JUAN CARLOS MARTÍN CASTILLO

ELECTRICIDAD Y ELECTRÓNICA

Sistemas secuenciales programables

EDITEX

JUAN CARLOS MARTÍN CASTILLO

Sistemas secuenciales programables

EDITEX

ÍNDICE

ORGANIZACIÓN DE LA UNIDAD

Cada unidad didáctica se inicia con una gran imagen motivadora, un breve índice de contenidos con los epígrafes que presenta la unidad en el apartado **Vamos a conocer**, y los objetivos a alcanzar al término de la misma en el apartado **Y al finalizar esta unidad**.

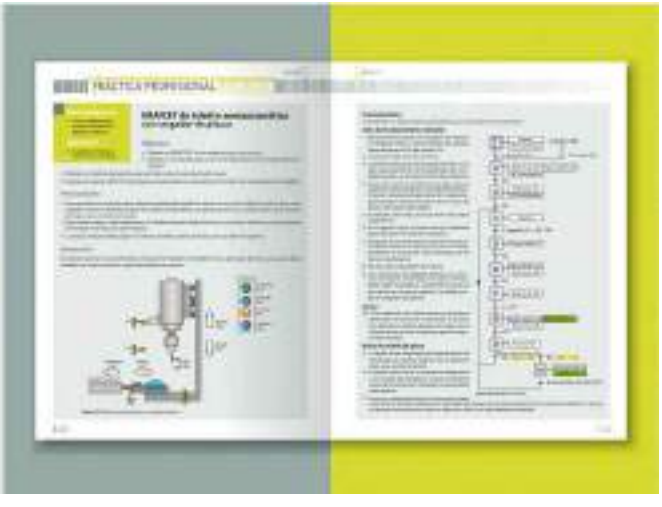
A continuación comienza el desarrollo de contenidos. Para apoyar y reforzar los contenidos se presentan ejemplos, tablas, esquemas y numerosas ilustraciones, seleccionadas de entre los equipos y herramientas más frecuentes que te vas a encontrar al realizar tu trabajo.

En los márgenes se desarrollan multitud de textos complementarios de ampliación de información, consejos de seguridad, vocabulario técnico, diccionario español-inglés y enlaces web, que permiten profundizar en los conocimientos expuestos.

A lo largo del texto se incorporan casos, ejemplos y actividades prácticas. Estas actividades por un lado, ayudan a asimilar los conceptos, y por otro, promueven la realización en el taller de los procesos explicados.

En la sección **Práctica profesional resuelta**, se plantea el desarrollo de un caso práctico, en el que se describen las operaciones que se realizan, se detallan las herramientas y el material necesario, y se incluyen fotografías que ilustran los pasos a seguir.

Estas prácticas profesionales representan los resultados de aprendizaje que debes alcanzar al terminar tu módulo formativo.

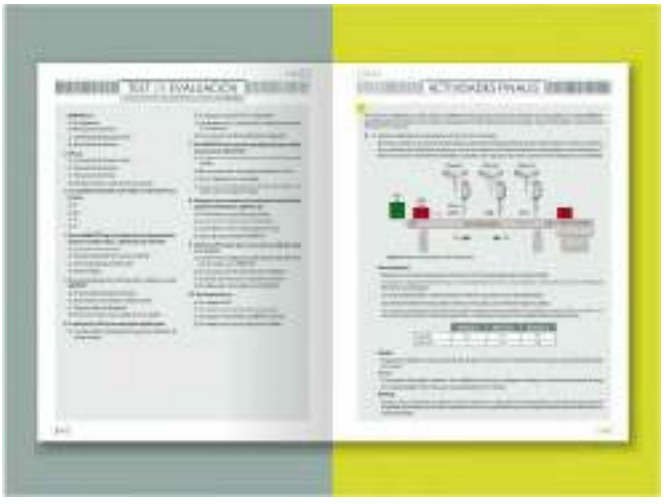


IMPORTANTE: Todas las actividades propuestas en este libro deben realizarse en un cuaderno de trabajo, nunca en el propio libro.

Regístrate en nuestra web y accede a los recursos adicionales: <www.editex.es>.

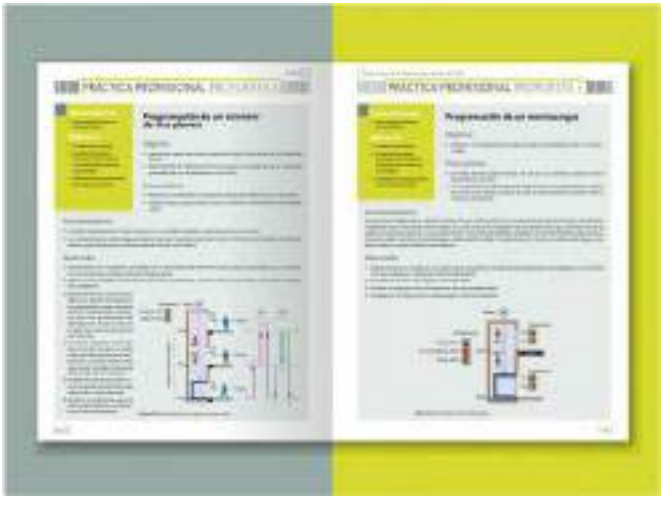
El **Test de evaluación** consta de una batería de preguntas centradas en los conceptos más importantes de la unidad. Este test permite comprobar el nivel de conocimientos adquiridos tras el estudio de la misma.

Tras ello se proponen una serie de **Actividades finales** para aplicar y repasar los conceptos y procedimientos explicados a lo largo de la unidad.

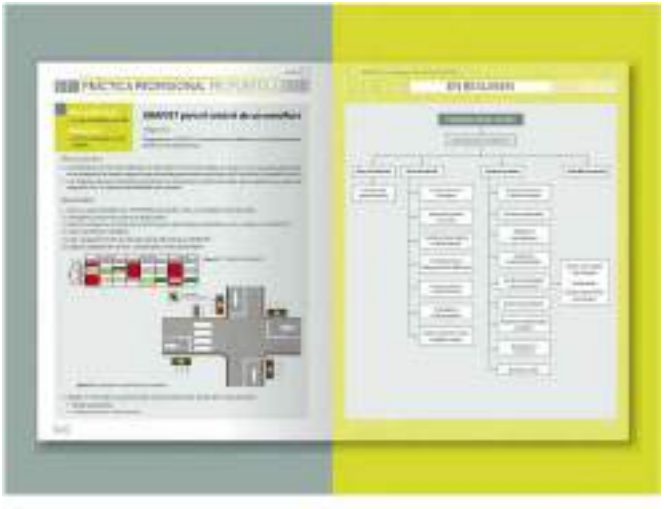


En la **Práctica profesional propuesta** se plantean actividades prácticas y, al igual que en la Práctica profesional resuelta, se detallan las herramientas y el material necesario para su desarrollo.

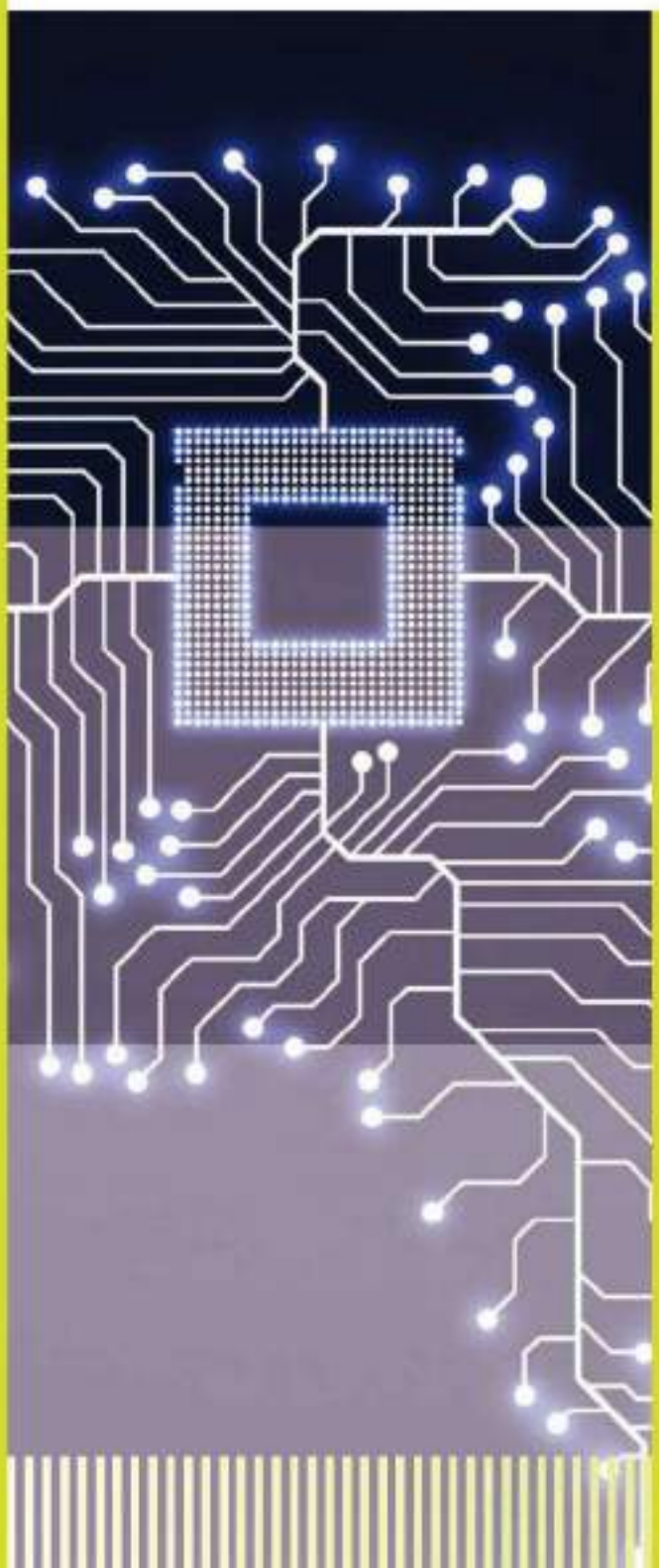
Con la práctica profesional propuesta se pretende potenciar tu autonomía y tu espíritu emprendedor, fomentando la metodología de aprender haciendo. Puedes descargarte estas páginas profesionales propuestas y otros recursos si te registras en nuestra web: <www.editex.es>



La unidad finaliza con el apartado **En resumen**, un mapa conceptual que relaciona los conceptos claves de la unidad. Este apartado sirve para recapitular todo lo tratado en la unidad.



1 Lógica digital



Vamos a conocer...

1. Tipos de señales
2. Sistemas y códigos de numeración.
3. Lógica digital
4. Circuitos secuenciales

PRÁCTICA PROFESIONAL RESUELTA

Análisis, simplificación y simulación de un circuito lógico

PRÁCTICAS PROFESIONALES PROPUESTAS

1. Comprobación de un circuito con puertas lógicas
2. Obtención de circuitos simplificados desde una tabla de la verdad
3. Resolución de problemas de lógica combinacional
4. Circuitos de lógica secuencial con biestables

Y al finalizar esta unidad...

- Identificarás números escritos en distintos sistemas y códigos de numeración.
- Conocerás los conceptos básicos de la lógica digital para representar y simplificar circuitos lógicos combinacionales.
- Sabrás cuáles son las puertas lógicas básicas, sus expresiones lógicas y su tabla de la verdad.
- Conocerás en qué consisten los circuitos lógicos combinacionales.
- Montarás y simularás circuitos de lógica digital combinacionales y secuenciales.

1. Tipos de señales

En un proceso industrial las señales pueden ser de dos tipos: digitales y analógicas.

Las **señales de tipo digital** trabajan con valores discretos, que solamente pueden tener dos estados o niveles posibles: todo o nada, 1 o 0.

Las **señales analógicas** trabajan con valores continuos, dentro de un rango entre un valor mínimo y un valor máximo.

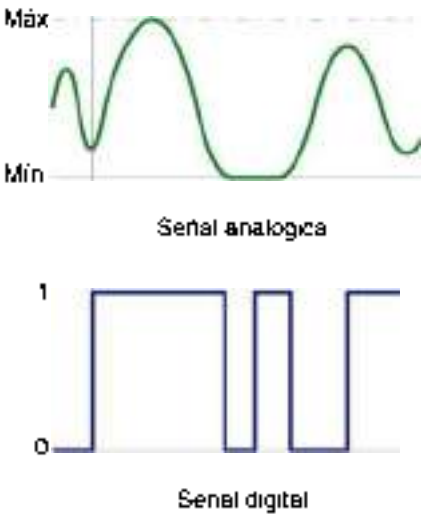


Figura 1.1. Tipos de señales.

Las señales digitales son las utilizadas en la lógica digital y son las que se estudiarán a continuación. Al final de este libro abordaremos el estudio básico y la aplicación de las señales de tipo analógico en los sistemas secuenciales programables.

Los estados de una señal digital se originan por niveles de tensión, denominados niveles lógicos, en los que, por lo general, el valor máximo se corresponde con el 1 y el valor mínimo con el 0.

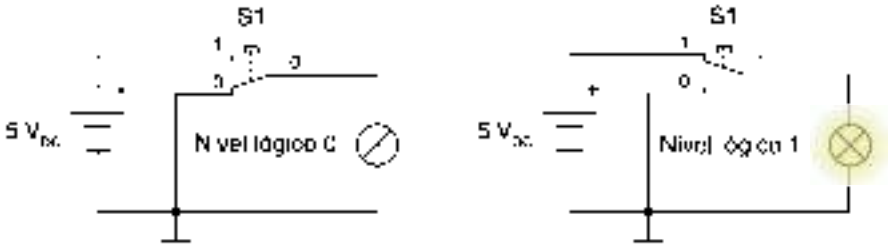


Figura 1.2. Niveles lógicos de una señal digital.

Los nombres que reciben los valores lógicos son: *verdadero-falso* (*true-false*), *alta-baja* (*high-low*), *todo-nada*.

El estudio de la lógica digital se basa en el **álgebra de Boole**, ya que con ella es posible la resolución de las operaciones lógicas y su posterior aplicación a diferentes circuitos o dispositivos de programación.

Saber más

George Boole fue un matemático británico del siglo XIX que desarrolló el álgebra que lleva su nombre y que actualmente se utiliza para estudiar todo lo relacionado con la electrónica digital y la informática.

2. Sistemas y códigos de numeración

2.1. Sistema de numeración

Un **sistema de numeración** está formado por un grupo de símbolos y unas normas que permiten componer números. De esta forma es posible expresar cantidades descifrables por las personas o por los sistemas electrónicos.

Los sistemas de numeración mayormente utilizados son:

- Decimal.
- Binario.
- Hexadecimal.
- Octal.

El número de símbolos de que dispone un sistema de numeración se denomina base; por lo tanto, a partir de ahora, al decir que un sistema de numeración está escrito en una base determinada estaremos hablando del número de dígitos que se utilizan para formar los números.

2.1.1. Sistema de numeración decimal

Es el sistema de numeración más utilizado en la vida cotidiana, ya que emplea diez símbolos que corresponden a la cantidad de dedos que los humanos tenemos en nuestras manos. Por tanto, tiene base diez y está compuesto por los siguientes símbolos: 0-1-2-3-4-5-6-7-8-9, cuya combinación permite formar cualquier tipo de número fácilmente entendible por las personas (12, 103, 6129, etc.).

Existen otros sistemas y códigos de numeración mucho más adecuados para trabajar y operar en los sistemas digitales y microinformáticos.

2.1.2. Sistema de numeración binario

Solamente utiliza dos símbolos, el 1 y el 0, por lo que la base del sistema de numeración es el dos. Es el sistema de numeración empleado en los sistemas digitales y, por tanto, el que mejor se adapta al estudio de la lógica digital.

Número en binario	Número en decimal
011100	28
10	2
101	5

Los números en binario se forman por una combinación de ceros y unos, y se leen dígito a dígito y de forma individual. Así, todos los números en binario tienen su equivalente en decimal y viceversa.

En un número binario cada dígito se denomina *bit*. Por tanto, los números en este sistema se forman en función de la cantidad de bits que los componen. Así, la cantidad de números que se pueden formar en binario, se calcula elevando el número de la base, que en este caso es 2, al número de bits de dicho número.

Por ejemplo, con tres bits podríamos formar los siguientes números:

Número en binario	Número en decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
110	7

Vocabulario

El bit de mayor peso también se denomina *bit más representativo* y el de menor peso *bit menos representativo*.

En un número binario, cada bit tiene un peso, siendo el bit de menor peso el que se encuentra a la derecha y el de mayor peso a la izquierda. Leyendo el número de derecha a izquierda, el peso de cada bit es el doble que el del anterior.

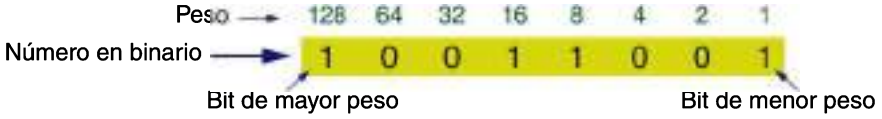


Figura 1.3. Peso de los bits de un número en binario.

Una forma sencilla de convertir un número binario, no demasiado grande, en decimal es sumar los pesos de aquellos bits que se encuentran a 1 en el número en binario. Así, el resultado de la suma es el número en decimal.

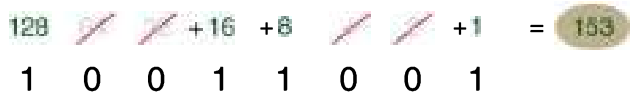


Figura 1.4. Ejemplo de conversión de un número binario en decimal por la suma de los pesos de sus bits.

2.1.3. Sistema de numeración hexadecimal

Los números se representan con dieciséis símbolos. Por tanto, se dice que este sistema de numeración trabaja en base dieciséis. Los diez primeros dígitos coinciden con los del sistema decimal, es decir, del 0 al 9, y para los siguientes, mayores de 9, se utilizan las primeras letras del abecedario, de la A a la F.

Así, los números en hexadecimal se forman de la siguiente manera:

Número en hexadecimal	Número en decimal
FF	255
1A	26
D24	3364

Sistema hexadecimal

Símbolos del sistema hexadecimal y su correspondencia con el sistema decimal:

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

2.1.4. Sistema de numeración octal

Es un sistema en base ocho, por lo que utiliza solamente ocho dígitos, del 0 al 7, los cuales coinciden con el sistema de numeración decimal. A partir del símbolo 8 la codificación es diferente:

0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, 23, 24, 25...

Ejemplos de número en octal y su equivalente en decimal:

Número en octal	Número en decimal
17	15
24	20
377	255
32	26

2.1.5. Representación de los números por su base

Una forma de escribir y diferenciar los números en los distintos sistemas de numeración consiste en escribir la base en forma de subíndice en el lado derecho del número:

- Decimal: $30041_{(10}$
- Binario: $10101_{(2}$
- Hexadecimal: $A34B_{(16}$
- Octal: $21015_{(8}$

Código BCD

El código BCD facilita la representación de números en decimal o hexadecimal, con displays basados en segmentos LED.

Figura 1.5. Segmentos LED..

2.2. Códigos de numeración

Un código de numeración es una forma codificada de utilizar un sistema de numeración para la representación de números, especialmente en los sistemas electrónicos e informáticos. Existen muchos códigos de numeración como, por ejemplo: BCD, Gray, exceso de 3, Aiken, ASCII, etc. No obstante, aquí solamente se estudiará el BCD, que es uno de los más utilizados en la técnica digital.

2.2.1. Código BCD

Debe su nombre a las iniciales de su denominación en inglés, *Binary-Coded Decimal*, que significa «decimal codificado en binario».

En el código BCD cada dígito está formado con un número en binario de cuatro bits. Así, los números se forman dígito a dígito, codificando independientemente cada uno de ellos en binario.

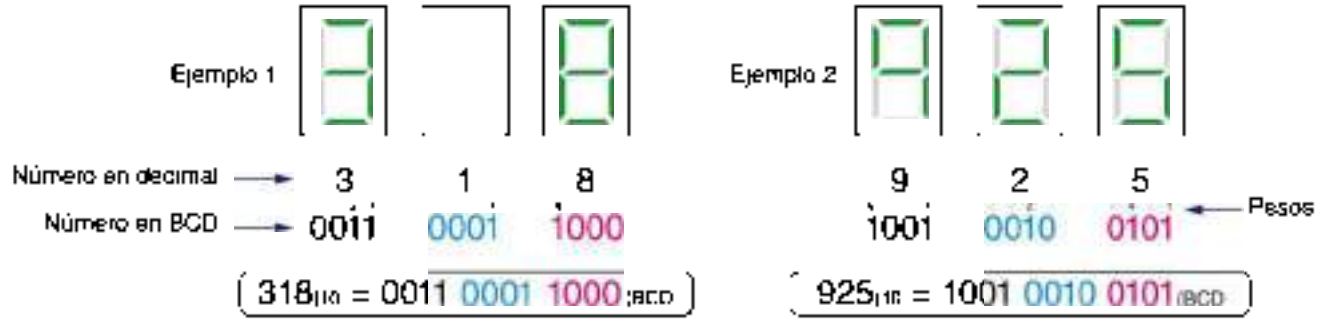


Figura 1.6. Dos ejemplos de uso del código BCD

Uso de la calculadora

Una forma sencilla y práctica de realizar la conversión de números entre los diferentes sistemas de numeración es utilizar una calculadora que lo permita, como puede ser la que ofrece el sistema operativo Windows.

Figura 1.7. Calculadora de programador del sistema operativo Windows.

Actividades

1. Utilizando una calculadora que permita la conversión entre sistemas de numeración, completa en tu cuaderno la tabla con el número correspondiente en el sistema indicado.

Decimal	Binario	Hexadecimal	Octal
450	....	....	....
....	101110	....	....
1010	....	....	....
....	....	FA45	....
....	....	....	2347
....	1111	....	....
....	....	FBBB	76
....	101010	....	....
....	....	DDD	....
100	....	....	....

2. Representa en sistema BCD los siguientes números escritos en sistema de numeración decimal:
   a) $300_{(10}$
   b) $1001_{(10}$
   c) $454_{(10}$
   d) $920_{(10}$
3. Indica a qué números en decimal corresponden cada uno de los siguientes números en BCD:
   a) $1010\ 1100_{(BCD}$
   b) $1111\ 0000\ 1010_{(BCD}$
   c) $0011\ 1011\ 1110_{(BCD}$
   d) $0110\ 1001\ 1000_{(BCD}$

3. Lógica digital

La **lógica digital** es la parte de la electrónica que estudia el comportamiento de los circuitos digitales basándose en el álgebra de Boole y los circuitos de puertas lógicas.

En la lógica digital se trabaja con dos niveles de tensión, donde el nivel más alto corresponde con un 1 lógico y el nivel más bajo con un 0 lógico.

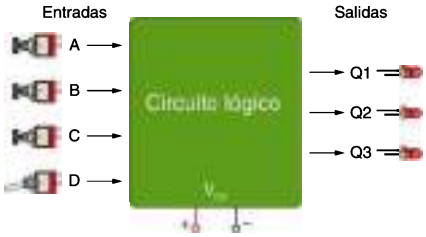


Figura 1.8. Circuito lógico.

3.1. Circuito lógico

Es un circuito electrónico destinado a realizar una serie de operaciones, basadas en valores discretos (lógicos) de tensión, para obtener también un resultado del mismo tipo.

Las señales lógicas se aplican mediante elementos de entrada, como pueden ser pulsadores, interruptores, etc., o señales provenientes de otros circuitos lógicos. El resultado del circuito lógico se muestra a través de las salidas, en cuyo caso se pueden utilizar diodos LED, lámparas, relés, etc.

3.2. Variables lógicas

Una variable es un elemento del circuito que puede cambiar de valor. En el caso de las variables lógicas, solamente es posible almacenar dos valores: el 1 o el 0.

Los circuitos lógicos disponen de variables de entrada y variables de salida. Las entradas se identifican mediante las letras del abecedario (A, B, C, etc.) y las salidas mediante Q1, Q2, Q3, etc.

Las entradas y salidas se suelen representar de forma simplificada, como se muestra a continuación:

Denominación	Símbolo IEC	Símbolo ANSI	Identificador
Entrada lógica	0		A, B, C...
Salida lógica			Q1, Q2, Q3...

3.3. Tabla de la verdad

La **tabla de la verdad** es una forma gráfica de representar el estado de las variables de salida de un circuito lógico en función del estado en el que se encuentran las entradas.

A modo de ejemplo, se pueden mostrar dos circuitos eléctricos básicos. Ambos son idénticos salvo porque en el primero el elemento de entrada es un pulsador normalmente abierto y en el segundo es un pulsador normalmente cerrado. En el primer caso, cuando el pulsador es accionado (1) la lámpara se enciende y cuando cesa la acción sobre él (0) la lámpara se apaga. Sin embargo, en el segundo circuito ocurre todo lo contrario, ya que el pulsador utilizado está normalmente cerrado en reposo. Así, la lámpara se enciende cuando no hay acción (0) sobre el pulsador y se apaga cuando se acciona (1).

Si el estado de ambos elementos se representa en formato de tabla en función de sus valores lógicos (0-1) se obtiene la denominada *tabla de la verdad*.

Vocabulary

Leyenda: *caption.*
Lógica: *logic.*
Entrada: *input.*
Salida: *output.*
Binario: *binary.*
Tabla de la verdad: *truth table.*
Prueba: *probe.*
Cable: *wire.*
Dispositivo: *device.*
Circuito integrado: *integrated circuit.*
Mejora: *improvement.*
Nivel lógico: *logic level.*
Lógica positiva: *logical ones.*
Lógica negativa: *logical zero.*
Alto: *high.*
Bajo: *low.*
Verdadero: *true.*
Falso: *false.*

Truco

Una forma rápida y simple de ordenar las filas de una tabla de la verdad consiste en hacer lo siguiente:

1. Calcula el número de posibilidades en función del número de variables.
2. Completa la columna de la variable que está más a la derecha, alternando en cada fila un 0 y un 1.
3. Completa hacia la derecha las siguientes columnas, poniendo de forma contigua el doble de unos y ceros que en la columna anterior.

De esta forma harás tablas de la verdad sin posibilidad de equivocarte.

Paso 1

A	B	C
		0
		1
		0
		1
		0
		1
		0
		1

Paso 2

A	B	C
	0	0
	0	1
	1	0
	1	1
	0	0
	0	1
	1	0
	1	1

Paso 3

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Figura 1.11. Método para crear a mano tablas de la verdad.

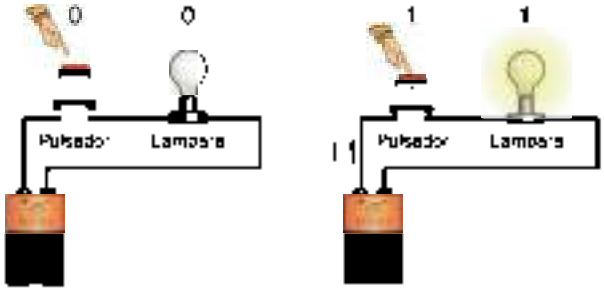


Tabla de la verdad

Pulsador	Lámpara
0	0
1	1

Figura 1.9. Tabla de la verdad de un circuito con pulsador normalmente abierto.

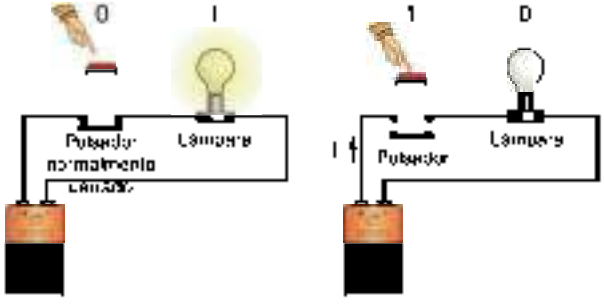


Tabla de la verdad

Pulsador	Lámpara
0	1
1	0

Figura 1.10. Tabla de la verdad de un circuito con pulsador normalmente cerrado.

En una tabla de la verdad deben contemplarse todas las combinaciones posibles que existan entre las variables de entrada. Así, para calcular el número de combinaciones posibles, se toma la base del sistema de numeración en el que se está trabajando, que en este caso es 2, por ser el binario, y se eleva al número de variables que se van a utilizar. El resultado es el número de combinaciones que se pueden conseguir usando todas las entradas sin que se repitan.

Dos variables	Tres variables	Cuatro variables
$2^2 = 4$	$2^3 = 8$	$2^4 = 16$

	A	B	Q
0	0	0	
1	0	1	
2	1	0	
3	1	1	

	A	B	C	Q
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

	A	B	C	D	Q
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	

Los valores de ceros y unos de las filas corresponden al número de orden en decimal codificado en binario. El número más bajo es el que está en la fila superior y el más alto el que está en la fila inferior.

Actividades

4. Dibuja en tu cuaderno una tabla de la verdad para cinco variables de entrada (A, B, C, D y E). ¿Cuántas combinaciones son posibles con este número de variables? Si el número de variables fuese seis, ¿cuántas combinaciones serían posibles para esta tabla de la verdad?

3.4. Funciones lógicas

Las funciones lógicas son operaciones del álgebra de Boole que permiten obtener un resultado sobre una salida en función de los estados de sus entradas.

A continuación, se indican las funciones lógicas más comunes y se muestra en cada una de ellas su función matemática dentro del álgebra de Boole, su representación gráfica en forma de puerta lógica, tanto en la simbología IEC como en la ANSI, su tabla de la verdad y su circuito eléctrico equivalente.

Debido a su amplia utilización, los esquemas de electrónica digital de este libro se han diseñado según la simbología ANSI en lugar de la IEC.

La funciones lógicas son conocidas por su denominación en inglés: AND, OR, etc., aunque también es posible utilizarlas por su traducción al castellano, Y, O, etc.

3.4.1. Función directa

También denominada *función igualdad* o *función SI*. Es el equivalente a un interruptor normalmente abierto de un circuito eléctrico.

En electrónica esta función es un amplificador o *buffer*.

3.4.2. Función NOT (NO o negación)

Es la operación inversa a la anterior, y es equivalente a utilizar un contacto normalmente cerrado en reposo.

3.4.3. Función AND (Y)

También denominada *operación producto,* tiene un comportamiento similar a interruptores en serie.

3.4.4. Función OR (O)

También conocida como *operación suma,* es equivalente a interruptores en paralelo.

3.4.5. Función NAND (NO Y)

Es la función inversa a la función AND. El símbolo es parecido al de esta, pero con una negación en su salida. Su comportamiento es similar a dos pulsadores en paralelo normalmente cerrados.

3.4.6. Función NOR (NO O)

Es la función inversa de la función OR.

3.4.7. Función XOR (O exclusiva)

Se representa con el símbolo de suma directa $\oplus$. Su funcionamiento es equivalente a un circuito conmutado. El resultado desarrollado de esta función es:

$$Q = A \oplus B = \overline{A}B + A\overline{B}$$

3.4.8. Función NXOR (NO O exclusiva)

Es la función inversa a la función XOR.

A continuación se muestra una tabla con los símbolos, tablas de la verdad, ecuaciones lógicas y circuitos equivalentes de cada una de las funciones lógicas anteriormente nombradas.

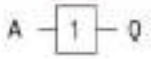
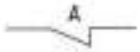
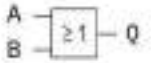
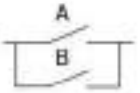
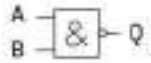
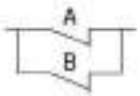
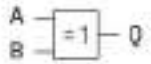
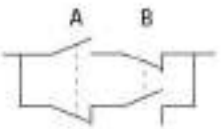
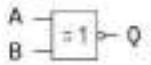
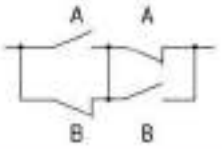
Función	Tabla de la verdad	Símbolo IEC	Símbolo ANSI	Ecuación lógica	Circuito equivalente
DIRECTA	A Q 0 0 1 1			$Q = A$	
NOT	A Q 0 1 1 0			$Q = \overline{A}$	
AND (Y)	A B Q 0 0 0 0 1 0 1 0 0 1 1 1			$Q = A \cdot B$	
OR (O)	A B Q 0 0 0 0 1 1 1 0 1 1 1 1			$Q = A + B$	
NAND	A B Q 0 0 1 0 1 1 1 0 1 1 1 0			$Q = \overline{A \cdot B}$	
NOR	A B Q 0 0 1 0 1 0 1 0 0 1 1 0			$Q = \overline{A + B}$	
XOR	A B Q 0 0 0 0 1 1 1 0 1 1 1 0			$Q = A \oplus B$	
NXOR	A B Q 0 0 1 0 1 0 1 0 0 1 1 1			$Q = \overline{A \oplus B}$	

Tabla 1.1. Funciones lógicas.

3.4.9. Funciones de más de dos entradas

Las funciones lógicas pueden disponer de más de dos entradas. En estos casos, al símbolo lógico se le añaden las líneas necesarias de entrada. La tabla de la verdad se debe construir con todas las posibles combinaciones que existen y en la ecuación lógica se representan las operaciones con todas las variables.

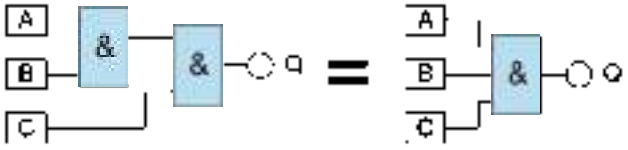
Una puerta lógica de tres o más entradas puede construirse conectando en cascada puertas de dos entradas.

A modo de ejemplo, a continuación se muestra una función AND de tres entradas.

Tabla de la verdad

A	B	C	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Circuito y símbolo lógico:



Ecuación lógica:

$Q = A \cdot B \cdot C$

Figura 1.12. Representación de una puerta lógica de tres entradas

3.5. Obtención de ecuaciones lógicas a partir de un circuito

Para obtener la ecuación lógica resultante de un circuito lógico previamente representado se lee el esquema de izquierda a derecha, escribiendo el resultado lógico que se va acumulando en cada bloque. De esta forma se consigue el valor de la función sin dar lugar a errores.

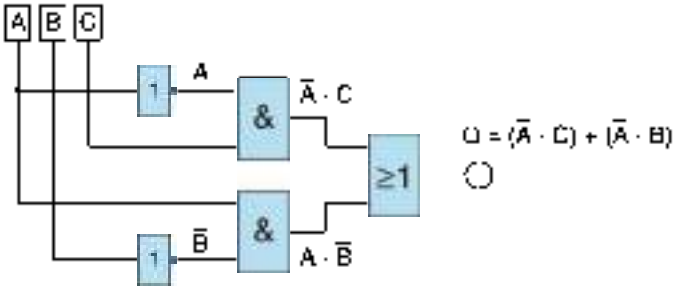


Figura 1.14. Ejemplo de la obtención de una ecuación lógica a partir de un circuito.

Simbología

La simbología IEC es la utilizada para representar circuitos lógicos en los sistemas de automatización industrial y, por tanto, será la usada en este libro.

Recuerda

Para facilitar el diseño de esquemas lógicos es aconsejable trazar líneas de señal como las mostradas en la figura. De esta forma el circuito «cuelga» de ellas en función de cómo se necesiten las señales, negadas o sin negar.

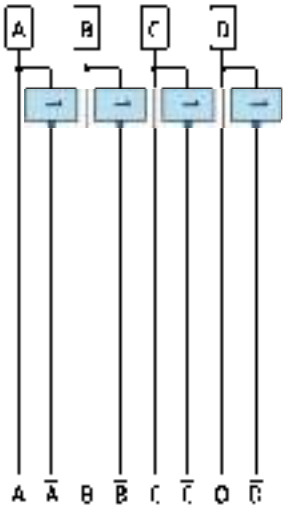


Figura 1.13. Líneas de señal para diseño de esquemas lógicos

Actividades

5. Escribe las ecuaciones lógicas de los siguientes circuitos:

Circuito 1

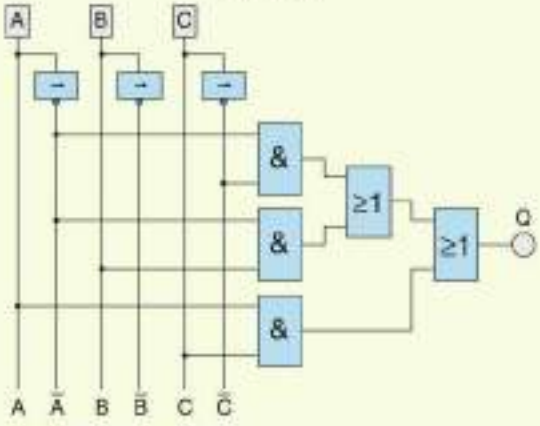


Figura 1.15. Circuito lógico 1.

