción. La etiqueta indicada debe encontrarse en el mismo bloque en el que se ejecuta la instrucción. Su nombre debe ser único en el bloque. En cada segmento no debe existir más de una bobina de salto.
- Si el resultado lógico (RLO) de la entrada de la instrucción es "1", se ejecuta el salto al segmento identificado por la etiqueta indicada. El salto puede realizarse hacia números de segmento superiores o inferiores.
- Si no se cumple la condición en la entrada de la instrucción (RLO = 0), la ejecución del programa continúa en el segmento siguiente.

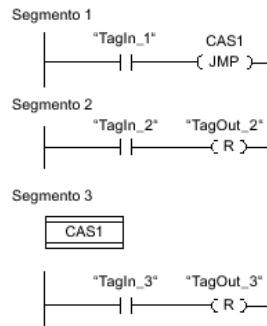


▼ Segmento 2: LABEL SALTO_1

- En este segmento hemos colocado la etiqueta que identifica el salto del segmento 1 (SALTO_1). Una etiqueta sirve para marcar el segmento de destino en el que debe continuar la ejecución del programa tras un salto.
- La etiqueta y la instrucción en la que se indica la etiqueta como destino del salto deben encontrarse en el mismo bloque. La denominación de una etiqueta debe ser única en el bloque. Puede declarar un máximo de 32 etiquetas en caso de utilizar una CPU S7-1200 y un máximo de 256 etiquetas en caso de utilizar una CPU S7-1500.
- En un segmento sólo se puede colocar una etiqueta. A toda etiqueta se puede acceder desde distintas posiciones.



Ejemplo

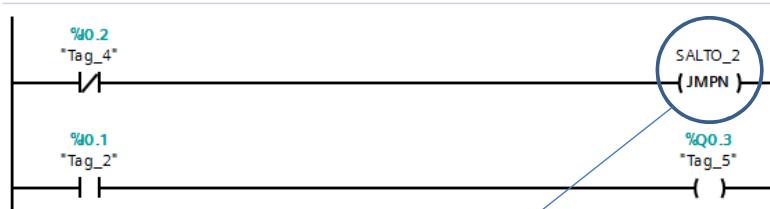


Si el operando "TagIn_1" devuelve el estado lógico "1", se ejecuta la instrucción "Saltar si RLO = 1". Por este motivo, se interrumpe la ejecución lineal del programa y se prosigue en el segmento 3, marcado por la etiqueta CAS1. Si la entrada "TagIn_3" devuelve el estado lógico "1", se desactiva la salida "TagOut_3".

135

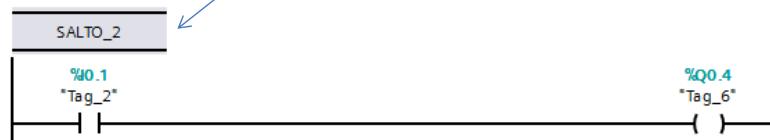
▼ Segmento 3: SALTAR SI EL RESULTADO LÓGICO RLO=0 (JMPN)

- La instrucción "Saltar si RLO = 0" permite interrumpir la ejecución lineal del programa y continuarla en un segmento diferente si el resultado lógico de la entrada de la instrucción es "0". El segmento de destino tiene que marcarse con una etiqueta (LABEL). El nombre de la etiqueta se indica en el comodín situado encima de la instrucción.
- La etiqueta indicada debe encontrarse en el mismo bloque en el que se ejecuta la instrucción. Su nombre debe ser único en el bloque. En cada segmento no debe existir más de una bobina de salto.
- Si el resultado lógico (RLO) de la entrada de la instrucción es "0", se ejecuta el salto al segmento identificado por la etiqueta indicada. El salto puede realizarse hacia números de segmento superiores o inferiores.
- Si el resultado lógico de la entrada de la instrucción es "1", la ejecución del programa continúa en el segmento siguiente.

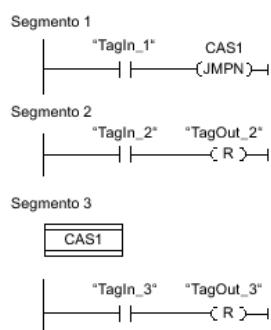


▼ Segmento 4: LABEL SALTO_2

- Una etiqueta sirve para marcar el segmento de destino en el que debe continuar la ejecución del programa tras un salto.
- La etiqueta y la instrucción en la que se indica la etiqueta como destino del salto deben encontrarse en el mismo bloque. La denominación de una etiqueta debe ser única en el bloque. Puede declarar un máximo de 32 etiquetas en caso de utilizar una CPU S7-1200 y un máximo de 256 etiquetas en caso de utilizar una CPU S7-1500.
- En un segmento sólo se puede colocar una etiqueta. A toda etiqueta se puede acceder desde distintas posiciones.



Ejemplo



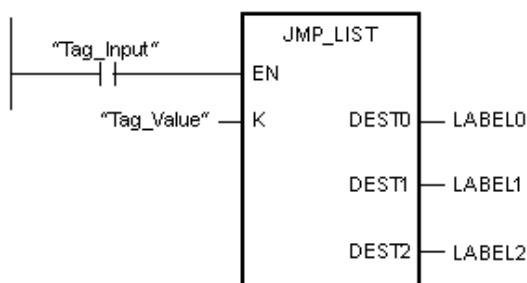
Si el operando "TagIn_1" devuelve el estado lógico "0", se ejecuta la instrucción "Saltar si RLO = 0". Por este motivo, se interrumpe la ejecución lineal del programa y se prosigue en el segmento 3, marcado por la etiqueta CAS1. Si la entrada "TagIn_3" devuelve el estado lógico "1", se desactiva la salida "TagOut_3".

136

JMP_LIST: Definir lista de saltos

Ejemplo

El siguiente ejemplo muestra el funcionamiento de la instrucción:



La tabla siguiente muestra el funcionamiento de la instrucción con valores de operandos concretos:

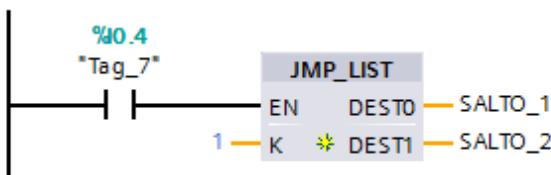
Parámetro	Operando/etiqueta	Valor
K	Tag_Value	1
Dest0	LABEL0	Salto al segmento que está marcado con la etiqueta "LABEL0".
Dest1	LABEL1	Salto al segmento que está marcado con la etiqueta "LABEL1".
Dest2	LABEL2	Salto al segmento que está marcado con la etiqueta "LABEL2".

Si el operando "Tag_Input" devuelve el estado lógico "1", se ejecuta la instrucción "Definir lista de saltos". La ejecución del programa continúa conforme al valor del operando "Tag_Value" en el segmento que está marcado con la etiqueta "LABEL1".

137

Segmento 5: DEFINIR LISTA DE SALTOS (JMP_LIST)

- La instrucción "Definir lista de saltos" permite definir varios saltos condicionados y continuar la ejecución del programa en un segmento determinado en función del valor del parámetro K. Los saltos se definen mediante etiquetas (LABEL) que se indican en las salidas del cuadro de la instrucción. El número de salidas del cuadro de la instrucción se puede ampliar. Puede declarar un máximo de 32 salidas en caso de utilizar una CPU S7-1200 y un máximo de 99 salidas en caso de utilizar una CPU S7-1500. La numeración de las salidas comienza por el valor "0" y continúa en orden ascendente con cada nueva salida. En las salidas de la instrucción únicamente se pueden indicar etiquetas. No está permitido indicar instrucciones u operandos. Con el valor del parámetro K se indica el número de la salida ya la vez la etiqueta en la que debe continuarse la ejecución del programa. Si el valor del parámetro K es mayor que el número de salidas disponibles, la ejecución del programa continúa en el siguiente segmento del bloque. La instrucción "Definir lista de saltos" se ejecuta solo si el estado lógico de la entrada de habilitación EN es "1".

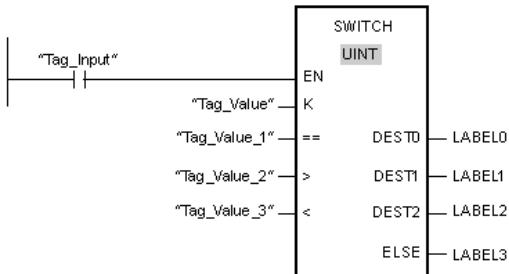


138

SWITCH: Distribuidor de saltos

Ejemplo

El siguiente ejemplo muestra el funcionamiento de la instrucción:

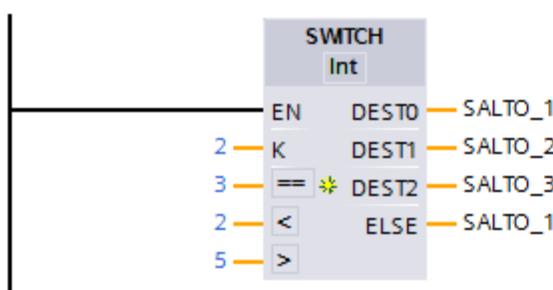


Parámetro	Operando/etiqueta	Valor
K	Tag_Value	23
==	Tag_Value_1	20
>	Tag_Value_2	21
<	Tag_Value_3	19
Dest 0	LABEL0	Salto a la etiqueta "LABEL0" si el valor del parámetro K es igual a 20.
Dest 1	LABEL1	Salto a la etiqueta "LABEL1" si el valor del parámetro K es mayor que 21.
Dest 2	LABEL2	Salto a la etiqueta "LABEL2" si el valor del parámetro K es menor que 19.
ELSE	LABEL 3	Salto a la etiqueta "LABEL3" si no se cumple ninguna de las condiciones de comparación.

Si el operando "Tag_Input" cambia al estado lógico "1", se ejecuta la instrucción "Distribuidor de saltos". La ejecución del programa continúa en el segmento que está marcado con la etiqueta "LABEL1".

Segmento 6: DISTRIBUIDOR DE SALTOS (SWTCH)

- La instrucción "Distribuidor de saltos" permite definir varios saltos de programa que se ejecutan en función del resultado de una o varias instrucciones de comparación.
- El valor que se va a comparar se especifica en el parámetro K. Este valor se compara con los valores que devuelven las distintas entradas. El tipo de comparación se selecciona individualmente para cada entrada. La disponibilidad de las diferentes instrucciones de comparación depende del tipo de datos de la instrucción.





Conversión de datos

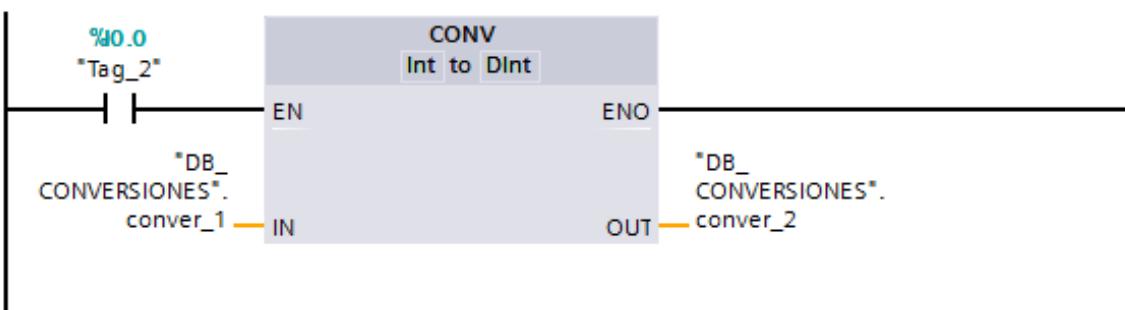
Visión Práctica S1200

- Segmento 1:** CONVERTIR VALOR (CONV)
- Segmento 2:** REDONDEAR NÚMERO (ROUND)
- Segmento 3:** REDONDEAR AL SIGUIENTE ENTERO SUPERIOR (CEIL)
- Segmento 4:** REDONDEAR AL SIGUIENTE ENTERO INFERIOR (FLOOR)
- Segmento 5:** TRUNCAR A ENTERO (TRUNC)
- Segmento 6:** ESCALAR (SCALE_X)
- Segmento 7:** NORMALIZAR (NORM_X)
- Segmento 8:** CONTROL ANALÓGICO CON SCALE_X / NORM_X

141

▼ Segmento 1: CONVERTIR VALOR (CONV)

- ▼ La instrucción "Convertir valor" lee el contenido del parámetro IN y lo convierte según los tipos de datos seleccionados en el cuadro de la instrucción. El valor convertido se deposita en la salida OUT.



Parámetros

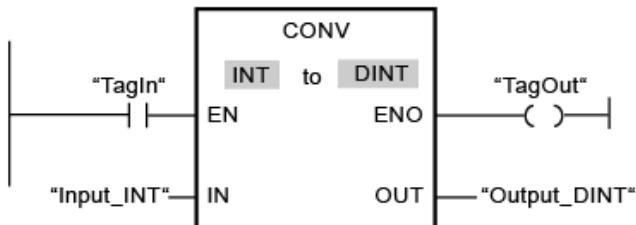
La tabla siguiente muestra los parámetros de la instrucción "Convertir valor":

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
EN	Input	BOOL	I, Q, M, D, L	Entrada de habilitación
ENO	Output	BOOL	I, Q, M, D, L	Salida de habilitación
IN	Input	Secuencias de bits, enteros, números en coma flotante, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P o constante	Valor que se convierte.
OUT	Output	Secuencias de bits, enteros, números en coma flotante, CHAR, WCHAR, BCD16, BCD32	I, Q, M, D, L, P	Resultado de la conversión

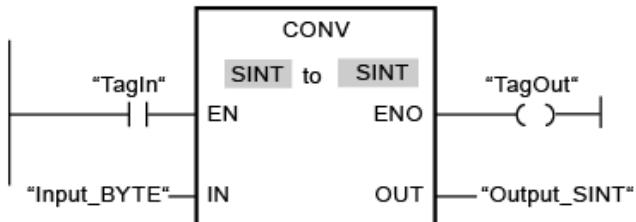
142

Ejemplos

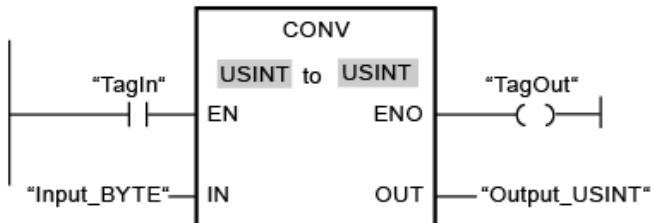
El ejemplo siguiente muestra la conversión de un entero (16 bits) a otro entero (32 bits):



El ejemplo siguiente muestra la conversión de un byte (8 bits) al entero SINT (8 bits):



El ejemplo siguiente muestra la conversión de un byte (8 bits) a un entero sin signo USINT (8 bits):

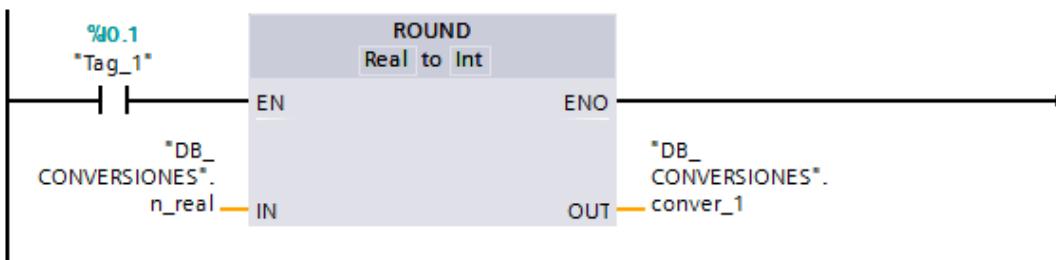


Las conversiones son posibles porque los operandos tienen la misma longitud.