Circuito 2

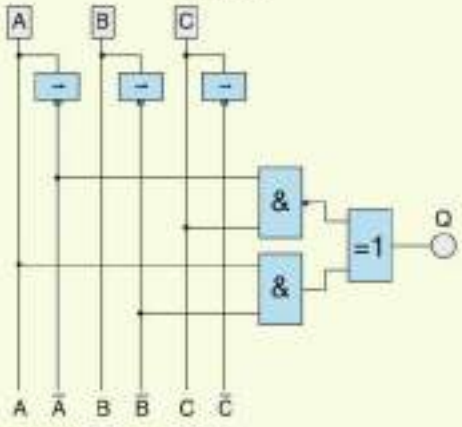


Figura 1.16. Circuito lógico 2.

3.6. Tabla de la verdad a partir de una ecuación lógica

Para conseguir una tabla de la verdad partiendo de una expresión booleana o ecuación lógica, lo que hay que hacer es sustituir el valor que tiene cada una de las variables en la fila de la tabla de la verdad y operar con ellas matemáticamente.

Sirva como ejemplo el paso a tabla de la verdad de la siguiente ecuación lógica:

$$Q = (\overline{A} \cdot B) + \overline{C}$$

A	B	C	Operación	Q
0	0	0	$(\overline{0} \cdot 0) + \overline{0} = (1 \cdot 0) + 1$	1
0	0	1	$(\overline{0} \cdot 0) + \overline{1} = (1 \cdot 0) + 0$	0
0	1	0	$(\overline{0} \cdot 1) + \overline{0} = (1 \cdot 1) + 1$	1
0	1	1	$(\overline{0} \cdot 1) + \overline{1} = (1 \cdot 1) + 0$	1
1	0	0	$(\overline{0} \cdot 1) + \overline{1} = (1 \cdot 1) + 0$	1
1	0	1	$(\overline{1} \cdot 0) + \overline{1} = (0 \cdot 0) + 0$	0
1	1	0	$(\overline{1} \cdot 1) + \overline{0} = (0 \cdot 1) + 1$	1
1	1	1	$(\overline{1} \cdot 1) + \overline{1} = (0 \cdot 1) + 0$	0

Importante

Propiedades del álgebra de Boole:

$$1 + 0 = 0$$
$$1 + 1 = 1$$
$$1 \cdot 1 = 1$$
$$0 \cdot 1 = 0$$
$$A + A = A$$
$$A \cdot A = A$$
$$\overline{\overline{A}} = A$$
$$A + \overline{A} = 1$$
$$A \cdot \overline{A} = 0$$
$$\overline{\overline{A}} = A$$

3.7. Ecuación lógica a partir de la tabla de la verdad

Partiendo de una tabla de la verdad es posible obtener su expresión lógica y con ella el circuito lógico correspondiente. Para ello se deben tener en cuenta las filas en las que la salida Q está a 1 o las filas en las que Q está a 0.

Si se eligen las filas donde Q = 1, de cada una de ellas se saca el producto de sus variables, teniendo en cuenta el signo (1: no negada y 0: negada). Cada uno de estos términos recibe el nombre de *término mínimo* o *minterm* y la expresión final es el resultado de sumar dichos términos. Es decir, se realiza una suma de productos.

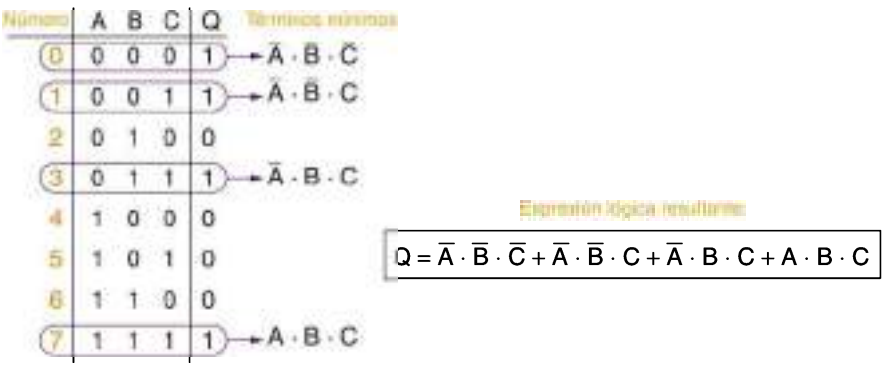


Figura 1.17. Ecuación lógica a partir de los términos mínimos de una tabla de la verdad.

Si, por el contrario, lo que se elige son las filas en las que Q = 0, las variables deben operar en formato de suma pero invirtiendo el signo respecto

Saber más

En ocasiones, la negación de las variables se expresa con el símbolo del apóstrofe:

$$A' = \overline{A}$$

Saber más

Los números de 8 bits se denominan *bytes*.

Los números de 16 bits se denominan *word*.

a la tabla de la verdad; es decir, donde hay un 0 la variable debe estar sin negar y donde hay un 1 la variable debe estar negada. Cada uno de estos resultados recibe el nombre de *término máximo* o *maxterm* y la expresión resultante es el producto de todos ellos. Es decir, se realiza un producto de sumas, con el signo de las variables cambiado respecto a la tabla de la verdad.

Número	A	B	C	Q	Términos máximos
0	0	0	0	1	
1	0	0	1	1	
2	0	1	0	0	$\rightarrow A + \overline{B} + C$
3	0	1	1	1	
4	1	0	0	0	$\rightarrow \overline{A} + B + C$
5	1	0	1	0	$\rightarrow \overline{A} + B + \overline{C}$
6	1	1	0	0	$\rightarrow \overline{A} + \overline{B} + C$
7	1	1	1	1	

Expresión lógica resultante:

$$Q = (A + \overline{B} + C)\,(\overline{A} + B + C)\,(\overline{A} + B + \overline{C})\,(\overline{A} + \overline{B} + C)$$

Figura 1.18. Ecuación lógica a partir de los términos máximos de una tabla de la verdad.

En ocasiones, para sintetizar las ecuaciones basadas en términos mínimos o en términos máximos se suelen utilizar expresiones abreviadas en las que solamente se indica el número de orden de la fila, que es en realidad el número de su codificación en binario que forman las variables. Así, para indicar que es una suma de productos se utiliza el singo sumatorio (Σ) y para indicar que es un producto de sumas se emplea el signo productorio (Π). En ambos casos, el número que aparece debajo de ellos es el número de variables de entrada que utiliza la expresión lógica.

Simbología

Forma abreviada de la ecuación de *minterms*:

$$Q = \sum_{3}(0, 1, 3, 7)$$

Forma abreviada de la ecuación de *maxterms*:

$$Q = \prod_{3}(2, 4, 5, 6)$$

Actividades

6. Obtén las tablas de la verdad de las siguientes ecuaciones lógicas:

a) $Q = (\overline{A \cdot B}) + A \cdot C$ b) $Q = \overline{(A + B) \cdot C}$ c) $Q = \overline{A} \cdot \overline{B} + \overline{A} \cdot C + \overline{B} \cdot C$

7. Escribe las ecuaciones lógicas de las siguientes tablas de la verdad expresadas en términos mínimos.

a)

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

b)

A	B	C	Q
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

c)

A	B	C	Q
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Escribe las ecuaciones lógicas de las tablas de la actividad anterior expresadas en términos máximos.

8. Representa la tabla de la verdad para que una salida Q, controlada con tres variables de entrada, esté a 1 lógico cuando al menos dos de las tres se encuentra a 1.

9. En un circuito con cuatro variables de entrada hay que encender una lámpara (Q1) siempre que dos de las variables sean distintas a las otras dos y encender otra lámpara (Q2) siempre que el número de variables de entrada que están a 0 sea impar.

3.8. Simplificación de ecuaciones lógicas

El uso de términos máximos o términos mínimos es una buena forma de obtener una expresión lógica a partir de una tabla de la verdad. No obstante, la ecuación resultante puede ser excesivamente larga. En ocasiones, dicha ecuación se puede simplificar y obtener otra mucho más corta, completamente equivalente y con el mismo resultado lógico.

El siguiente ejemplo muestra la ecuación de términos mínimos utilizada en el ejemplo anterior y su equivalente simplificada.

- No simplificada: $Q = (\overline{A} \cdot \overline{B} \cdot \overline{C}) + (\overline{A} \cdot \overline{B} \cdot C) + (\overline{A} \cdot B \cdot C) + (A \cdot B \cdot C)$
- Simplificada: $Q = (\overline{A} \cdot \overline{B}) + (B \cdot C)$

Existen varios métodos para simplificar ecuaciones lógicas; no obstante, el más utilizado y más sencillo de implementar es el denominado *simplificación mediante mapas de Karnaugh,* y es el que se va a describir a continuación.

3.8.1. Simplificación con mapas de Karnaugh

Es un método gráfico que permite simplificar con facilidad sistemas de hasta cuatro variables.

Lo primero que hay que realizar es el mapa o tabla en el que se representen todas las posibilidades de combinación entre las variables. De esta forma, si se dispone de dos variables la tabla tiene cuatro celdas, si se dispone de tres variables dispone de ocho celdas y si es de cuatro variables tiene un total de dieciséis celdas.

Recuerda

Al elaborar los mapas de Karnaugh es aconsejable escribir en la celda el número en decimal que corresponde con su codificación en binario. De esta forma será mucho más sencillo y rápido ubicar los valores de la salida desde la tabla de la verdad.

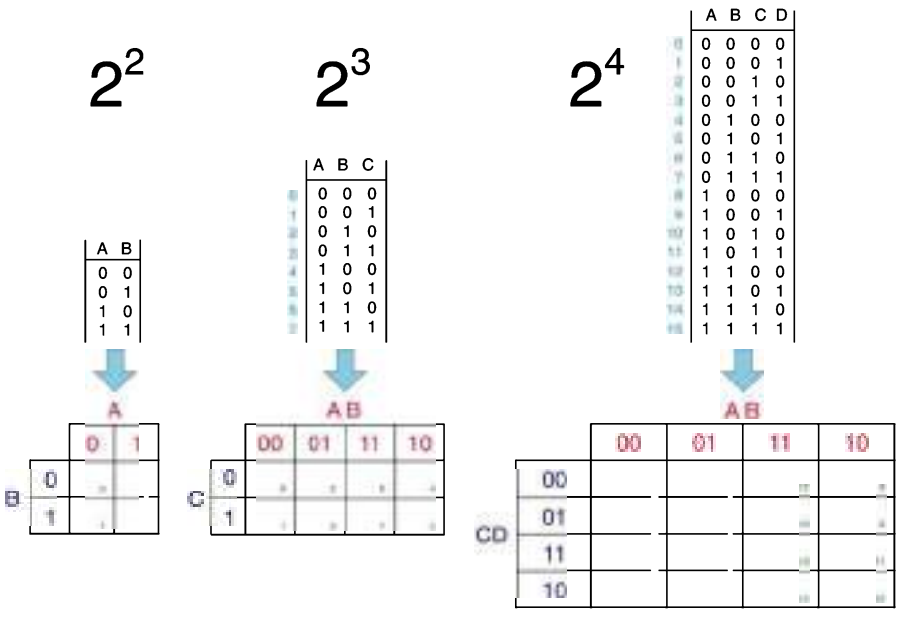


Figura 1.19. Mapas de Karnaugh de 2, 3 y 4 variables.

Saber más

Los mapas de Karnaugh se pueden utilizar también para resolver sistemas de más de cuatro variables. No obstante, debido a su largo y elaborado desarrollo, no se estudiarán en este libro. Para la resolución de estos sistemas es aconsejable utilizar cualquiera de las aplicaciones informáticas que existen diseñadas para tal efecto.

Si se toma como ejemplo el mapa de Karnaugh de cuatro variables, dos de ellas se colocan en horizontal y las otras dos en vertical. El orden de dos en dos de estas variables se hace de la siguiente forma: 00-01-11-10, en la que se observa que el cambio de valor entre columnas (o filas cuando corresponda) se consigue haciendo coincidir el valor de la variable de la columna anterior. De esta forma, si se lee consecución de variables, se debe comprobar que las variables que unen las columnas (o filas) deben coincidir en valor. Esto es debido a que utilizan un código de numeración, denominado *Gray,* que atiende a dicho patrón.

Desarrollo de la simplificación

1. Se dibuja el mapa en función del número de variables.
2. Partiendo de la tabla de la verdad, se ubican en ella los unos del valor de la salida. Se puede trabajar por términos máximos, teniendo en cuenta los ceros, o por términos mínimos, teniendo en cuenta los unos. No obstante, aquí se ha optado por realizar la simplificación por minterms, ya que es más inmediata y sencilla de implementar.
3. Se realizan grupos de 1, 2, 4, 8 o 16 números uno que estén en celdas contiguas, teniendo en cuenta que solamente se pueden efectuar en horizontal y en vertical, nunca en diagonal.

   Hay que intentar realizar grupos con el mayor número de unos posible, ya que cuanto mayor es el grupo, más simplificado es el resultado, pudiéndose formar grupos con números uno que se encuentran incluidos en otros.

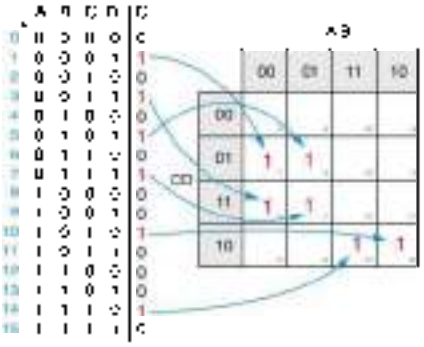


Figura 1.20. Ubicación de los resultados con valor 1 en un mapa de Karnaugh.

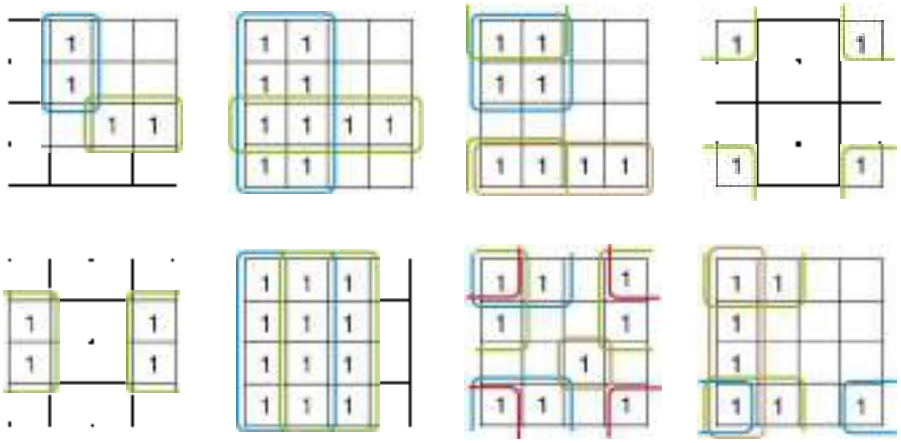


Figura 1.21. Ejemplos de posibles agrupaciones en mapas de Karnaugh.

4. Los minterms de un mismo grupo se escriben uno debajo de otro y se eliminan aquellos términos que tienen valores negados y sin negar en la misma columna. Las variables que no cambian de valor se muestran como el resultado del grupo en forma de producto.

   Con la práctica, esta operación puede resolverse mentalmente.
5. El valor de la expresión para la salida Q se obtiene sumando los resultados simplificados de cada grupo.

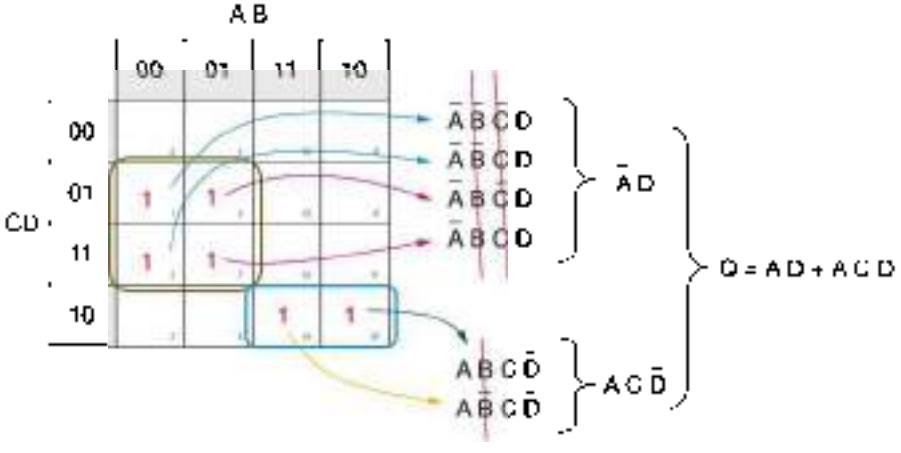


Figura 1.22. Simplificación de los grupos y obtención de la expresión resultante.

3.9. Cronogramas

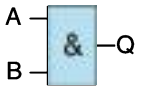
Recuerda

Los cronogramas son especialmente útiles para mostrar el funcionamiento tanto de los circuitos combinacionales como de los secuenciales y a menudo son utilizados por los fabricantes en las hojas de características para mostrar el comportamiento de sus componentes.

Un cronograma es un gráfico en el que se muestra cómo evolucionan una o más señales, en este caso digitales, en función del tiempo. También son conocidos con el nombre de *diagramas de tiempo*.

La siguiente figura representa el cronograma con el funcionamiento de una puerta lógica AND de dos entradas. En él se observa cómo la señal de salida Q solamente tiene valor lógico 1 cuando coincide en el tiempo que las dos entradas, A y B, también lo tienen.

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



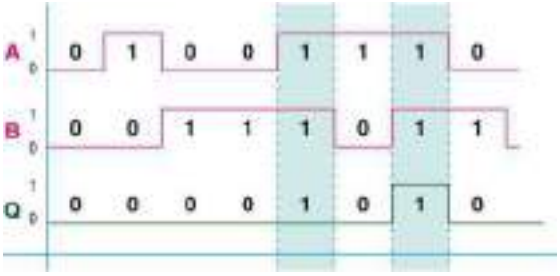


Figura 1.23. Ejemplo de cronograma de una función lógica AND.

Figura 1.24. Flancos de una señal digital.

En los cronogramas, las señales digitales se muestran con dos posibles valores, el 0 y el 1. Así, el flanco que pasa de 0 a 1 se denomina *flanco ascendente* o de subida y el que pasa de 1 a 0 *flanco descendente* o de bajada.

3.10. Teoremas de Morgan

Importante

$$\overline{A \cdot B} \neq \overline{A} \cdot \overline{B}$$
$$\overline{A + B} \neq \overline{A} + \overline{B}$$

Es un método del álgebra de Boole que permite simplificar funciones lógicas retirando las negaciones que afectan a las operaciones, lo que las hace más manejables.

Los teoremas de Morgan son dos:

Teorema 1: $Q = \overline{A \cdot B} = \overline{A} + \overline{A}$ **Teorema 2:** $Q = \overline{A + B} = \overline{A} \cdot \overline{B}$

Ejemplo

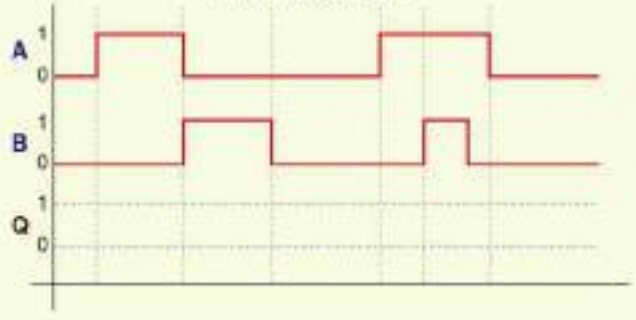
Ejemplos de aplicación de los teoremas de Morgan:

1. $Q = \overline{(A \cdot B) + C} = (\overline{A} + \overline{B}) \cdot \overline{C}$
2. $Q = \overline{A \cdot \overline{B} + \overline{C}} = \overline{A} + B \cdot C$
3. $Q = \overline{(\overline{A} \cdot \overline{D}) + (B \cdot \overline{C})} = \overline{(\overline{A} \cdot \overline{D})} \cdot \overline{(B \cdot \overline{C})} = (A + D) \cdot (\overline{B} + C)$
4. $Q = \overline{A \oplus B} = \overline{\overline{A} \cdot B + A \cdot \overline{B}} = \overline{\overline{A} \cdot B} \cdot \overline{A \cdot \overline{B}} = (A + \overline{B}) \cdot (\overline{A} + B)$

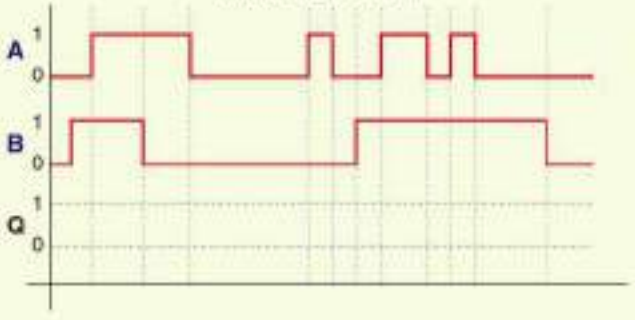
Actividades

10. Representa en tu cuaderno, sobre las gráficas, las señales de salida correspondientes a las siguientes funciones lógicas:

Función lógica OR:



Función lógica XOR:



4. Circuitos secuenciales

En la técnica digital podemos diferenciar dos tipos de circuitos, los denominados *circuitos combinacionales* y los *circuitos secuenciales*.

En los **circuitos combinacionales,** que se han estudiado hasta el momento, el estado de las salidas depende única y exclusivamente del estado de las entradas. Sin embargo, en los **circuitos secuenciales** los valores de las salidas, además de depender de las entradas, dependen del estado anterior de ellas mismas.

Figura 1.25. Circuito combinacional.

Figura 1.26. Circuito secuencial.

En esta unidad se estudiarán, de forma básica, algunos de los circuitos secuenciales más significativos y que posteriormente será necesario conocer para programar dispositivos, como los denominados *autómatas programables*.

Recuerda

Se puede decir que un circuito secuencial tiene memoria y uno combinacional no la tiene.

4.1. Circuito con realimentación

Así, una primera aproximación a la lógica secuencial es la siguiente:

Supóngase la operación OR de dos señales de entrada (1) en la que el resultado lógico de la salida es 1 cuando cualquiera de ellas lo es también. Si una de las señales de entrada se sustituye por la señal de la propia salida de la función (2) cuando la entrada tiene valor 1, la salida se pone a nivel alto y por tanto se aplica un 1 lógico en el otro terminal de la puerta OR. De esta forma, la salida se mantiene activada de forma permanente a pesar de que la entrada que provocó esta acción deje de estar a nivel alto en la puerta lógica. Con esto se consigue hacer una función de realimentación y así memorizar el estado de la propia salida, aplicando su nivel lógico como si fuese una entrada.

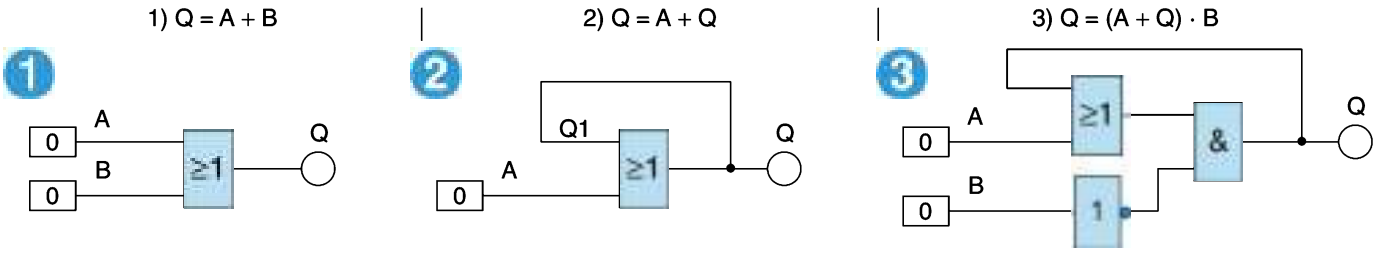


Figura 1.27. Circuito con memoria.

Dicha operación no tiene demasiado sentido si no es posible desactivar el nivel lógico de la salida de alguna manera. Para ello, simplemente se realiza una operación AND de una entrada negada con la operación OR anterior (3), de forma que dicha entrada se utiliza para poner a valor de 0 lógico la salida y desactivar así la realimentación.

De esta forma, una entrada activa la salida y se mantiene en ese estado aunque cese la acción sobre ella, y la otra la desactiva y la mantiene en ese estado, aunque su valor lógico retorne a 0.

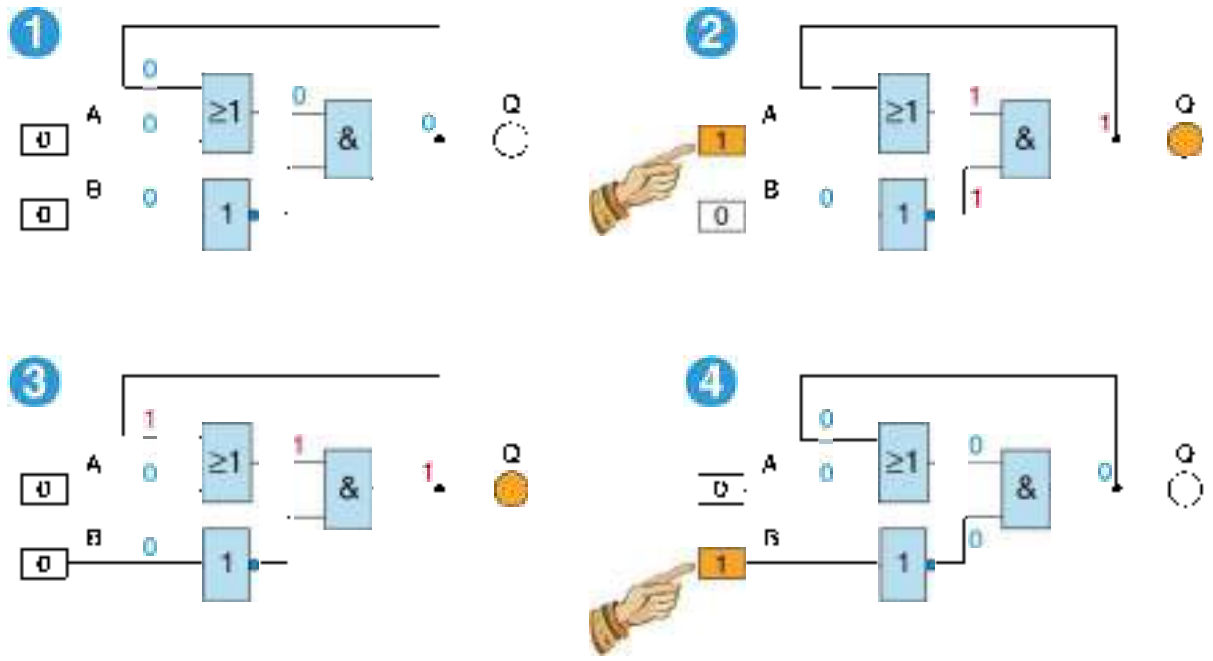


Figura 1.28. Secuencia de funcionamiento de un circuito con realimentación

Autómatas programables (PLC)

Esta forma de realizar circuitos con memoria no es muy utilizada en electrónica digital; sin embargo, está ampliamente difundida en los sistemas secuenciales basados en autómatas programables (PLC), que serán motivo de estudio en próximas unidades.

Por tanto, en el caso de la figura, se puede decir que la entada A es la activación o SET y la entrada B es la desactivación o RESET.

4.2. Circuitos secuenciales con realimentación

El circuito con memoria visto anteriormente puede ser la base para resolver sencillos circuitos secuenciales mediante métodos intuitivos basados en cuatro reglas básicas.

4.2.1. Activación de señales (puesta en marcha)

Siempre que se desee activar una salida mediante una o más variables de entrada, esta o estas se deben sumar a la realimentación. En el ejemplo se muestra como la salida Q se puede activar desde tres variables de entrada: A, B o C.

4.2.2. Desactivación de señales (parada)

Siempre que se desee desactivar (parar) una salida mediante una o más variables de entrada, esta o estas se deben multiplicar, negando cada variable individualmente, al grupo de la realimentación. En el siguiente ejemplo se muestra como la salida Q se puede desactivar desde B, desde C o desde D.

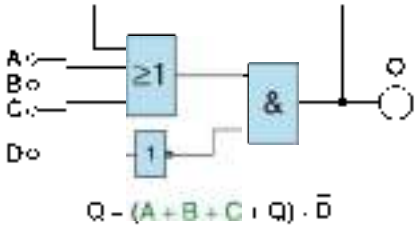


Figura 1.29. Activación de una salida con realimentación desde tres puntos

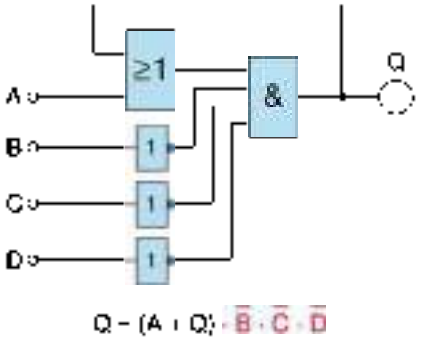


Figura 1.30. Desactivación de una salida desde tres puntos

4.2.3. Activación de una salida condicionada a la activación de otra

Si se desea poner como condición que una variable de salida se active si previamente lo está otra, se multiplica, sin negar, la variable de la segunda en la ecuación de la primera. De esta forma, Q1 solamente se podrá activar si previamente lo ha hecho Q2.

4.2.4. Activación de una salida condicionada a la NO activación de otra

Si se desea poner como condición que una variable de salida NO se active si previamente lo está otra, se multiplica la variable negada de la segunda salida en la ecuación de la primera. En este caso, si Q2 está activada Q1 no podrá hacerlo.

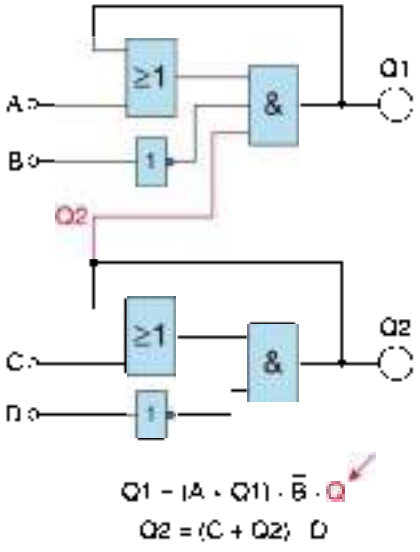


Figura 1.31. Activación de una salida al funcionamiento de otra

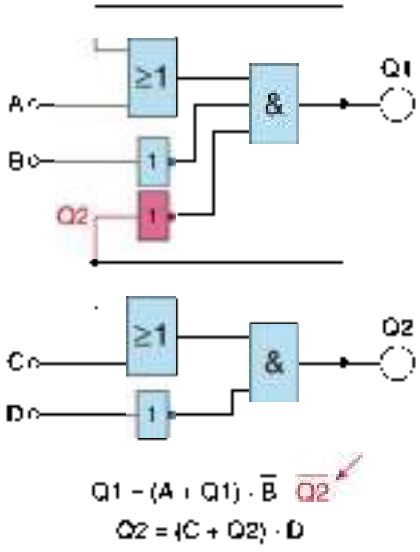


Figura 1.32. Activación de una salida al NO funcionamiento de otra

4.3. El biestable

Un biestable es un circuito electrónico que tiene dos posibles estados estables. Se puede decir que un biestable es un circuito con memoria, que permite almacenar un dato en binario y utilizarlo cuando sea necesario en el circuito en el que se encuentra implementado.

Se podría decir que el biestable es como si se encapsulara en un único bloque el circuito de realimentación o memoria visto anteriormente. De esta forma, habría una entrada para la activación de la salida, denominada *SET*, y otra para la desactivación, denominada *RESET*.

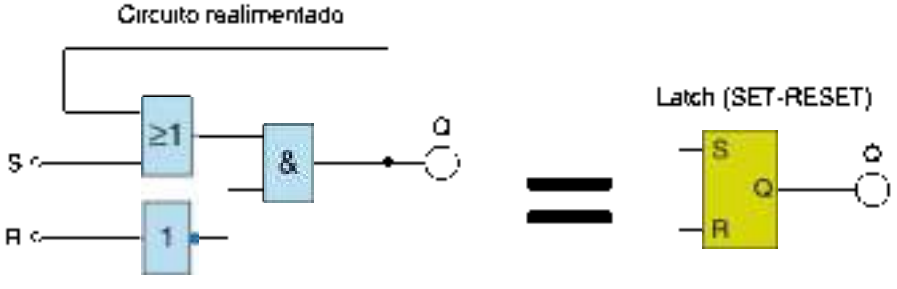


Figura 1.33. Circuito equivalente de un biestable SET-RESET.

Biestables asíncronos y síncronos

En electrónica digital es habitual clasificar los biestables en función de si la conmutación de sus salidas atiende o no a una señal de reloj (clock). Así, los biestables se pueden clasificar en **biestables asíncronos** y **biestables síncronos**. De ambos, los primeros son los de mayor interés para la programación de autómatas programables o PLC, y por lo tanto los que aquí se estudian.

R	S	Q
0	0	Valor anterior
0	1	1
1	0	0
1	1	0

Tabla de la verdad biestable SET-RESET

Saber más

En numerosas ocasiones en los biestables, además de la salida Q, se representa una salida Q negada, cuyo estado es inverso a la anterior.

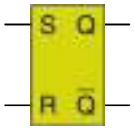


Figura 1.35. Símbolo del biestable SR.

Un biestable tiene dos entradas, por tanto, se deben representar dos ecuaciones lógicas, una para el SET y otra para el RESET.

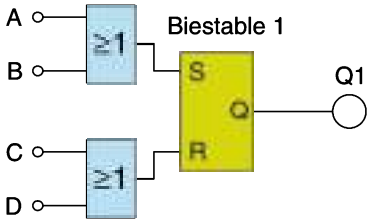


Figura 1.34. Ecuaciones lógicas de un biestable SR.

4.4. Circuitos secuenciales con biestables

De igual forma a como se ha estudiado anteriormente para los circuitos con realimentación, es posible resolver sencillos circuitos secuenciales con biestables aplicando cuatro reglas básicas.

4.4.1. Activación (marcha)

Para activar la salida de un biestable desde una o más variables de entrada se han de operar en formato de suma (OR) en la entrada del SET.

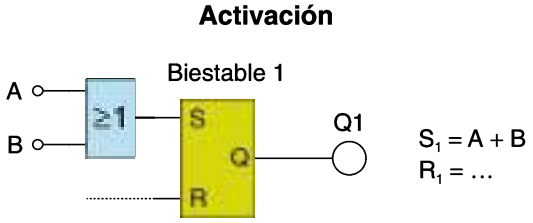


Figura 1.36. Activación desde varias entradas.

4.4.2. Desactivación (paro)

Para desactivar la salida de un biestable desde una o más variables de entrada se han de operar en formato de suma (OR) en la entrada del RESET.

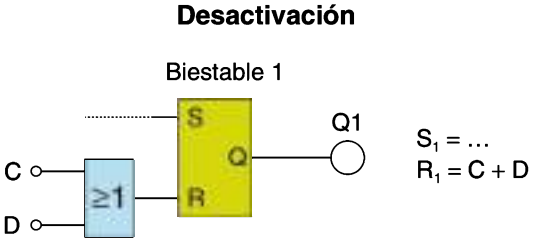


Figura 1.37. Desactivación desde varias entradas.

4.4.3. Activación de una salida condicionada a la activación de otra

Para establecer esta condición se ha de operar, sin negar, la variable de la segunda salida en la ecuación de SET de la que se desea condicionar.

4.4.4. Activación de una salida condicionada a la NO activación de otra

Para establecer esta condición se ha de operar negando la variable de la segunda salida en la ecuación de SET de la que se desea condicionar.

Salida condicionada a la activación de otra

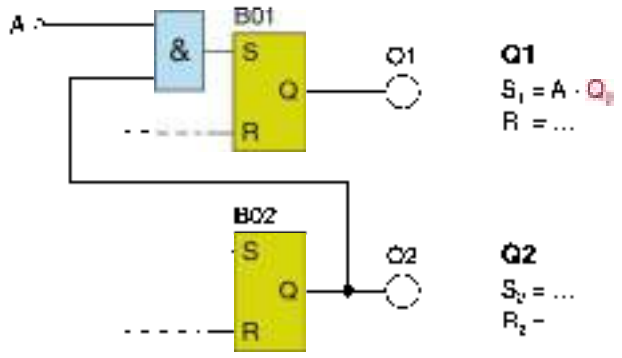


Figura 1.38. Circuito con biestable condicionado 1.

Salida condicionada a la NO activación de otra

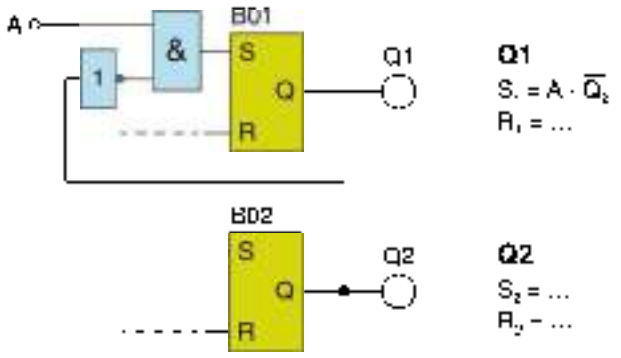


Figura 1.39. Circuito con biestable condicionado 2.

Ejemplo

Dibuja los esquemas de un circuito con realimentación y biestables que cumpla las siguientes condiciones:

- **La salida Q1 se activa desde A y Q2 desde B. Ambas se descactiva.**
- **Ambas salidas se desactivan desde C.**
- **Si una de las salidas está activa, no puede estarlo la otra.**

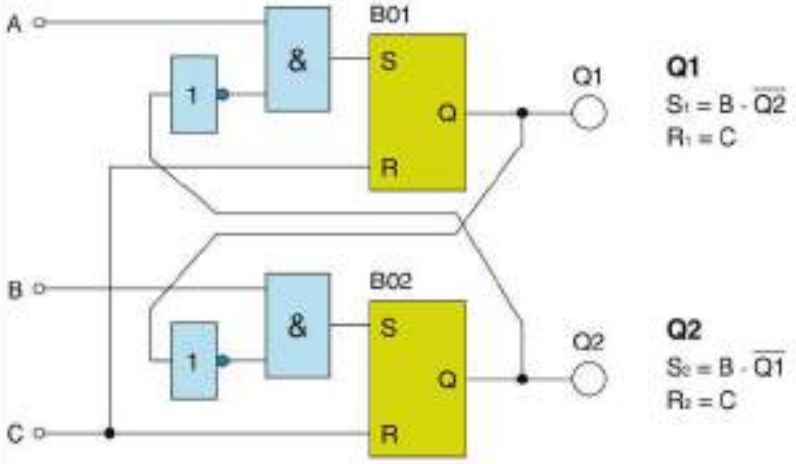


Figura 1.40. Esquemas de circuitos con realimentación y biestables.

Actividades

11. Obtén las ecuaciones lógicas y dibuja los esquemas de los siguientes circuitos con realimentación y/o biestables:

a) La salida Q1 se activa desde los pulsadores A o C y se desactiva desde B, D o E.

b) La salida Q1 se activa desde A y se para desde C o D, y la salida Q2 se pone en marcha desde C o desde B y se para desde D.

c) Tres lámparas se deben activar con sus respectivos pulsadores, A-B-C, y todas ellas se deben desactivar mediante D. El encendido de las lámparas debe hacerse en el orden 1-2-3, de forma que la segunda no se pueda activar si no lo ha hecho previamente la primera, y que la tercera no se encienda si no lo ha hecho la segunda.

d) Tres salidas, Q1, Q2 y Q3, se activan con A, B y C, respectivamente, y se desactivan con D. Se debe establecer la condición de funcionamiento de forma que Q3 solamente se pueda activar si alguna de las otra dos no lo ha hecho previamente.

PRÁCTICA PROFESIONAL RESUELTA

Herramientas

- PC y simulador de electrónica digital

Material

- Cuaderno de trabajo

Análisis, simplificación y simulación de un circuito lógico

Objetivo

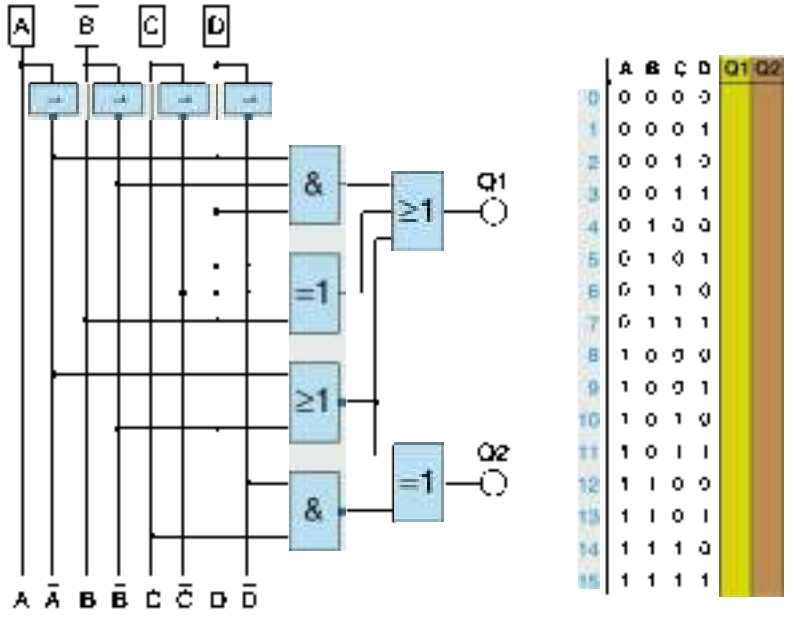
- Completar una tabla de la verdad basándose en una propuesta de funcionamiento
- Obtener las ecuaciones basadas en minterms desde una tabla de la verdad.
- Simplificar ecuaciones mediante mapas de Karnaugh.
- Dibujar y simular circuitos lógicos partiendo de ecuaciones lógicas.

Precauciones

- Hay que poner especial atención para no olvidar ninguna de las negaciones de las variables en todo el proceso de resolución, ya que un cambio de valor dará un resultado completamente diferente

Desarrollo

1. Utilizando un simulador de electrónica digital, hay que obtener los resultados del circuito de la figura en la tabla de la verdad para las salidas Q1 y Q2
2. Comprobar con el simulador cada una las posibles combinaciones de las entradas



	A	B	C	D	Q1	Q2
0	0	0	0	0		
1	0	0	0	1		
2	0	0	1	0		
3	0	0	1	1		
4	0	1	0	0		
5	0	1	0	1		
6	0	1	1	0		
7	0	1	1	1		
8	1	0	0	0		
9	1	0	0	1		
10	1	0	1	0		
11	1	0	1	1		
12	1	1	0	0		
13	1	1	0	1		
14	1	1	1	0		
15	1	1	1	1		

Figura 1.41 Circuito y tabla completa

3. Anotar los resultados en la tabla de la verdad

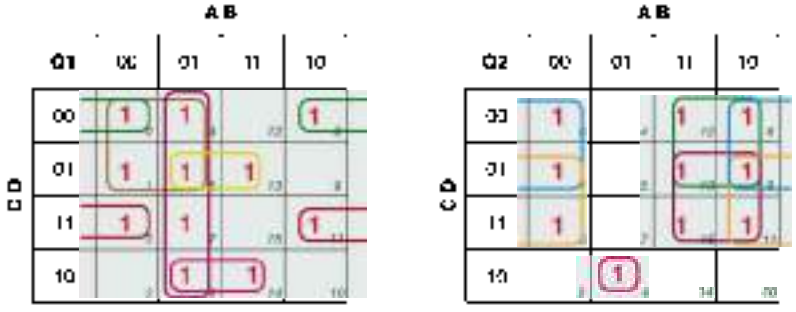
4. Obtener las ecuaciones de minterms de las dos salidas digitales.

	A	B	C	D	Q1	Q2
0	0	0	0	0	1	1
1	0	0	0	1	1	1
2	0	0	1	0	0	0
3	0	0	1	1	1	1
4	0	1	0	0	1	0
5	0	1	0	1	1	0
6	0	1	1	0	1	1
7	0	1	1	1	1	0
8	1	0	0	0	1	1
9	1	0	0	1	0	1
10	1	0	1	0	0	0
11	1	0	1	1	1	1
12	1	1	0	0	0	1
13	1	1	0	1	1	1
14	1	1	1	0	1	0
15	1	1	1	1	0	1

$$Q1 = \bar{A}\cdot\bar{B}\cdot\bar{C}\cdot\bar{D} + \bar{A}\cdot\bar{B}\cdot\bar{C}\cdot D + \bar{A}\cdot\bar{B}\cdot C\cdot\bar{D} + \bar{A}\cdot\bar{B}\cdot C\cdot D + A\cdot\bar{B}\cdot\bar{C}\cdot\bar{D} + A\cdot\bar{B}\cdot\bar{C}\cdot D + A\cdot\bar{B}\cdot C\cdot\bar{D} + A\cdot\bar{B}\cdot C\cdot D + A\cdot B\cdot\bar{C}\cdot\bar{D} + A\cdot B\cdot\bar{C}\cdot D + A\cdot B\cdot C\cdot\bar{D}$$

$$Q2 = \bar{A}\cdot\bar{B}\cdot\bar{C}\cdot\bar{D} + \bar{A}\cdot\bar{B}\cdot\bar{C}\cdot D + \bar{A}\cdot\bar{B}\cdot C\cdot\bar{D} + \bar{A}\cdot\bar{B}\cdot C\cdot D + \bar{A}\cdot B\cdot\bar{C}\cdot\bar{D} + A\cdot\bar{B}\cdot\bar{C}\cdot\bar{D} + A\cdot\bar{B}\cdot\bar{C}\cdot D + A\cdot\bar{B}\cdot C\cdot\bar{D} + A\cdot\bar{B}\cdot C\cdot D + A\cdot B\cdot C\cdot D$$

5. Simplificar por Karnaugh el resultado de ambas ecuaciones lógicas



6. Dibujar los circuitos simplificados y comprobar con el simulador que la tabla de la verdad es la correcta.

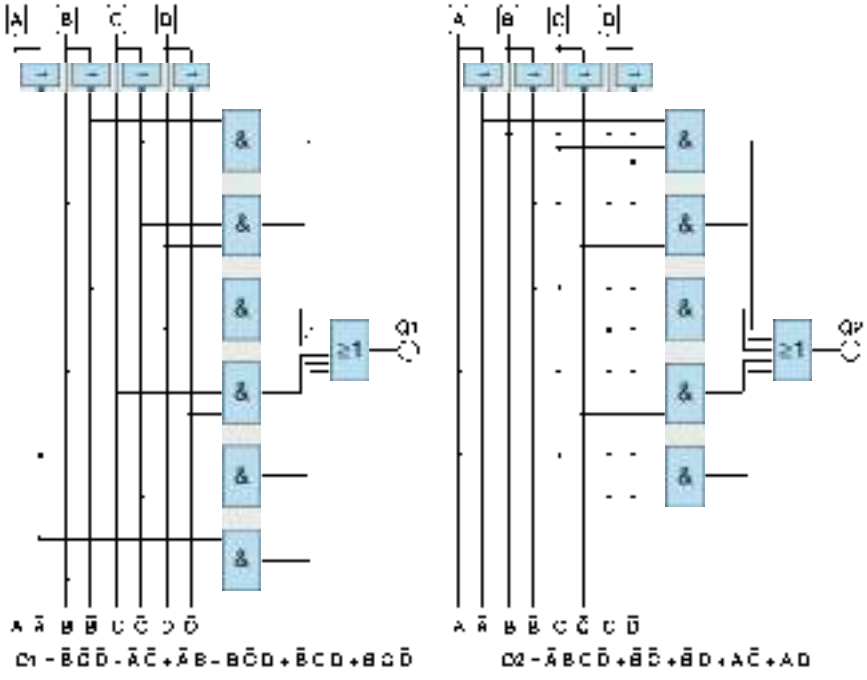


Figura 1.42. Circuitos equivalentes simplificados que simular.

TEST DE EVALUACIÓN

RESUELVE EN TU CUADERNO O BLOC DE NOTAS

1. ¿Con qué valores trabaja una señal digital?

a) Entre 0 y 5 voltios.
b) Menos de 10 voltios.
c) Solo 0 y 1.
d) Menos de 18 voltios si la frecuencia es de 10 kHz.

2. Las señales lógicas, ¿qué otro nombre reciben?

a) Señales booleanas.
b) Señales analógicas.
c) Señales de alta frecuencia.
d) Señales rectificadas.

3. ¿Cuál de estos sistemas de numeración trabaja con dieciséis símbolos?

a) El binario.
b) El decimal.
c) El octal.
d) El hexadecimal.

4. ¿A qué número en decimal corresponde este número en binario: 1001?

a) 1001.
b) 9.
c) 10.
d) 19.

5. ¿Cuántos números en decimal se pueden formar con un número en binario de 4 bits?

a) 4.
b) 8.
c) 16.
d) 32.

6. Si nos encontramos con la siguiente expresión lógica: $Q = A + B + C$, estamos hablando de una función lógica:

a) AND.
b) XOR.
c) NXOR.
d) OR.

7. La expresión lógica de una función NAND es:

a) $Q = \overline{A \cdot B}$.
b) $Q = \overline{A + B}$.
c) $Q = A \oplus B$.
d) $Q = \overline{A \oplus B}$.

8. La expresión $Q = A \oplus B$ es lo mismo que:

a) $Q = A\overline{B} + \overline{A}B$.
b) $Q = \overline{A}\overline{B} + \overline{A}\overline{B}$.
c) $Q = \overline{A}\overline{B} + AB$.
d) $Q = \overline{\overline{A}\overline{B} + AB}$.

9. Una ecuación con términos mínimos es:

a) Una suma de sumas.
b) Una suma de productos.
c) Un producto de productos.
d) Un producto de sumas.

10. ¿A qué equivale un circuito con realimentación?

a) A un biestable.
b) A una puerta XOR.
c) A interruptores en paralelo.
d) A una función OR con negación en una de sus entradas.

ACTIVIDADES FINALES

1. Escribe las ecuaciones lógicas de los siguientes circuitos lógicos:

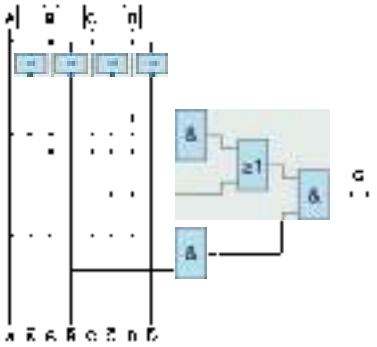


Figura 1.43. Circuito lógico 1.

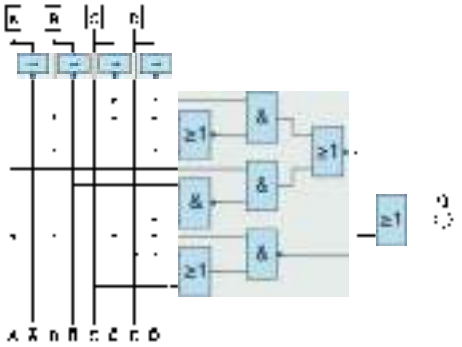


Figura 1.44. Circuito lógico 2.

2. Utilizando un programa de simulación electrónica digital, representa la tabla de la verdad para Q. Coteja el resultado con tu compañero para comprobar que es correcto.

3. Dibuja los circuitos a partir de las siguientes ecuaciones lógicas:

a) $Q = (A + B) \cdot (\overline{B} - C)$

b) $Q = \overline{A \cdot B} + \overline{A} \cdot C$

c) $Q = \overline{A \cdot B} + \overline{A} \cdot C$

d) $Q = A\ B + A \cdot C + \overline{B\ C}$

e) $Q = (\overline{B} + \overline{C} + A) \oplus (\overline{A} \cdot D)$

4. Obtén las tablas de la verdad de las ecuaciones lógicas A, B, C y D de la actividad anterior y, con un software de simulación de electrónica digital, comprueba que son correctas.

5. Simplifica por Karnaugh las siguientes tablas de la verdad y dibuja el circuito resultante.

1

A	B	C	Q
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

2

A	B	C	D	Q
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

3

A	B	C	D	Q
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Figura 1.45. Tablas de la verdad para simplificar por Karnaugh.

6. Aplica los teoremas de Morgan a las siguientes ecuaciones lógicas hasta que no existan operaciones lógicas afectadas por negaciones.

Q1 = $\overline{A + B \cdot C}$

Q2 = $\overline{A + B \cdot (C\ D)}$

Q3 = $\overline{(A \cdot B) + (C \cdot D)}$

Q4 = $\overline{((A + \overline{B}) \cdot \overline{D}) + \overline{C}}$

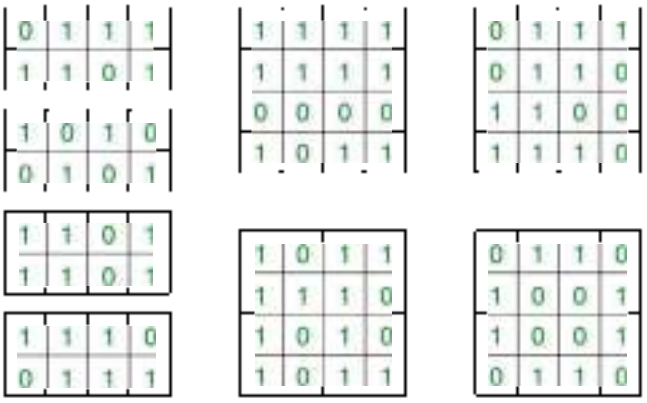
Q5 = $\overline{(A + B) \cdot C \cdot D - E}$

Q6 = $\overline{A \oplus \overline{B}\ (C + \overline{D})}$

ACTIVIDADES FINALES

continuación

7. Dibuja en tu cuaderno los siguientes mapas de Karnaugh y realiza las agrupaciones de tal forma que el número de grupos sea el menor posible, incluyendo dentro de ellos el mayor número de unos en agrupaciones de 1, 2, 4, 8 y 16



8. Dibuja en tu cuaderno el cronograma correspondiente a la salida Q de la siguiente ecuación lógica:

$$Q = (\bar{A} + B) \cdot C$$

Figura 1.46. Cronograma actividad 8

9. Representa y simula el circuito secuencial, con realimentaciones, que permita el control de dos salidas digitales, Q1 y Q2, según lo mostrado a continuación:

Activación	Desactivación
Q1 = A, B o C	Q1 = D o E
Q2 = A o E	Q2 = B, F o G

10. Tres salidas, Q1, Q2 y Q3, se activan y se desactivan según la siguiente tabla:

Salida	Activar	Desactivar
Q1	A o B	C
Q2	E o F	C
Q3	A o G	C o H

Se han de establecer las siguientes condiciones de funcionamiento: Q3 no se puede activar hasta que lo haga Q2 y Q2 no lo puede hacer si esta activada Q1.

11. Representa y simula los circuitos de las actividades 9 y 10 mediante biestables.

PRÁCTICA PROFESIONAL PROPUESTA 1

Herramientas

- PC y simulador de electrónica digital

Material

- Cuaderno de trabajo

Comprobación de un circuito con puertas lógicas

Objetivo

- Obtener la ecuación lógica partiendo de un circuito.
- Obtener la tabla de la verdad partiendo de la ecuación lógica.
- Simplificar el resultado, si es posible, y representar el esquema correspondiente.

Precauciones

- Pon especial atención para no olvidar ninguna de las negaciones de las variables en todo el proceso de resolución, ya que un cambio de valor dará un resultado completamente diferente.

Desarrollo

1. Fíjate en el siguiente circuito lógico y obtén su ecuación lógica.

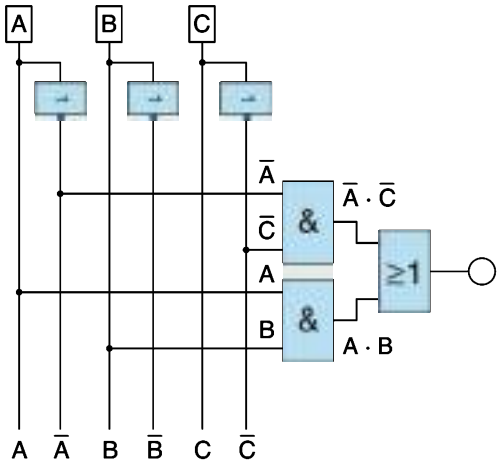


Figura 1.47. Circuito.

2. Partiendo de la ecuación lógica, obtén la tabla de la verdad de tres variables y calcula el resultado para la salida.

$$Q = \overline{A} \cdot \overline{C} + A \cdot B$$

A	B	C	Operación	Q
0	0	0	$(\overline{0} \cdot \overline{0}) + (0 \cdot 0) = (1 \cdot 1) + (0 \cdot 0)$	
0	0	1	$(\overline{0} \cdot \overline{1}) + (0 \cdot 0) = (1 \cdot 0) + (0 \cdot 0)$	
0	1	0	$(\overline{0} \cdot \overline{0}) + (0 \cdot 1) = (1 \cdot 1) + (0 \cdot 1)$	
0	1	1	$(\overline{0} \cdot \overline{1}) + (0 \cdot 1) = (1 \cdot 0) + (0 \cdot 1)$	
1	0	0	$(\overline{1} \cdot \overline{0}) + (1 \cdot 0) = (0 \cdot 1) + (1 \cdot 0)$	
1	0	1	$(\overline{1} \cdot \overline{1}) + (1 \cdot 0) = (0 \cdot 0) + (1 \cdot 0)$	
1	1	0	$(\overline{1} \cdot \overline{0}) + (1 \cdot 1) = (0 \cdot 1) + (1 \cdot 1)$	
1	1	1	$(\overline{1} \cdot \overline{1}) + (1 \cdot 1) = (0 \cdot 0) + (1 \cdot 1)$	

PRÁCTICA PROFESIONAL PROPUESTA 2

Herramientas

- PC y simulador de electrónica digital

Material

- Cuaderno de trabajo

Obtención de circuitos simplificados desde una tabla de la verdad

Objetivo

- Obtener las ecuaciones basadas en minterms desde una tabla de la verdad.
- Simplificar ecuaciones mediante mapas de Karnaugh.
- Dibujar y simular circuitos lógicos partiendo de ecuaciones lógicas.

Precauciones

- Pon especial atención para no olvidar ninguna de las negaciones de las variables en todo el proceso de resolución, ya que un cambio de valor dará un resultado completamente diferente.

Desarrollo

1. Fíjate en la siguiente tabla de la verdad.

A	B	C	D	Q1	Q2	Q3
0	0	0	0	0	1	1
0	0	0	1	1	0	0
0	0	1	0	1	1	1
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	1
0	1	1	0	0	0	1
0	1	1	1	0	1	1
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	0	1	1
1	0	1	1	1	0	0
1	1	0	0	1	0	1
1	1	0	1	0	1	0
1	1	1	0	1	0	1
1	1	1	1	0	1	0

2. Escribe las ecuaciones de minterms de cada una de las salidas de la tabla.
3. Simplifica por Karnaugh las ecuaciones de cada una de las salidas.
4. Dibuja los circuitos de las ecuaciones simplificadas y comprueba con un *software* que su funcionamiento tiene como resultado el propuesto en la tabla de la verdad.

PRÁCTICA PROFESIONAL PROPUESTA 3

Herramientas

- PC y simulador de electrónica digital

Material

- Cuaderno de trabajo

Resolución de problemas de lógica combinacional

Objetivo

- Obtener las ecuaciones basadas en minterms desde una tabla de la verdad.
- Simplificar ecuaciones mediante mapas de Karnaugh.
- Dibujar y simular circuitos lógicos partiendo de ecuaciones lógicas.

Precauciones

- Pon especial atención para no olvidar ninguna de las negaciones de las variables en todo el proceso de resolución, ya que un cambio de valor dará un resultado completamente diferente.

Desarrollo

1. Fíjate en la siguiente tabla de la verdad.

ABCD	Q1	Q2	Q3	Q4	Q5
0000	1	1	0		
0001	0	1	0		
0010	1	1	0		
0011	0	1	1		
0100	1	0	1		
0101	0	0	0		
0110	1	0	0		
0111	0	0	0		
1000	0	1	1		
1001	1	1	1		
1010	1	1	0		
1011	0	1	0		
1100	1	0	0		
1101	0	0	1		
1110	1	0	1		
1111	1	0	0		

2. Escribe los resultados de la tabla de la verdad para Q4 sabiendo que es 1 siempre que tres de las variables tengan un mismo valor (0 o 1) y la cuarta sea diferente al valor de las otras tres.
3. Escribe el resultado de la tabla de la verdad para Q5 correspondiente a la siguiente ecuación lógica: $Q5 = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + \overline{A}\overline{B}\overline{C}\overline{D}$.
4. Simplifica por Karnaugh y obtén los resultados de las ecuaciones lógicas de todas las salidas (Q1, Q2, Q3, Q4 y Q5).
5. Comprueba con un *software* de simulación que los resultados son correctos.

PRÁCTICA PROFESIONAL PROPUESTA 4

Herramientas

- PC y simulador de electrónica digital

Material

- Cuaderno de trabajo

Circuitos de lógica secuencial con biestables

Objetivo

- Utilizar biestables para resolver problemas de lógica secuencial.
- Dibujar y simular circuitos lógicos partiendo de ecuaciones lógicas.

Precauciones

- Ten en cuenta que cuando en el enunciado se dice *o* corresponde con una función OR, y cuando se dice *y* corresponde con una función AND.
- Las entradas deben comportarse como pulsadores y no como interruptores.

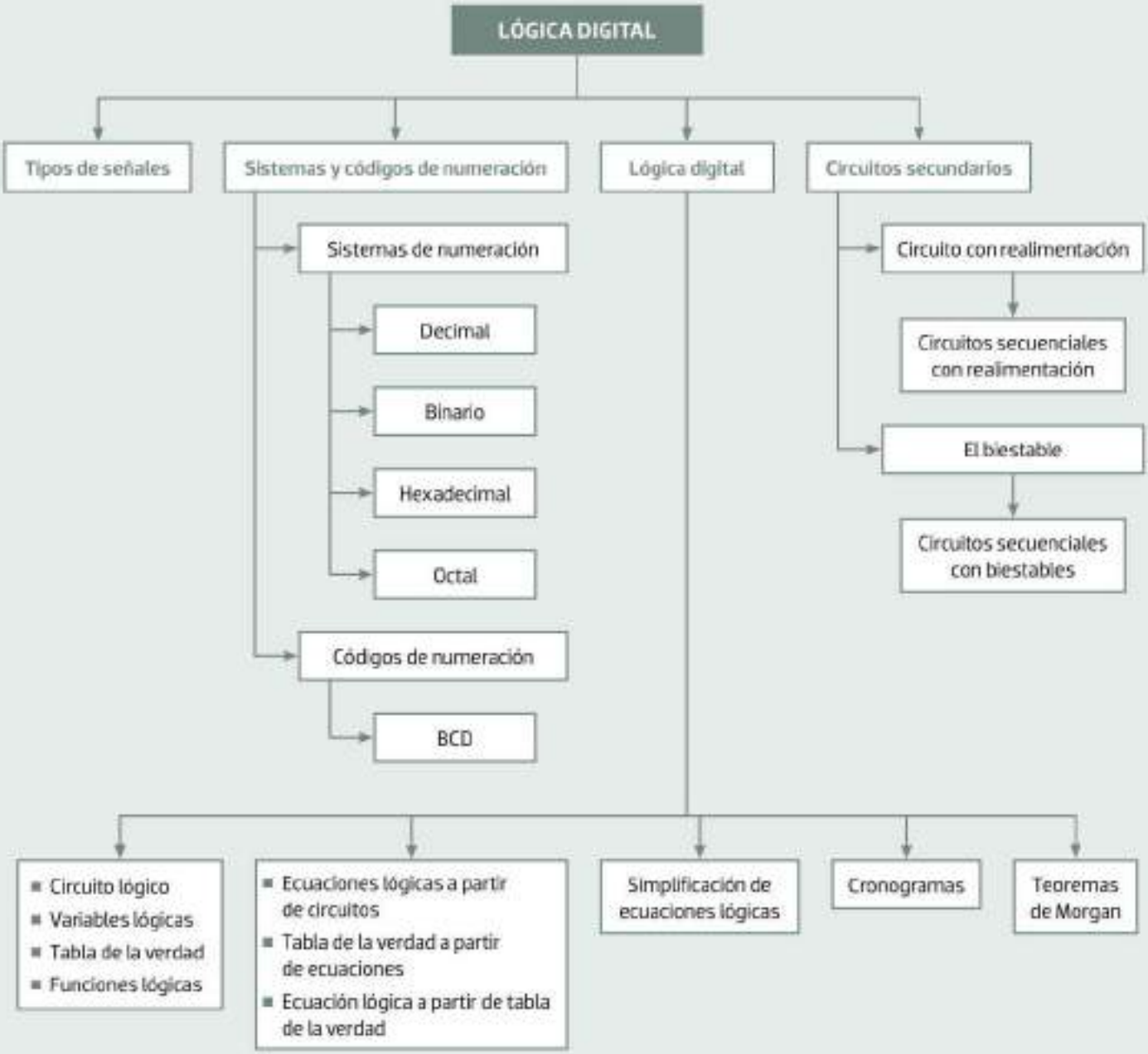
Desarrollo

1. Fíjate en la siguiente tabla con las condiciones de funcionamiento de un circuito secuencial de cinco salidas, controlado con ocho entradas.

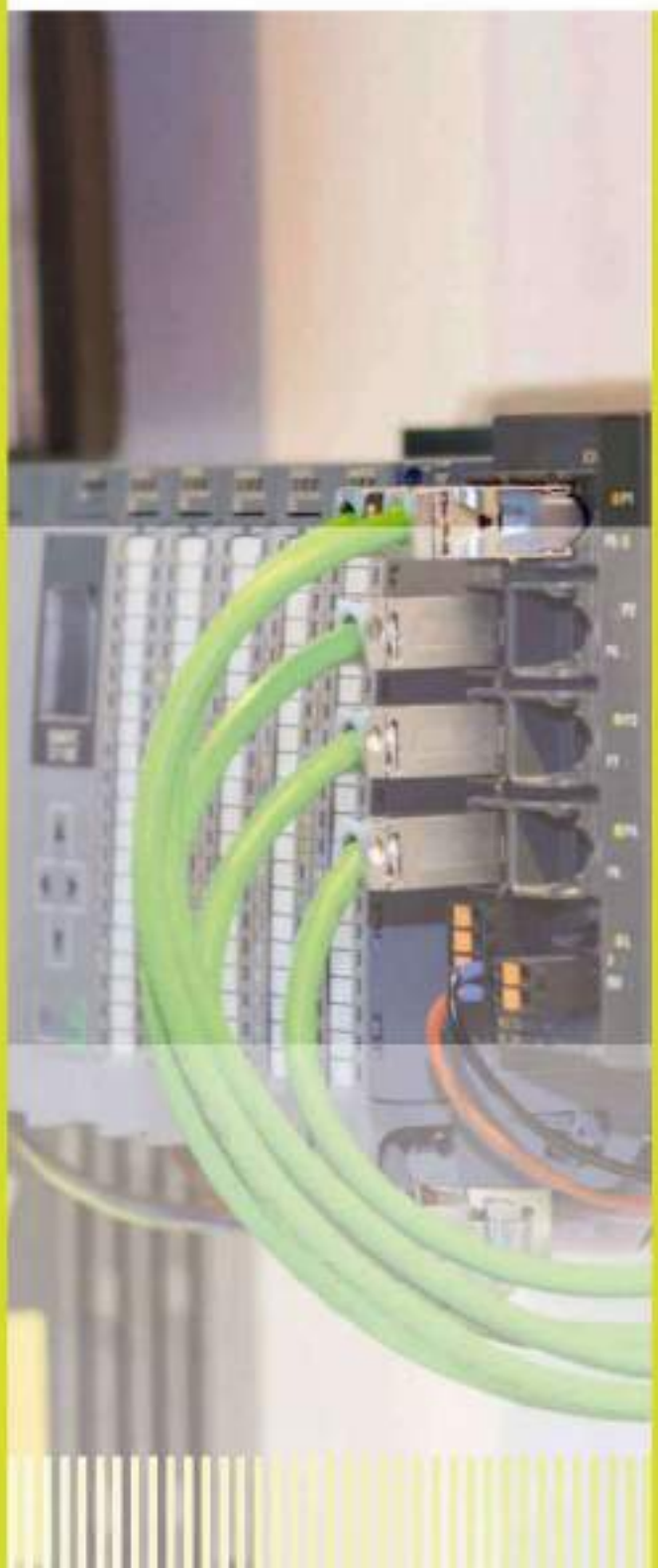
Salidas	Se activa desde	Se desactiva desde	Condición
Q1	A o B	G o C	No se activa si está Q2
Q2	C	G o A	No se activa si está Q1
Q3	D y E	G	Se activa si está Q1 o Q2 (no es necesario que estén las dos)
Q4	F	G a la vez que H	Solamente se activa si está Q3
Q5	A y C	G	No se activa si está Q4

2. Escribe las ecuaciones lógicas sabiendo que cada salida se controla con un biestable SR.
3. Comprueba su funcionamiento con un *software* de simulación electrónica, teniendo en cuenta que las entradas funcionan como pulsadores y no como interruptores.

EN RESUMEN


LÓGICA DIGITAL
Tipos de señales
Sistemas y códigos de numeración
Lógica digital
Circuitos secundarios
Sistemas de numeración
Decimal
Binario
Hexadecimal
Octal
Códigos de numeración
BCD
Circuito con realimentación
Circuitos secuenciales con realimentación
El biestable
Circuitos secuenciales con biestables
Circuito lógico
Variables lógicas
Tabla de la verdad
Funciones lógicas
Ecuaciones lógicas a partir de circuitos
Tabla de la verdad a partir de ecuaciones
Ecuación lógica a partir de tabla de la verdad
Simplificación de ecuaciones lógicas
Cronogramas
Teoremas de Morgan

2 Autómatas programables industriales



Vamos a conocer...

1. Sistemas automatizados
2. El autómata programable industrial

PRÁCTICA PROFESIONAL RESUELTA

Diseño del esquema de un automatismo con relé programable

PRÁCTICAS PROFESIONALES PROPUESTAS

1. Conexión de sensores y actuadores a un relé programable para el control de un montacargas
2. Conexión de sensores y actuadores a un PLC para el control de un taladro semiautomático

Y al finalizar esta unidad...

- Sabrás qué es un autómata programable y para qué se utiliza.
- Identificarás las partes que componen los autómatas programables.
- Reconocerás los diferentes tipos de PLC que existen en el mercado.
- Conocerás cómo se cablean los sensores y actuadores a los diferentes interfaces de E/S que disponen los autómatas programables.
- Representarás esquemas para la conexión de sensores y actuadores a relés y autómatas programables de diferentes sistemas y automatismos industriales.

1. Sistemas automatizados

En la unidad anterior has conocido los principios básicos de la lógica digital y cómo aplicarlos a la resolución de circuitos combinacionales y secuenciales. También has visto que es posible comprobar dichos circuitos con *software* de simulación específico; sin embargo, quizá te preguntes cómo se aplican dichos circuitos a máquinas y sistemas automáticos industriales.

Los circuitos lógicos, tanto combinacionales como secuenciales, se pueden implementar de forma «física» con diferentes tecnologías: eléctrica, neumática, hidráulica o programada. El uso de unas u otras dependerá del proceso que se vaya a controlar; no obstante, las técnicas programadas, al ser más flexibles y versátiles, son las más usadas en la actualidad, en combinación con las demás.