143

Segmento 2: REDONDEAR NÚMERO (ROUND)

- La instrucción "Redondear número" permite redondear el valor de la entrada IN al siguiente número entero. La instrucción interpreta el valor de la entrada IN como número en coma flotante y lo convierte a un número entero del tipo de datos DINT. Si el valor de entrada se encuentra entre un número par y uno impar, se selecciona el número par. El resultado de la instrucción se deposita en la salida OUT y se puede consultar allí.



Parámetros

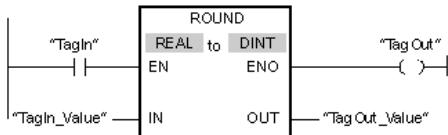
La tabla siguiente muestra los parámetros de la instrucción "Redondear número":

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
EN	Input	BOOL	I, Q, M, D, L	Entrada de habilitación
ENO	Output	BOOL	I, Q, M, D, L	Salida de habilitación
IN	Input	Números en coma flotante	I, Q, M, D, L, P o constante	Valor de entrada que se debe redondear.
OUT	Output	Enteros, números en coma flotante	I, Q, M, D, L, P	Resultado del redondeo

144

Ejemplo

El siguiente ejemplo muestra el funcionamiento de la instrucción:



La tabla siguiente muestra el funcionamiento de la instrucción con valores de operandos concretos:

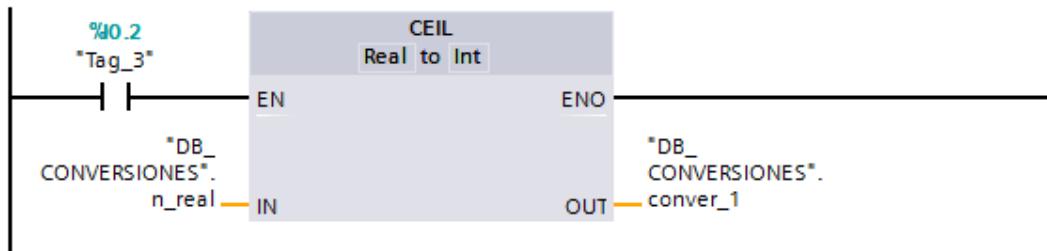
Parámetro	Operando	Valor	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

Si el operando "TagIn" devuelve el estado lógico "1", se ejecuta la instrucción. El número en coma flotante de la entrada "TagIn_Value" se redondea al número entero par más próximo y se deposita en la salida "TagOut_Value". Si no se producen errores al ejecutar la instrucción, se activa la salida "TagOut".

145

▼ Segmento 3: REDONDEAR AL SIGUIENTE ENTERO SUPERIOR (CEIL)

- La instrucción "Redondear un número en coma flotante al siguiente entero superior" permite redondear el valor de la entrada IN al siguiente número entero superior. La instrucción interpreta el valor de la entrada IN como número en coma flotante y lo convierte en el siguiente número entero superior. El resultado de la instrucción se deposita en la salida OUT y se puede consultar allí. El valor de salida puede ser mayor o igual al valor de entrada.



Parámetros

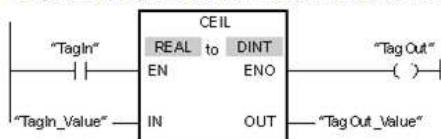
La tabla siguiente muestra los parámetros de la instrucción "Redondear un número en coma flotante al siguiente entero superior":

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
EN	Input	BOOL	I, Q, M, D, L	Entrada de habilitación
ENO	Output	BOOL	I, Q, M, D, L	Salida de habilitación
IN	Input	Números en coma flotante	I, Q, M, D, L, P o constante	Valor de entrada
OUT	Output	Enteros, números en coma flotante	I, Q, M, D, L, P	Resultado con el siguiente entero superior

146

Ejemplo

El siguiente ejemplo muestra el funcionamiento de la instrucción:



La tabla siguiente muestra el funcionamiento de la instrucción con valores de operandos concretos:

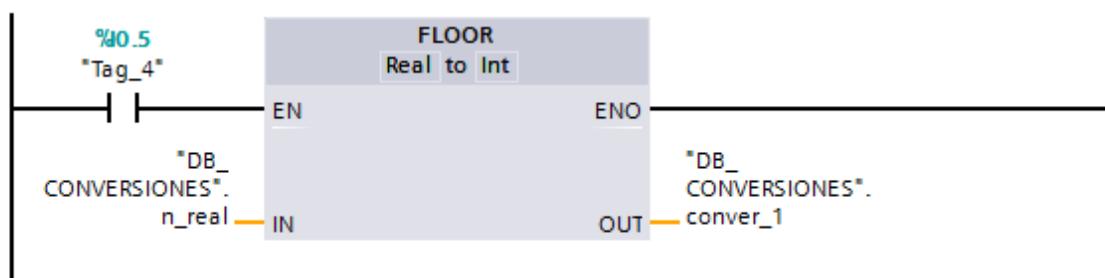
Parámetro	Operando	Valor	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

Si el operando "TagIn" devuelve el estado lógico "1", se ejecuta la instrucción. El número en coma flotante de la entrada "TagIn_Value" se redondea al siguiente número entero superior y se devuelve en la salida "TagOut_Value". Si no se producen errores al ejecutar la instrucción, se activa la salida "TagOut".

147

Segmento 4: REDONDEAR AL SIGUIENTE ENTERO INFERIOR (FLOOR)

- La instrucción "Redondear un número en coma flotante al siguiente entero inferior" permite redondear el valor de la entrada IN al siguiente número entero inferior. La instrucción interpreta el valor de la entrada IN como número en coma flotante y lo convierte en el siguiente número entero inferior. El resultado de la instrucción se deposita en la salida OUT y se puede consultar allí. El valor de salida puede ser menor o igual al valor de entrada.



Parámetros

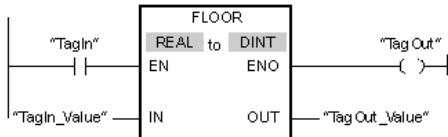
La tabla siguiente muestra los parámetros de la instrucción "Redondear un número en coma flotante al siguiente entero inferior":

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
EN	Input	BOOL	I, Q, M, D, L	Entrada de habilitación
ENO	Output	BOOL	I, Q, M, D, L	Salida de habilitación
IN	Input	Números en coma flotante	I, Q, M, D, L, P o constante	Valor de entrada
OUT	Output	Enteros, números en coma flotante	I, Q, M, D, L, P	Resultado con el siguiente entero inferior

148

Ejemplo

El siguiente ejemplo muestra el funcionamiento de la instrucción:



La tabla siguiente muestra el funcionamiento de la instrucción con valores de operandos concretos:

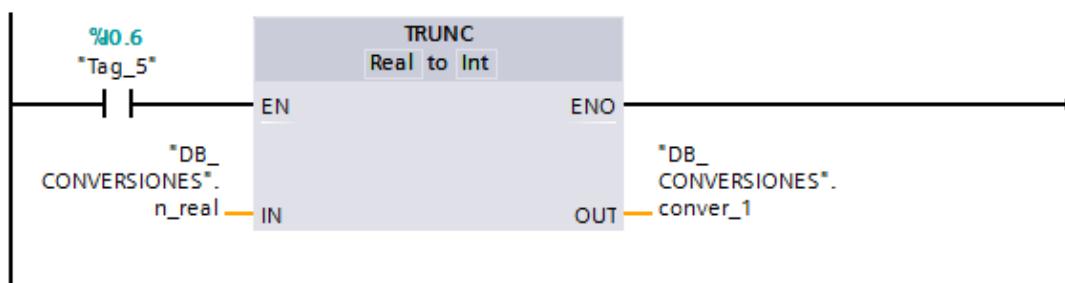
Parámetro	Operando	Valor	
IN	TagIn_Value	0.5000000	-0.5000000
OUT	TagOut_Value	0	-1

Si el operando "TagIn" devuelve el estado lógico "1", se ejecuta la instrucción. El número en coma flotante de la entrada "TagIn_Value" se redondea al número entero inferior más próximo y se deposita en la salida "TagOut_Value". Si no se producen errores al ejecutar la instrucción, se activa la salida "TagOut".

149

Segmento 5: TRUNCAR A ENTERO (TRUNC)

- La instrucción "Truncar a entero" permite generar un valor entero a partir del valor de la entrada IN. El valor de la entrada IN se interpreta como número en coma flotante. La instrucción selecciona solo la parte entera del número en coma flotante y la deposita sin decimales en la salida OUT.



Parámetros

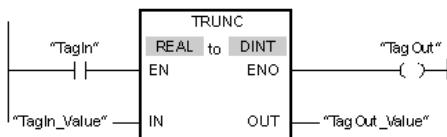
La tabla siguiente muestra los parámetros de la instrucción "Truncar a entero":

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
EN	Input	BOOL	I, Q, M, D, L	Entrada de habilitación
ENO	Output	BOOL	I, Q, M, D, L	Salida de habilitación
IN	Input	Números en coma flotante	I, Q, M, D, L o constante	Valor de entrada
OUT	Output	Enteros, números en coma flotante	I, Q, M, D, L	Parte entera del valor de entrada

150

Ejemplo

El siguiente ejemplo muestra el funcionamiento de la instrucción:



La tabla siguiente muestra el funcionamiento de la instrucción con valores de operandos concretos:

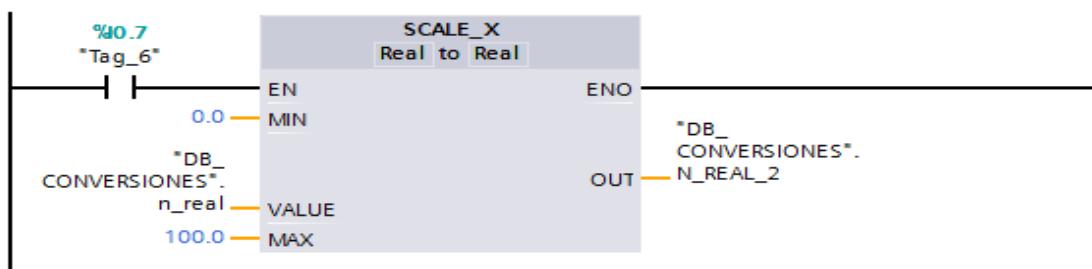
Parámetro	Operando	Valor	
IN	TagIn_Value	1.5000000	-1.5000000
OUT	TagOut_Value	1	-1

Si el operando "TagIn" devuelve el estado lógico "1", se ejecuta la instrucción. La parte entera del número en coma flotante de la entrada "TagIn_Value" se convierte en un número entero y se deposita en la salida "TagOut_Value". Si no se producen errores al ejecutar la instrucción, se activa la salida "TagOut".

151

Segmento 6: ESCALAR (SCALE_X)

- ▼ La instrucción "Escalar" escala el valor de la entrada VALUE mapeándolo en un determinado rango de valores. Al ejecutar la instrucción "Escalar", el número en coma flotante de la entrada VALUE se escala al rango de valores definido por los parámetros MIN y MAX. El resultado de la escala es un número entero que se deposita en la salida OUT.



Parámetros

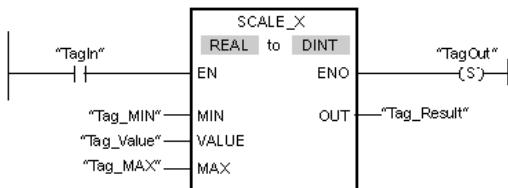
La tabla siguiente muestra los parámetros de la instrucción "Escalar".

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
EN	Input	BOOL	I, Q, M, D, L	Entrada de habilitación
ENO	Output	BOOL	I, Q, M, D, L	Salida de habilitación
MIN	Input	Enteros, números en coma flotante	I, Q, M, D, L o constante	Límite inferior del rango de valores
VALUE	Input	Números en coma flotante	I, Q, M, D, L o constante	Valor que se escala. Si se indica una constante, esta debe declararse.
MAX	Input	Enteros, números en coma flotante	I, Q, M, D, L o constante	Límite superior del rango de valores
OUT	Output	Enteros, números en coma flotante	I, Q, M, D, L	Resultado de la escala

152

Ejemplo

El siguiente ejemplo muestra el funcionamiento de la instrucción:



La tabla siguiente muestra el funcionamiento de la instrucción con valores de operandos concretos:

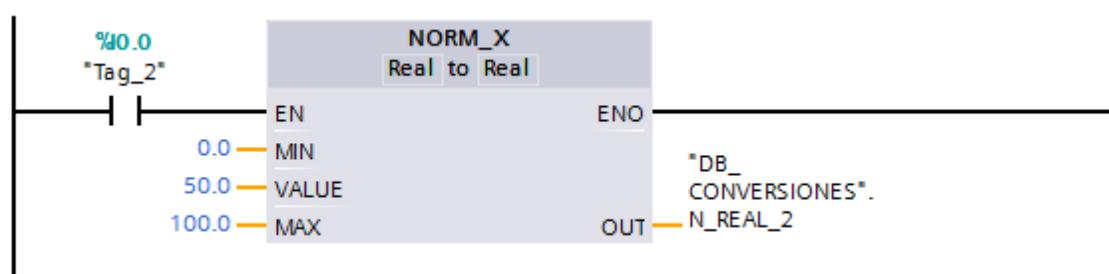
Parámetro	Operando	Valor
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5
MAX	Tag_MAX	30
OUT	Tag_Result	20

Si el operando "TagIn" devuelve el estado lógico "1", se ejecuta la instrucción. El valor de la entrada "Tag_Value" se escala al rango de valores definido por los valores de las entradas "Tag_MIN" y "Tag_MAX". El resultado se deposita en la salida "Tag_Result". Si no se producen errores al ejecutar la instrucción, la salida de habilitación ENO devuelve el estado lógico "1" y se activa la salida "TagOut".