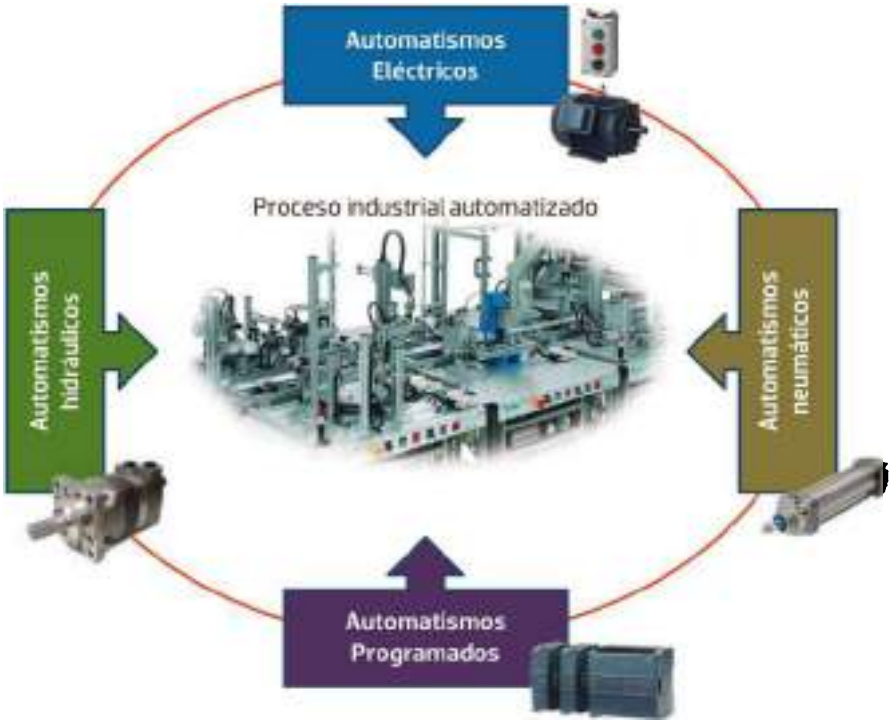


Figura 2.1. Formas de automatizar un proceso industrial.

El uso exclusivo de sistemas secuenciales cableados (eléctricos, neumáticos o hidráulicos) tiene ciertas limitaciones tanto en su concepción como en su posible ampliación o expansión, por lo que en la actualidad se usan en combinación con sistemas programados.

El dispositivo por excelencia para dar soluciones a procesos industriales automatizados programados es el denominado *autómata programable industrial,* que se estudiará a continuación, y es en el que se basan todos los contenidos de este libro.

El autómata programable nació a principios de los años setenta del siglo xx para flexibilizar la adaptación de circuitos de automatismos eléctricos, que se utilizaban en la cada vez más exigente y cambiante industria del automóvil. En aquella época cualquier modificación en el proceso automatizado significaba recablear grandes cuadros eléctricos basados en relés electromecánicos. Esta terea, además de tediosa, resultaba enormemente cara, por lo cual se comenzó a investigar una forma más flexible de implementar este tipo de circuitos manteniendo fijo el cableado de los cuadros eléctricos.

2. El autómata programable industrial

Un autómata programable, también denominado *PLC (programmable logic controller)* o *controlador lógico programable*, es un dispositivo electrónico capaz de gestionar sistemas de lógica digital de forma programada. Se puede decir que un autómata programable es un ordenador, o computadora, diseñado específicamente para realizar tareas de control en entornos industriales que necesitan un alto grado de automatización.

Figura 2.3. Montaje E-S con PLC.

Figura 2.2. Diferentes modelos de autómatas programables (Fuente Siemens AG).

Los autómatas programables leen señales de los sensores y envían información a los actuadores en función de un programa de usuario.

En la actualidad, el uso de los autómatas programables está generalizado en todos los sectores y permite resolver cualquier aplicación automatizada, desde las más sencillas hasta las más complejas.

Los autómatas deben estar diseñados para trabajar en entornos industriales, ya que su funcionamiento no debe verse afectado por las perturbaciones que este tipo de ambientes pueden ocasionar.

Ya que los autómatas programables nacieron para sustituir los grandes cuadros eléctricos de relés y contactores, es habitual iniciar su estudio comparándolos con los sistemas cableados.

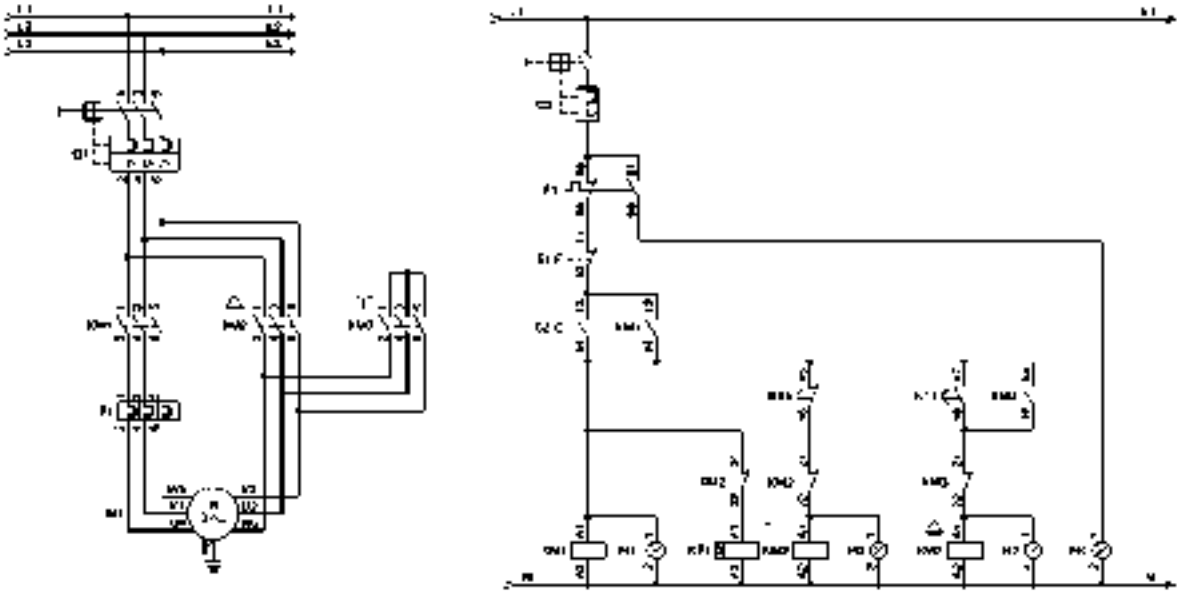
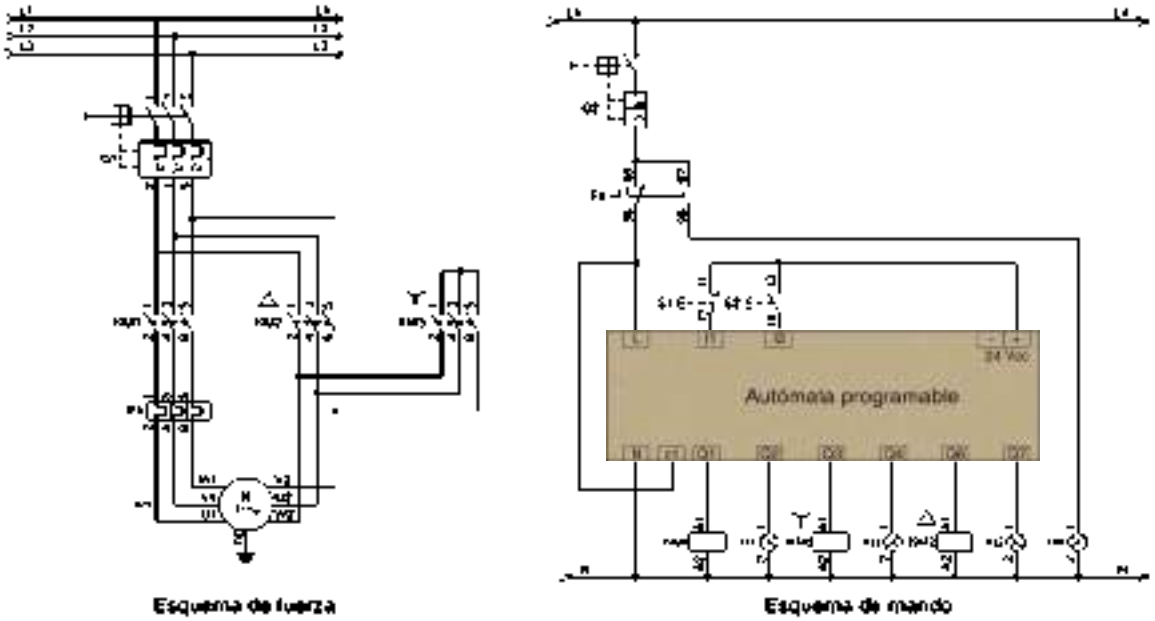


Figura 2.4. Arranque de un motor estrella-triángulo con lógica cableada.

En el ejemplo de la figura se muestra cómo se gestiona el arranque estrella-triángulo de un motor trifásico con rotor en jaula de ardilla. En él se observa que el circuito de mando es «cerrado» y que, una vez finalizado, cualquier cambio en el funcionamiento requiere recableario. Sin embargo, si se sustituye el circuito de mando por un autómata programable, una vez cableados los elementos de entradas y salidas, cualquier modificación en su forma de funcionar es mucho más simple que su equivalente cableado, ya que solamente requiere reprogramar el dispositivo.

Evidentemente, en cualquiera de los dos casos, el circuito de potencia sigue siendo el mismo.



Esquema de fuerza — Esquema de mando

Figura 2.5. Arranque de un motor estrella-triángulo con lógica programada.

2.1. Clasificación de los autómatas programables

Atendiendo a su modularidad, los autómatas pueden ser clasificados en dos tipos: compactos y modulares.

2.1.1. Autómatas programables compactos

Son aquellos que contienen todos sus elementos: E/S, CPU, fuente de alimentación, etc., en una misma envolvente.

S7-1200 de Siemens — Omron CP-2E

Figura 2.6 Diferentes modelos de autómatas compactos

Tienen la ventaja de ser más económicos que sus equivalentes modulares; sin embargo, presentan un gran inconveniente, y es que si alguna de sus partes se avería o deteriora es necesario sustituir todo el dispositivo.

Hace algunos años, muchos d
na posibilidad de expan-
sión. En la actualidad la
mayoría de ellos pueden
ser ampliados mediante
módulos de todo tipo: en-
tradas, salidas, tarjetas de
comunicación, etc.

Figura 2.7. PLC semicompacto con módulos de ampliación (Fuente Siemens AG)

Figura 2.8. Relé programable ZELIO (Fuente Schneider Electric)

Dentro de este grupo cabe destacar los que se han denominado *relés progra mables*, que algunos fabricantes están desarrollando con gran éxito para aplicaciones domésticas y gestión de pequeña maquinaria. Con un teclado básico (seis u ocho teclas) situado directamente en su frontal, es posible realizar tareas de programación y parametrización de una forma rápida y sencilla. Además, presentan la posibilidad de ser conectados a un ordenador personal para la edición, monitorización y salvaguarda de los programas del usuario.

Actualmente muchos de estos relés programables suelen disponer de posibilidades de expansión y comunicación.

Figura 2.9. Relé programable LOGO (Fuente Siemens AG)

2.1.2. Autómatas programables modulares

En estos dispositivos cada uno de sus elementos está ubicado en un módulo diferente, módulos que se instalan sobre un rack común. Las posibilidades de expansión son enormes si se comparan con los de tipo compacto, pero su coste es mucho más elevado.

Figura 2.10. Autómata modular S7-1500 (Fuente Siemens AG).

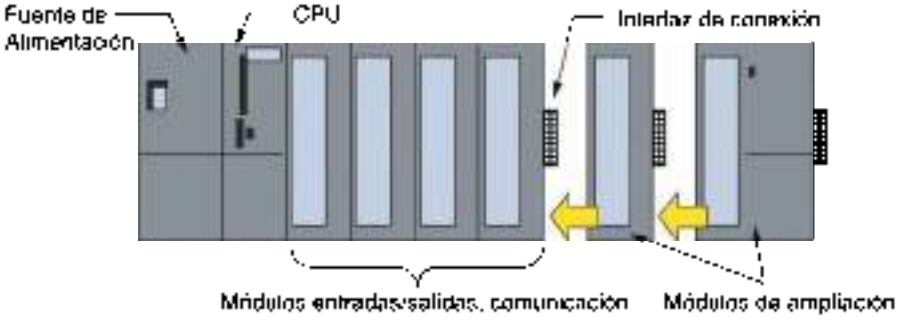


Figura 2.11. Ejemplo de autómata programable modular

2.2. Estructura del autómata programable

El autómata programable es un dispositivo electrónico basado en un microprocesador (CPU) instalado en una placa base que está conectada a los interfaces de entradas y salidas, por los cuales se reciben las señales de los sensores y se envían las órdenes de control a los actuadores.

En el siguiente diagrama de bloques se muestra la estructura básica de un autómata programable:

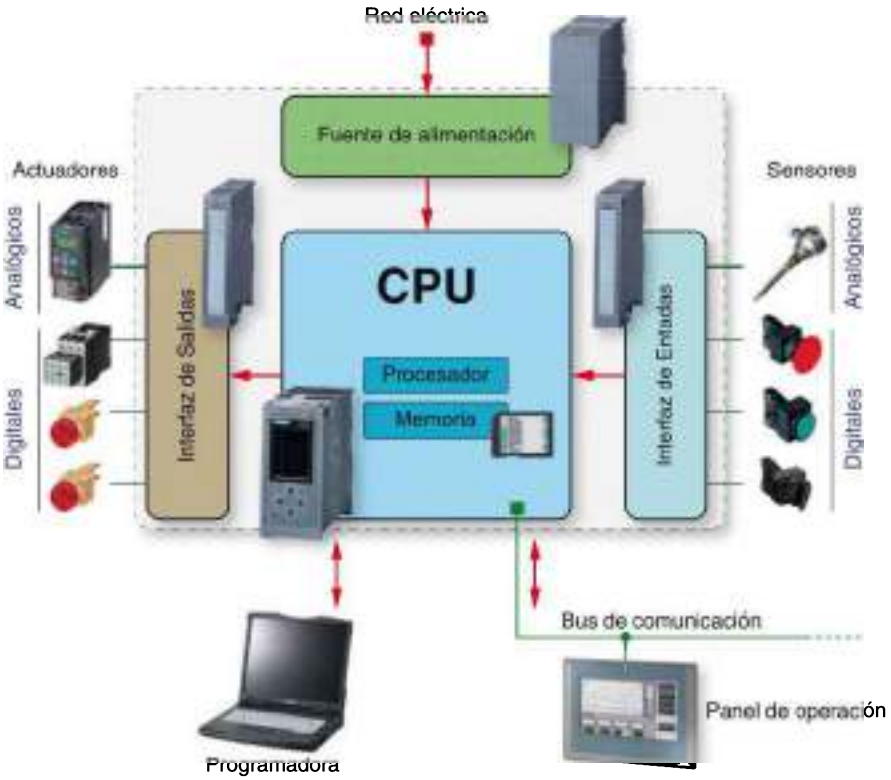


Figura 2.13. Estructura básica de un autómata programable (Fuente Siemens AG).

Todo el conjunto se alimenta con tensión de corriente continua, normalmente 24 V_{DC}, procedente de una fuente de alimentación. Esta puede estar integrada en la misma placa electrónica del circuito principal o puede ser un dispositivo externo.

A continuación se describe cada una de las partes de un PLC.

2.2.1. Unidad central de procesos

La unidad central de procesos (CPU) es el cerebro del autómata. Está constituida, principalmente, por el microprocesador y la memoria.

Tiene como misión procesar el programa de usuario y, externamente, dispone de un conjunto de indicadores led que muestran información sobre su estado de funcionamiento (RUN/STOP) o alguna posible advertencia o error que se ha producido en su funcionamiento. Los modelos más avanzados disponen de pequeñas pantallas LCD con las que se facilita la configuración y la comunicación con el operario.

Suele disponer de un interruptor (RUN/STOP) para poner en marcha y detener la ejecución del programa. No obstante, algunos modelos prescinden de dicho interruptor y es necesario activar los modos de funcionamiento RUN/STOP por *software*.

Figura 2.12. CPU de un S7–300 de Siemens.

Figura 2.14. Led indicadores del modo de funcionamiento de la CPU.

Figura 2.15. Dos tipos de CPU S7-1500 (Fuente Siemens AG).

En el módulo de la CPU suele estar ubicado el interfaz de conexión por el que se realiza la comunicación con la programadora. Además, puede disponer de diferentes puertos para su comunicación con otros dispositivos de campo.

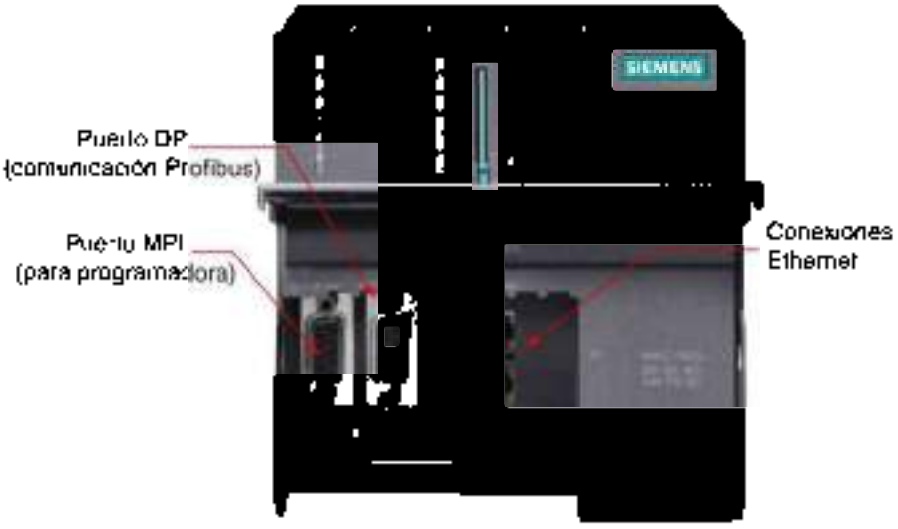


Figura 2.16. Ejemplo de posibles puertos de comunicación en una CPU de un autómata programable S7-300 (Fuente Siemens AG).

En la actualidad, tanto la comunicación con otros dispositivos industriales como con la programadora, suele hacerse mediante conexiones Ethernet que, al ser un sistema estandarizado, facilita el cableado y la configuración.

Los programas se almacenan y se ejecutan en la memoria del PLC, que puede ser de dos tipos:

- **Memoria volátil**: también denominada *memoria de trabajo* o *memoria RAM*, es donde se ejecuta el programa del usuario. Al ser volátil su contenido se pierde cuando el dispositivo deja de recibir la alimentación de la red eléctrica. Los autómatas programables suelen disponer de una batería (los más antiguos) o un condensador de alta capacidad (los más modernos) para evitar que los datos de la memoria se pierdan ante un corte de tensión.
- **Memoria no volátil**: es la memoria que no se pierde, aunque falle la alimentación del dispositivo. Tradicionalmente, para este tipo de memoria se han estado utilizando cartuchos EEPROM, que debían instalarse en el PLC bien de forma fija o extraíble. Sin embargo, en la actualidad, es habitual el uso de tarjetas de memoria (*memory card*), que suelen ser más económicas y fáciles de instalar. Dichas tarjetas funcionan como memoria de carga, almacenándose en ella la configuración de *hardware* del equipo, el programa de usuario y otros tipos de datos que interese guardar, evitando así su pérdida ante un inesperado corte en la alimentación del dispositivo.

Figura 2.17. Diferentes tipos de tarjetas de memoria para autómatas programables (Fuente Siemens AG).

Actividades

1. Utilizando varios modelos de PLC que poseas en el aula, localiza en ellos los puertos de que dispongan para su comunicación con el exterior. Haz una foto de ellos y, utilizando la documentación del fabricante, busca para qué se utiliza cada uno de ellos.

2.2.2. Fuente de alimentación *(power supply)*

Tiene la misión de convertir la corriente alterna de la red eléctrica en la corriente continua, a 24 V_{DC}, necesaria para alimentar los circuitos electrónicos del interior del autómata.

Si la fuente de alimentación tiene la potencia adecuada también se puede usar para alimentar los sensores y actuadores conectados al PLC. En este caso es necesario tener en cuenta la corriente que consume cada uno de ellos para evitar sobrecargarla.

Este tipo de fuentes son similares a las utilizadas para aplicaciones de propósito general, por lo que podría utilizarse cualquiera que cubra las necesidades eléctricas del sistema en tensión y en corriente.

Figura 2.18. Fuentes de alimentación de autómatas (Fuente Siemens AG).

2.2.3. Interfaces de entradas y salida digitales (I/O)

Están formadas por un conjunto de módulos, bornes de conexionado y soportes cuyas principales funciones son:

- Adaptar la tensión de trabajo de los sensores y actuadores a los dispositivos electrónicos del autómata.
- Recibir señales de los sensores, en el caso de las entradas, y enviar señales a los actuadores, en el caso de las salidas.

Vocabulario

Los módulos de entradas/salidas se denominan, de forma abreviada, E/S (o, en inglés, I/O (*input/ouput*).

2.2.4. Módulo de entradas digitales *(digital input)*

Este módulo tiene como misión recibir las señales de los sensores del proceso o de la máquina. La información del estado de dichas señales se almacena en una zona de memoria que posteriormente es procesada por la CPU, según el programa del usuario residente en la memoria.

A este módulo se unen eléctricamente los captadores (interruptores, finales de carrera, pulsadores, sensores, detectores de posición, etc.).

Recuerda

Las entradas se identifican con el símbolo nemotécnico «I» y las salidas lo hacen con el símbolo «Q».

Figura 2.19. Módulo de entradas de un S7–1500 (Fuente Siemens AG).

Las **entradas digitales** captan señales de tipo discreto que varían su estado ante cambios de tensión todo o nada. Esto permite evaluar si la señal está a 0 o a 1 lógico. El valor de tensión depende del modelo del módulo de entradas y, aunque es habitual trabajar con valores de corriente continua a 24 V, existen autómatas que pueden trabajar a otras tensiones, como 12 V_{DC} e incluso entre 120 y 240 V_{AC}.

Entradas a 24 V_{DC}

Son los tipos de señales más utilizadas en la industria. Se aplican desde una fuente de alimentación con salida a 24 V_{DC}, interna en el caso de los PLC compactos y externa en los modulares.

En este caso, el valor lógico de la señal digital de entrada puede hacerse con lógica positiva o con lógica negativa. Así, los valores lógicos, en función de la polaridad de entrada, se representan en la siguiente tabla:

	Lógica positiva	Lógica negativa
1 lógico	Positivo (+)	Negativo (–)
0 lógico	Negativo (–)	Positivo (+)

Recuerda

Cuando se hable de conexión PNP o NPN, recuerda que la primera letra es la que define la polaridad que sale del dispositivo o sensor. Es decir, **P**NP–positivo, **N**PN–negativo.

Recuerda

No todos los PLC permiten trabajar con los dos tipos de polaridad (lógica positiva o negativa), por lo que siempre es necesario consultar el manual de operación antes de realizar el cableado.

La lógica positiva también se denomina PNP o *sinking* y la lógica negativa NPN o *sourcing*.

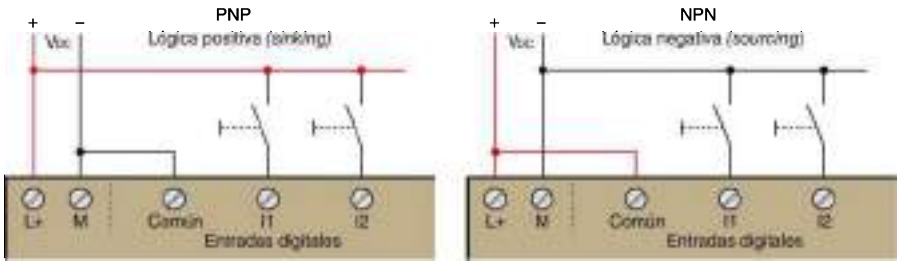


Figura 2.20. Tipo de lógica de las entradas digitales.

Algunos equipos tienen la posibilidad de realizar un tipo de conexión u otra. En este caso será necesario conectar el borne común de las entradas al positivo o al negativo, según indique el fabricante en el manual de instalación del dispositivo.

Ejemplo

A continuación se muestra cómo configurar el cableado de las entradas de un PLC S7-1200 de Siemens para trabajar en lógica positiva (sinking) o en lógica negativa (sourcing).

Lógica positiva *(sinking)*:

Lógica negativa *(sourcing)*:

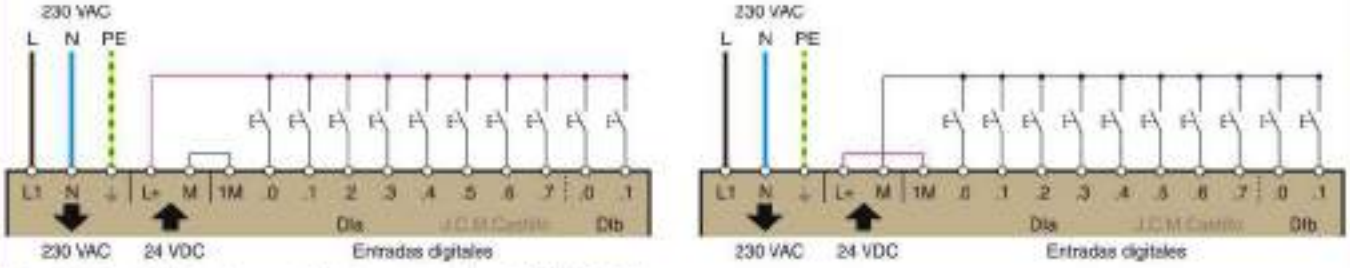


Figura 2.21. Posibilidades de conexión de la entradas de un S7-1200 de Siemens.

En ambos casos, la configuración de un tipo u otro se determina según la forma de referenciar el borne común de las entradas (1M). Si se hace desde el negativo (–) de la fuente, las entradas funcionan con lógica positiva, por lo cual todos los sensores deben conectarse al positivo. Si, por el contrario, el borne 1M se referencia al positivo (+), el modo de funcionamiento es en lógica negativa y los sensores deben conectarse al negativo. En ambas configuraciones, todas las entradas integradas de este PLC quedan condicionadas a uno de los modos de funcionamiento.

Los PLC de tipo compacto, especialmente los que se conectan directamente a la red de 230 V_{AC}, suelen disponer de una fuente de alimentación de salida a 24 V_{DC} que puede ser utilizada para conectar los sensores pasivos electromecánicos que envían señales a las entradas, además de detectores activos (inductivos, capacitivos, fotoeléctricos...) que requieren alimentación para su funcionamiento. Estos últimos tienen un determinado consumo de corriente, por lo que se debe tener en cuenta para no sobrecargar la salida de esta fuente de alimentación.

Si el número de sensores activos es elevado, la mejor solución consiste en poner una fuente externa para alimentarlos. En este caso, el cable de referencia, como puede ser la masa en un sistema de lógica positiva, debe unirse a la masa de la otra fuente de alimentación.

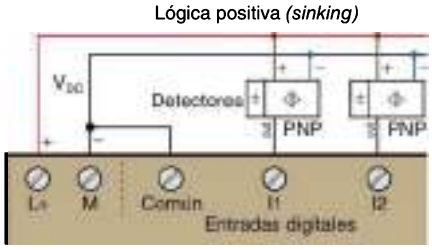


Figura 2.23. Conexión de detectores de tres hilos a la fuente de salida del propio PLC.

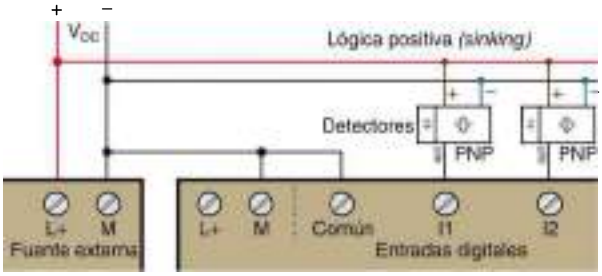


Figura 2.24. Conexión de detectores activos de tres hilos a una fuente de alimetacióne externa.

Entradas a 110–240 V_{AC}

No es una configuración tan habitual como la que ya se ha visto para las entradas a 24 V_{DC}, pero algunos autómatas, especialmente de relés programables, disponen de modelos cuyas entradas consiguen su valor lógico mediante la aplicación de la fase (L) del sistema de alimentación de 230 V_{AC}.

Estos equipos tienen amplia utilización en aplicaciones domóticas y pequeños automatismos industriales.

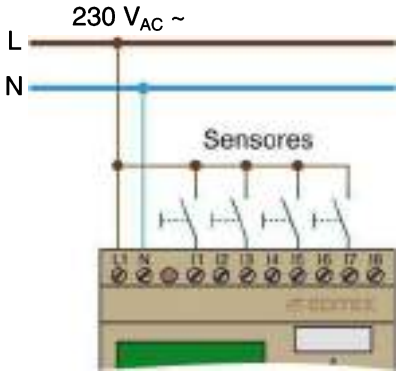


Figura 2.26. Conexión de captadores pasivos a 230 V_{AC}.

Detector de tres hilos

Los detectores de tres hilos son de tipo PNP o NPN. Los dos tipos requieren alimentación eléctrica en V_{DC}, que se aplica a los terminales + (color marrón) y – (color azul). Cuando el detector se activa, la señal de salida se presenta en el cable *Out* (color negro), siendo de valor positivo en los de tipo PNP y de valor negativo en los NPN.

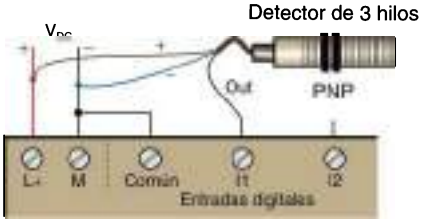


Figura 2.22. Ejemplo de conexión de un detector de tres hilos tipo PNP.

Conexión de entradas libres

Algunos autómatas disponen de dos bornes para cada captador.

Este método es rápido de implementar, ya que no es necesario tener en cuenta la fase de la alimentación que se debe aplicar a los captadores; sin embargo, requiere más cableado en la instalación.

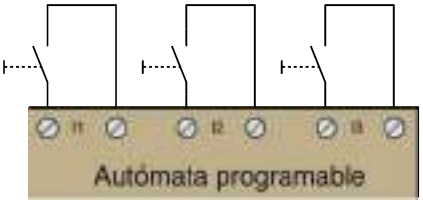


Figura 2.25. Entradas libres de tensión.

2.2.5. Módulo de salidas digitales *(digital output)*

Este módulo tiene como misión enviar las señales de activación y desactivación a los actuadores (bobinas de contactores y relés, electroválvulas, lámparas de señalización, etc.).

Figura 2.27. Módulo de salidas digitales.

Las salidas tienen una zona de memoria en la que se almacena su valor una vez que ha sido procesado el programa de usuario.

Las salidas digitales pueden ser de dos tipos: a relés y a transistor.

Salidas a relés *(relay output)*

Se denominan de esta manera debido a que su funcionamiento se basan en el disparo interino de un relé electromecánico cuyo contacto es usado para el control externo de los actuadores.

Son libres de tensión y, por tanto, se pueden utilizar para cualquier sistema de alimentación (AC, DC), incluso en la misma aplicación.

Los contactos de salida se pueden presentar como contactos independientes o con un borne común. En este segundo caso, todos los actuadores que se conectan al mismo bloque deben utilizar el mismo tipo de alimentación.

Importante

Es importante conocer el poder de corte de las salidas para conectar en ellas los actuadores adecuados. Si esto no se tiene en cuenta, es posible que se deterioren o destruyan.

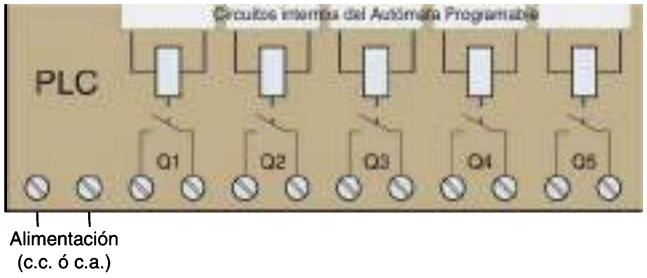


Figura 2.28. Salidas a relés independientes.

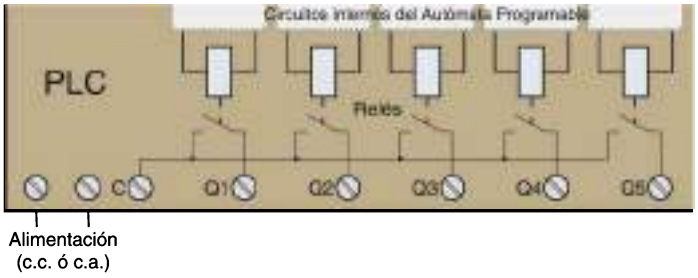


Figura 2.29. Salidas a relés con borne común.

Salidas a Transistor

Las salidas a transistor, también denominadas *DC* o *salidas a colector abierto,* se utilizan para activar actuadores de corriente continua.

Son salidas que pueden realizar operaciones de conmutación muy rápidas, por lo que pueden ser usadas para generar trenes de pulsos.

De igual forma que ocurre con las entradas, las salidas a transistor pueden ser de lógica positiva (PNP) o de lógica negativa (NPN). En el primer caso, cuando la salida está activa saca por ella el positivo de la alimentación, y en el segundo caso, saca el negativo.

Preactuadores

Un preactuador es un actuador que, a su vez, se encarga de gestionar un dispositivo de mayor potencia o diferente tecnología (neumática o hidráulica).

La conexión de las cargas a las salidas se realiza como se muestra en las siguientes figuras:

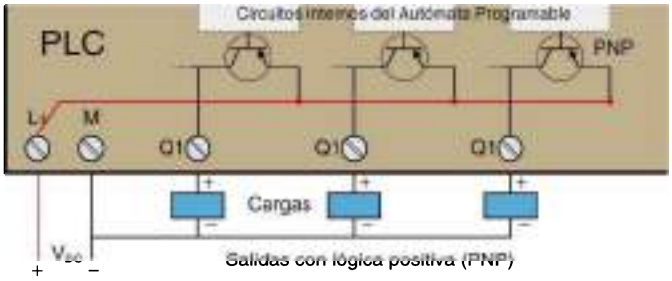


Figura 2.30. Salidas DC con lógica positiva.

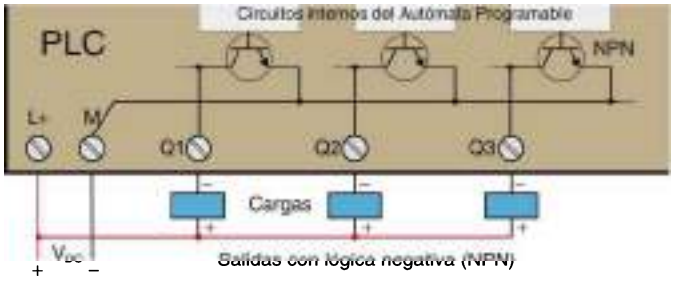


Figura 2.31. Salidas DC con lógica negativa.

Hay que tener en cuenta que la conmutación de cargas inductivas como relés, electroválvulas, etc., a través de las salidas DC de los autómatas programables, producen picos de tensión que pueden dañar de forma irremediable sus transistores finales.

La conexión de un diodo volante, conectado en paralelo y polarizado en inversa, con la bobina de la carga, amortigua dichos picos y protege las salidas del PLC.

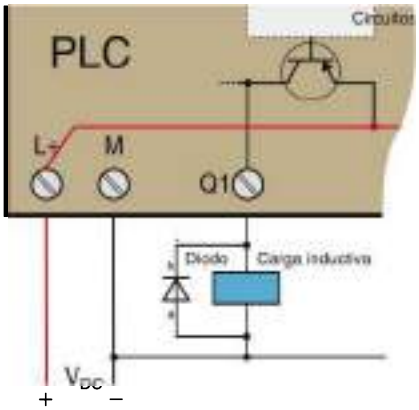


Figura 2.32. Protección de las salidas DC con diodo volante con cargas inductivas.