153

Segmento 7: NORMALIZAR (NORM_X)

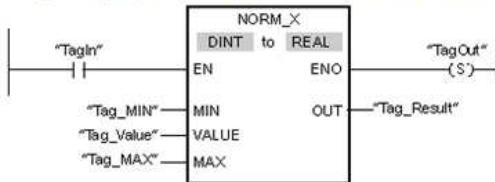
- La instrucción "Normalizar" normaliza el valor de la variable de la entrada VALUE representándolo en una escala lineal. Los parámetros MIN y MAX sirven para definir los límites de un rango de valores que se refleja en la escala. En función de la posición del valor que se debe normalizar en este rango de valores, se calcula el resultado y se deposita como número en coma flotante en la salida OUT. Si el valor que se debe normalizar es igual al valor de la entrada MIN, la salida OUT devuelve el valor "0.0". Si el valor que se debe normalizar es igual al valor de la entrada MAX, la salida OUT devuelve el valor "1.0".



154

Ejemplo

El siguiente ejemplo muestra el funcionamiento de la instrucción:



La tabla siguiente muestra el funcionamiento de la instrucción con valores de operandos concretos:

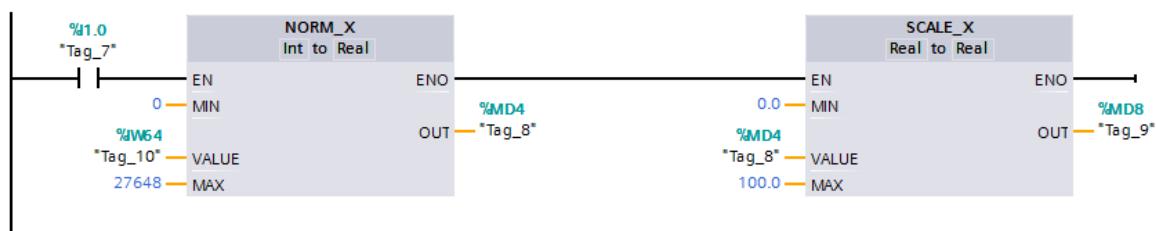
Parámetro	Operando	Valor
MIN	Tag_MIN	10
VALUE	Tag_Value	20
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

Si el operando "TagIn" devuelve el estado lógico "1", se ejecuta la instrucción. El valor de la entrada "Tag_Value" se asigna al rango de valores definido por los valores de las entradas "Tag_MIN" y "Tag_MAX". El valor de la variable de la entrada "Tag_Value" se normaliza conforme al rango de valores definido. El resultado se deposita como número en coma flotante en la salida "Tag_Result". Si no se producen errores al ejecutar la instrucción, la salida de habilitación ENO devuelve el estado lógico "1" y se activa la salida "TagOut".

155

Segmento 8: CONTROL ANALÓGICO CON SCALE_X / NORM_X

- Normalizado y escalado de la entrada analógica AI0. Primero se normaliza el dato de la entrada analógica según la resolución del convertidor analógico del PLC y después se escala según el rango de valores máximos y mínimos en los que nos debemos mover para el control.



156



Reloj en Tiempo Real

Visión Práctica S1200

Segmento 1: LEER LA HORA DEL RELOJ DE LA CPU (RD_SYS_T)

Segmento 2: LEER LA HORA ADAPTADA A LA ZONA HORARIA Y HORARIOS DE VERANO/INVIERNO (RD_LOC_T)

Segmento 3: EJEMPLO DE PROGRAMACIÓN DEL RTC (RD_SYS_T)

157

RD_SYS_T: Leer la hora

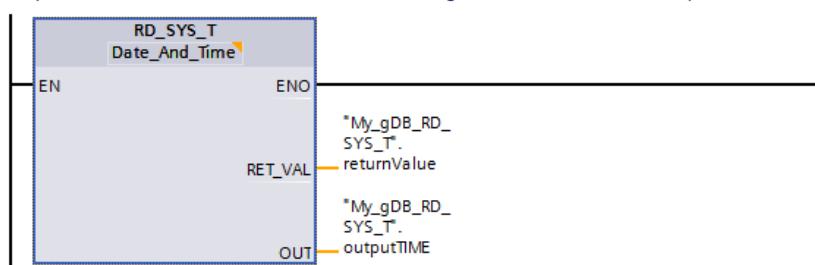
Ejemplo

En el ejemplo siguiente se lee la hora del módulo del reloj de la CPU. El tipo de datos utilizado es DATE AND TIME.

Para almacenar los datos se crean dos variables en un bloque de datos global.

	Name	Data type	Start value
1	Static		
2	outputTIME	Date_And_Time	DT#1990-01-01-00:00:00
3	returnValue	Int	0

Los parámetros de la instrucción se interconectan del siguiente modo. Seleccione el tipo de datos DATE AND TIME.



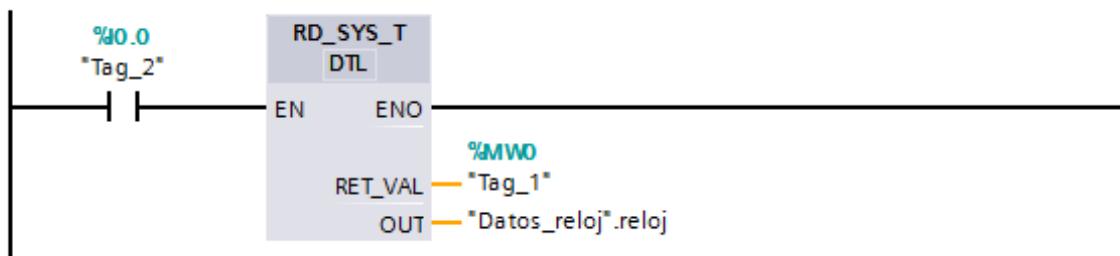
La hora del módulo del reloj de la CPU se lee y se indica en el parámetro de salida OUT ("outputTIME"). En el parámetro de salida RET_VAL ("returnValue") se indica que la ejecución se ha realizado sin errores.

	Name	Data type	Start value	Monitor value
1	Static			
2	outputTIME	Date_And_Time	DT#1990-01-01-00:00:00	DT#2014-08-04-15:15:15
3	returnValue	Int	0	0

158

▼ Segmento 1: LEER LA HORA DEL RELOJ DE LA CPU (RD_SYS_T)

- La instrucción lee la fecha y hora actuales del reloj de la CPU. Los datos leídos se devuelven en el parámetro de salida OUT de la instrucción, y este queda almacenado en el bloque de datos (Datos_reloj) en la variable DTL (reloj). En la salida RET_VAL se puede comprobar si se han producido errores durante la ejecución de la instrucción.



159

RD_LOC_T: Leer hora local

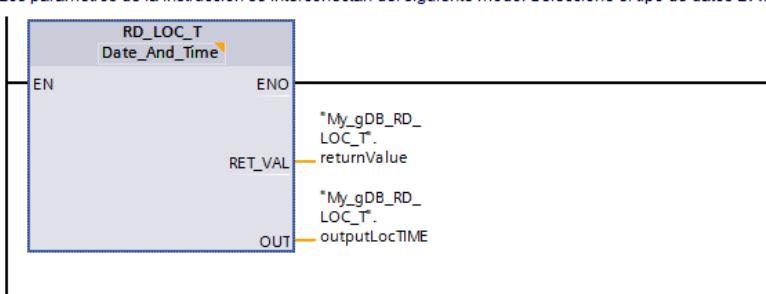
Ejemplo

En el siguiente ejemplo se lee la hora local del reloj de la CPU. El tipo de datos utilizado es DATE AND TIME.

Para almacenar los datos se crean dos variables en un bloque de datos global.

My_gDB_RD_LOC_T		
	Name	Data type
1	Static	
2	outputLocTIME	Date_And_Time
3	returnValue	Int

Los parámetros de la instrucción se interconectan del siguiente modo. Seleccione el tipo de datos DATE AND TIME.



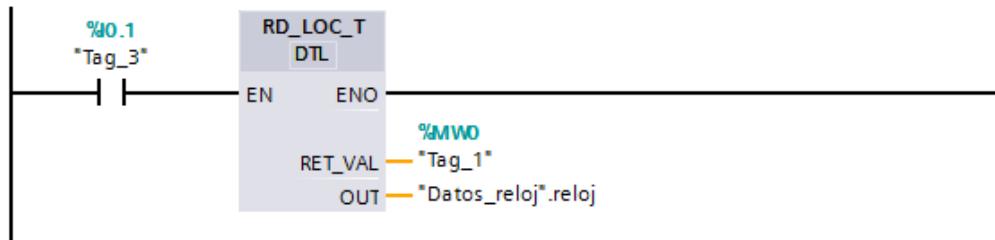
La hora local del reloj de la CPU se lee y se indica en el parámetro de salida OUT ("outputLocTIME"). En el parámetro de salida RET_VAL ("returnValue") se indica que la ejecución se ha realizado sin errores y que la hora local se emite como horario de verano con esta llamada.

My_gDB_RD_LOC_T				
	Name	Data type	Start value	Monitor value
1	Static			
2	outputLocTIME	Date_And_Time	DT#1990-01-01-00:00:00	DT#2014-08-04-16:15:15.001
3	returnValue	Int	0	1

160

Segmento 2: LEER LA HORA ADAPTADA A LA ZONA HORARIA Y HORARIOS DE VERANO/INVIERNO (RD_LOC_T)

- La instrucción lee la fecha y hora actuales del reloj de la CPU. Los datos leídos se devuelven en el parámetro de salida OUT de la instrucción, y este queda almacenado en el bloque de datos (Datos_reloj) en la variable DTL (reloj). En la salida RET_VAL se puede comprobar si se han producido errores durante la ejecución de la instrucción. Para emitir la hora local se utilizan las indicaciones relativas a la zona horaria y al comiendo de los horarios de verano e invierno, que se han ajustado al configurar el reloj de la CPU.



161

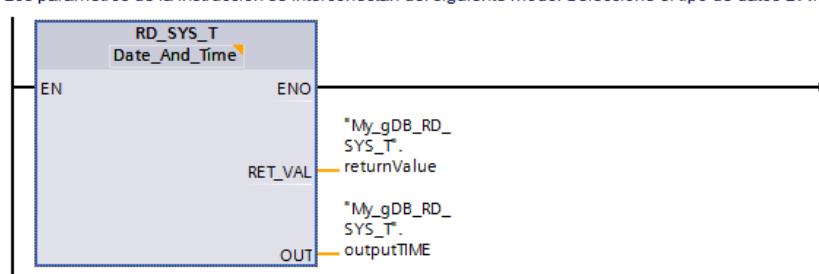
RD_SYS_T: Leer la hora

Ejemplo

En el ejemplo siguiente se lee la hora del módulo del reloj de la CPU. El tipo de datos utilizado es DATE AND TIME. Para almacenar los datos se crean dos variables en un bloque de datos global.

	Name	Data type	Start value
1	Static		
2	outputTIME	Date_And_Time	DT#1990-01-01-00:00:00
3	returnValue	Int	0

Los parámetros de la instrucción se interconectan del siguiente modo. Seleccione el tipo de datos DATE AND TIME.



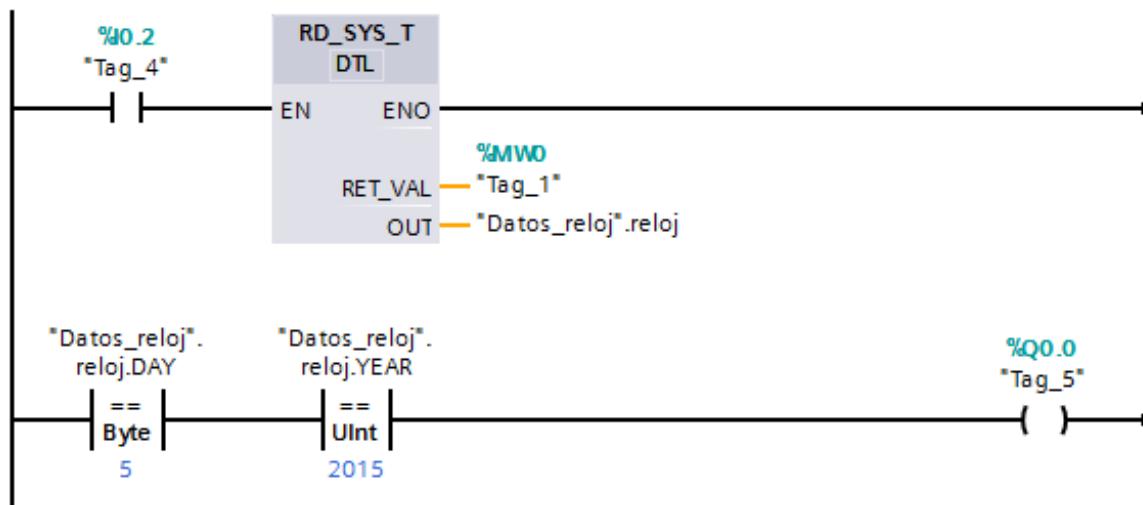
La hora del módulo del reloj de la CPU se lee y se indica en el parámetro de salida OUT ("outputTIME"). En el parámetro de salida RET_VAL ("returnValue") se indica que la ejecución se ha realizado sin errores.

	Name	Data type	Start value	Monitor value
1	Static			
2	outputTIME	Date_And_Time	DT#1990-01-01-00:00:00	DT#2014-08-04-15:15:15
3	returnValue	Int	0	0

162

▼ Segmento 3: EJEMPLO DE PROGRAMACIÓN DEL RTC (RD_SYS_T)

- ▼ Cuando se llama al RTC hay que generar un dato DTL en un bloque de datos. Una vez generado y transferido se pueden observar los datos en la table de observación y en el bloque de datos. En nuestro ejemplo cuando activamos la entrada lee el reloj y lo envia al bloque de datos. Si las condiciones de comparación son TRUE entonces la salida (Q0.0) se activa.



163



Programación Estructurada Visión Práctica S1200

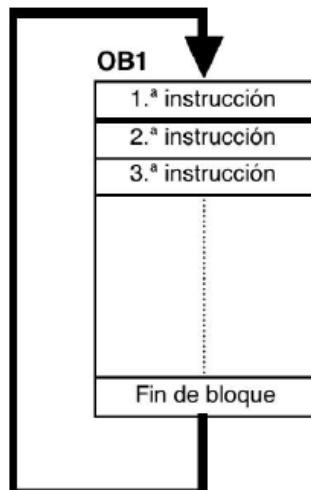
164

Organización de programas

- Programación lineal

- Un programa lineal ejecuta todas las instrucciones de la tarea de automatización de forma secuencial, es decir, una tras otra.

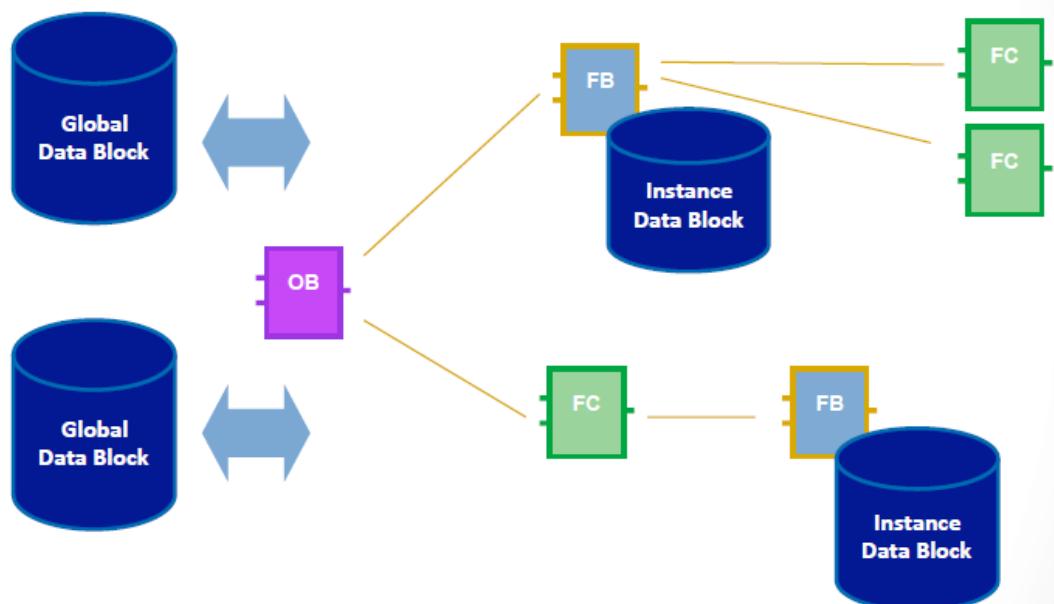
Generalmente, el programa lineal deposita todas las instrucciones del programa en un OB de ciclo (como OB 1), encargado de la ejecución cíclica del programa.



165

Organización de programas

- Programación estructurada



La profundidad máxima de anidamiento es de 16

Organización de programas

- Al crear el programa de usuario para las tareas de automatización, las instrucciones del programa se insertan en bloques lógicos (OB, FB o FC). De manera estándar está disponible el bloque de organización Main [OB1].

OB (bloques de organización)



FC (función)

No tiene bloque de datos de instancia



FB (bloque de función) tiene un bloque de datos de instancia para almacenamiento temporal



DB (bloque de datos) para acceso global y almacenamiento permanente



167

Organización de programas

• Programación estructurada

- En el caso de tareas de control amplias, se subdivide el programa en bloques de programa más pequeños, abarcables y ordenados por funciones. Esto presenta la ventaja de permitir la comprobación de las partes del programa de forma independiente y ejecutarlas como una función global durante el funcionamiento.
- Los bloques de programa deben ser llamados por el bloque de orden superior. Si se detecta un fin de bloque (BE), el programa continuará ejecutándose en el bloque que llama, detrás de la llamada.

• Programación estructurada

- Diseñando FBs y FCs que ejecuten tareas genéricas, se crean bloques lógicos modulares.
- El programa de usuario se estructura luego, de manera que otros bloques lógicos llamen estos bloques modulares reutilizables. El bloque que efectúa la llamada transfiere los parámetros específicos del dispositivo al bloque llamado. Cuando un bloque lógico llama otro bloque lógico, la CPU ejecuta la lógica de programa contenida en el bloque llamado. Una vez finalizada la ejecución del bloque llamado, la CPU reanuda la ejecución del bloque que ha efectuado la llamada. El procesamiento continúa con la ejecución de la instrucción siguiente a la llamada de bloque.

168

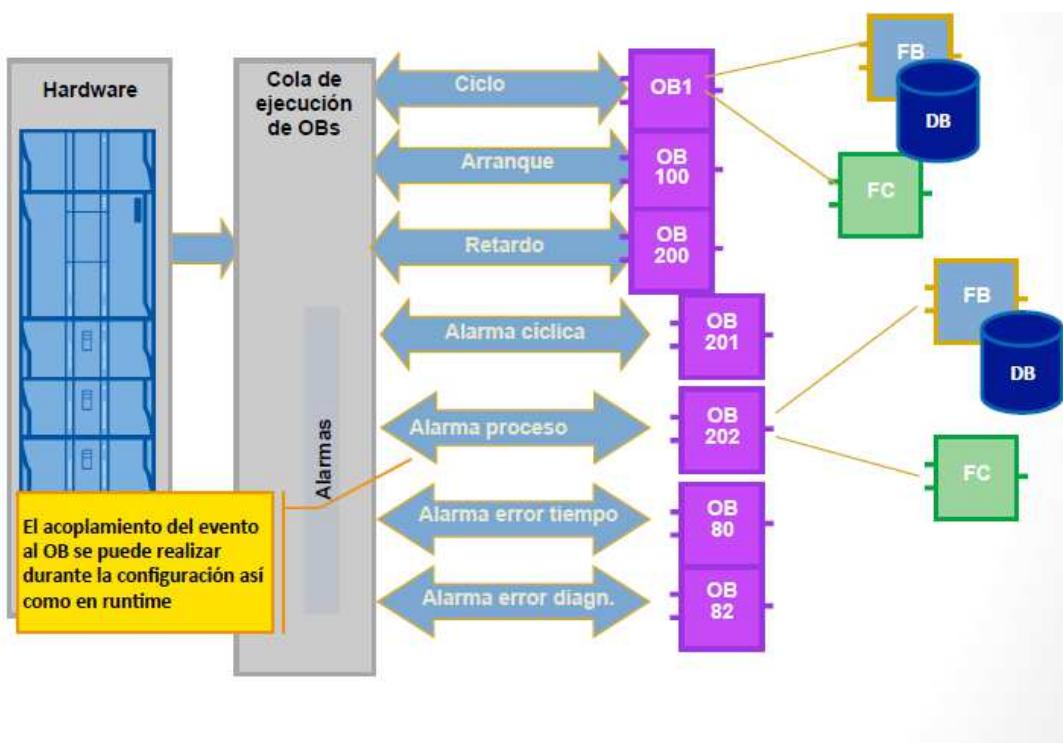
Tipos de módulos

• Bloque de organización (OB)

- Un bloque de organización (OB) reacciona a un evento específico en la CPU y puede interrumpir la ejecución del programa de usuario. El bloque predeterminado para la ejecución cíclica del programa de usuario (OB 1) ofrece la estructura básica y es el único bloque lógico que se requiere para el programa de usuario. Si se incluyen otros OBs en el programa, éstos interrumpen la ejecución del OB 1. Los demás OBs ejecutan funciones específicas, tales como tareas de arranque, procesamiento de alarmas y tratamiento de errores, o ejecución de un código de programa específico en determinados intervalos.
- Los OBs son controlados por eventos. Un evento, p. ej. una alarma de diagnóstico o un intervalo, hace que la CPU ejecute un OB. Algunos OBs tienen eventos de arranque y comportamiento en arranque predefinidos.
- La CPU determina el orden de procesamiento de eventos de alarma según la prioridad asignada a cada OB. Todo evento tiene una prioridad de procesamiento propia. El nivel de prioridad correspondiente dentro de una clase de prioridad determina el orden en que se ejecutan los OB.

169

Tipos de módulos



170

Tipos de módulos

• Función (FC)

- Una función (FC) es un bloque lógico que, por lo general, realiza una operación específica en un conjunto de valores de entrada. La FC almacena los resultados de esta operación en posiciones de memoria. Por ejemplo, las FC se utilizan para ejecutar operaciones estándar y reutilizables (como cálculos matemáticos) o funciones tecnológicas (como para controles individuales que utilizan lógica de bits). Una FC también se puede llamar varias veces en diferentes puntos de un programa. Esto facilita la programación de tareas que se repiten con frecuencia.
- Una FC no tiene ningún bloque de datos instancia asociado (DB). La FC usa la pila de datos locales para los datos temporales utilizados para calcular la operación. Los datos temporales no se almacenan. Para almacenar los datos de forma permanente es preciso asignar el valor de salida a una posición de memoria global, p. ej. el área de marcas o un DB global.

171

Tipos de módulos

• Bloque de función (FB)

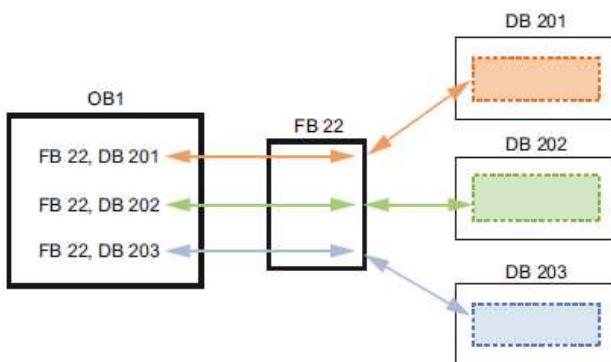
- Un bloque de función (FB) es un bloque lógico que utiliza un bloque de datos instancia para sus parámetros y datos estáticos.
- Los FBs tienen una memoria variable ubicada en un bloque de datos (DB) o DB "instancia". El DB instancia ofrece un bloque de memoria asociado a esa instancia (o llamada) del FB y almacena datos una vez que haya finalizado el FB.
- Es posible asociar distintos DBs de instancia a diferentes llamadas del FB. Los DBs instancia permiten utilizar un FB genérico para controlar varios dispositivos.

172

Tipos de módulos

• Bloque de función (FB)

- La figura siguiente muestra un OB que llama un FB tres veces, utilizando un bloque de datos diferente para cada llamada. Esta estructura permite que un FB genérico controle varios dispositivos similares (p. ej. motores), asignando un bloque de datos instancia diferente a cada llamada de los distintos dispositivos. Cada DB instancia almacena los datos (p. ej. velocidad, tiempo de aceleración y tiempo de operación total) de un dispositivo en particular.



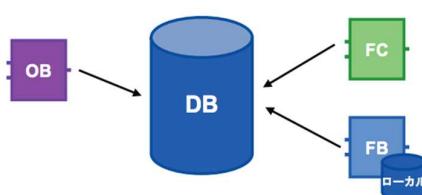
En este ejemplo, el FB 22 controla tres dispositivos diferentes. El DB 201 almacena los datos operativos del primer dispositivo, el DB 202, los del segundo y, el DB 203, los del tercero.

173

Tipos de módulos

• Bloque de datos (DB)

- Los bloques de datos (DB) se crean en el programa de usuario para almacenar los datos de los bloques lógicos. Todos los bloques del programa de usuario pueden acceder a los datos en un DB global. En cambio, un DB instancia almacena los datos de un bloque de función (FB) específico.
- Los datos almacenados en un DB no se borran cuando finaliza la ejecución del bloque lógico asociado. Hay dos tipos de DBs, a saber:
 - Un DB global almacena los datos de los bloques lógicos en el programa. Cualquier OB, FB o FC puede acceder a los datos en un DB global.
 - Un DB instancia almacena los datos de un FB específico. La estructura de los datos en un DB instancia refleja los parámetros (Input, Output e InOut) y los datos estáticos del FB. (La memoria temporal del FB no se almacena en el DB instancia.)