Ejemplo

A continuación, se muestra diferentes formas de conectar actuadores a las salidas de los PLC:

1. Conexión de actuadores de corriente continua a salidas y a transistor con borne común.
2. Conexión de actuadores de corriente alterna a salidas y a relés con borne común.

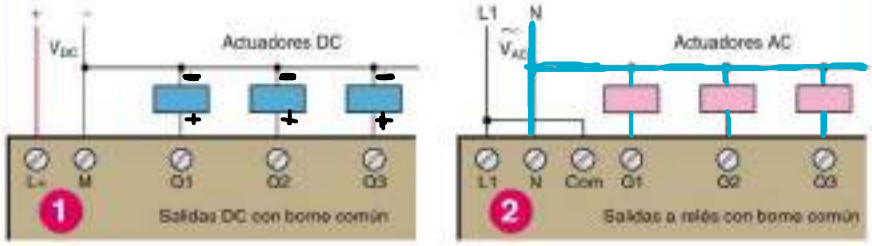


Figura 2.33. Ejemplos de conexión de actuadores a salidas.

3. Conexión de actuadores de corriente continua a salidas DC con bornes independientes.
4. Conexión de actuadores de correinte alterna y corriente continua a salidas a relés con bornes independientes.

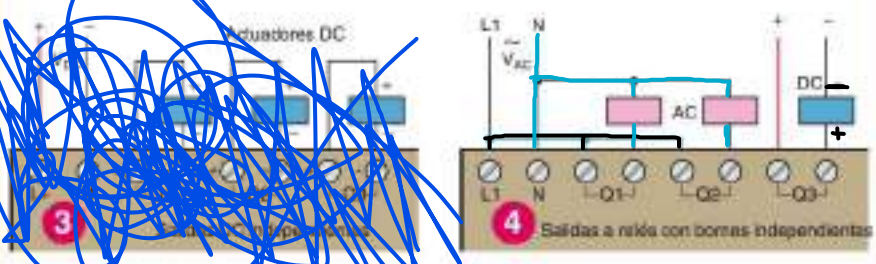


Figura 2.34. Ejemplos de conexión de actuadores a salidas.

2.3. Señales analógicas

Las señales analógicas permiten transferir valores de señales físicas, como temperatura, distancia, masa, etc., a través señales eléctricas que se procesan como datos numéricos en el interior del PLC.

Las señales eléctricas analógicas pueden ser en tensión o en corriente. Estas se encuentran estandarizadas y las más comunes son las que se enumeran a continuación.

Señales bipolares y unipolares

Las señales analógica bipolares procesan señales eléctricas con valores negativos y positivos. Las unipolares solo procesan señales con valores positivos.

Estándar de tensión	Estándar de corriente
■ ±10 V_{DC} (Bipolar) ■ 0-10 V_{DC} ■ 0-5 V_{DC}	■ 0 a 20 mA ■ 4 a 20 mA

Ejemplo

A continuación se muestra el escalado de dos señales analógicas, una para medir distancia mediante un sensor inductivo de proximidad y otra para asignar velocidad a un motor eléctrico a través de un variador de frecuencia.

En el primer caso, la señal procede de un detector de proximidad analógico cuya salida es de 4 mA a 20 mA. El rango de medida del sensor es de 1 mm a 10 mm. Así, el valor entregado por el sensor para el valor mínimo será de 4 mA y para el valor máximo será de 20 mA.

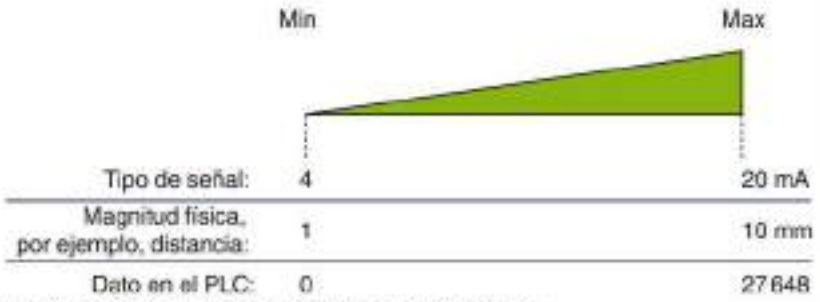


Figura 2.35. Escalado de una señal analógica para medir distancia.

En el segundo caso la señal analógica es de tipo bipolar, ya que el rango de la señal está entre –10 V y +10 V. Así, con valores negativos el variador hace girar el motor en un sentido y con positivos en sentido contrario. El escalado de la señal física se hace en rpm (revoluciones por minuto), de forma que el valor de tensión mínimo (–10 V) corresponde con –1500 rpm y el valor máximo con +100 rpm. Si la salida analógica entrega 0 V el motor se para.

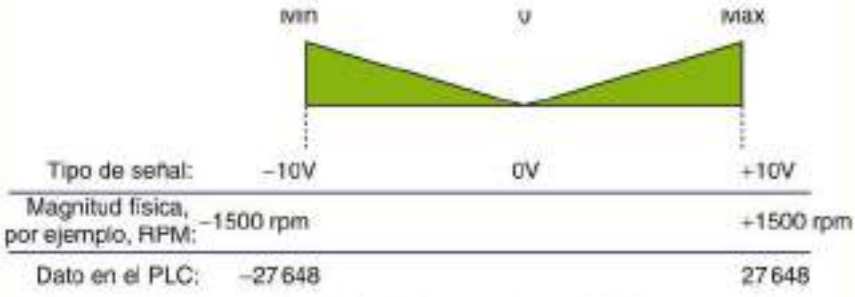


Figura 2.36. Escalado de una señal analógica bipolar para asignar velocidad a un motor eléctrico.

Los autómatas programables suelen disponer tanto de entradas como salidas analógicas. Estas pueden estar integradas en el módulo de la propia CPU o en módulos de expansión.

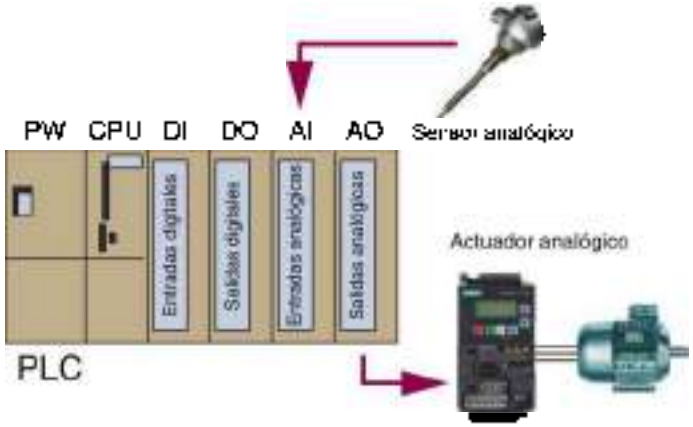


Figura 2.37. Entradas y salidas analógicas en un PLC.

El cableado de las señales analógicas puede ser crítico, ya que las interferencias generadas en entornos industriales pueden falsearlas. Los cables deben tener una malla de apantallamiento conectada a la masa del sistema eléctrico y no deben superar la distancia aconsejada por el fabricante.

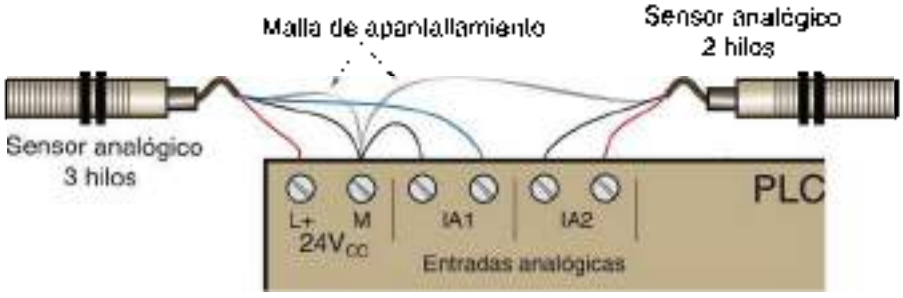


Figura 2.38. Ejemplo de conexión de sensores a entradas analógicas.

Tanto las entradas como las salidas analógicas de los autómatas programables suelen ser configurables, por lo que se puede elegir el tipo de señal mediante *software*, conmutación de microrruptores o cableado.

Algunos módulos de entradas/salidas analógicas disponen de más de dos bornes para la conexión de sensores/actuadores. Estos se utilizan para conectar dispositivos de más de dos hilos o elegir el tipo de señal que se va a procesar (tensión o corriente).

Importante

Debes seguir las indicaciones del fabricante para conectar las entradas/salidas analógicas de tus dispositivos.

Figura 2.39. Conexión en tensión o en corriente en la misma entrada analógica.

2.4. Programación de los autómatas programables

La programación de autómatas requiere disponer de un *software* diseñado por el fabricante y un ordenador en el que ejecutarlo.

El equipo de programación se comunica con el PLC mediante un cable que permite la transferencia bidireccional de la configuración y los programas de usuario.

Hasta no hace mucho tiempo, el cable de programación debía ser el que el propio fabricante comercializaba para tal fin. Actualmente, casi todos los PLC del mercado disponen de una toma RJ-45 que facilita su conexión con el dispositivo de programación a través del estándar Ethernet.

Redes Ethernet

Ethernet es un estándar para la creación de redes entre equipos informáticos. Desde el punto de vista del cableado, se caracteriza por utilizar latiguillos con conectores RJ-45, que son muy económicos y facilitan la conexión entre dispositivos.

Figura 2.40. Cable de programación específico

Figura 2.41. Cable de programación Ethernet

Figura 2.42. Diferentes modelos de cables y adaptadores para la programación de autómatas a través de PC (Fuente: Siemens AG)

Los equipos de programación permiten la comunicación entre el usuario y el autómata.

Las principales funciones de un equipo de programación, entre otras muchas, son:

- Creación y edición de programas.
- Transferencia y descarga de programas.
- Configuración de los equipos y su comunicación con otros dispositivos.
- Creación de tabla de variables.
- Monitorización y observación del funcionamiento de los programas.

Figura 2.43. Software de configuración y programación TIA Portal.

2.4.1. Lenguajes de programación

El lenguaje de programación es el conjunto de instrucciones que utiliza el usuario para crear secuencias ejecutables en la memoria del autómata.

La norma IEC 1131-3 establece cinco lenguajes de programación que los fabricantes pueden implementar en sus dispositivos. De estos lenguajes, dos son textuales y tres de tipo gráfico.

Lenguajes textuales:

- **IL** *(Instruction List):* lista de instrucciones. Es el lenguaje más antiguo, ya que fue el que se utilizó, allá por los años setenta, en los primeros autómatas programables.
- **ST** *(Structured Text):* texto estructurado. Es un lenguaje que utiliza funciones de programación similares a los lenguajes de alto nivel usados para la creación de aplicaciones informáticas.

Leguajes gráficos:

- **LD** *(Ladder Diagram):* diagrama a contactos o escalera. Es un lenguaje similar al utilizado para representar los esquemas de automatismos industriales, por eso es tan popular entre los técnicos con conocimientos eléctricos.
- **FBD** *(Function Block Diagram):* diagrama de bloques funcionales. Utiliza bloque de funciones lógicas.
- **SFC** *(Secuential Function Chart):* es un diagrama que industriales. También es conocido como *GRAFCET*.

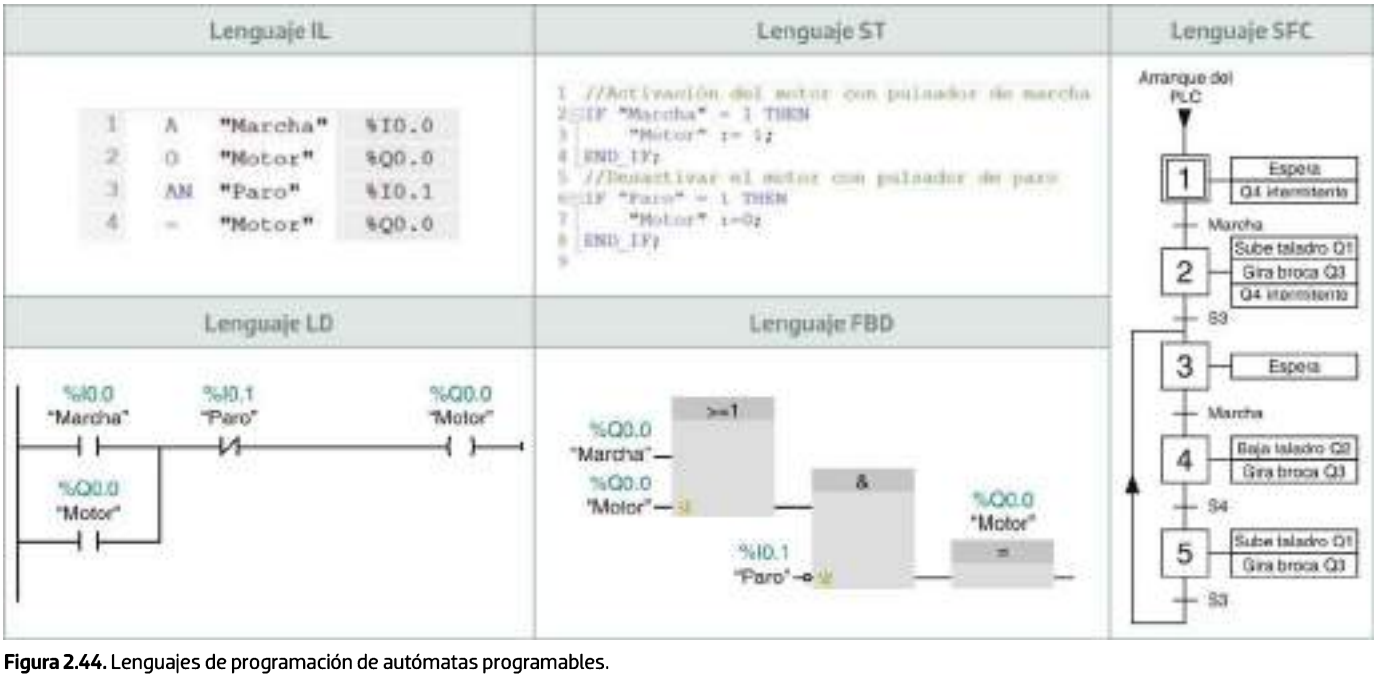


Figura 2.44. Lenguajes de programación de autómatas programables.

PRÁCTICA PROFESIONAL RESUELTA

Herramientas

- Material de dibujo

Material

- Cuaderno de trabajo
- Hoja de características de un relé programable comercial con entradas a 230 V y salidas a relés

Diseño del esquema de un automatismo con relé programable

Objetivo

- Identificar cada uno de los elementos que intervienen en el automatismo
- Representar el cableado de los sensores y actuadores en un relé programable de un automatismo industrial.

Precauciones

- Representar las conexiones de los sensores a las entradas y de los actuadores a las salidas.
- La apertura de la puerta se hace con un motor eléctrico gobernado por dos contactores. KM1 hace girar el motor para abrirla y KM2 para cerrarla
- El detector fotoeléctrico dispone de un circuito de salida basado en un relé. Así, cuando la barrera fotoeléctrica detecta algún vehículo, el contacto del relé se cierra. Por el contrario, cuando no se detecta ninguno, el contacto del relé se abre
- No es necesario representar el circuito de fuerza para controlar el motor eléctrico

Desarrollo

1. Fijándose en el automatismo de la figura, realizar una lista con los sensores y actuadores utilizados en el automatismo. Añadir en ella la cantidad necesaria de cada uno, su denominación y una breve descripción.

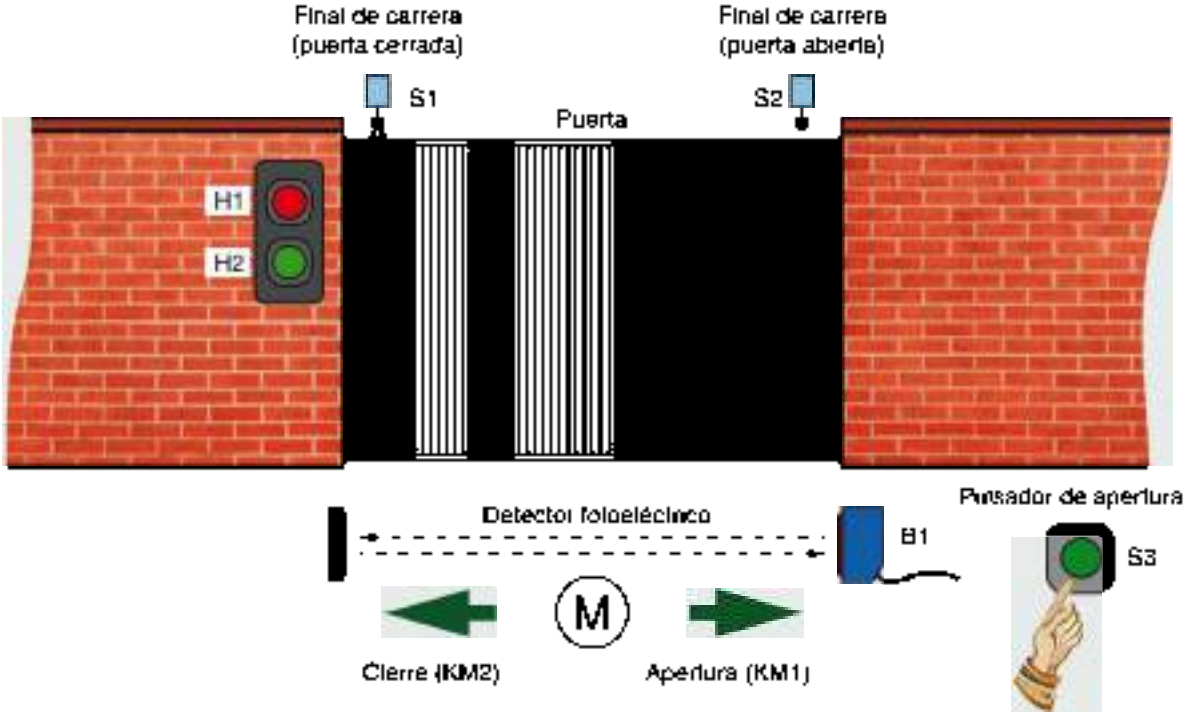


Figura 2.45. Automatismo de una puerta automática.

2. Añadir a la lista las entradas y salidas de relé programable utilizadas para los sensores y los actuadores.

Cantidad	Elemento	Identificador	Entrada	Salida	Descripción
1	Pulsador	S3	I3		Pulsador normalmente abierto para abrir la puerta.
2	Final de carrera	S1 / S2	I1 / I2		Finales de carrera para detectar si la puerta está abierta o cerrada.
1	Detector fotoeléctrico	B1	I4		Detector fotoeléctrico para comprobar la presencia de vehículos en la puerta.
2	Contactor	KM1/KM2		Q1/Q2	Contactores para apertura y cierre de la puerta.
2	Lámparas	H1/H2		Q3/Q4	Lámparas del semáforo.

3. Representar un esquema de conexión de los sensores y actuadores al relé programable sabiendo que:

a) El relé programable:

i) Se alimenta de la red eléctrica de 230 V_{AC}.

ii) Las entradas son de 230 V_{AC}.

iii) Las salidas son a relé con contactos independientes.

b) Todos los sensores tienen contactos normalmente abiertos.

c) El detector fotoeléctrico tiene una salida a relé que se debe conectar como cualquier otro sensor a su entrada correspondiente.

d) Las bobinas de los contactores son de 230 V_{AC}.

e) Las lámparas del semáforo también son de 230 V_{AC}.

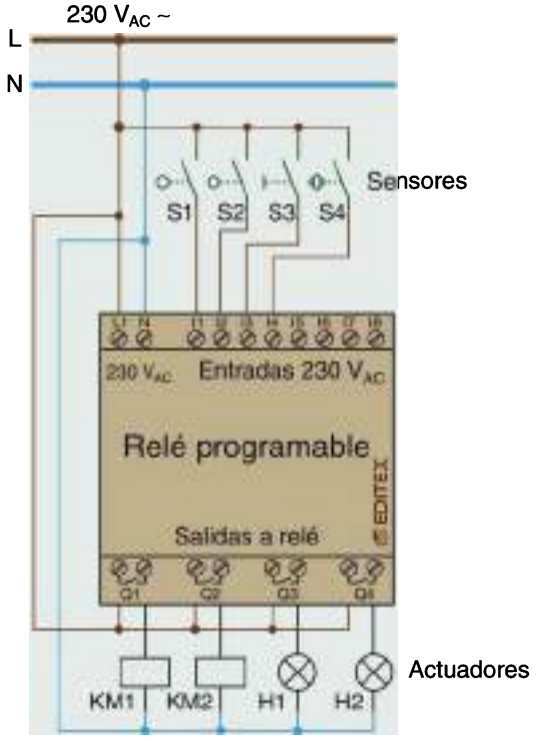


Figura 2.46. Esquema de conexión de actuadores y sensores al relé programable.

TEST DE EVALUACIÓN

RESUELVE EN TU CUADERNO O BLOC DE NOTAS

1. **Los autómatas programables se conocen como:**
   a) PC.
   b) PLC.
   c) PCL.
   d) Microcontroladores.

2. **Las señales de los sensores se conectan en un autómata programable:**
   a) Al módulo de entradas.
   b) Al módulo de salidas.
   c) Es indiferente.
   d) A la red eléctrica de 230 V_{AC}.

3. **El «cerebro» de un autómata programable está en:**
   a) La fuente de alimentación.
   b) Los módulos de E/S.
   c) El puerto de comunicación.
   d) La CPU.

4. **La memoria EEPROM es:**
   a) Un tipo de memoria volátil.
   b) Un tipo de memoria no volátil.
   c) Una parte de la fuente de alimentación.
   d) Un dispositivo que se instala en los módulos de entrada para protegerlos.

5. **Si se habla de I/O se está haciendo referencia a:**
   a) Un sistema de comunicación.
   b) Entradas y salidas.
   c) Zona de memoria de un PLC.
   d) Puerto para programar un autómata.

6. **En relación a la lógica positiva, ¿cuál de estas afirmaciones es correcta?**
   a) El 1 lógico se consigue con el negativo de la fuente de alimentación.
   b) Es un sistema NPN.
   c) Es un sistema PNP.
   d) Solo se pueden utilizar dispositivos de tres hilos.

7. **El denominado *diodo volante* se utiliza para:**
   a) Proteger las bobinas de los contactores.
   b) Para evitar picos de corriente en la conmutación de bobinas.
   c) Para evitar picos de tensión en la conmutación de bobinas.
   d) Para evitar excesivo consumo de potencia de las cargas inductivas.

8. **¿Cuál de estos datos no se corresponde con un estándar de señal analógica?**
   a) 1 a 25 mA.
   b) 0-10 V.
   c) 4-20 mA.
   d) 0-20 mA.

9. **¿Cuál de estos lenguajes de programación es el que más similitudes tiene con los circuitos eléctricos basados en contactos?**
   a) IL.
   b) SCL.
   c) FBD.
   d) LD.

10. **¿Cuál de estos lenguajes de programación utiliza puertas lógicas de forma gráfica?**
    a) IL.
    b) SCL.
    c) FBD.
    d) LD.

ACTIVIDADES FINALES

1. Busca en internet información sobre el autómata S7-1200 de la firma Siemens y di qué tipo de bloques son los marcados en la figura.

Figura 2.47. Autómata programable S7-1200 (Siemens AG)

2. Se desea invertir el sentido de giro de un motor trifásico cuyo esquema de fuerza se representa a continuación. Para ello son necesarios tres pulsadores: dos normalmente abiertos, uno para cada sentido de giro, y un pulsador normalmente cerrado para el paro. Dibuja el esquema de conexión de dichos pulsadores y las bobinas de los contactores requeridos, para hacer el control mediante un relé programable con entradas a 230 V_{AC} y salidas independientes a relés. Añade dos lámparas de señalización a dicho esquema, una para saber cuándo el motor gira a izquierdas y la otra a derechas.

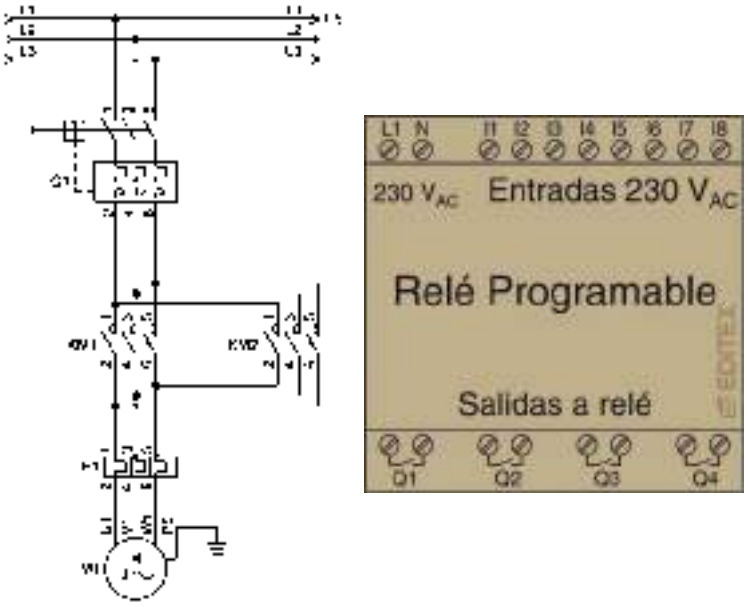


Figura 2.48. Esquema de fuerza para la inversión del sentido de giro de un motor trifásico

3. Observa el proceso industrial de la figura para el control de tres motores mediante pulsadores y dibuja el esquema de conexión de los sensores y los actuadores a un relé programable similar al de la actividad anterior.

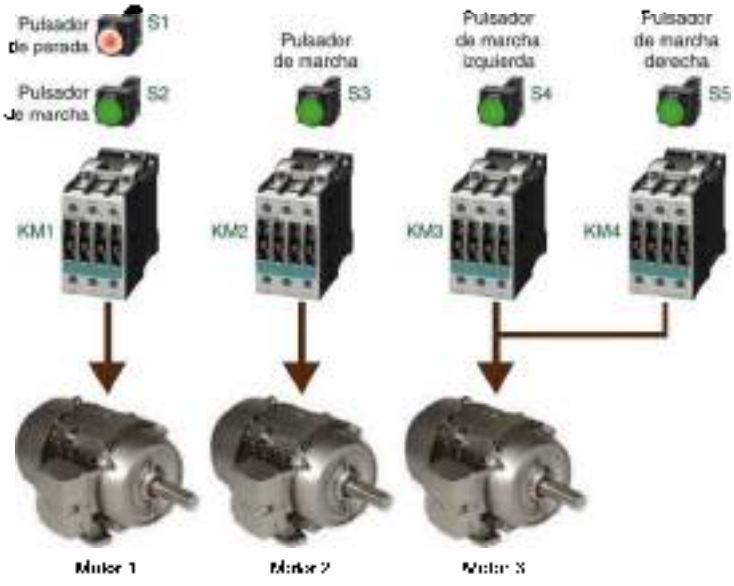


Figura 2.49. Proceso industrial para el control de tres motores mediante pulsadores.

ACTIVIDADES FINALES

continuación

4. Fíjate en el PLC de la figura y representa en tu cuaderno cómo conectarías en él los sensores y actuadores de la puerta automática de la Práctica Profesional Resuelta de esta unidad. El autómata dispone de nueve entradas digitales de 24 V_{DC} en lógica positiva y diez salidas digitales a relé, un borne común para las cinco primeras y otro para las cinco últimas.

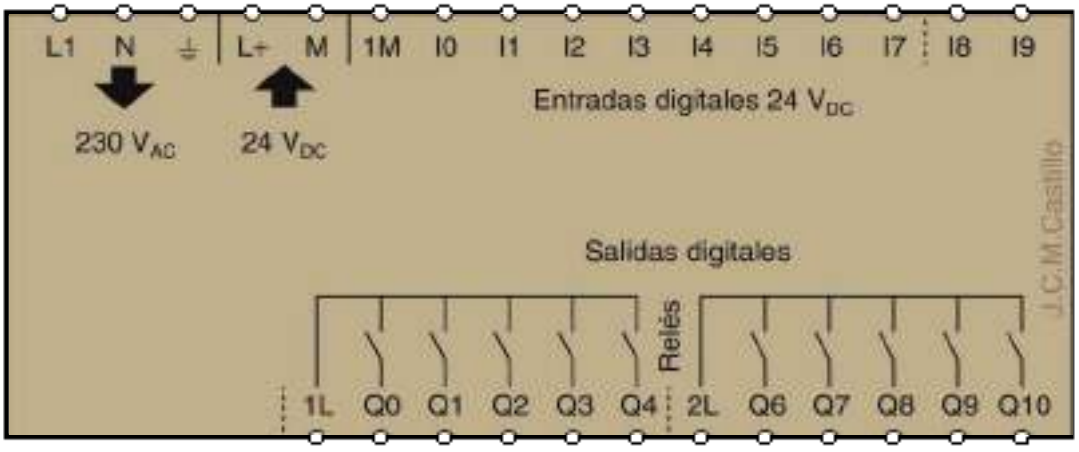


Figura 2.50. PLC con entradas DC y salidas a relé.

5. En el autómata de la actividad anterior, dibuja cómo conectarías dos detectores activos de tres hilos sabiendo que son NPN. Además, en el mismo esquema, conecta tres pulsadores normalmente abiertos para que sean compatibles con la polaridad de los detectores. ¿Sería posible conectar, además, un detector tipo PNP? Razona la respuesta.

6. En el autómata de la actividad número 4 se desea conectar dos contactores con bobinas a 250 V_{AC} y tres pilotos de señalización de 24 V_{AC}. ¿Sería posible? ¿Cómo? Represéntalo en un esquema.

7. Fíjate en el PLC de la siguiente figura y representa en tu cuaderno cómo conectarías los sensores y actuadores a sus respectivas entradas y salidas del automatismo del taladro propuesto en la Práctica Profesional Propuesta 2 de esta unidad. Debes elegir adecuadamente los sensores y actuadores, sabiendo que tanto las entradas como las salidas del autómata programable son tipo DC en lógica positiva.

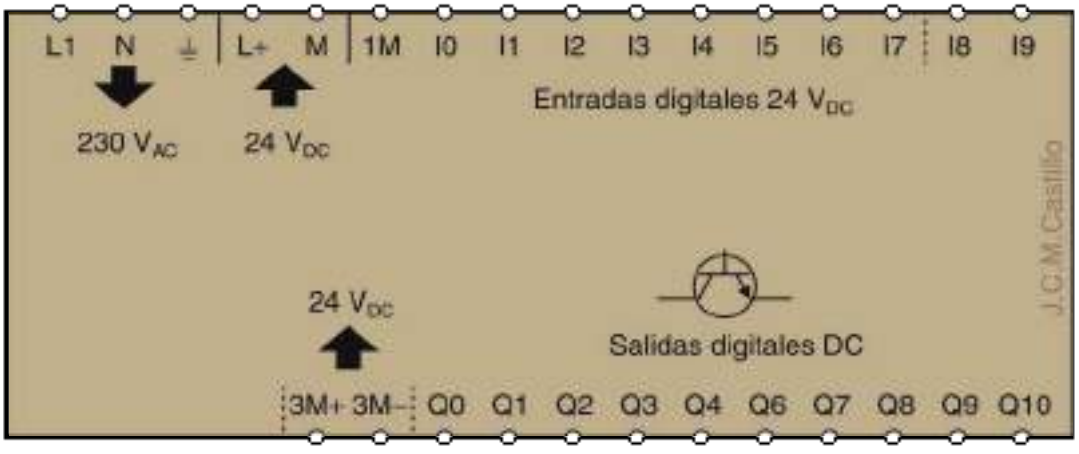


Figura 2.51. PLC con entradas y salidas DC.

8. ¿Cómo serían las conexiones del circuito de la actividad anterior sabiendo que el PLC, tanto para las entradas como para las salidas, funciona en lógica negativa?

9. Dibuja el esquema de conexión de sensores y actuadores del proceso de la figura sabiendo que la detección de las posiciones del taladro y del cargador de piezas se hace con detectores inductivos de tres hilos tipo PNP.

PRÁCTICA PROFESIONAL PROPUESTA 1

Herramientas

- Material de dibujo

Material

- Cuaderno de trabajo
- Hoja de características de un relé programable comercial con entradas a 230 V y salidas a relés

Conexión de sensores y actuadores a un relé programable para el control de un montacargas

Objetivo

- Identificar cada uno de los elementos que intervienen en el automatismo.
- Representar el cableado de los sensores y actuadores en un relé programable de un automatismo industrial.

Precauciones

- La subida y bajada de la cabina del montacargas se realiza mediante un motor gobernado por dos contactores. KM1 la sube y KM2 la baja.
- En cada planta hay una botonera doble que permite subir y bajar el montacargas.
- El relé programable es el mismo que el usado en la Práctica Profesional Resuelta de esta unidad. Es decir, entradas a 230 V_{AC} y salidas independientes a relés.
- No es necesario representar el circuito de fuerza para controlar el motor eléctrico.

Desarrollo

1. Fíjate en el automatismo de la figura y realiza una lista con los sensores y actuadores que se necesitan. Añade en ella su denominación, según el gráfico, y una breve descripción.

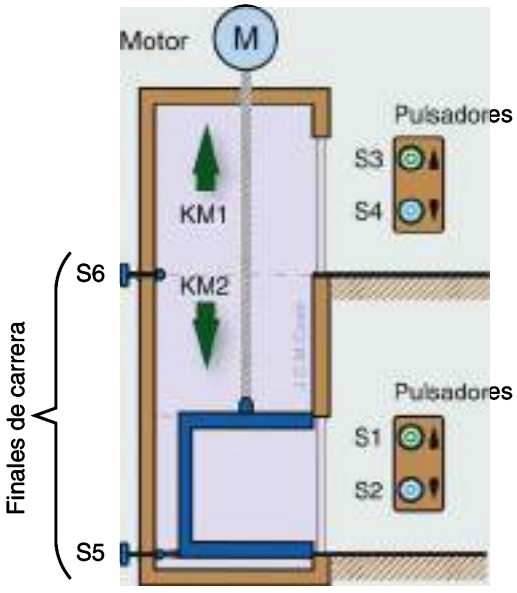


Figura 2.52. Automatismo de un montacargas.

2. Dibuja el esquema de conexiones de los actuadores y sensores al PLC sabiendo que se ha utilizado el mismo relé programable que en la Práctica Profesional Resuelta de esta unidad.

PRÁCTICA PROFESIONAL PROPUESTA 2

Herramientas

- Material de dibujo

Material

- Cuaderno de trabajo
- Hoja de características de un relé programable comercial con entradas a 24 VDC y salidas a relés

Conexión de sensores y actuadores a un PLC para el control de un taladro semiautomático

Objetivo

- Identificar cada uno de los elementos que intervienen en el automatismo
- Representar el cableado de los sensores y actuadores en un relé programable de un automatismo industrial.

Precauciones

- Si utilizas un relé programable comercial es posible que sean necesarios módulos de ampliación de E/S.
- Ya que las entradas son de 24 V_{DC}, es necesario disponer de una fuente de alimentación externa.
- No es necesario representar los circuitos de fuerza que se requieren para controlar este automatismo

Desarrollo

1. Fíjate en el automatismo de la figura y realiza una lista con los sensores y actuadores que se necesitan. Añade en ella su denominación, según el gráfico, y una breve descripción

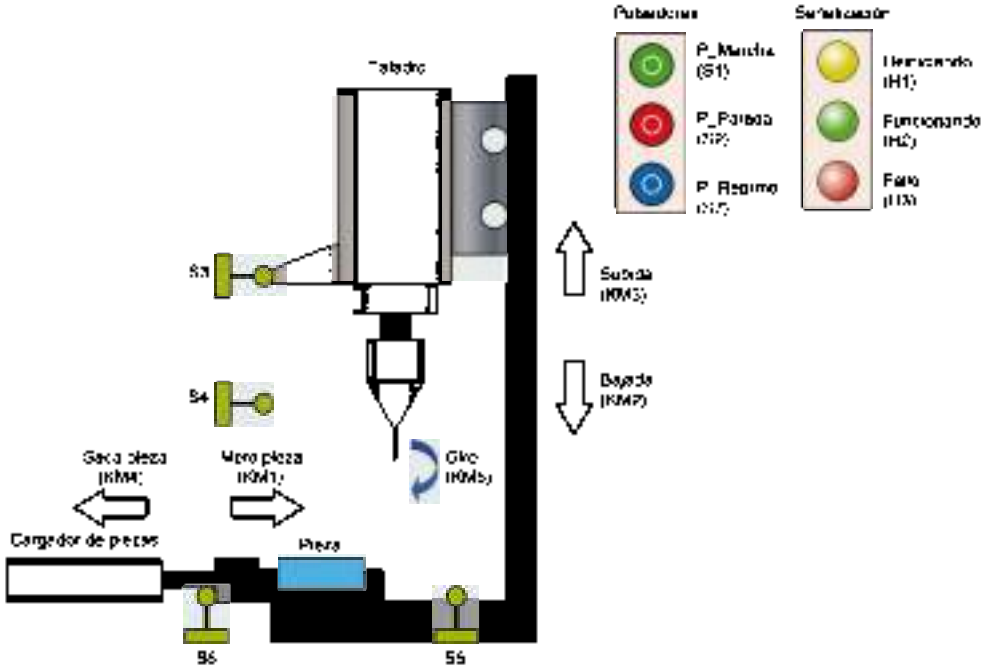
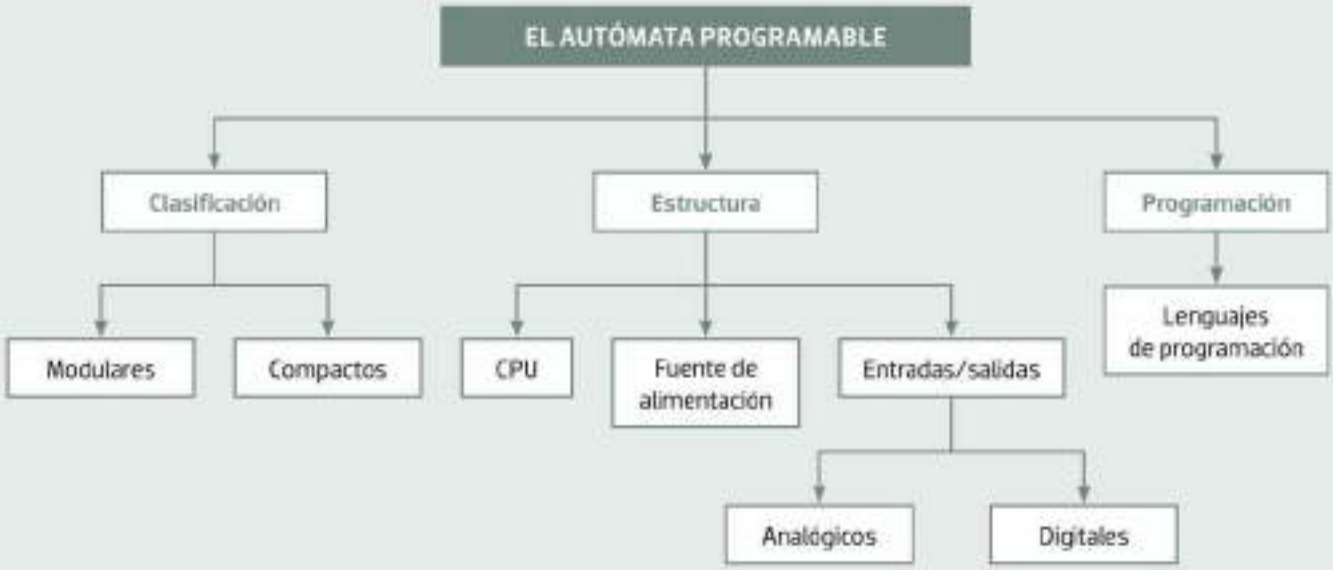


Figura 2.55. Automatismo de un taladro semiautomático.

2. Una vez contabilizadas todas las entradas y salidas necesarias, elige el relé programable adecuado y los módulos de expansión requeridos
3. Dibuja el esquema de conexiones de los actuadores y sensores al PLC sabiendo que el relé programable tiene entradas a 24 V_{DC} en lógica positiva y salidas individuales a relé

EN RESUMEN


EL AUTÓMATA PROGRAMABLE
Clasificación
Estructura
Programación
Modulares
Compactos
CPU
Fuente de alimentación
Entradas/salidas
Lenguajes de programación
Analógicos
Digitales

3 Programación de relés programables en FBD

Vamos a conocer...

1. Lenguaje de programación FBD para LOGO
2. Programación básica
3. Funciones de tiempo (temporizadores)
4. Función contador/descontador
5. Detección de flancos en señales digitales
6. Marcas internas

PRÁCTICA PROFESIONAL RESUELTA

Puesta en servicio de un relé programable

PRÁCTICAS PROFESIONALES PROPUESTAS

1. Programación de un montacargas
2. Programación de un ascensor de tres plantas

Y al finalizar esta unidad...

- Conocerás las funciones de programación del lenguaje FBD del relé programable LOGO.
- Identificarás los bloques especiales de este lenguaje de programación.
- Usarás las marcas para realizar operaciones internas.
- Sabrás cómo se detectan los flancos de las señales digitales en su aplicación práctica.
- Montarás un relé programable para comprobar los programas propuestos en esta unidad.
- Programarás diferentes procesos industriales reales.

1. Lenguaje de programación FBD para LOGO

El lenguaje FBD está basado en las funciones lógicas que se han estudiado en la primera unidad de este libro. A continuación se estudia su uso de forma práctica para resolver circuitos secuenciales básicos.

La programación se basará en el relé programable LOGO de la firma Siemens, ya que el FBD es su lenguaje natural.

El uso del relé programable LOGO será el medio para aplicar este lenguaje de programación ampliamente utilizado en la industria, pero el manejo del relé en profundidad no será el objetivo principal de esta unidad.

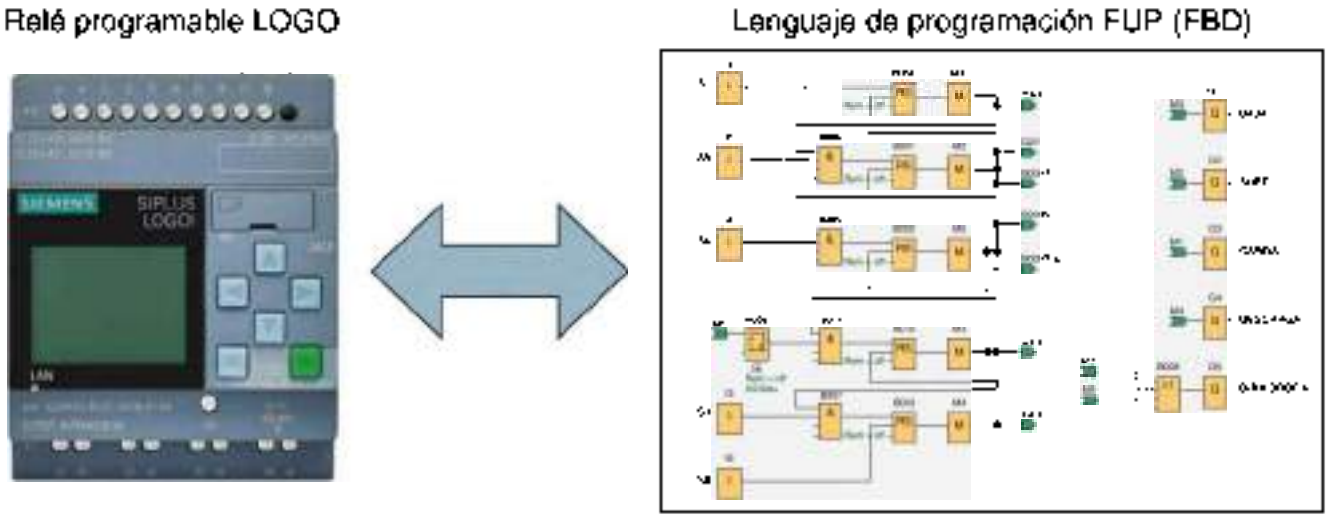


Figura 3.1. Relé programable LOGO de Siemens y su lenguaje de programación FUP

1.1. Lenguaje de programación FBD para LOGO

Siemens llama *FUP* al lenguaje FBD, así que desde este momento se utilizará esta denominación para hacer referencia a dicho lenguaje.

El lenguaje FUP, de la controladora LOGO, está basado en bloques de funciones que están organizados en tres categorías:

- Conectores y constantes (CO).
- Funciones generales (GF).
- Funciones especiales (SF).

1.2. Conectores y constantes (CO)

Permiten realizar la conexión entre el LOGO y su entorno, además de almacenar valores de señales y datos.

En esta categoría se encuentran, entre otras, las señales de entradas, salidas, marcas, bits de las teclas de cursor, variables de comunicación, etc. Aquí principalmente se utilizarán las de entradas, salidas y marcas digitales.

Figura 3.2. Símbolos de conectores básicos

Actividades

1. Monta el panel de pruebas mostrado en la Práctica Profesional Resuelta de esta unidad. Te servirá para comprobar los ejemplos y actividades aquí propuestas.

Los bloques de salidas y de marcas tienen dos terminales de conexión: uno por el lado izquierdo, para habilitar la señal, y otro por el lado derecho, para tomar el valor lógico de dicha señal, como puede ser para realizar un circuito con realimentación.

1.3. Funciones generales (GF)

Son las funciones básicas basadas en puertas lógicas (AND, OR, NOT, etc.). Con ellas es posible realizar las combinaciones requeridas por el proceso para activar salidas, marcas o entradas de disparo de bloques especiales como, por ejemplo, temporizadores y contadores.

Disponen de un número fijo de entradas y una salida.

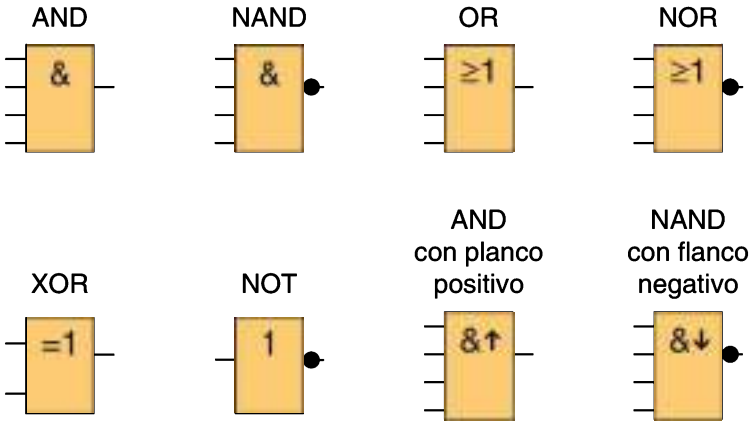


Figura 3.4. Funciones generales.

Además de las funciones generales utilizadas en la lógica digital, también hay funciones que detectan flancos, las cuales se describirán más adelante.

1.4. Funciones especiales (SF)

Son aquellas que tienen una funcionalidad muy específica, como puede ser temporizar, contar, comparar, memorizar el valor de una señal, etc.

El número de funciones especiales disponibles depende de la versión del relé programable. Aquí solamente se utilizarán algunas de ellas.

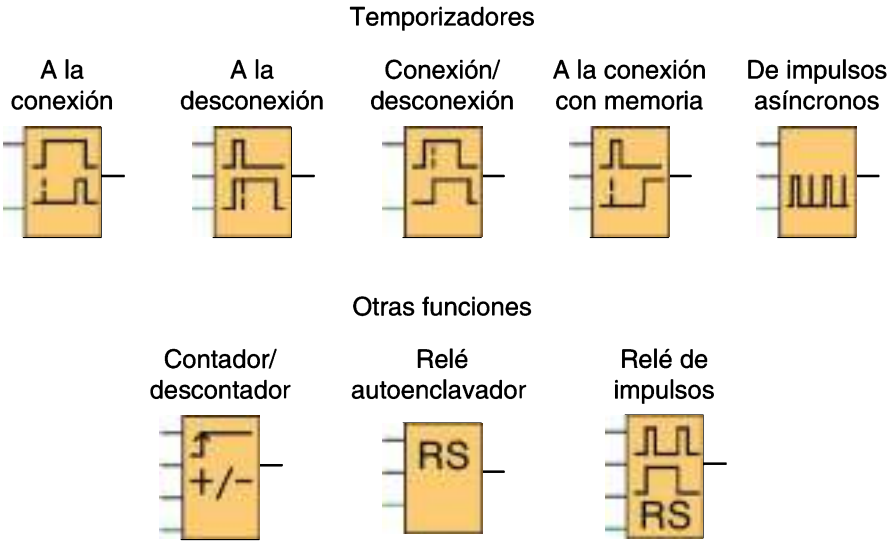


Figura 3.5. Funciones especiales.

Entradas de LOGO

El número máximo de entradas que tienen los bloques del LOGO es cuatro. Si se desea programar un bloque con más entradas se requiere conectar dos o más del mismo tipo en cascada.

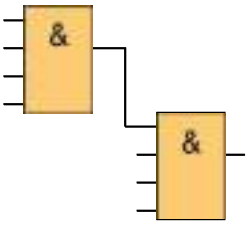


Figura 3.3. Conexión de bloques en cascada.

Remanencia

Muchos bloques de programación permiten activar la denominada *remanencia*. Esto posibilita guardar el dato de una señal de forma permanente en la memoria, de manera que si se produce una parada del programa, por ejemplo, por un corte de la red eléctrica, cuando se restaure de nuevo su ejecución el valor de la señal será el que tenían antes de producirse la incidencia.

2. Programación básica

Para la programación básica de circuitos, tanto con realimentación como con biestables, se pueden utilizar las reglas que se estudiaron en la unidad 1. A continuación se muestran algunos equipos de su uso en la programación FUP.

$$Q1 = (I1 + I2 + I3 + Q1) \cdot \overline{I4} \cdot \overline{I5}$$

Activación — Desactivación

Figura 3.6. Ecuación para circuito de realimentación.

2.1. Circuitos realimentados

Se deben aplicar las mismas reglas ya utilizadas en la primera unidad de este libro. Así, las señales de activación se suman a la realimentación y las de desactivación se multiplican, negándolas individualmente, al bloque de activación.

Ejemplo

En el siguiente circuito se muestra cómo una salida (Q1) se activa desde tres puntos (I1, I2, I3) y se desactiva (I4, I5) desde otros dos.

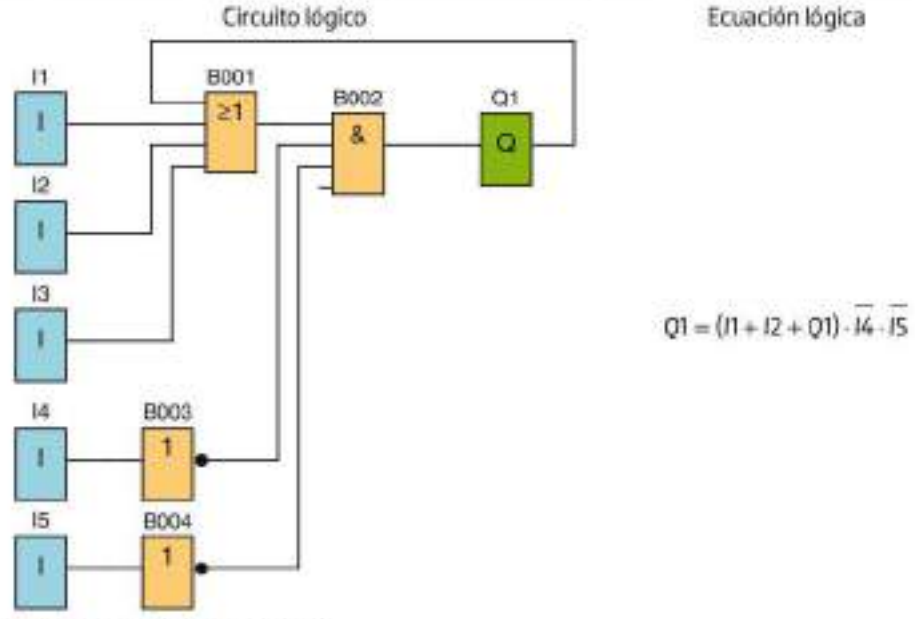


Figura 3.7. Circuito con realimentación.

De esta forma se pueden establecer condiciones de parada y puesta en marcha de una forma muy sencilla.

Actividades

2. Se desea que un programa con relé programable haga lo siguiente:

Salida	Activar con	Desactivar con
Q1	I1 o I2	I5 o I3
Q2	I3 o I4	I5 o I1
Q3	I5	I6
Q4	I1	I6

Escribe las ecuaciones lógicas, dibuja el circuito en el software de programación y simula su funcionamiento.

Recuerda

Según los teoremas de Morgan:

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

Las condiciones entre salidas pueden hacerse de diferentes maneras.

2.1.1. Condicionar la activación de una salida a otra

En esta situación se desea que la salida condicionada no se pueda activar hasta que la salida condicionante no esté activada:

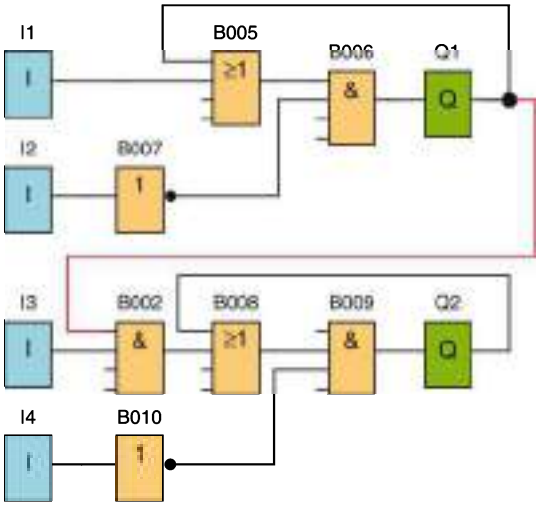
Caso 1

La salida que hay que condicionar solamente está afectada a su activación.

En el siguiente ejemplo se observa que Q2 solamente se podrá activar mediante I3 si previamente se ha activado Q1.

Importante

En la programación del LOGO, las condiciones de las salidas se deben tomar del terminal de la derecha de los bloques Q.



$$Q1 = (I1 + Q1) \cdot \overline{I2}$$

$$Q2 = (I3 \cdot Q1 + Q2) \cdot \overline{I4}$$

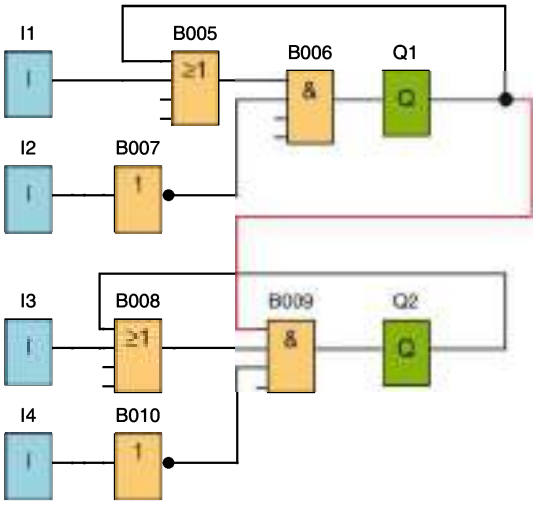
Figura 3.8. Condición para la activación.

Si ambas salidas están a «1» y se desactiva Q1, la salida Q2 se mantiene activa.

Caso 2

La salida que hay que condicionar está afectada tanto a su activación como a su desactivación.

En el ejemplo se muestra que Q1 condiciona la activación de Q2. Sin embargo, respecto al caso anterior, en esta situación, si ambas salidas están a «1» lógico y se desactiva Q1, también lo hace Q2.



$$Q1 = (I1 + Q1) \cdot \overline{I2}$$

$$Q2 = (I3 + Q2) \cdot \overline{I4} \cdot Q1$$

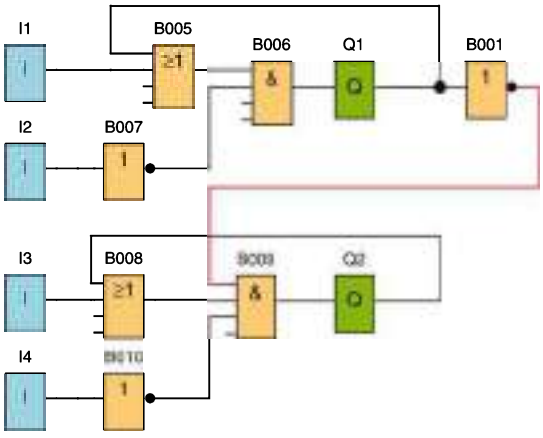
Figura 3.9. Condición para la activación/desactivación.

2.1.2. Condicionar la activación de una salida a la NO activación de otra

La salida condicionada no se puede activar si la salida condicionadora está previamente activada.

Caso 1

En este caso, la salida condicionante (Q1) se multiplica, negándola, a la ecuación de la realimentación.



$$Q1 = (I1 + Q1) \cdot \overline{I2}$$

$$Q2 = (I3 + Q2) \cdot \overline{I4} \cdot \overline{Q1}$$

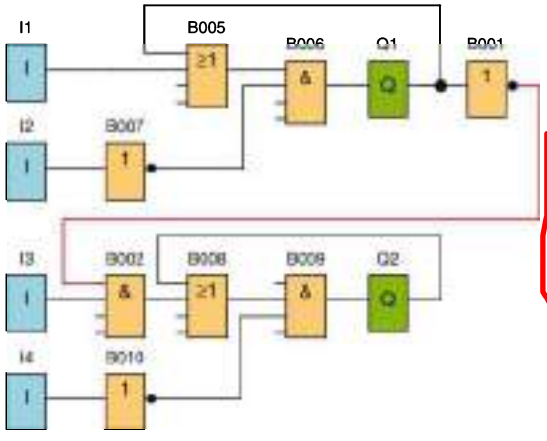
Figura 3.10. Condición para la activación.

Q2 no se activará si lo está Q1. Sin embargo, si Q2 está activada y lo hace Q1, esta desactivará la anterior.

En esta situación nunca podrán estar las dos salidas funcionando al mismo tiempo, ya que Q1 desactiva siempre Q2.

Caso 2

La variable de la salida condicionante (Q1) se multiplica, negándola, con la variable de entrada que activa la salida condicionada (Q2), que se encuentra en el bloque de realimentación.



$$Q1 = (I1 + Q1) \cdot \overline{I2}$$

$$Q2 = (I3 \cdot \overline{Q1} + Q2) \cdot \overline{I4}$$

Figura 3.11. Condición para la activación/desactivación.

De igual forma que en el caso anterior, en esta situación Q2 no se activará si lo está previamente Q1. Sin embargo, la diferencia en esta combinación lógica radica en que una vez se ha avivado Q2, aunque lo haga posteriormente Q1, Q2 ya no parará.

En este caso las dos salidas podrían estar activadas al mismo tiempo, ya que Q1 solamente condiciona la puesta en marcha de Q2, pero no la parada.

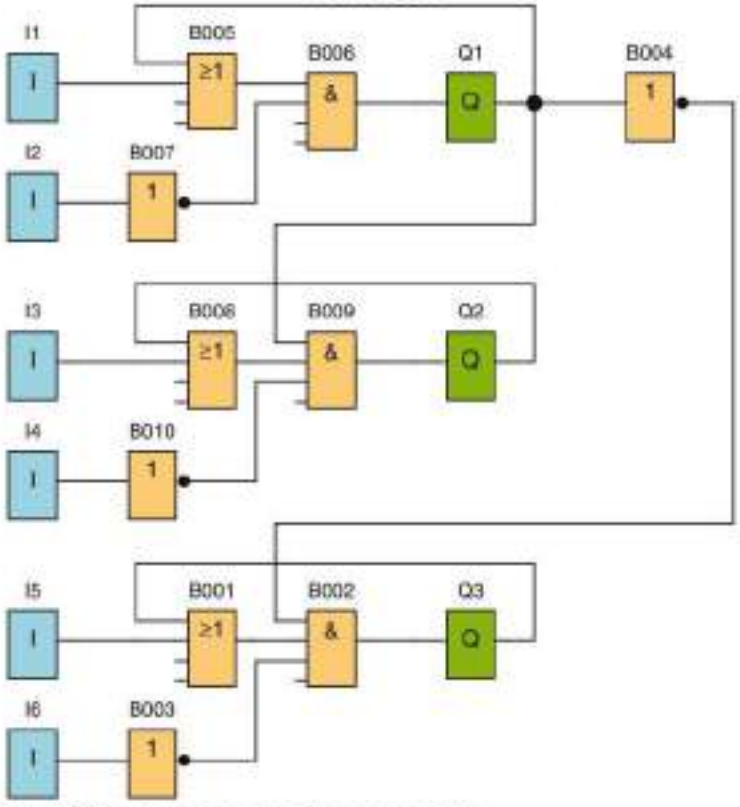
Vocabulary

Entrada: *input*.
Salida: *output*.
Marca: *mark*.
Configurar: *setup*.
Fuente de alimentación: *power supply*.
Conmutación: *switching*.
Inicio: *run*.
Parada: *stop*.
Borrar: *delete*.
Insertar: *insert*.
Tiempo: *time*.
Retardo a la conexión: *on-delay*.
Retardo a la desconexión: *off-delay*.
Con memoria: *retentive*.
Flanco: *edge (flank)*.
Dispositivo: *device*.
Temporizadores: *timmers*.
Contadores: *counters*.
Contador: *count up*.
Descontador: *count down*.
Biestable: *latching relay*.
Cableado: *wiring*.

Ejemplo

El siguiente ejemplo muestra un circuito con realimentación en el que la salida Q2 solamente se activará si previamente lo está Q1, y la salida Q3 no lo hará si Q1 está activada. En ambos casos la desconexión de las salidas condicionadas depende de las salidas condicionadoras. Así, Q2 se desactivará si lo hace Q1 y Q3 se desactivará si se pone a «1» lógico Q1.

Circuito lógico



Ecuación lógica

$$Q1 = (I1 + Q1) \cdot \overline{I2}$$
$$Q2 = (I3 + Q2) \cdot \overline{I4} \cdot \overline{Q1}$$
$$Q3 = (I5 + Q3) \cdot \overline{I6} \cdot \overline{Q1}$$

Figura 3.12. Circuito lógico con condiciones entre salidas.

Actividades

3. Se desea que un programa con relé programable haga lo siguiente:

Salida	Activar con	Desactivar con	Condición para activar salida
Q1	I1	I4	Q2 no esté activada
Q2	I2	I4	Q1 no esté activada
Q3	I3	I4	Estén activadas Q1 o Q2
Q4	I5	I6	Q3 no esté activada

Condiciones de funcionamiento entre salidas:

- Q1 se activa si Q2 no está activada previamente.
- Q2 se activa si Q1 no está activada previamente.
- Si se activa Q1 se debe desactivar Q2 y viceversa.
- Q3 se activará si previamente están Q1 o Q2 activadas. Esta condición solamente afecta a su activación.
- Todas las salidas se desactivan, o no se pueden activar, si se pone a «1» lógico Q4.

Escribe las ecuaciones lógicas, dibuja el circuito en el software de programación y simula su funcionamiento.

2.2. Circuitos con biestables

El uso de biestables simplifica las tareas de programación, ya que para realizar la misma función solamente se utiliza un bloque en lugar de tres, como ocurre en el circuito de realimentación.

En el lenguaje de programación FUP del relé programable LOGO de Siemens, el biestable recibe el nombre de *relé autoenclavador* y dispone de dos entradas, una para el SET y la otra para el RESET.

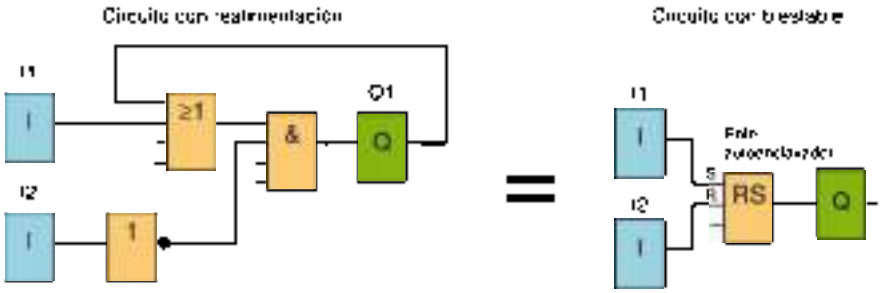


Figura 3.13. Circuito con realimentación y su equivalente con biestable

Para la obtención de circuitos con biestables se pueden utilizar las reglas estudiadas en la primera unidad. A continuación se muestra un ejemplo de aplicación práctica.

Ejemplo

Varias salidas digitales se activan y desactivan mediante entradas digitales, según las condiciones indicadas en la siguiente tabla.

Salida	Activar con	Desactivar con	Condición para activar salida
Q1	I1	I4	Si Q2 no está activada
Q2	I2 o I3	I4	Si Q1 no está activada
Q3	I5	I4	Si están activadas Q1 o Q2 previamente
Q4	I1	I4 o I6	Si Q3 no está activada

Cada salida se asocia a un relé autoenclavador (biestable); por tanto, se deben obtener dos ecuaciones lógicas, una para la entrada SET y otra para la entrada RESET.

Q1	Q2
$S = I1 \cdot \overline{Q2}$ $R = I4$	$S = (I2 + I3) \cdot \overline{Q1}$ $R = I4$
Q3	**Q4**
$S = I5 \cdot (Q1 + Q2)$ $R = I4$	$S = I1 \cdot \overline{Q3}$ $R = I4 + I6$

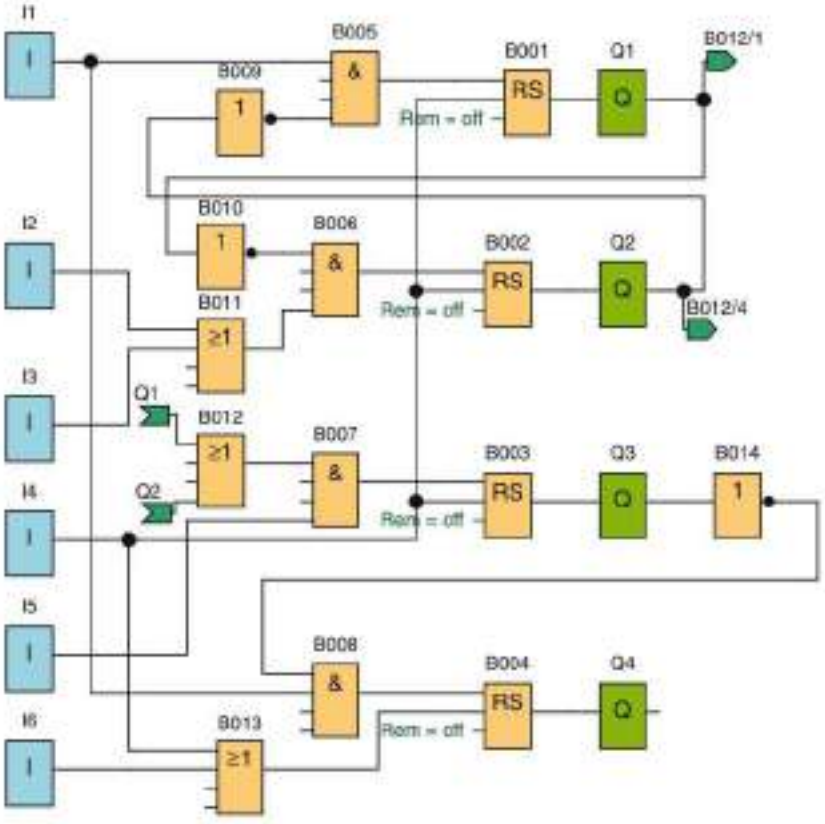


Figura 3.14. Circuito lógico con relés autoenclavadores.

3. Funciones de tiempo (temporizadores)

Son bloques de funciones especiales (SF) que permiten generar eventos en función de un tiempo prefijado.

El relé programable LOGO dispone de varios tipos de temporizadores, de los cuales aquí solamente se utilizan cinco de ellos.

Por su similitud con la lógica cableada, se insistirá especialmente en los denominados *a la conexión* y *a la desconexión*.

Los bloques de temporizadores tienen una o más entradas (por ejemplo: *Trg*, RESET...) para su control y una salida que conmuta en función del tiempo preseleccionado y el modo de funcionamiento del temporizador.

Todos los temporizadores disponen del parámetro de tiempo, que se presenta de forma similar a una entrada pero en color verde.

3.1. Utilización de las señales de tiempo en las ecuaciones lógicas

Entrada Trg

La mayoría de los temporizadores del LOGO disponen de una entrada *Trg* (*triger*) que se utiliza para efectuar su disparo.

La entrada de disparo del temporizador es el equivalente a la bobina en los sistemas cableados y la salida al contacto.

En un temporizador, a la entrada *Trg* se le asigna una combinación lógica para su activación (T=). La salida de utilización del temporizador se usa como operando en las ecuaciones lógicas de otros bloques de asignación.

En el siguiente ejemplo, el temporizador se activa con la combinación OR de las entradas I1 o I2 (T1 = I1 + I2). La salida Q1 conmuta en función del tiempo programado en el temporizador, siempre que, en este caso, el valor lógico de la entrada I3 sea «1» (Q1 = T1 · I3).

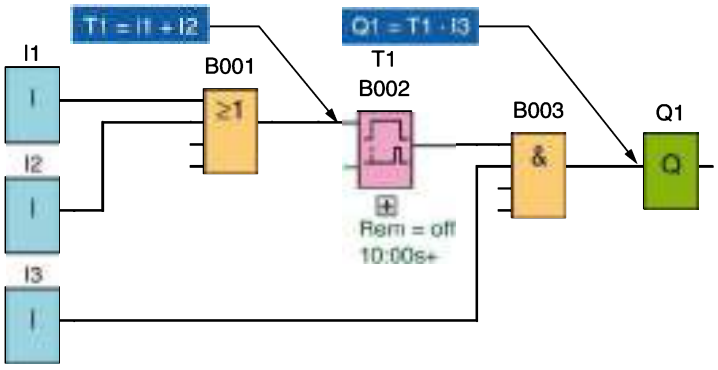


Figura 3.15. Ecuaciones lógicas en un circuito con temporizador.

3.2. Tipos de temporizadores

A continuación se describe el funcionamiento de los diferentes tipos de temporizadores que se pueden utilizar en el LOGO. Aquí se utilizarán cinco de ellos, los denominados *con retardo a la conexión, con retardo a la desconexión, con retardo a la conexión con memoria, generador de impulsos asíncronos* (intermitente) y *con retardo a la conexión/desconexión*.

3.2.1. Temporizador con retardo a la conexión (TON)

Temporizador a la conexión

El temporizador a la conexión es el que más se utilizará en las actividades prácticas propuestas a lo largo de todo el libro.

Activa su salida una vez transcurrido el tiempo de preselección. Dispone de una entrada Trg que dispara el temporizador cuando se pone a «1» lógico. Si la alimentación de dicha entrada pasa a 0 antes de que transcurra el tiempo preseleccionado, el temporizador se resetea y comienza desde 0 en la próxima activación.

La salida se activa una vez transcurrido el tiempo preprogramado y se mantiene en esta situación siempre que *Trg* esté a «1» lógico.

El cronograma de funcionamiento de este temporizador es el siguiente.