174

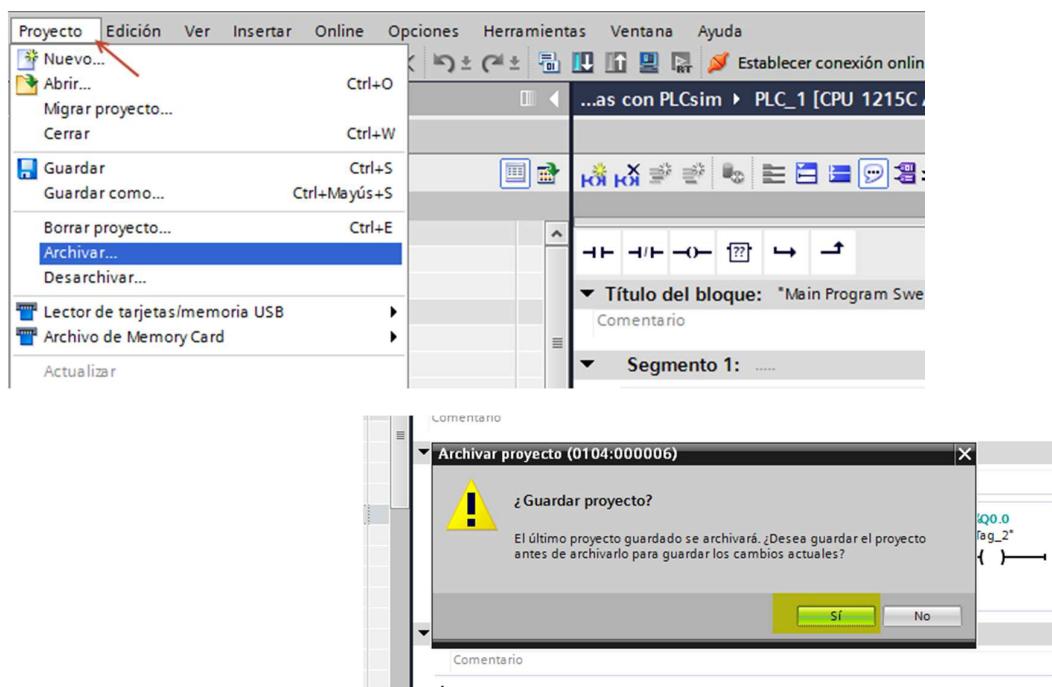


Archivar / Desarchivar

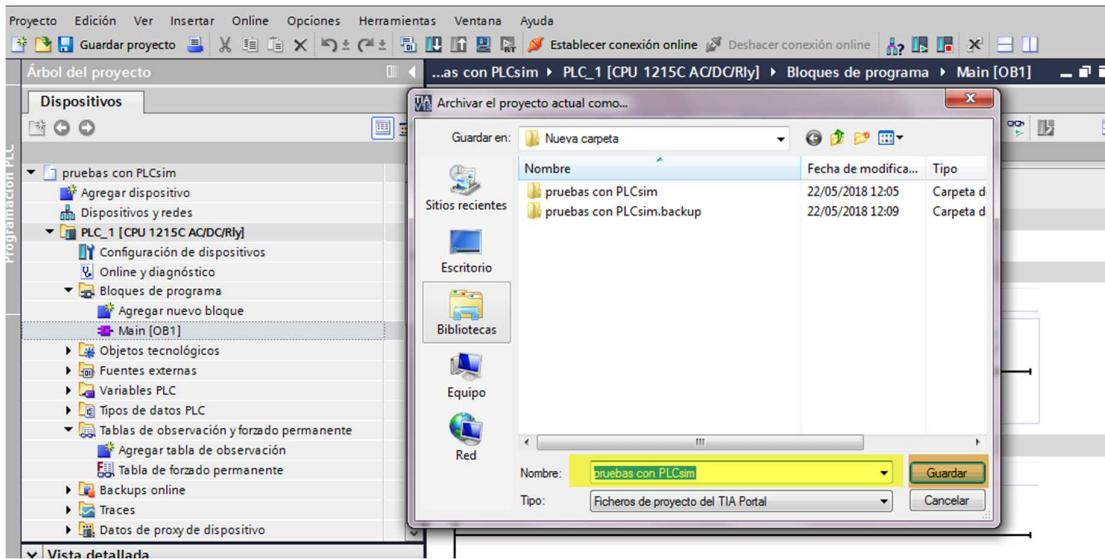
Visión Práctica S1200

175

La opción archivar y desarchivar la utilizamos para archivar un documento con la intención de resumirlo todo en una carpeta, la cual sea útil para almacenar , enviar a otras personas, etc...

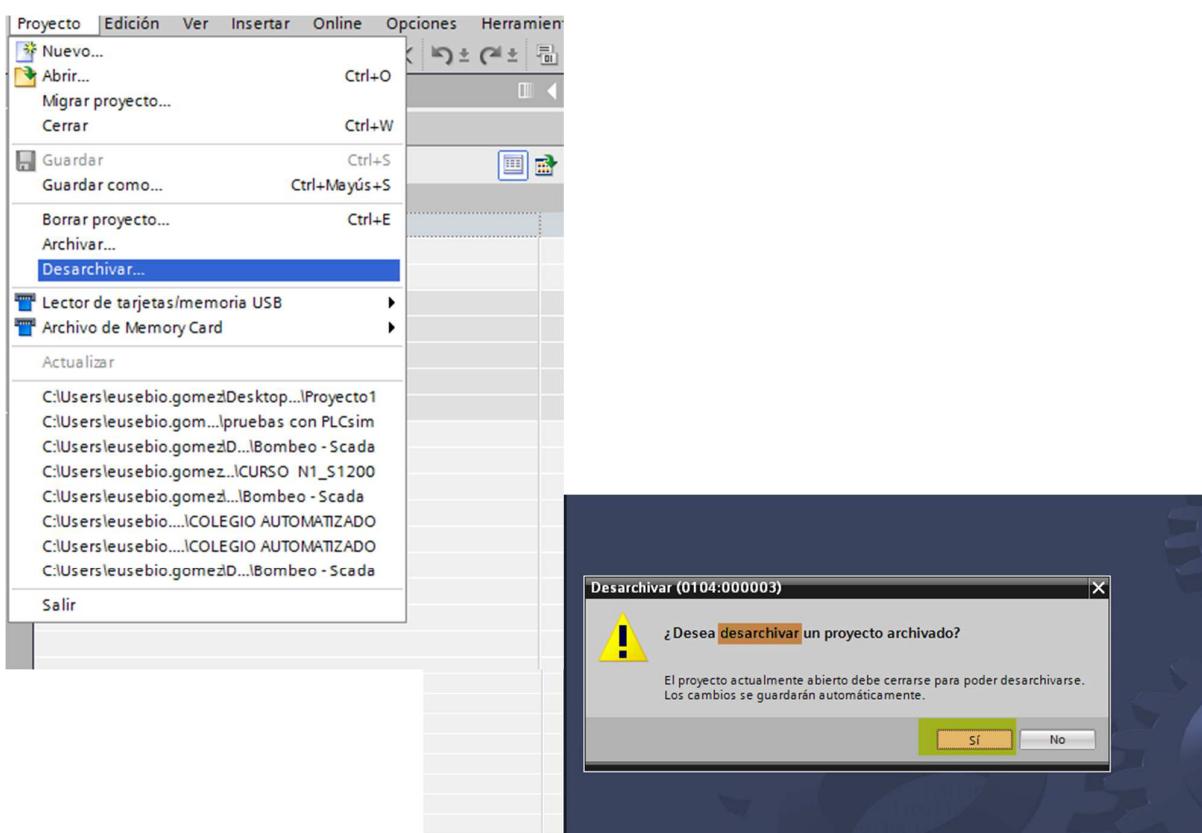


176

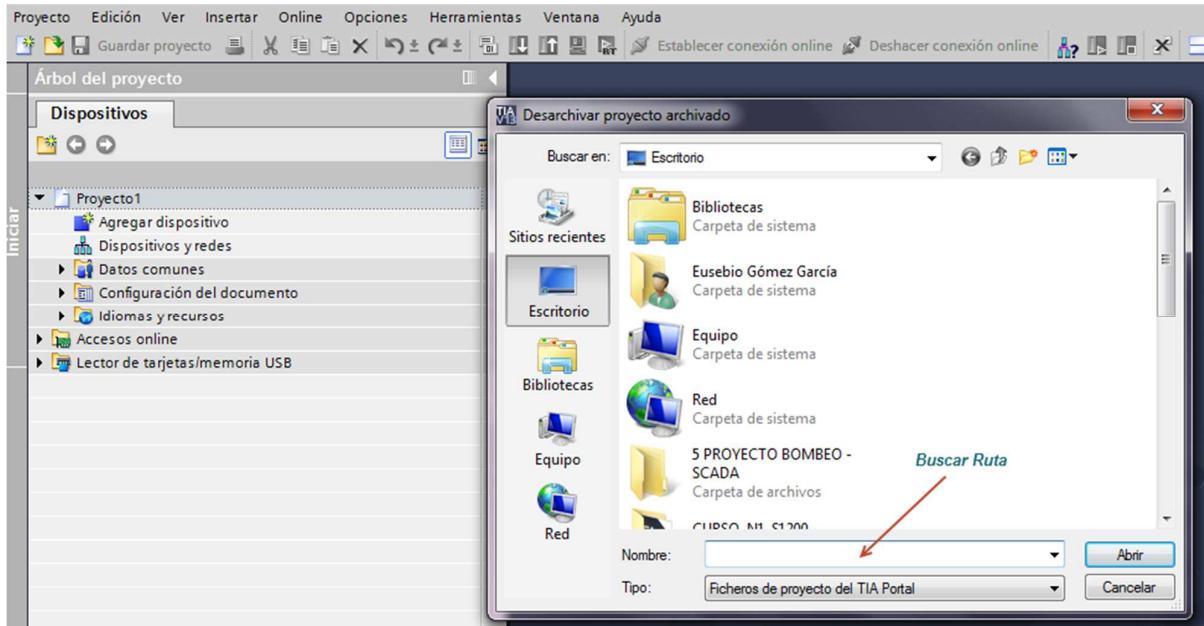


Una vez archivado el proyecto, nos genera una carpeta comprimida con todas las propiedades del proyecto. Elegimos una ruta donde ubicar el archivo. El archivo queda a disposición del programador para su uso de almacenamiento, envío, etc..

177



178



Una vez seleccionada la ruta donde se encuentra el proyecto archivado, el programa los desarchiva y lo pone a disposición del TIA portal para su uso.

179

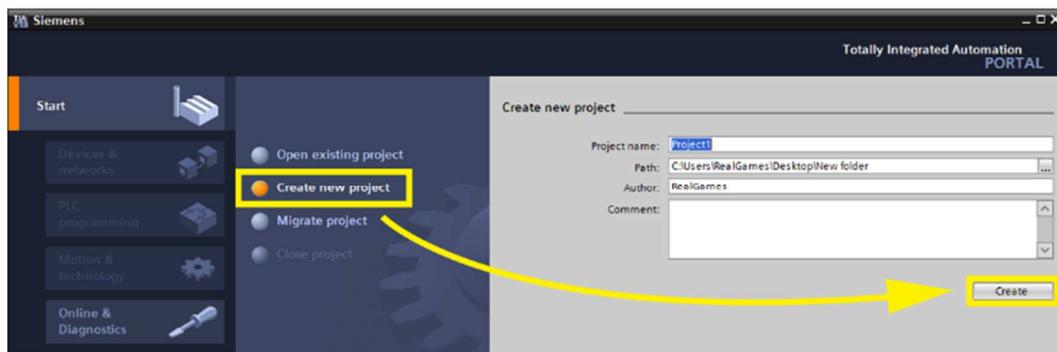


CONEXIÓN S1200 con FACTORY I/O Visión Práctica S1200

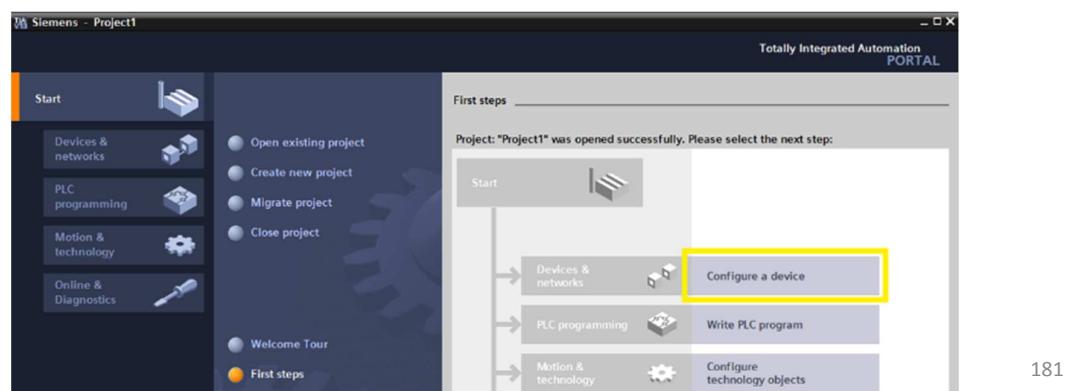
180

COMUNICACIÓN ENTRE PC Y PLC

- 1.- Conectar el PLC a la red.
- 2.- Crear un nuevo proyecto en TIA Portal.

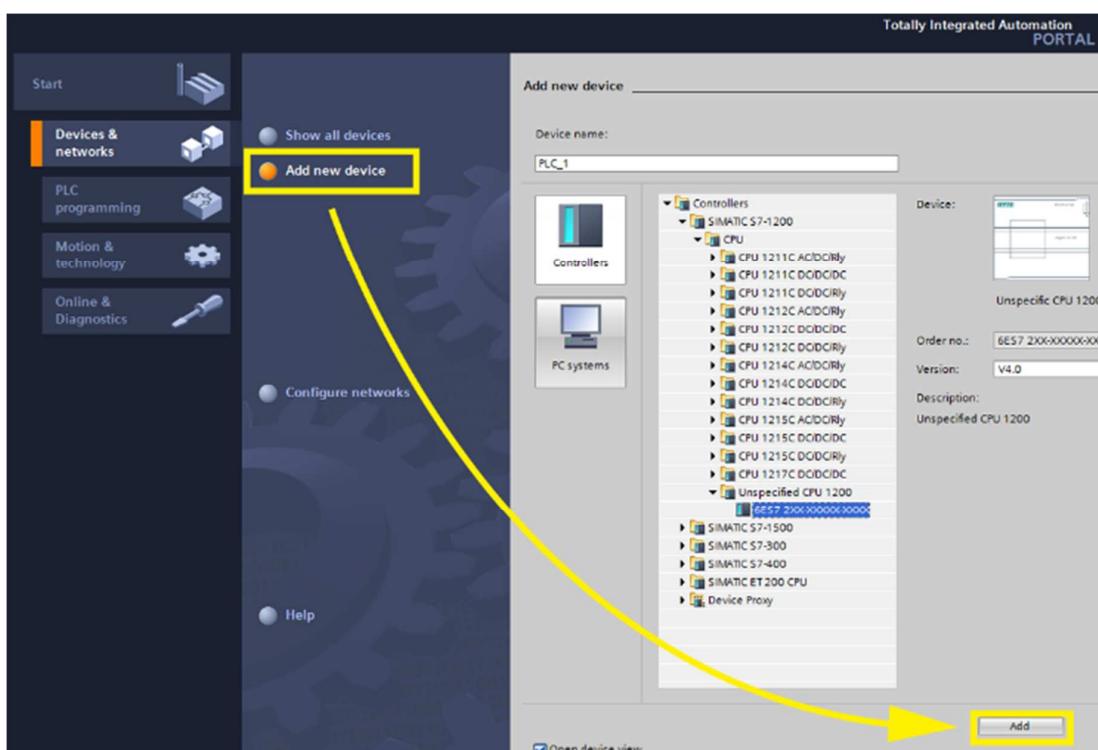


- 3.- Seleccionar configuración de dispositivos.



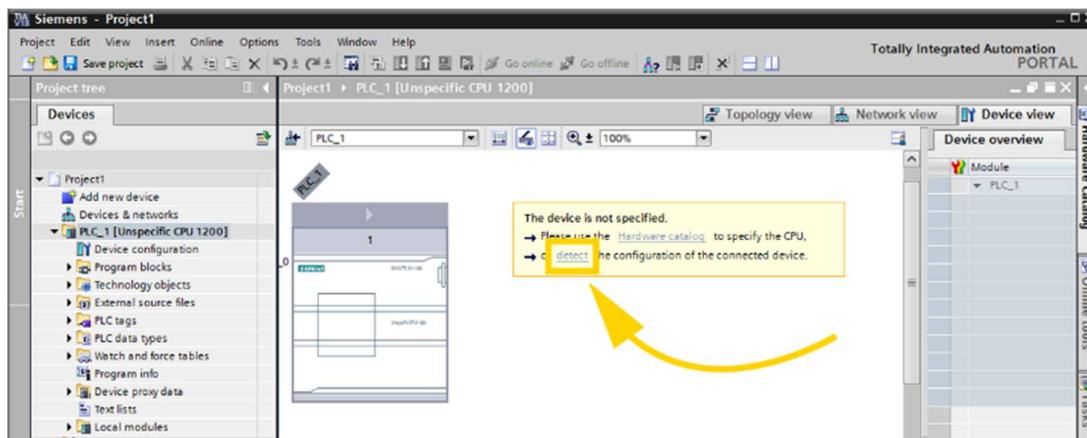
181

- 4.- Clicar en Añadir dispositivos nuevos y elegir el controlador que necesitemos, en nuestro caso un CPU 1200.

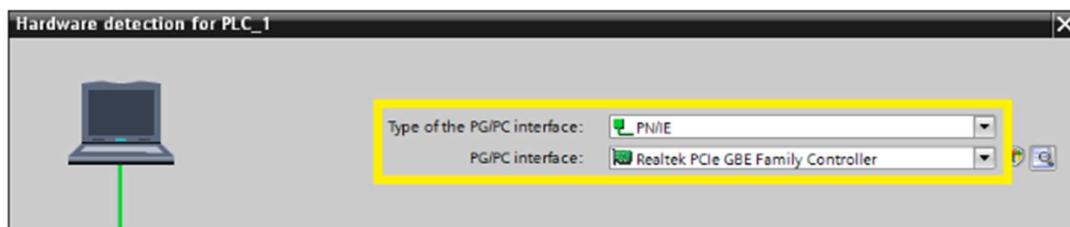


182

5.- Podemos utilizar la opción de detectar automáticamente el hardware de la CPU con la que queremos conectar. Esta operación es una opción.

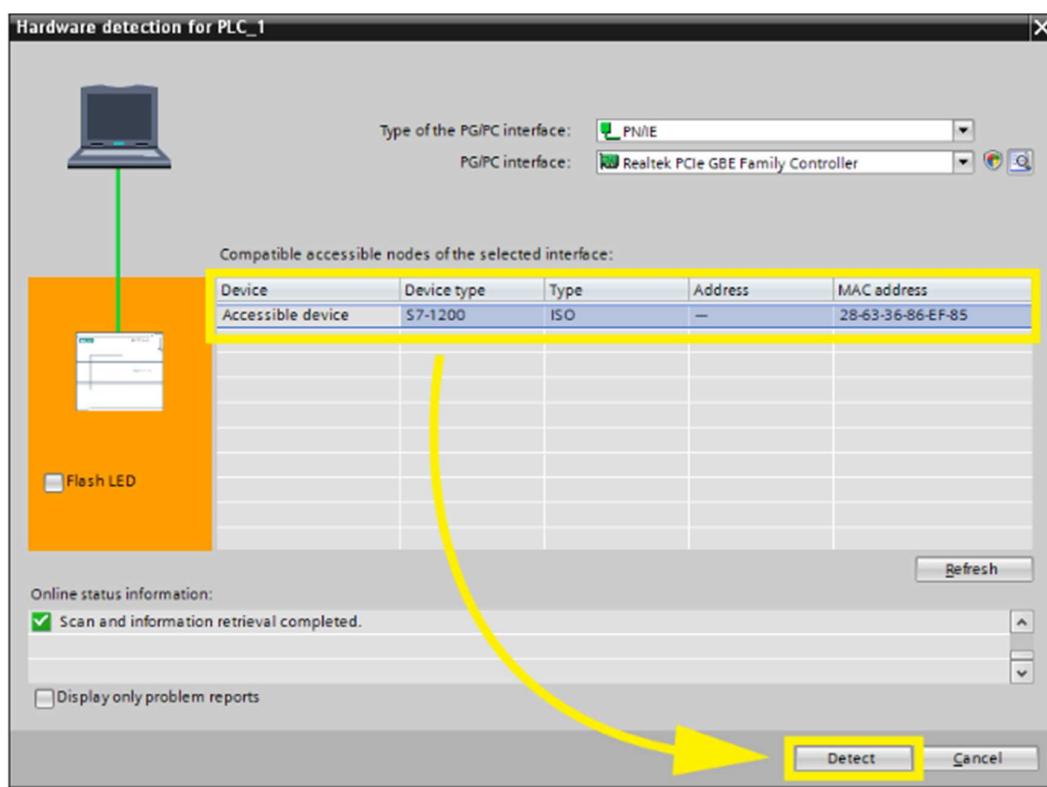


6.- Elige PN/IE y el interfaz del adaptador de red para conectar con el PLC.



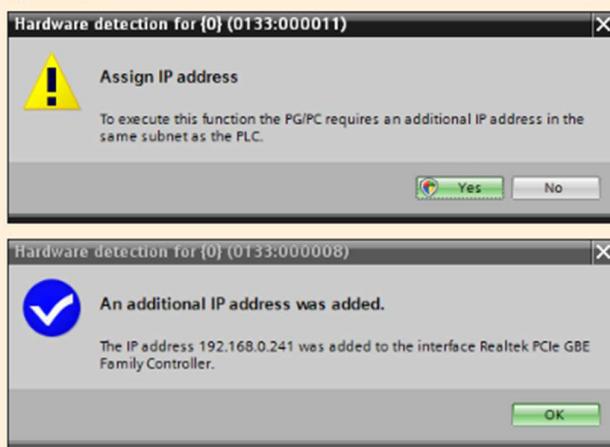
183

7.- Al scanear la red seleccionamos el PLC de la lista para conectar.



184

⚠ If you are adding a PLC that is not in the same subnet as your computer, you will be prompted about assigning a new IP address to the network interface; click on Yes.



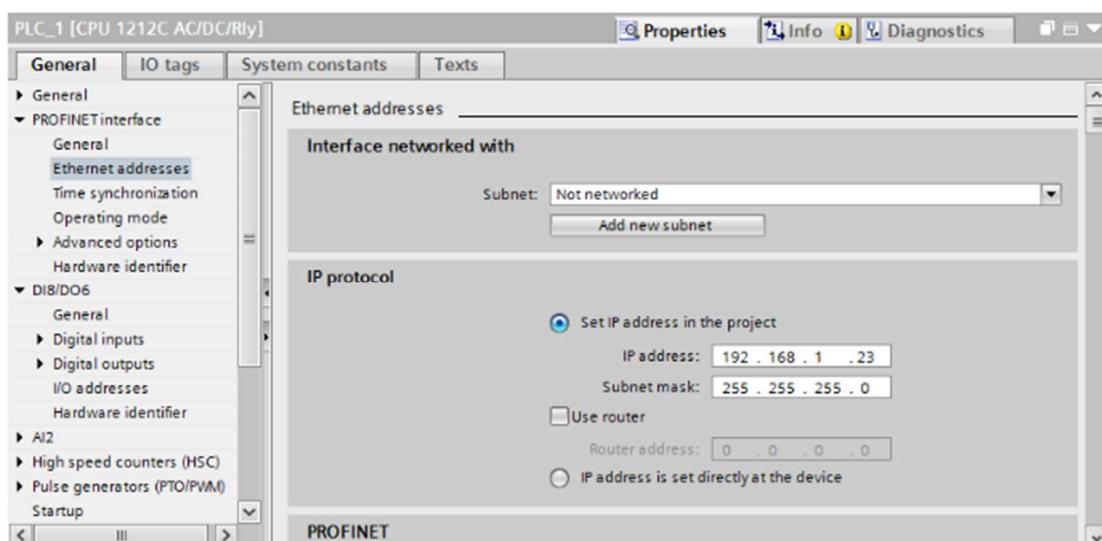
Later you may have to change the PLC's IP address to one in the same subnet as your computer, otherwise FACTORY I/O might not be able to connect to it.



8.- El PLC es detectado para el proyecto. Algunas propiedades serán necesario configurar para que las comunicaciones con el programa Factory I/O sea posible.

185

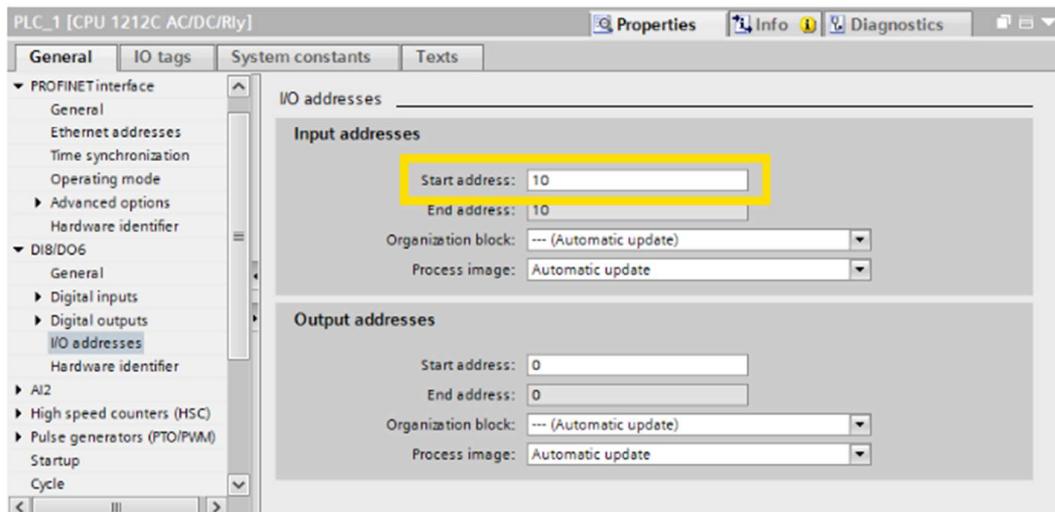
9.- Asignamos al PLC la dirección IP. Esta acción se realiza en General / Propiedades.



⚠ If you were prompted about assigning a new IP address to the network interface, you should now assign an IP address to the PLC that is in the same subnet as your computer.

186

10. PLC physical inputs use by default the first memory addresses of %I. For FACTORY I/O to be able to write sensors values to %I you must offset the input addresses, we recommend an offset of 10. Click on I/O addresses and change the Input addresses > Start address to 10.

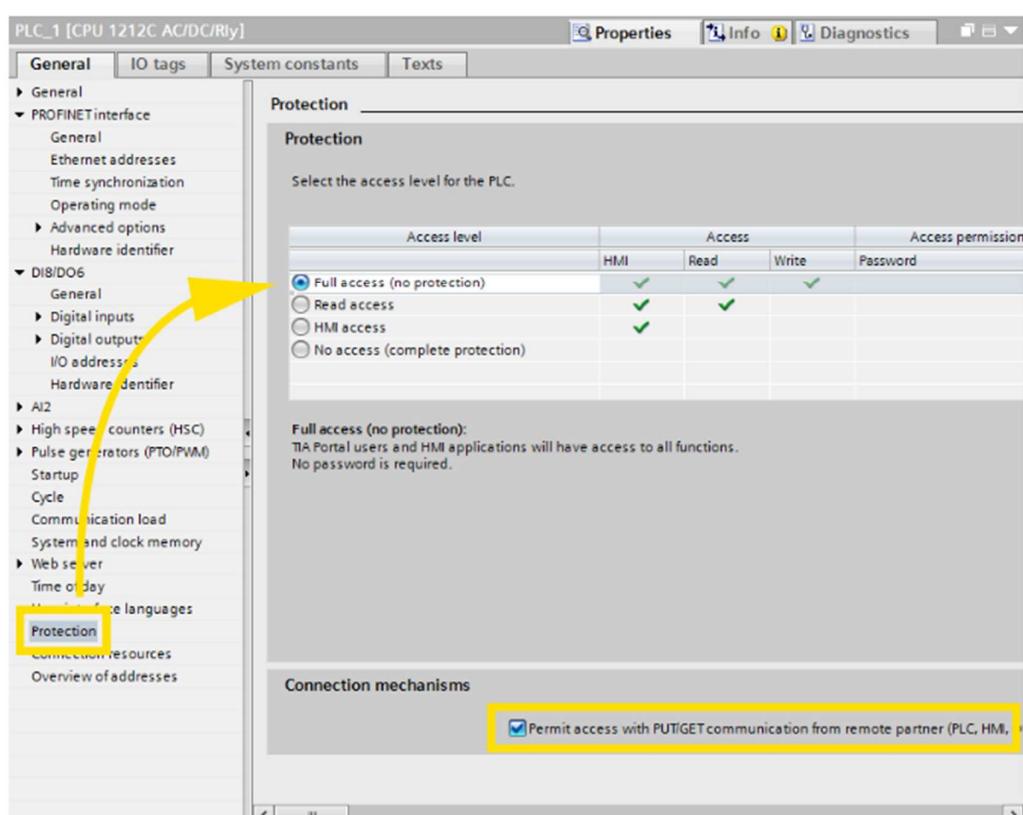


FACTORY I/O should not use input addresses that are assigned to physical inputs

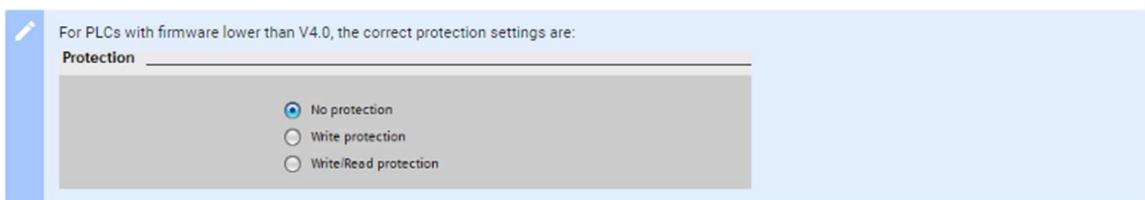
Otherwise, the values written by FACTORY I/O will be overwritten as the state of physical inputs is copied to I memory.