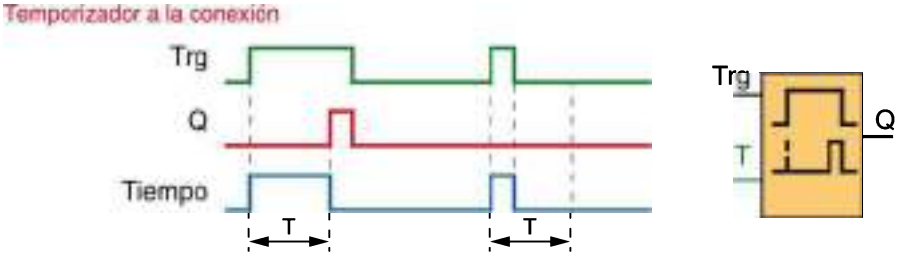
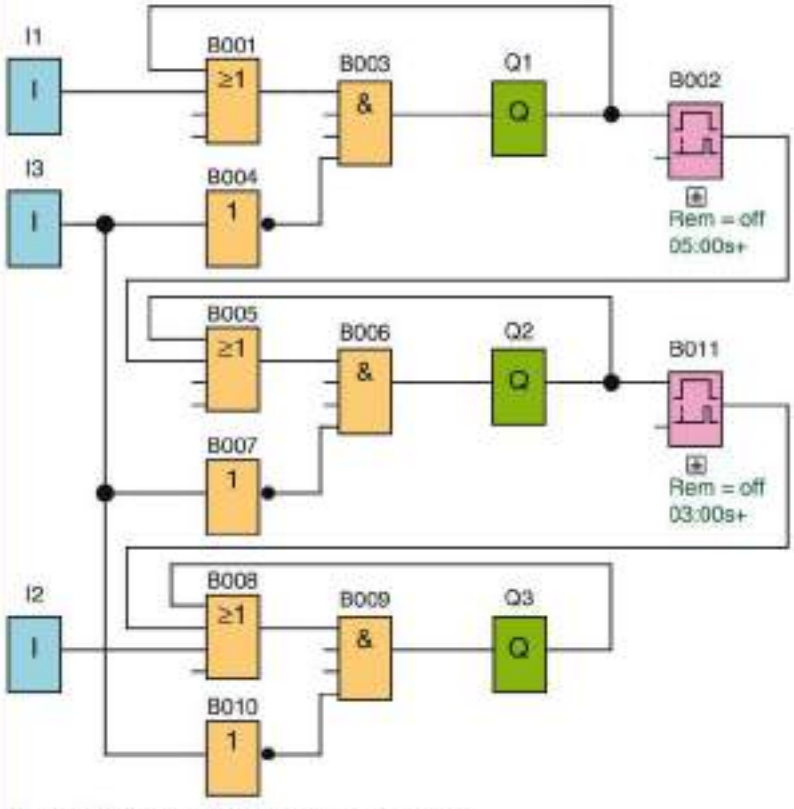


Figura 3.16. Cronograma del temporizador a la conexión.

Ejemplo

Funcionamiento:

- El circuito consta de tres salidas, Q1, Q2 y Q3, que se controlan mediante circuitos con realimentación.
- La Q1 se pone en marcha con el pulsador I1.
- La Q2 se pone en marcha a los 5 segundos de haberse activado Q1.
- La Q3 se activa a los 3 segundos de hacerlo Q2 o de forma manual mediante I2.
- Todas las salidas se desactivan mediante el pulsador I3.



Ecuaciones lógicas:

- $Q1 = (I1 + Q1) \cdot \overline{I3}$
- $Q2 = (T1 + Q2) \cdot \overline{I3}$
- $Q3 = (T2 + I2 + Q3) \cdot \overline{I3}$
- $T1 = Q1\,(5\ s)$
- $T2 = Q2\,(3\ s)$

Figura 3.17. Ejemplo con temporizadores a la conexión

3.2.2. Temporizador con retardo a la desconexión (TOF)

Este tipo de temporizador activa su salida cuando se alimenta la entrada *Trg*. Si dicha entrada se mantiene a «1» lógico, la salida también lo hace de forma permanente. La temporización comienza cuando el valor en la entrada *Trg* pasa de «1» a «0» y desactiva la salida una vez transcurrido el tiempo preseleccionado.

Este temporizador dispone de una entrada RESET para inicializar la temporización. Si esta entrada se mantiene a «1», la salida del temporizador se conmuta a 0

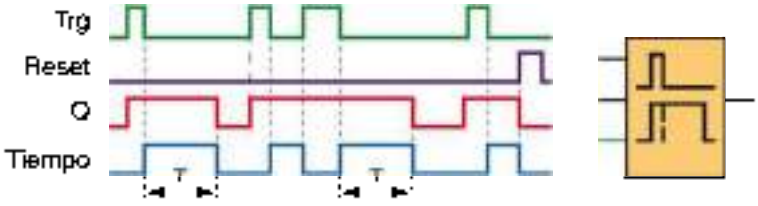


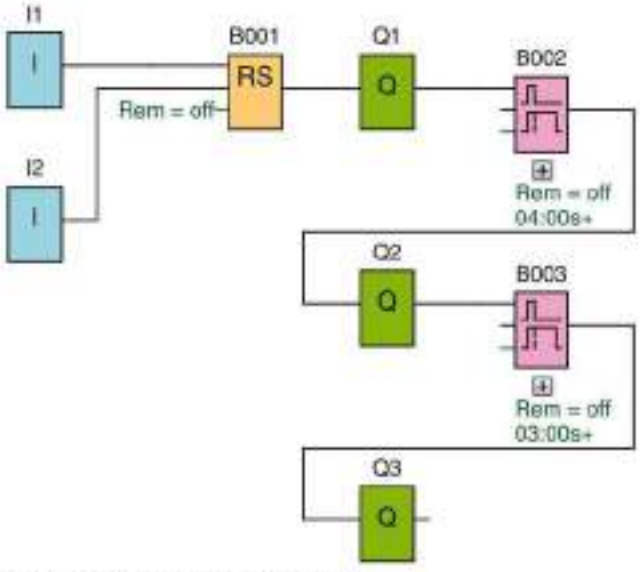
Figura 3.18. Cronograma del temporizador a la desconexión

Ejemplo

Funcionamiento:

En un circuito con tres salidas y dos entradas:

- La salida Q1 se activa mediante I1 y se desactiva mediante I2.
- Las salidas Q2 y Q3 se controlan a través dos temporizadores a la desconexión.
- Cuando se activa Q1, las otras dos salidas también lo hacen.
- Cuando la salida Q1 se desactiva mediante I2, la salida Q2 debe hacerlo a los 4 segundos de la primera y la salida Q3 a los 3 segundos de la segunda.



Ecuaciones lógicas:

- Q1
  - S = I1
  - R = I2
- Q2 = T1
- Q3 = T2
- $T1_{(desc)} = Q1_{(5\ s)}$
- $T2_{(desc)} = Q2_{(3\ s)}$

Figura 3.19. Ejemplo con temporizadores a la conexión.

3.2.3. Temporizador con retardo a la desconexión con memoria

El funcionamiento es idéntico a. de retardo a la conexión, con la diferencia de que el disparo de la temporización se hace simplemente con un pulso en la entrada *Trg*, no siendo necesario mantenerla a «1». La entrada RESET permite desactivar la temporización una vez ha sido lanzada.

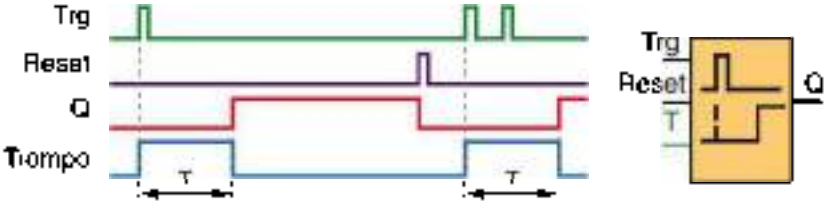


Figura 3.20. Cronograma del temporizador a la desconexión con memoria

Se puede decir que este tipo de temporizador es el circuito equivalente a poner un biestable a la entrada *Trg* de un temporizador a la conexión.

3.2.4. Generador de impulsos asíncronos

Activa su salida intermitentemente mediante un tren de pulsos. En este temporizador se debe ajustar el tiempo que la señal de salida debe estar a 1 (T_H) y el tiempo que debe estar a 0 (T_L).

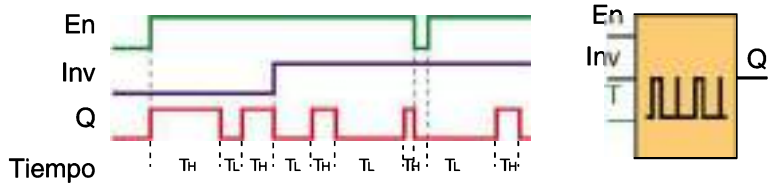


Figura 3.21. Cronograma del generador de impulsos asíncrono.

Dispone de una entrada inversora (Inv), que invierte el tiempo T_H por el T_L.

3.2.5. Temporizador con retardo a la conexión/desconexión

El temporizador con retardo a la conexión/desconexión es lo mismo que conectar en cascada un temporizador con retardo a la conexión con otro a la desconexión. De esta forma se consiguen dos acciones sobre la salida, una de activación después de un tiempo de habilitar la entrada de disparo y otra de desactivación después de un tiempo de deshabilitar dicha entrada.

Señales intermitentes

El generador de impulsos asíncronos se puede utilizar para generar señales intermitentes, como puede ser en lámparas de señalización.

Actividades

4. Uso del generador de impulsos asíncrono.

Programa y comprueba el funcionamiento del siguiente circuito lógico en el que se han utilizado dos generadores de impulsos asíncronos. Cuando se activa la entrada I1, la salida Q1 parpadea de forma rápida (0,3 s). Cuando lo hace la entrada I2, el parpadeo es lento (1 s). Si las dos entradas se activan a la vez, la salida permanece desactivada.

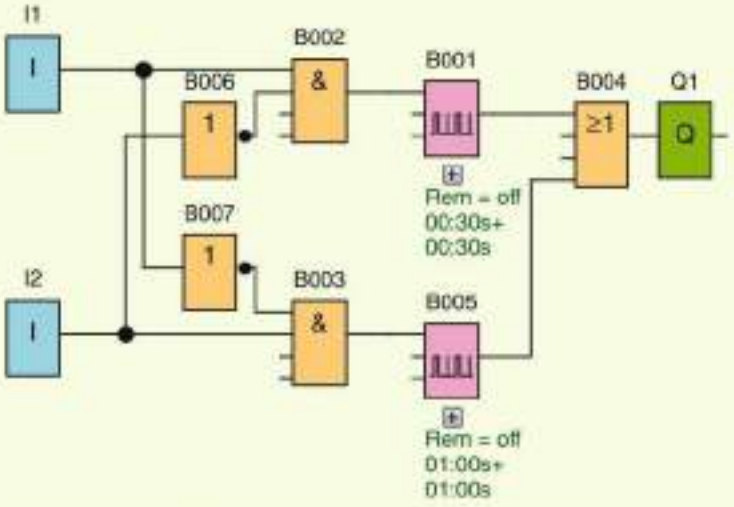


Figura 3.22. Ejemplo de uso de los generadores de impulsos asíncronos.

Usos del temporizador a la conexión/desconexión

Utilizando dos salidas (Q1 y Q2) y dos entradas (I1 e I2), realiza y comprueba el programa para el siguiente funcionamiento: la salida Q1 se activa después de 5 segundos de mantener a «1» la entrada I5 y se desactiva a los 10 segundos de poner a «0» la misma entrada. La salida Q2 se activa a los 3 segundos de mantener a «1» la entrada I2 y se desactiva a los 15 segundos de pasar I2 a valor «0».

4. Función contador/descontador

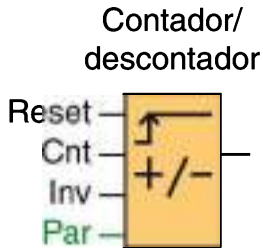


Figura 3.23. Contador/descontador.

La función «contador» permite registrar eventos que se producen en sus entradas. Estos se almacenan de número, de forma que, cuando se haya alcanzado el valor preajustado por el usuario, la salida se activa.

El contador del relé programable LOGO permite contar y descontar.

Su control se hace mediante tres entradas y un parámetro:

- **RESET:** pone el contador a «0» y desactiva su salida.
- **CNT:** incrementa (o reduce) el valor del contador cada vez que recibe un pulso.
- **DIR:** si se encuentra a «0», los pulsos en la entrada CNT incrementan el valor del contador. Si se encuentra a «1», los pulsos en CNT decrementan dicho valor.
- **PAR:** es el valor preseleccionado como límite para que se active la salida del contador.

A continuación se muestra un ejemplo de uso del contador. En este caso, cada vez que la entrada I1 recibe un pulso, el cómputo del contador se incrementa. Sin embargo, cuando el pulso lo recibe la entrada I2, el valor se decrementa. Así, la salida se activa cuando el valor del contador es igual o superior al valor preseleccionado en PAR.

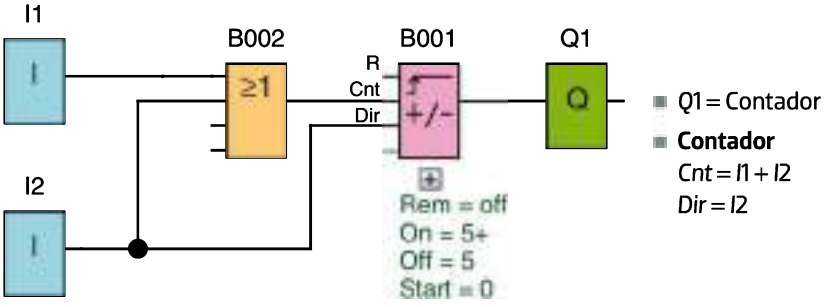


Figura 3.24. Contador/descontador en LOGO.

Un contador debe tener tantas ecuaciones lógicas como entradas del mismo se utilicen.

Actividades

5. Utilizando la función contador/descontador, realiza el programa para el control de las plazas de un aparcamiento de vehículos sabiendo que:

- El número de plazas es 20.
- Un semáforo señaliza si el aparcamiento tiene plazas libres (lámpara verde) o está completo (lámpara roja).
- En las puertas de entrada y de salida del aparcamiento hay instaladas dos barreras fotoeléctricas, una para controlar los vehículos que entran y otra para controlar los que salen. Cada una de ellas se encuentra conectada a una entrada diferente del relé programable.

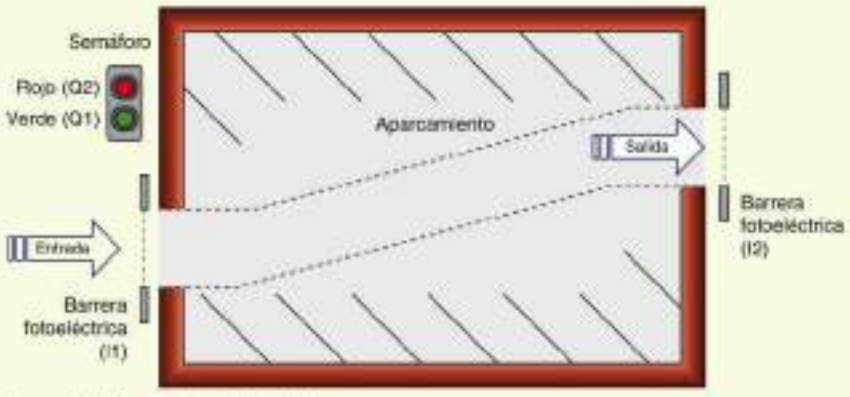


Figura 3.25. Aparcamiento de vehículos.

5. Detección de flancos en señales digitales

Las funciones de flanco permiten detectar cuándo una variable digital cambia de valor. Cuando esto ocurre se genera un impulso de una duración de tiempo muy breve, que puede ser utilizada en el programa de usuario.

Los flancos pueden ser de dos tipos: positivos o negativos.

En el lenguaje FUP del LOGO de Siemens los flancos están asociados a bloques lógicos AND, para los ascendentes, y NAND, para los descendentes. En ambos casos su salida está flanqueada con el signo que le corresponde y se representa con una flecha.

Figura 3.26. Flancos de una señal digital.

5.1. Flanco positivo o ascendente

Detecta cuándo la señal pasa de «0» a «1». En este caso, la señal en la salida de flanco se mantiene a «1» solamente un instante después de que la señal de entrada haya cambiado de valor.

5.2. Flanco negativo o descendente

Detecta cuándo la señal pasa de «1» a «0». En este caso, la señal en la salida de flanco se mantiene a «1» solamente un instante después de que la señal de entrada haya cambiado de valor.

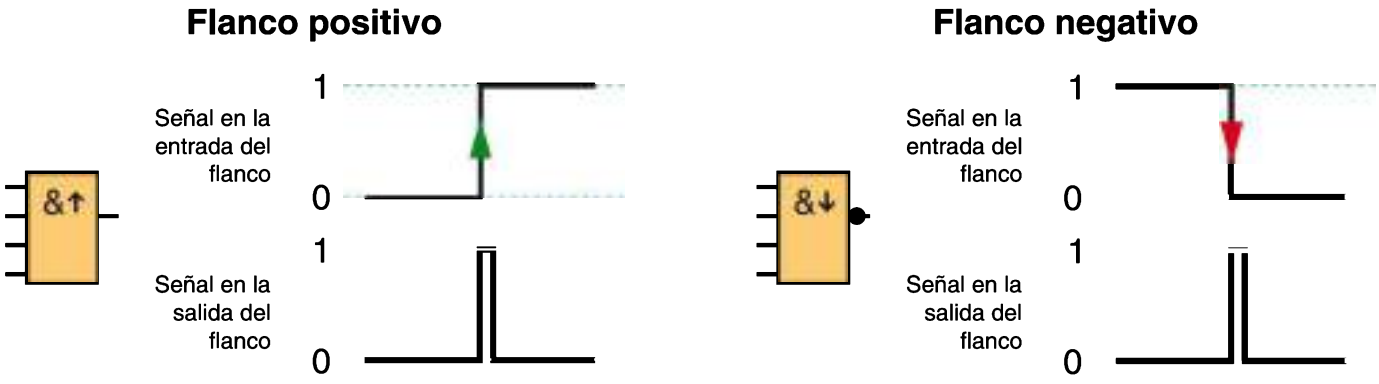


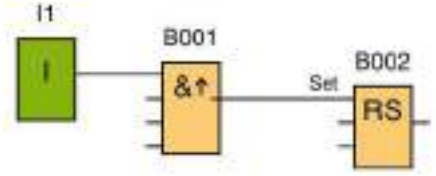
Figura 3.27. Señales en flancos.

En las ecuaciones lógicas los flancos se representan junto a la variable afectada con flechas verticales. La que apunta hacia arriba representa el flanco positivo y la que apunta hacia abajo el flanco negativo.

Ejemplo

Flanco positivo

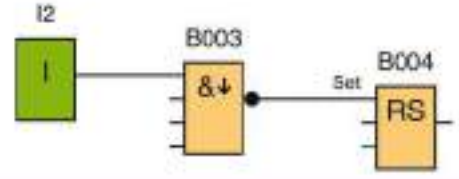
El SET del biestable recibe «1» de la señal de entrada I1 cuando esta pasa de «0» a «1».



$S = \uparrow I_1$

Flanco negativo

El SET del biestable recibe «1» de la señal de entrada I2 cuando esta pasa de «0» a «1».



$S = \downarrow I_2$

Ejemplo

Uso del flanco negativo

Recuerda la actividad que has realizado anteriormente relacionada con el aparcamiento de vehículos. Imagina que en ambas puertas se han instalado dos barreras, que están bajadas para evitar el paso de vehículos y que solamente suben cuando corresponde.

Además, en cada una de ellas, se han instalado detectores fotoeléctricos. Los cuales permiten conocer si el coche está debajo de la puerta de barrera o si ya la ha cruzado.

Si las señales de los detectores fotoeléctricos no tienen flanco, justo en el instante en que sea detectada la parte delantera del coche, el programa iniciará la temporización para cerrar la puerta.

Si el vehículo se detiene de forma no deseada, la barrera se puede cerrar sin que la haya cruzado por completo, golpeándolo y causando desperfectos con las correspondientes consecuencias.

El programa del PLC debe saber cuándo el vehículo ha abandonado por completo la zona de la barrera fotoeléctrica y, justo en ese instante, ejecutar la secuencia para cerrar la puerta de barrera.

Para ello, en este caso, se debe utilizar un flanco negativo asociado a la señal de entrada del detector fotoeléctrico.

De esta forma, hasta que dicha señal no detecte el flanco de bajada (de 1 a 0), el programa de usuario no cerrará la puerta, ya que ese es el momento exacto en el que el vehículo ha abandonado la zona crítica.

En las siguientes figuras, podemos observar el funcionamiento de la barrera con detección sin flanco y detección con flanco negativo.

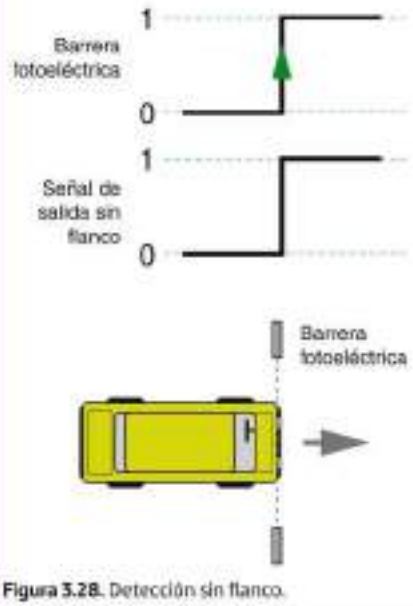


Figura 3.28. Detección sin flanco.

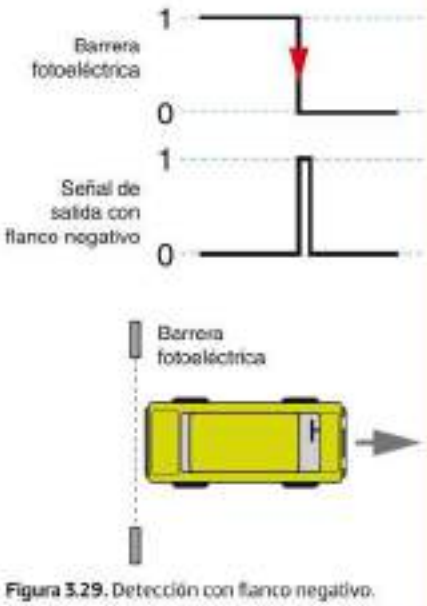


Figura 3.29. Detección con flanco negativo.

Actividades

6. Utiliza el programa que realizaste para el control de los vehículos que entran y salen del aparcamiento de coches y añádele las señales de flanco correspondiente a las barreras fotoeléctricas de la entrada y la salida.

5.3. Uso de los flancos para evitar señales permanentes

Las señales por nivel mantienen su valor lógico de forma permanente. Esta situación puede generar conflictos en determinados procesos automatizados. En esos casos es necesario evitar que se produzcan mediante la activación por flancos.

El uso de flancos, tanto positivos como negativos, permite detectar cuándo se activa o se desactiva una señal independientemente de cómo quede accionado el sensor que genera la acción.

Ejemplo

Obsérvese el funcionamiento del siguiente proceso industrial perteneciente a un taladro automático con cargador de piezas.

Al accionar el pulsador de marcha (S1), el cargador se desplaza con Q1 hasta colocar la pieza debajo del taladro. En esta situación, el taladro baja hasta accionar el final de carrera S4, que coincide cuando la pieza se ha taladrado por completo. Una vez realizado esto, el taladro sube hasta S3 y descarga la pieza con Q4, hasta que toca S6.

Este proceso tiene dos señales permanentes, las procedentes de S3 y S5. Ambos finales de carrera se quedan accionados en determinados estados de la secuencia, de forma que, manteniendo su valor a «1» lógico, provocan un funcionamiento no deseado del proceso. Ambas señales deben procesarse con detectores de flanco (en este caso de tipo positivo), de forma que su valor solamente sea detectado cuando se alcance el final de carrera, ignorando si está permanentemente presionado.

Q1	Q2	Q3	Q4
S = I1 · I6	S = I5 ↑	S = I4	S = I3 ↑
R = I5 ↑	R = I4	R = I3 ↑	R = I6

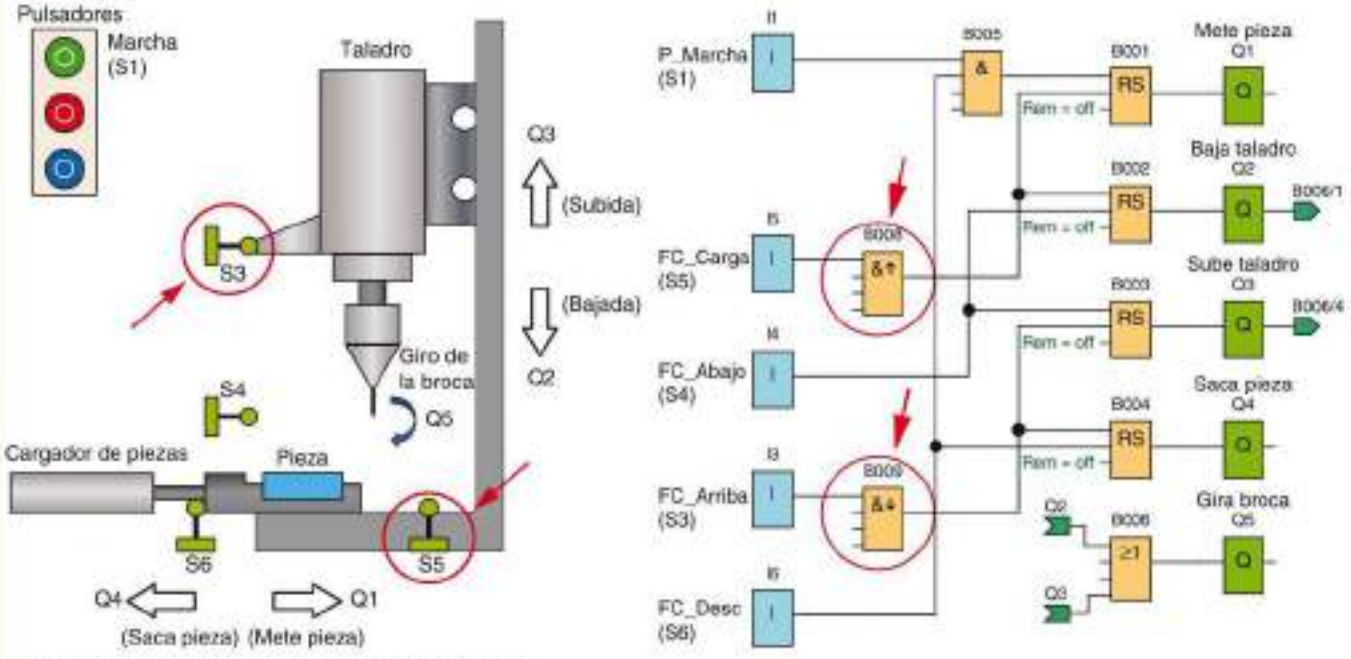


Figura 3.30. Taladro semiautomático con cargador de piezas.

¡Importante! La solución del proceso se ha simplificado al máximo para entender el uso de los flancos y evitar señales permanentes. No obstante, se debe considerar que no es definitiva. Si la subida y bajada del taladro se hace con un motor eléctrico será necesario prever que la inversión de giro no se hace de forma instantánea, ya que provocaría un cortocircuito a través del circuito de fuerza.

6. Marcas internas

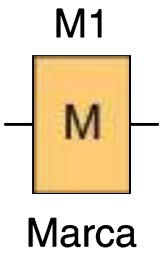


Figura 3.31. Bloque de marca en el LOGO.

Las marcas o bits auxiliares son utilizadas para operaciones internas en la memoria del autómata y tienen, en lo básico, un comportamiento similar a los relés auxiliares en la lógica cableada.

Se utilizan de igual forma que las salidas, con la diferencia de que no existe una conexión física (borne de conexión) para sacar su resultado a exterior del PLC.

En el lenguaje FUP las marcas se identifican con la letra M seguida de un número (M1, M2, M3, etc.).

En el siguiente ejemplo la salida Q1 se activa con la entrada I3, pero solo si previamente se han activado las marcas M1 y M2, con sus respectivas entradas (I1-I2). La desactivación, tanto de la salida como de las marcas, se hace con I4.

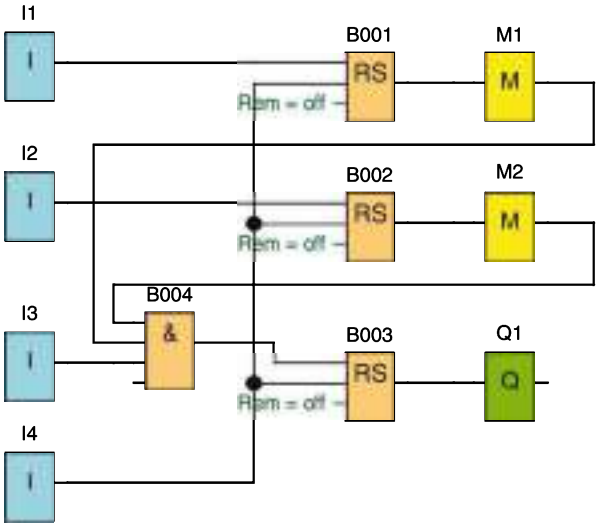


Figura 3.32. Ejemplo de uso de las marcas.

Importante

La marca especial M8 no debe disponer de ninguna señal conectada a su entrada.

El número de marcas disponibles dependerá del modelo de autómata o relé programable. En ocasiones, como ocurre en el LOGO de Siemens, algunas marcas o bits de sistema tienen asignadas funciones especiales. Por ejemplo, en este dispositivo la marca M8 aplica un flanco positivo a su salida cuando el relé programable pasa de STOP a RUN. Esta es especialmente útil para realizar acciones después de un arranque en caliente del sistema.

En el siguiente ejemplo se muestra cómo la salida Q1 se acciona de forma intermitente (mediante un generador de impulsos asíncrono) siempre que se pasa el autómata de STOP a RUN. La entrada I1 permite su desactivación.

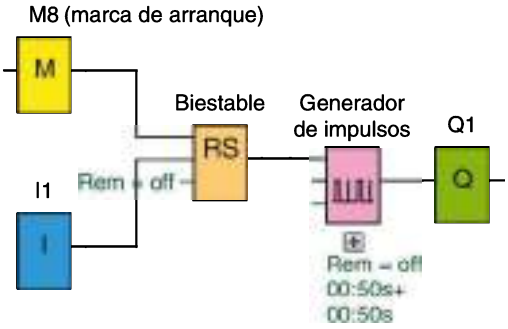
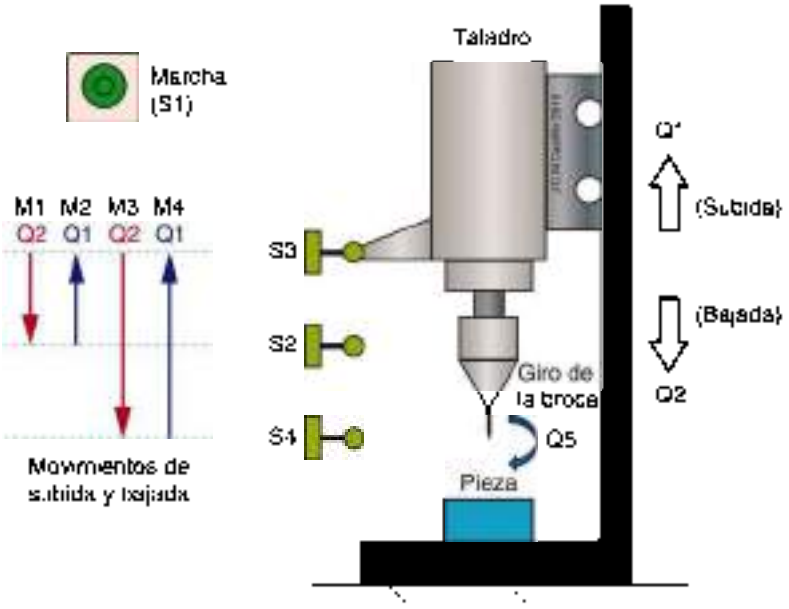


Figura 3.33. Ejemplo de la marca especial M8.

6.1. Uso de las marcas en secuencias básicas

En los autómatas programables el direccionamiento de las salidas es unívoco. Esto quiere decir que solamente se pueden llamar una vez en el programa. Sin embargo, es habitual realizar su conmutación en situaciones diferentes de una misma secuencia. En este caso es necesario plantear el programa en función de movimientos o estados basados en marcas, que posteriormente se encargarán de gestionar las salidas según las diferentes situaciones que se pueden presentar.

Sirva como ejemplo el siguiente proceso industrial de un taladro con desahogo, en el que se observa como el cuerpo del taladro baja y sube en dos ocasiones para conseguir objetivos diferentes: primero taladrar la pieza hasta la mitad y, posteriormente, hacerlo por completo. En este caso la secuencia se ha dividido en cuatro movimientos, dos de bajada con Q1 y dos de subida con Q2. Cada uno de estos movimientos se asocia a una marca (M) que se gestiona mediante biestables. Para encadenar de forma secuencial dichos movimientos se ha puesto la condición de que un movimiento no se puede ejecutar si no se está ejecutando el anterior. Además, se ha previsto que Q1 y Q2 no se puedan activar a la vez.



Ecuaciones lógicas
Secuencia de movimientos:

M1	M2
S = I1 R = S2	S = I2 · M1 R = S3
M3	M4
S = S5 · M2 R = S4	S = S4 · M3 R = S3

Acciones sobre las salidas

$$Q1 = (M2 + M4)\ \overline{Q2}$$
$$Q2 = (M2 + M3)\ \overline{Q1}$$
$$Q5 = Q1 + Q2$$

Figura 3.34. Taladro con desahogo

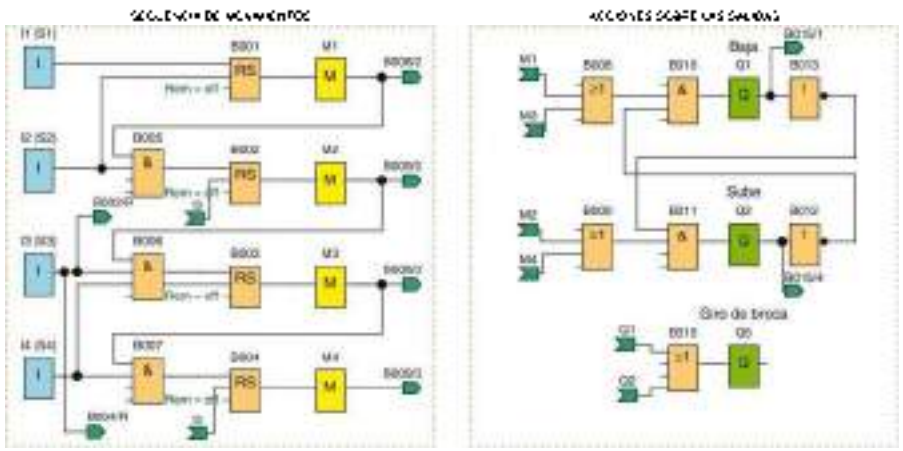


Figura 3.35. Programa en lenguaje FUP

PRÁCTICA PROFESIONAL RESUELTA

Herramientas

- Herramientas básicas del electricista

Material

- Cuaderno de trabajo
- Panel de pruebas utilizado en las prácticas de automatismos cableados
- Carril normalizado
- Canal ranurada
- Bornes para rail
- Magnetotérmico bipolar
- Autómata programable o relé programable
- Cable de línea de 1,5 mm^2
- Manguera de 5 × 2,5 mm^2
- 6 pulsadores, normalmente abiertos, para rail normalizado
- 4 lámparas de 230 V_{CA} para rail normalizado
- Software del relé programable

Puesta en servicio de un relé programable

Objetivo

- Identificación de las partes de un autómata programable y puesta en servicio del mismo

Precauciones

- Importante: en esta actividad se ha utilizado un relé programable de alimentación a 230 V_{CA}, con entradas a 230 V_{CA} y salidas a relés libres de tensión. El autómata que se utilice puede ser diferente al que aquí se propone. Por tanto, es absolutamente necesario consultar la hoja de características del fabricante para realizar correctamente las conexiones y no deteriorar de forma irremediable la electrónica del autómata.
- No manipular las conexiones con el panel conectado a la red de alimentación.