187

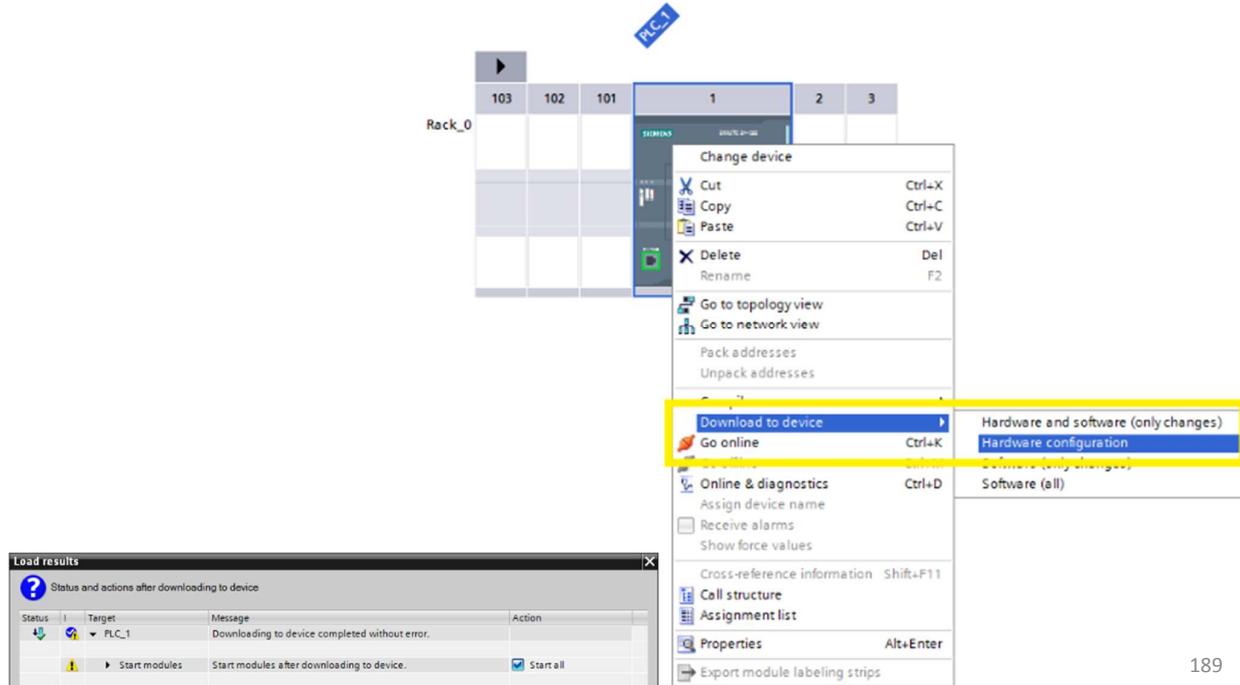
11. Finally, you must give full access to the PLC memory. Click on Protection and select Full access (no protection). Also, under Connection mechanisms check Permit access with PUT/GET communication from remote partner.



188



12. Right-click on the device and select Download to device > Hardware configuration. Next, Start the CPU.



189

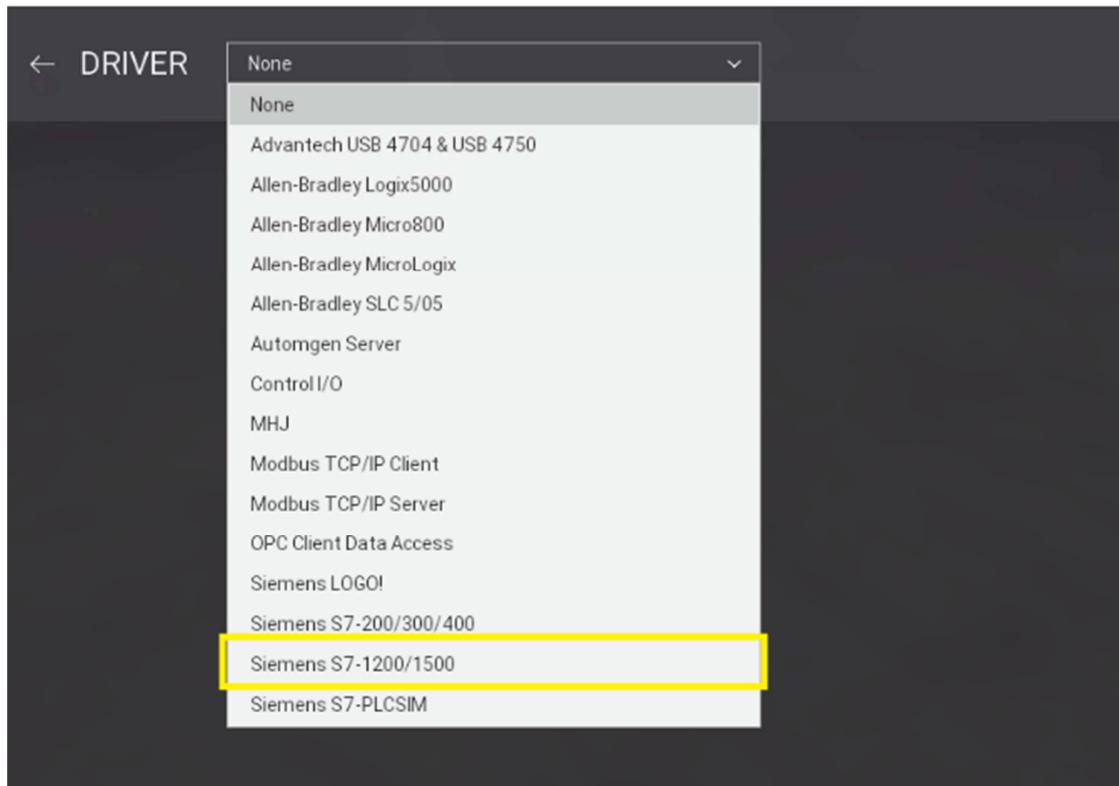
CONECCIÓN FACTORY I/O CON EL PLC

1.- En Factory I/O clicar en FILE / Para configurar los drives del dispositivo con el que vamos a trabajar.



190

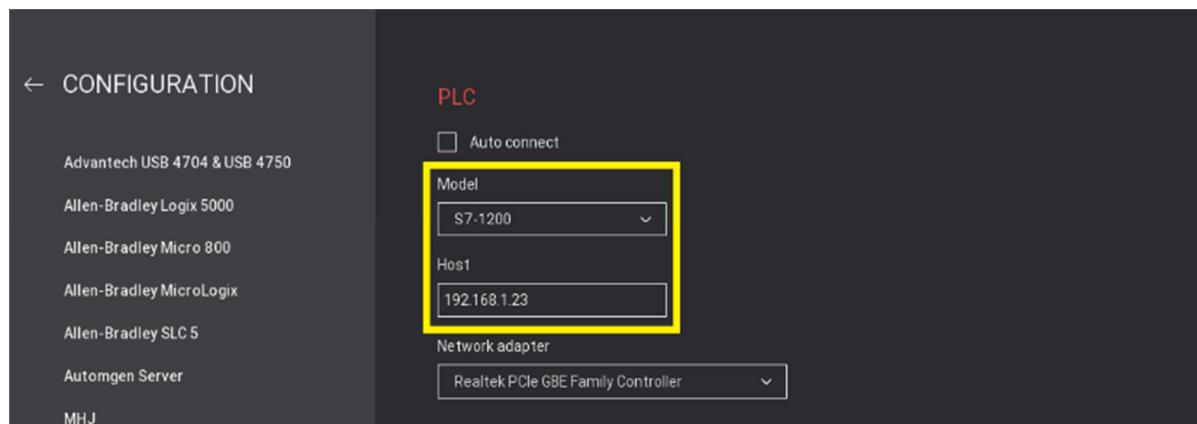
2.- Seleccionamos el Driver de S7-1200/1500 de la lista desplegable.



191

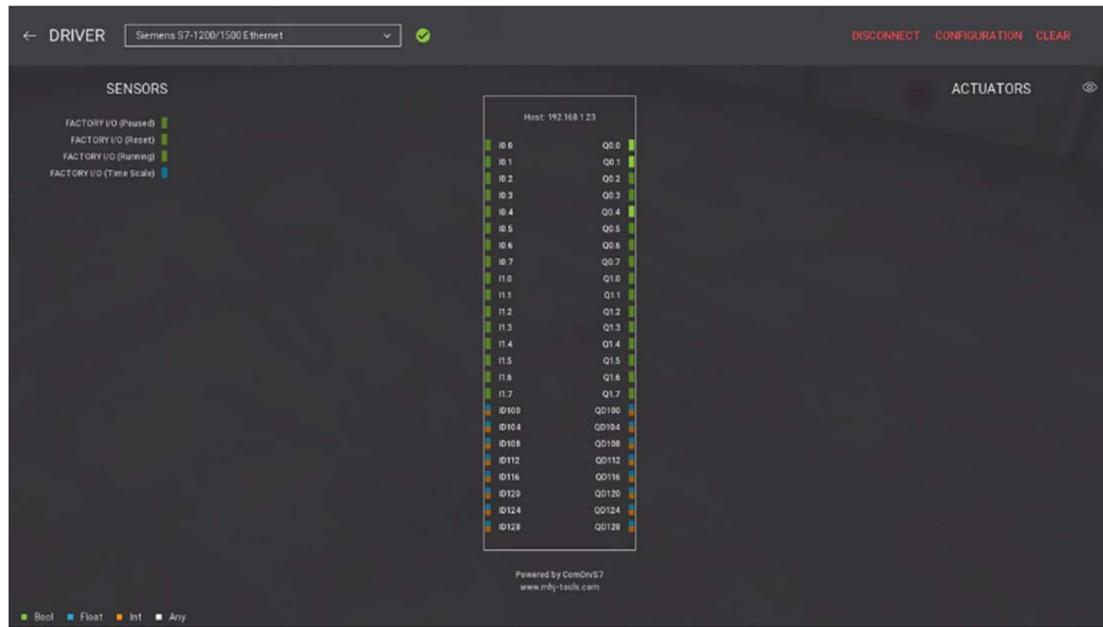
3.- Abrir el panel de configuración para establecer los parámetros de modelo de PLC y de direcciones IP para la comunicación.

4.- Elegir en este caso S7 1200 y colocar la dirección IP física del PLC real, en el ejemplo de la figura es 192.168.1.23



192

5.- Presionar ESC para retornar al menú principal de los drives. Seguidamente pulsamos en CONECTAR para establecer una conexión entre el programa y el PLC real. Si la conexión es OK se enciende una marca de color verde.



193

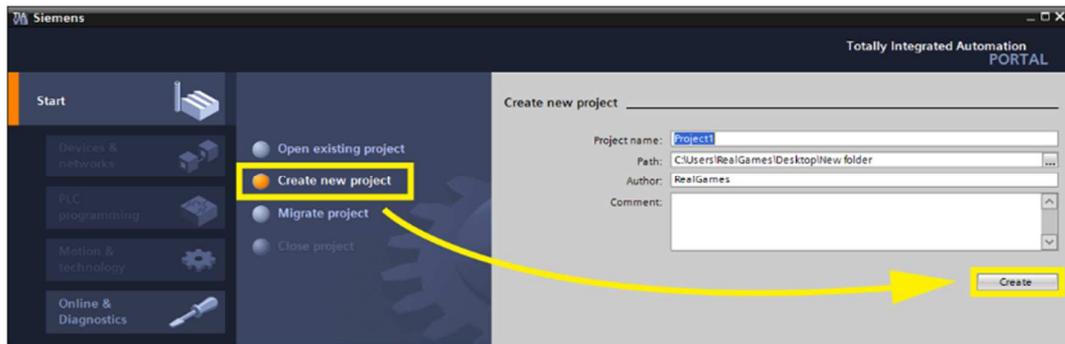


CONEXIÓN PLCSIM con FACTORY I/O Visión Práctica S1200

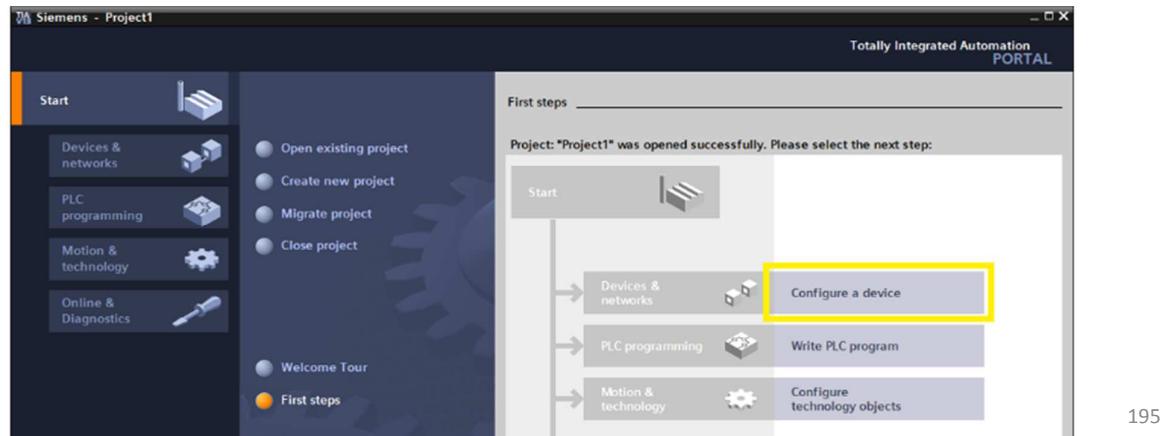
194

Activar el PLCSIM con TIA Portal

1.- Crear un proyecto nuevo en TIA POrtal

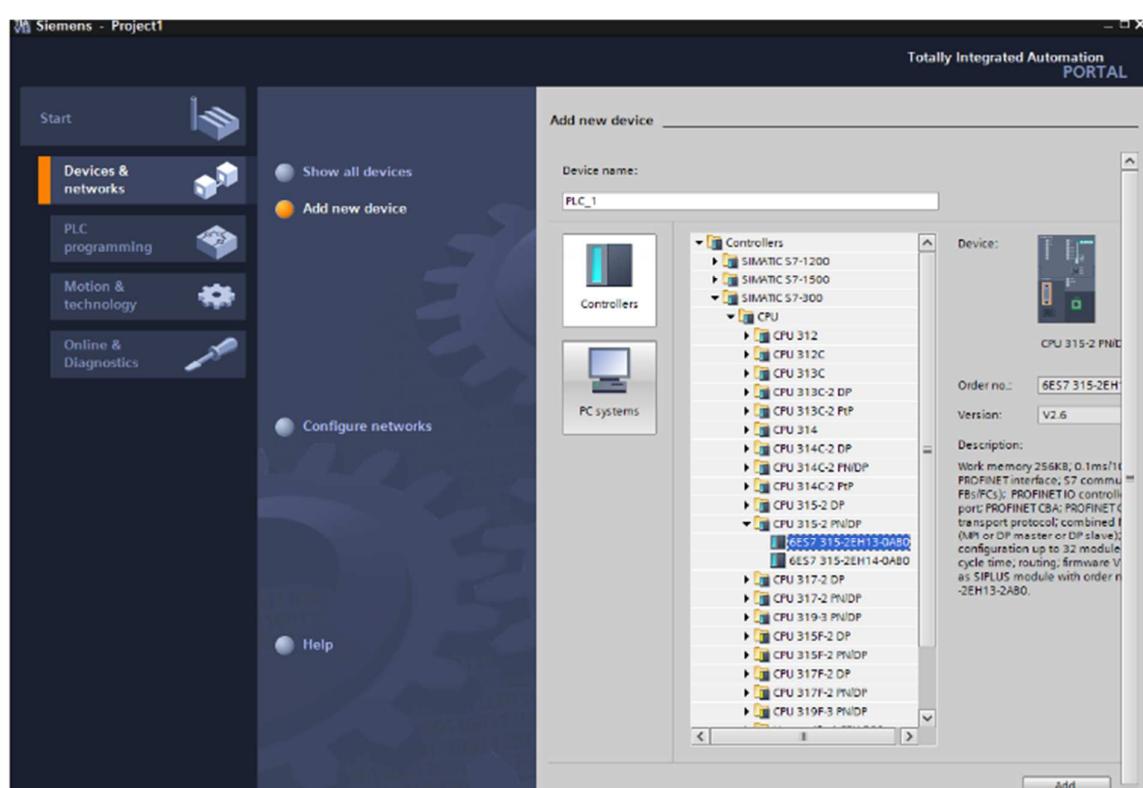


2.- Seleccionar Configuración de dispositivos.



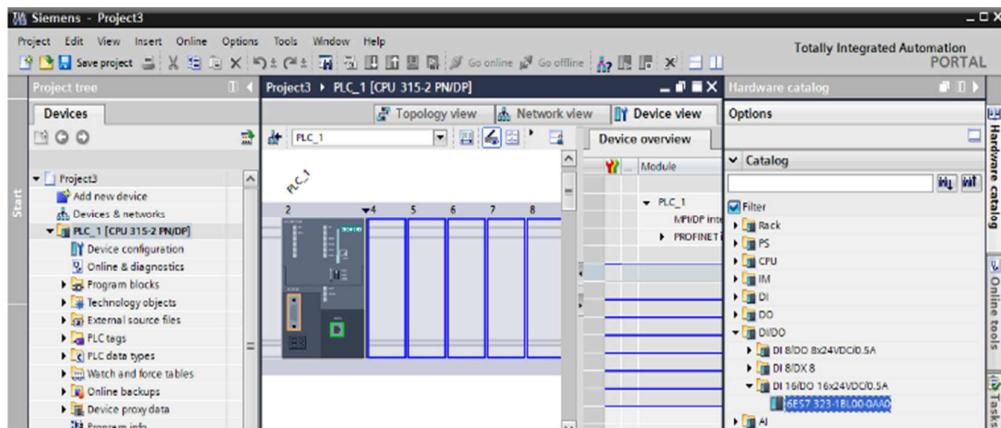
195

3.- Clicar en nuevo dispositivos y elegir el controlador que necesitemos.

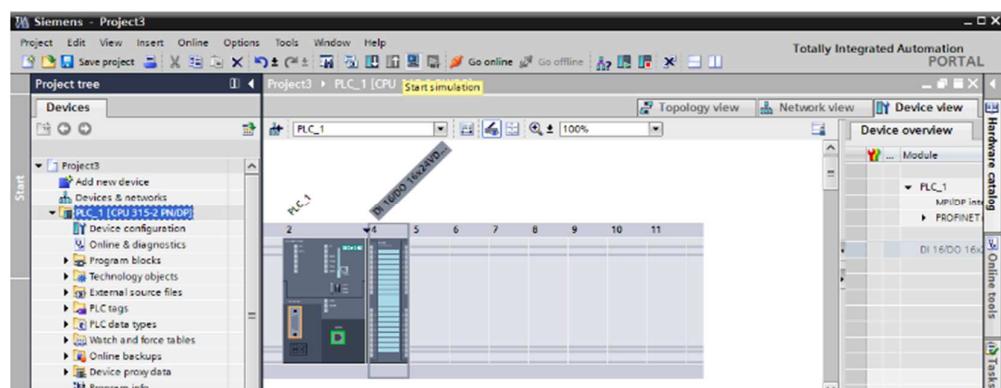


196

4.- Configurar los módulos necesarios para el bastidor

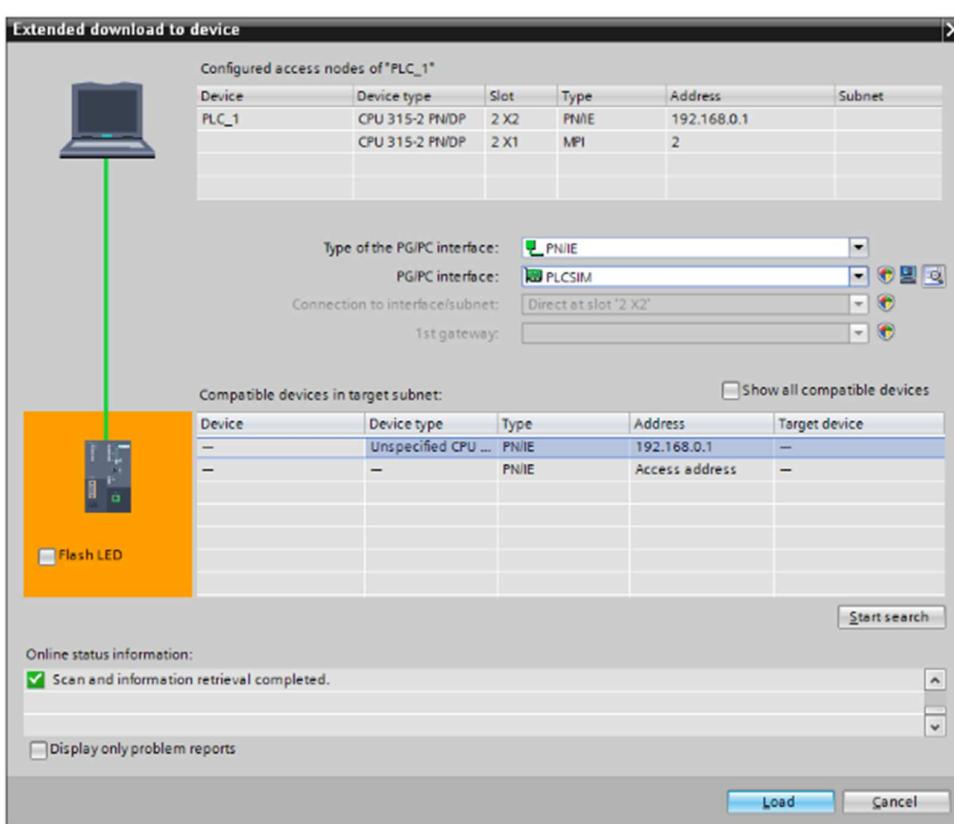


5.- Activar la simulación del proyecto presionando el botón de simulación.



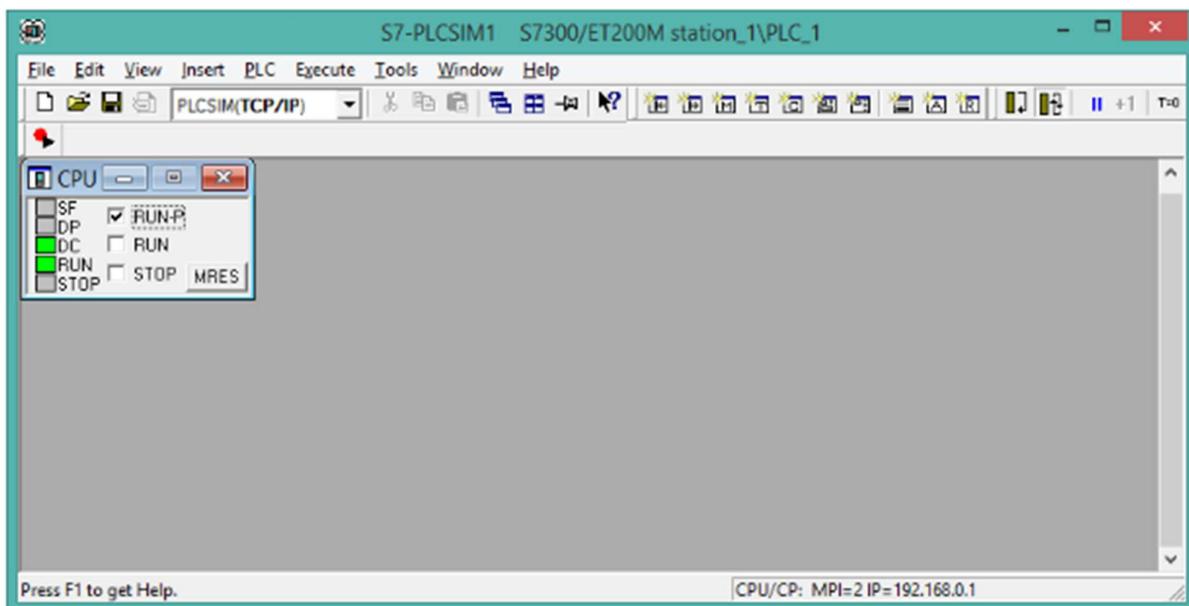
197

6.- Conectar con el PLCSIM y ponerse online para verificar que la conexión es OK.



198

7.- Poner el Simulador en RUN



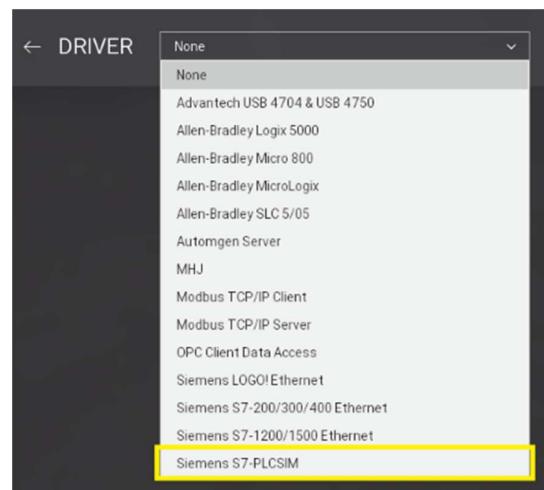
199

CONECTAR FACTORY I/O CON S7-PLCSIM

- 1.- Entrar en el menú FILE para configurar los drivers del dispositivo con el que trabajaremos.

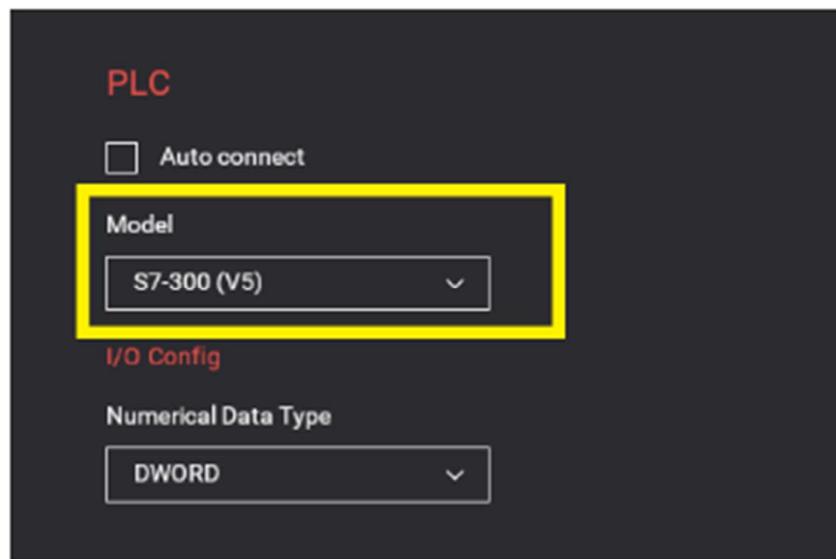


- 2.- Seleccionar el simulador Siemens S7-PLCSIM en la lista de drivers.



200

- 3.- Abrir el panel de configuración de drivers clicando en CONFIGURACIÓN
- 4.- Marcar el dispositivo con el que estamos trabajando en la lista desplegable que aparece en modelo.



- 5.- Presionar ESC para retornan al menú principal de drivers. Ahora clicar en CONECTAR para hacer la conexión entre el programa Factory I/O y el simulador de siemens.