Desarrollo

1. Sobre un panel de madera hay que montar los carriles y las canaletas, tal y como se muestra en la figura.

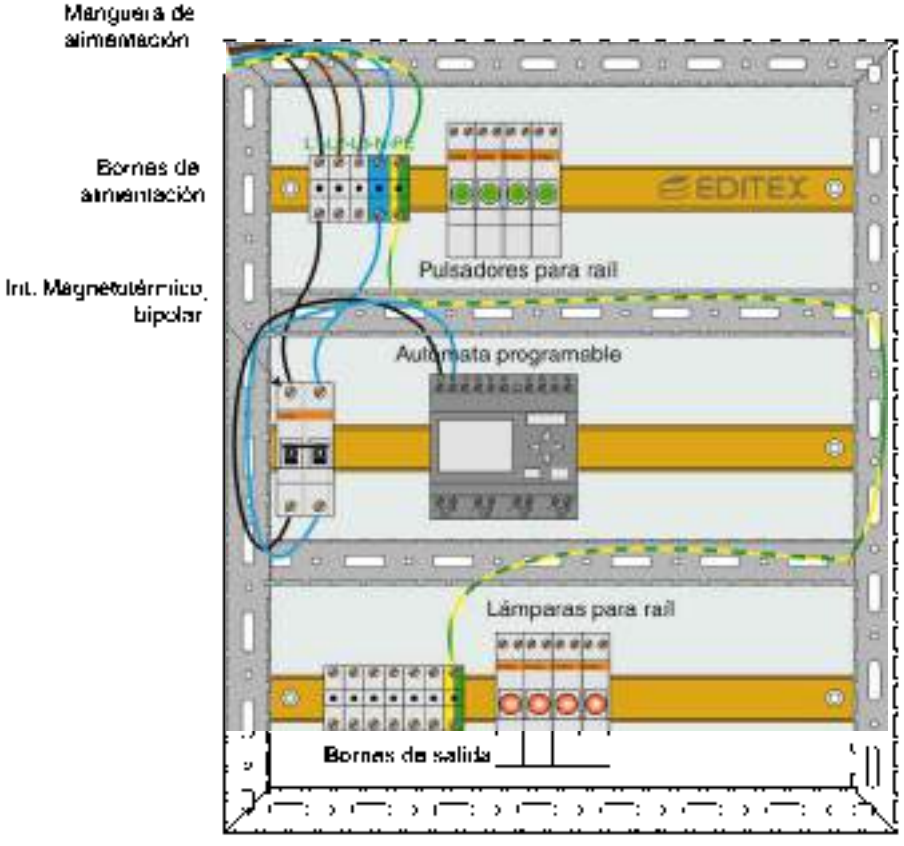


Figura 3.56. Cuadro con relé programable

2. Hay que fijar un magnetotérmico bipolar y el relé programable en el carril central, los pulsadores sobre el superior y las lámparas en el inferior.
3. Se debe conectar una de las fases y el neutro a la alimentación del autómata, pasando ambos conductores a través del interruptor magnetotérmico.
4. Hay que conectar la manguera a la red de alimentación y accionar el magnetotérmico para probar que el autómata se enciende correctamente.

5. Dibuja un esquema con las conexiones de los pulsadores a las entradas del autómata y las lámparas a las salidas.

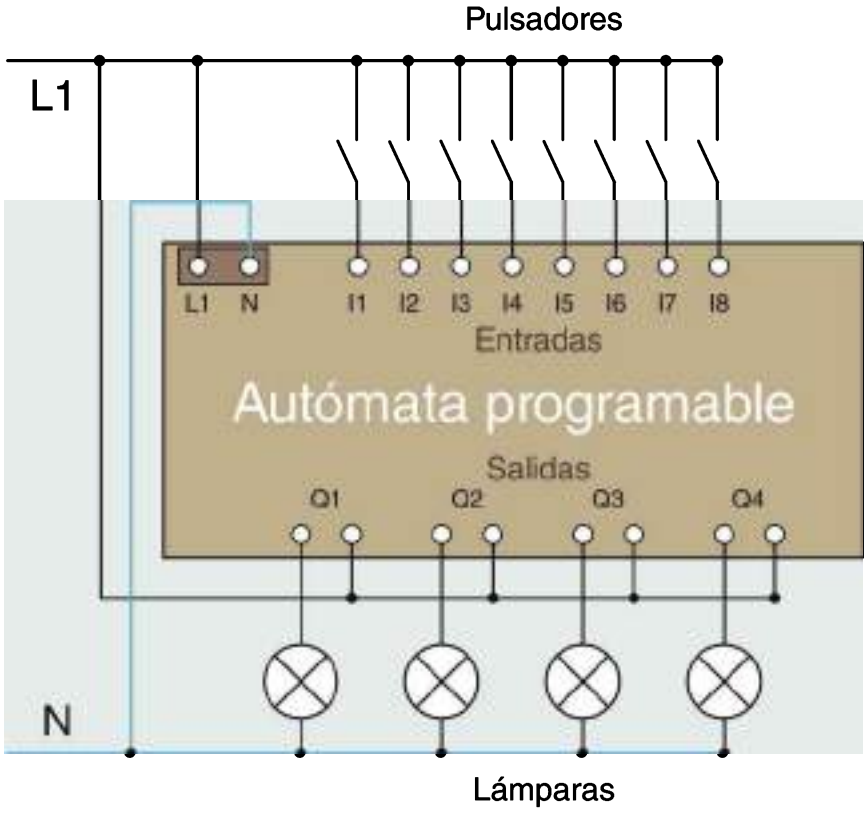


Figura 3.37. Conexiones del relé programable.

6. Hay que realizar, sobre el panel, las conexiones de los pulsadores a las entradas y las lámparas a las salidas.
7. Luego tapar la canaleta y conectar el panel de pruebas a la red de alimentación.
8. Una vez instalado el *software* de programación en un ordenador, conectar el cable de comunicación entre él y el autómata y realizar una prueba de comunicación.
9. Hay que transferir un sencillo programa de puertas lógicas y comprobar su funcionamiento accionando los pulsadores y observando lo que ocurre en sus salidas. Se puede utilizar cualquiera de los ejemplos mostrados a lo largo de esta unidad o realizar uno nuevo, pero hay que intentar que disponga del mayor número de E/S posible para llevar a cabo una óptima comprobación del cableado.

TEST DE EVALUACIÓN

RESUELVE EN TU CUADERNO O BLOC DE NOTAS

1. **¿Cómo denomina Siemens al lenguaje FBD?**
   a) KOP.
   b) FUP.
   c) PCL.
   d) SCL.
2. **Una marca en el relé LOGO es:**
   a) Un conector.
   b) Una función básica.
   c) Una función especial.
   d) Una función lógica.
3. **Relé autoenclavador es sinónimo de:**
   a) Temporizador.
   b) Contador.
   c) Biestable.
   d) Función lógica con flanco.
4. **Si un pulsador está asociado a un flanco positivo, el pulso se detecta:**
   a) Mientras el pulsador está accionado.
   b) Cuando el pulsador no está accionado.
   c) Justo en el momento de accionar el pulsador.
   d) Justo en el momento de soltar el pulsador.
5. **¿Cuando comienza la temporización de un temporizador a la desconexión?**
   a) Nada más habilitar su entrada.
   b) Cuando se mantiene a «1» lógico su entrada.
   c) Cuando se ha dado dos pulsos a su entrada.
   d) Cuando el valor de su entrada pasa de «1» a «0».
6. **TON es sinónimo de:**
   a) Temporizador a la desconexión.
   b) Temporizador a la conexión.
   c) Generador de impulsos.
   d) Temporizador con memoria.
7. **Si es necesario registrar el número de objetos que pasan por una cinta transportadora se debe usar un:**
   a) Contador.
   b) Temporizador de impulsos.
   c) Detector de flancos.
   d) Relé autoenclavador.
8. **En el relé LOGO, el detector de flanco negativo está asociado a un bloque:**
   a) AND.
   b) OR.
   c) NOR.
   d) NAND.
9. **Para evitar señales permanentes se utilizan:**
   a) Contadores.
   b) Temporizadores de impulsos.
   c) Flancos.
   d) Biestables.
10. **En el relé LOGO, la marca M8:**
    a) Activa un biestable.
    b) Resetea todas las salidas de PLC.
    c) Genera un pulso cuando se pasa el programa de STOP a RUN.
    d) Genera pulsos cada vez que se pone a «1» su entrada.

ACTIVIDADES FINALES

En todas las actividades, escribe las ecuaciones lógicas, programa el circuito lógico en el *software* de programación del relé programable y comprueba su funcionamiento, bien con el simulador o bien con el panel de pruebas montado en la práctica profesional resuelta de esta unidad.

1. Resuelve mediante circuitos con realimentación:
   - Dos salidas Q1 y Q2 se activan con las siguientes variables de entrada:
     - Q1 – a, b, c
     - Q2 – a, e
   - Y se desactivan con:
     - Q1 – d, e
     - Q2 – b, f, g

2. Resuelve mediante circuitos con realimentación:

   Tres salidas Q1, Q2 y Q3 se activan y se desactivan según la siguiente tabla:

Salida	Activar con	Desactivar con	Condiciones para que la salida se active
Q1	A o B	C	— —
Q2	E o F	C, D	Q2 no se puede activar Q1 lo está previamente
Q3	A o G	C	Q3 no se puede activar hasta que lo haga Q2

3. Realiza mediante biestables (relés autoencalvadores) la actividad 1 de esta página.
4. Haz lo mismo para la actividad 2.
5. **Luces en cascada.** Hacer que las salidas las salidas Q1, Q2, Q3 y Q4 se activen secuencialmente cada 0,5 segundos. La secuencia debe repetirse cíclicamente siempre que la entrada I1 se mantenga activada.
6. **Uso del temporizador con retardo a la desconexión.**

   En un circuito con tres salidas y dos entradas:
   - La salida Q1 se activa mediante I1 y se desactiva mediante I2.
   - Las salidas Q2 y Q3 se controlan a través dos temporizadores con retardo a la desconexión.
   - Cuando se activa Q1, las otras dos salidas también lo hacen.
   - Cuando la salida Q1 se desactiva mediante I2, la salida Q2 debe hacerlo a los 4 segundos de la primera y la salida Q3 a los 3 segundos de la segunda.
7. **Inversor del sentido de giro de un motor pasando por paro.** Hacer que un motor trifásico gire a izquierdas mediante I1 y a derechas mediante I2. El motor debe pararse mediante I3 cuando gira en cualquiera de los dos sentidos. Si el motor gira en un sentido, no debe poder hacerlo en el otro hasta que no se haya accionado el pulsador de paro.
8. **Inversor del sentido de giro de un motor sin pasar por paro.** El funcionamiento es similar a la actividad anterior pero, en este caso, si el motor gira en un sentido y si se acciona el pulsador de sentido contrario, el motor debe ser capaz de invertir el giro.

   Importante: hay que evitar que las dos salidas funcionen a la vez, ya que se produciría un cortocircuito en el circuito de fuerza. Debes solucionarlo poniendo una temporización muy breve entre la desactivación de una salida y la activación de la otra.

ACTIVIDADES FINALES

continuación

9. Con cuatro entradas, a las que se han conectado sendos pulsadores, y cuatro salidas, hacer que cada vez que se accione un pulsador se active la salida correspondiente, poniendo a «0» lógico las demás. Se debe prever la desactivación completa de todas las salidas mediante un quinto pulsador.

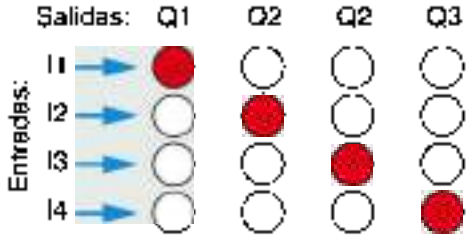


Figura 3.38. Actividad 9.

10. Tres salidas se activan en cascada en el orden Q1-Q2-Q3 transcurrido un intervalo de tiempo entre ellas. I1 es el pulsador que lanza la secuencia y I2 el que la detiene.

Las salidas Q4 y Q5 se activan con I3 y I4, respectivamente, y se paran con I5 o con S2. Dichas salidas solamente se podrán activar si el circuito de arranque en cascada (el de Q1, Q2 y Q3) ha llegado al final de su secuencia.

Si Q5 se deja activado durante más de 20 segundos todas las salidas se desactivan.

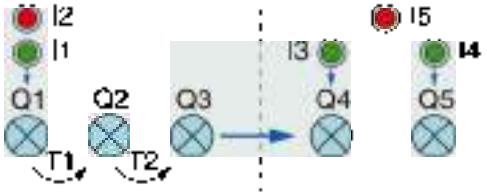


Figura 3.39. Actividad 10.

11. Uso del temporizador con retardo a la conexión con memoria.

En un circuito de tres salidas y tres pulsadores conectados a las entradas digitales del relé programable, se desea el siguiente funcionamiento: la salida Q1 se activa después de 10 segundos de pulsar I1, la salida Q2 lo hace a los 10 segundos de pulsar I2 y ambas se desactivan a los 5 segundos de haber pulsado sobre I3.

12. Puerta automática para aparcamiento de coches.

Completa el programa de la actividad del aparcamiento de coches propuesta en esta unidad. Añade una puerta de barrera a su entrada, de forma que la puerta esté elevada cuando haya plazas libres y cerrada cuando el aparcamiento esté completo.

Los movimientos de subida y bajada se controlan con dos salidas (Q1 y Q2). Para la detección de la posición de la puerta se utilizan dos finales de carrera que se conectan a las entradas del relé programable.

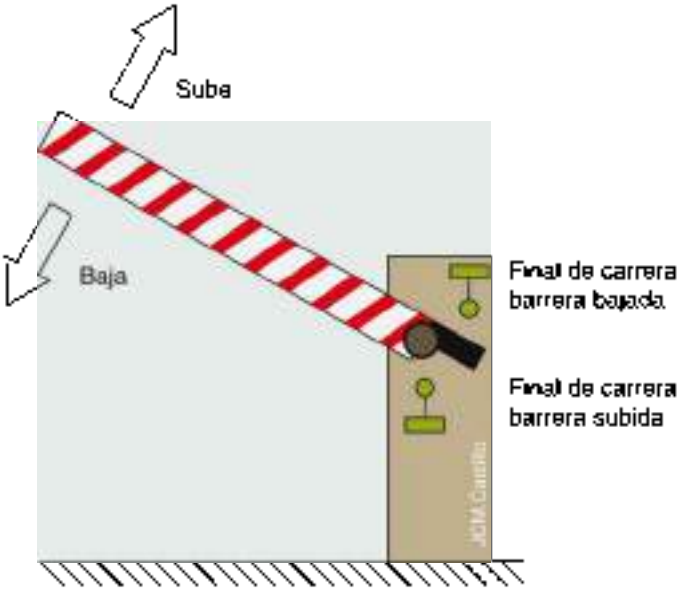


Figura 3.40. Detalle de la puerta para el aparcamiento de coches.

PRÁCTICA PROFESIONAL PROPUESTA 1

Herramientas

- Herramientas básicas del electricista

Material

- Cuaderno de trabajo
- El panel de pruebas montado en la Práctica Profesional Resuelta de esta unidad
- Software de programación del relé programable

Programación de un montacargas

Objetivo

- Realizar y comprobar el programa del automatismo de un montacargas.

Precauciones

- Se debe prever que el motor no gire en un sentido cuando lo está haciendo en el otro.
- La comprobación del programa debe hacerse manteniendo los finales de carrera activados cuando la cabina está tanto en la parte inferior como en la inferior.

Funcionamiento

La subida y bajada de la cabina se hace con un motor eléctrico, cuyos sentidos de giro están controlados mediante dos contactores (KM1 y KM2). En cada una de las plantas hay dos botoneras que permiten enviar la cabina a la parte superior o a la parte inferior. Tres lámparas deben señalizar el funcionamiento del proceso: H1 para saber cuándo el montacargas está parado arriba, H3 para hacer lo mismo cuando está abajo y H2 para indicar cuándo está en movimiento.

Desarrollo

1. Representa en tu cuaderno una tabla con la asociación de elementos del proceso de la figura y su conexión con las entradas y salidas del relé programable.
2. Escribe las ecuaciones lógicas con biestables.
3. Diseña el programa con el *software* del relé programable.
4. Cárgalo en el dispositivo y comprueba su funcionamiento.

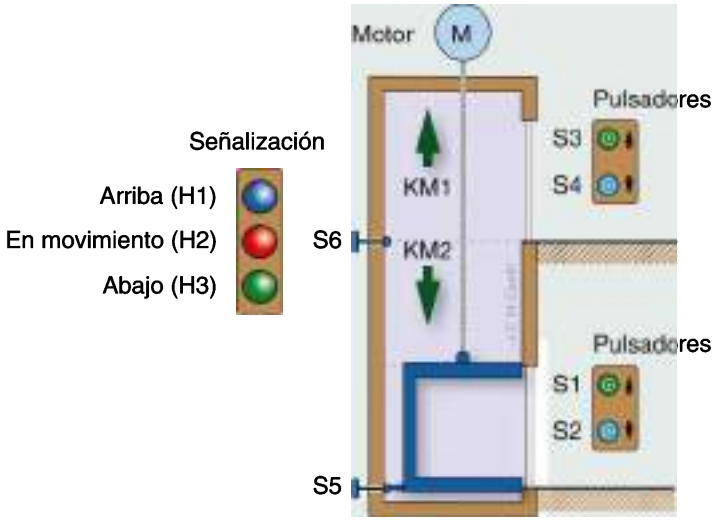


Figura 3.41. Automatismo de un montacargas.

PRÁCTICA PROFESIONAL PROPUESTA 2

Herramientas

- Herramientas básicas del electricista

Material

- Cuaderno de trabajo
- El panel de pruebas montado en la Práctica Profesional Resuelta de esta unidad
- Software de programación del relé programable

Programación de un ascensor de tres plantas

Objetivo

- Identificar cada uno de los elementos que intervienen en el automatismo.
- Representar el cableado de los sensores y actuadores en un relé programable de un automatismo industrial.

Precauciones

- Realizar y comprobar el programa del automatismo de un ascensor.
- Utilizar marcas para diferenciar los distintos movimientos del dispositivo.

Funcionamiento

- Se debe prever que el motor no gire en un sentido cuando lo está haciendo en el otro.
- La comprobación del programa debe hacerse manteniendo los finales de carrera activados cuando la cabina esté ubicada en la planta desde la que se le llama.

Desarrollo

1. Representa en tu cuaderno una tabla con la asociación de elementos del proceso de la figura y su conexión con las entradas y salidas del relé programable.
2. Observa los posibles movimientos que tiene el proceso y asocia cada uno de ellos a una marca interna del programa.
3. Representa las ecuaciones lógicas y realiza el programa, con biestables, para controlar dichos movimientos, sabiendo que los pulsadores de llamada son S1, S2 y S3 y los finales de carrera de planta S4, S5 y S6.
4. El motor requiere inversión del sentido de giro y debe estar controlado por dos contactores, uno para subir y otro para bajar. Se debe evitar que ambos funcionen a la vez.
5. Añade dos lámparas para indicar cuándo el ascensor está subiendo o está bajando.
6. Realiza el programa para el relé programable y comprueba su funcionamiento.

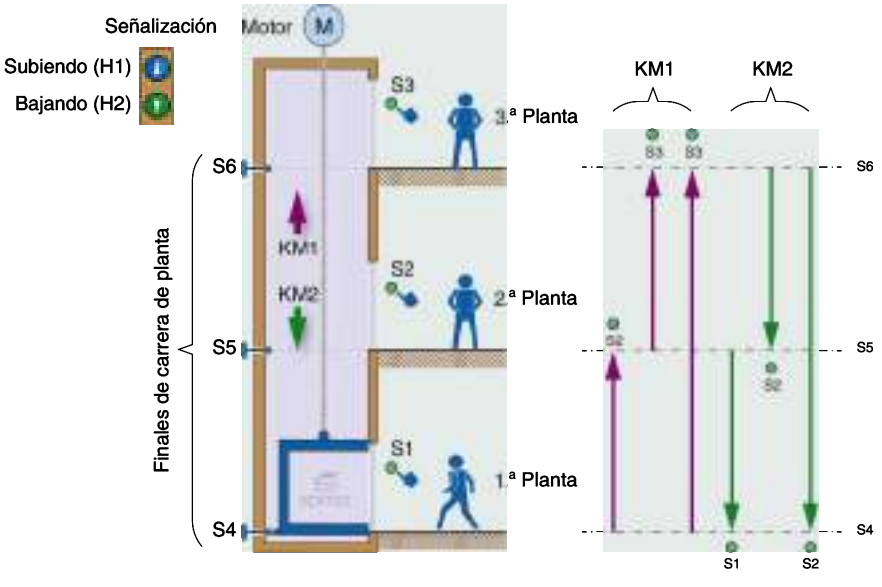
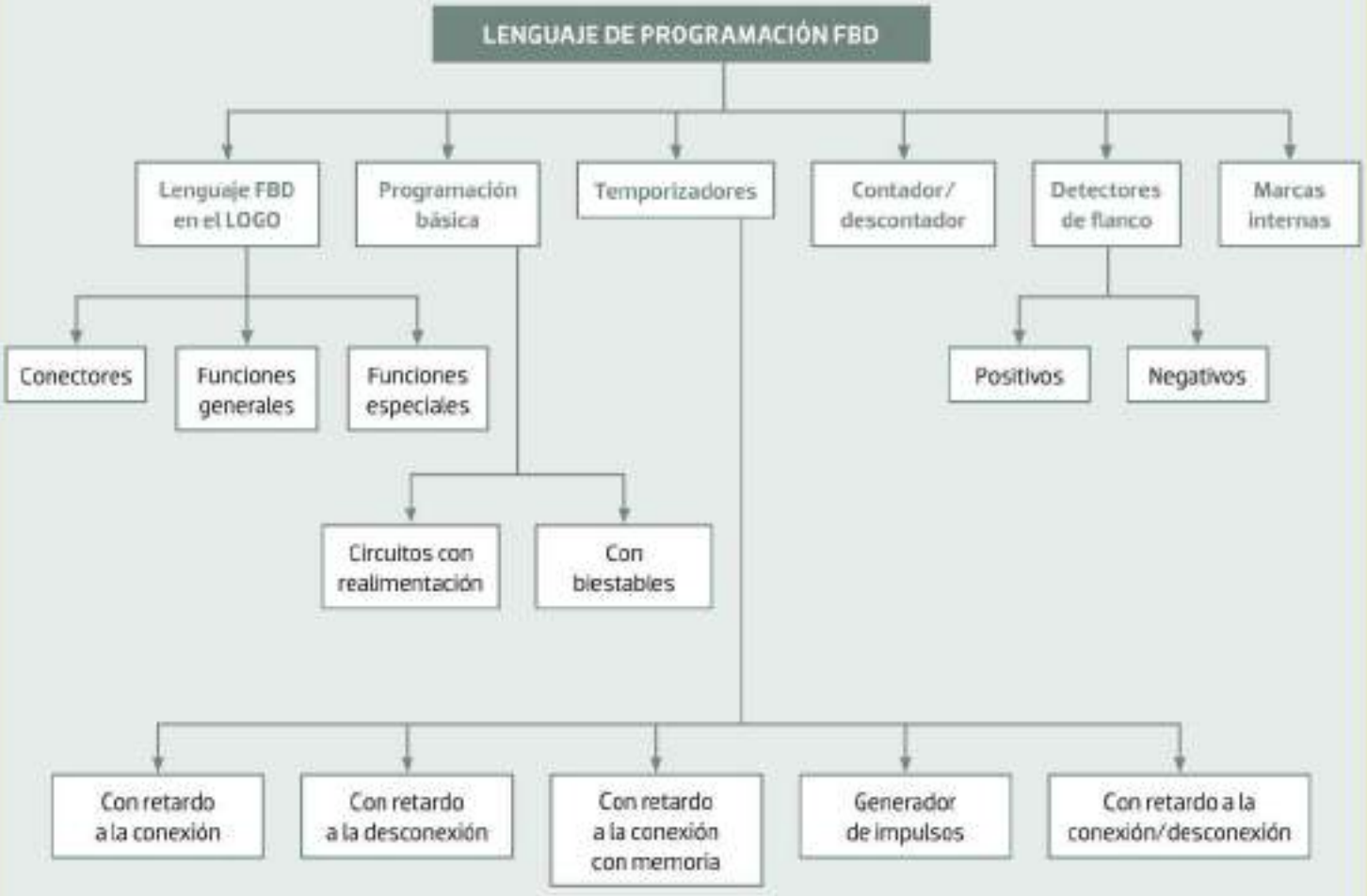


Figura 3.42. Automatismo de un taladro semiautomático.

EN RESUMEN


LENGUAJE DE PROGRAMACIÓN FBD
Lenguaje FBD en el LOGO
Programación básica
Temporizadores
Contador/ descontador
Detectores de flanco
Marcas internas
Conectores
Funciones generales
Funciones especiales
Positivos
Negativos
Circuitos con realimentación
Con biestables
Con retardo a la conexión
Con retardo a la desconexión
Con retardo a la conexión con memoria
Generador de impulsos
Con retardo a la conexión/desconexión

4 Programación en STEP 7 (I)

Vamos a conocer...

1. Introducción
2. Tamaño de los tipos de datos en STEP 7
3. Direccionamiento de entradas y salidas
4. Programación estructurada
5. Ciclo de SCAN
6. Variables en STEP 7
7. Programación en lenguaje de contactos (LD)

PRÁCTICA PROFESIONAL RESUELTA

Comunicación de un PLC mediante TIA PORTAL

PRÁCTICAS PROFESIONALES PROPUESTAS

1. Tratamiento de piezas con rociadores
2. Taladro con cargador de piezas en KOP

Y al finalizar esta unidad...

- Sabrás qué es STEP 7 y los diferentes lenguajes de programación que soporta.
- Direccionarás entradas y salidas en los PLC de Siemens.
- Identificarás los diferentes tipos de variables que se puede utilizar en STEP 7.
- Programarás de forma estructurada con bloques.
- Realizarás programas en lenguaje KOP de STEP 7 con TIA PORTAL.
- Aplicarás los conceptos de lógica digital y circuitos secuenciales estudiados en unidades anteriores, en lenguaje de contactos (KOP).

1. Introducción

El lenguaje STEP 7, tambien conocido como S7, es un conjunto de herramientas de programacion para los automatas programables y dispositivos de automatización de la firma Siemens.

El S7 esta diseñado para trabajar con la gama de productos SIMATIC, por lo que es muy habitual encontrar esta denominación en el contexto de su programacion.

En la actualidad está adaptado a la norma IEC 61131-3 y acepta los lenguajes de programación que esta propone, pero con diferente denominación.

Lenguaje	Denominación IEC	Denominación SIMATIC
Lista de instrucciones	IL	AWL
Texto estructurado	ST	SCL
Bloques de funciones	FBD	FUP
De contactos	LD	KOP
GRAFCET	SFB	S7-Graph

En STEP 7, se pueden programar los PLC de la firma Siemens correspondientes a las gamas: S7-300, S7-400, S7-1200 y S7-1500. Aunque presentan algunas diferencias en lo basico, la programación es muy similar en todos ellos, especialmente cuando se trabaja con la programación en lenguaje a contactos LD (KOP).

La programación en STEP 7 se puede realizar con dos entornos de programación diferentes, el administrador SIMATIC o el TIA PORTAL.

- El administrador SIMATIC (SIMATIC Manager) es un entorno antiguo que solamente permite trabajar los modelos de automatas S7-300 y S7-400.
- TIA PORTAL es un entorno de programación moderno que permite la programacion tanto de equipos clásicos, como los S7-300 y los S7-400, como de los actuales S7-1200 y S7-1500, entre otros dispositivos

Figura 4.2. Autómatas programables SIEMENS programables en STEP 7

STEP 7

STEP 7 es un conjunto de herramientas de programación que se pueden utilizar tanto en el SIMATIC Manager, como en TIA PORTAL

Figura 4.1. SIMATIC Manager y TIA PORTAL.

Importante

TIA PORTAL no acepta versiones de firmware de los dispositivos S7-300 y S7-400 desarrolladas antes de 2007.

Los lenguajes de programación aceptados por los diferentes modelos de PLC de Siemens son los que se muestran en la siguiente tabla:

Gama de PLC		KOP	FUP	AWL	SCL	S7-GRAPH
S7-300 S7-400		SÍ	SÍ	SÍ	SÍ	SÍ
S7-1200		SÍ	SÍ	**NO**	SÍ	**NO**
S7-1500		SÍ	SÍ	SÍ	SÍ	SÍ

En esta unidad se va a utilizar el lenguaje a contactos, que la norma IEC denomina *LD* y Siemens denomina *KOP*, esta última denominación es la que se usará a partir de ahora.

2. Tamaño de los tipos de datos en STEP 7

En función de su organización binaria, los datos en STEP 7 pueden tener los siguientes formatos:

- **Bit**: también denominado *booleano* o *bool*, es la unidad básica de información binaria. Solamente puede tener dos valores, el 0 o el 1.
- ***Byte***: está formado por ocho bits.
- ***Word*** ('palabra'): consta de dos *bytes* (16 bits).
- ***Double word*** ('doble palabra'): consta de dos *words* (32 bits).

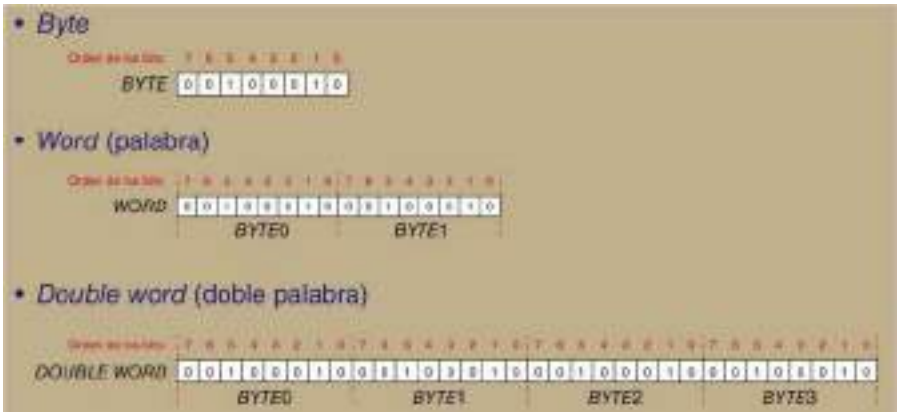


Figura 4.3. Organización de los tipos de datos en STEP 7.

Existen otros formatos, pero su estudio sale de los objetivos de este libro.

Actividades

1. Responde a las siguientes cuestiones:
   a) ¿Cuántos *bytes* y cuantos bits tiene un dato que ocupa tres dobles palabras?
   b) ¿Y si el dato tiene cinco palabras?

3. Direccionamiento de entradas y salidas

El direccionamiento es la forma de «apuntar» a las diferentes posiciones de las zonas de memoria del PLC.

El direccionamiento de las entradas y salidas depende en gran medida del modelo del autómata programable y del fabricante. Este puede hacerse de forma absoluta o simbólica; el direccionamiento absoluto apunta a la dirección física del canal. El simbólico lo hace a través de una variable creada por el usuario y que está asociada al canal físico.

Importante

El direccionamiento de las entradas y salidas es unívoco.

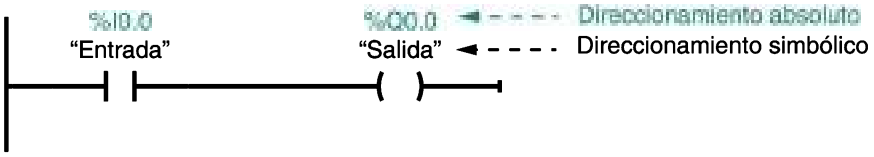


Figura 4.4. Direccionamiento de E/S en STEP 7.

A las entradas y salidas se puede acceder en formato de bit, *byte* o *word*, siendo el primero el que se va a utilizar a continuación.

Es los dispositivos de Siemens las entradas y salidas están organizadas en grupos de ocho bits, numerados del 0 a 7.

El direccionamiento absoluto se realiza mediante un operando que consta de los siguientes elementos:

- %: símbolo que indica que es un operando absoluto.
- Letra: indica si es una entrada (I) o una salida (Q).
- *Byte*: número que indica el grupo al que pertenece el operando.
- Bit: número que hace el operando dentro del *byte*.
- Punto: se utiliza como separador entre el número del *byte* y del bit.

Nomenclatura

En los dispositivos de Siemens, para identificar las zonas de memoria de entradas y salidas se puede utilizar la nomenclatura internacional o la alemana, también conocida como SIMANTIC.

Durante años ha sido habitual hacer los programas para los S7-300 utilizando la nomenclatura alemana.

Ejemplos

Direccionamiento en un S7-1200

El siguiente ejemplo muestra cómo se direccionan las entradas y salidas en un autómata programable S7-1214C. Este dispositivo consta de dos bytes de entradas y dos bytes de salidas. En ambos casos, el segundo byte (Byte 1) está incompleto. En el caso de las entradas, solamente dispone de seis bits. En el de las salidas, de dos bits.

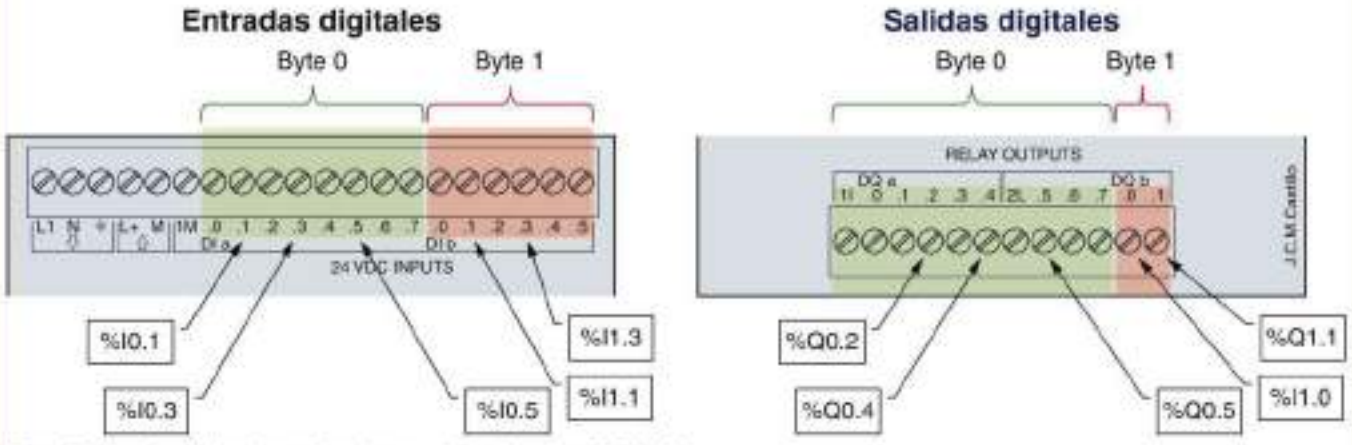


Figura 4.5. Ejemplo de direccionamiento de entradas y salidas en un S7-1214C.

continúa ›

Direccionamiento basado en bytes

Un *byte* tiene ocho bits, por lo que en el direccionamiento basado en *bytes* que utiliza STEP 7 solamente se pueden utilizar los bits del 0 al 7. Los bits 8 y 9 no son aceptados.

Así, no es posible realizar un direccionamiento como el que se muestra a continuación:

- %Q0.8
- %Q0.9

Modificar por software

En la mayoría de los dispositivos es posible modificar por *software* el direccionamiento de los *bytes* de entradas y salidas.

E/S en los S7-300

En los módulos mixtos de E/S de los S7-300 es posible repetir el número del *byte* para utilizarlo tanto con entradas como con salidas. Así, en un mismo módulo puede haber una entrada direccionada como E4.3 y una salida como A4.3 sin interferir entre ellas.

» continuación

Si a un PLC se le añaden módulos de expansión, estos se direccionan a partir del siguiente *byte* que quedó libre en el módulo anterior. Sirva como ejemplo el autómata nombrado anteriormente, al que se le ha añadido un módulo de expansión con ocho entradas y ocho salidas digitales.

Módulo principal (CPU)				Módulo de expansión	
Entradas		Salidas		Entradas	Salidas
%I0.0	%I1.0	%Q0.0	%Q1.0	%I2.0	%Q2.0
%I0.1	%I1.1	%Q0.1	%Q1.1	%I2.1	%Q2.1
%I0.2	%I1.2	%Q0.2	...	%I2.2	%Q2.2
%I0.3	%I1.3	%Q0.3	...	%I2.3	%Q2.3
%I0.4	%I1.4	%Q0.4	...	%I2.4	%Q2.4
%I0.5	%I1.5	%Q0.5	...	%I2.5	%Q2.5
%I0.6	...	%Q0.6	...	%I2.6	%Q2.6
%I0.7	...	%Q0.7	...	%I2.7	%Q2.7

Figura 4.6. Ejemplo de direccionamiento con módulo de expansión.

Direccionamiento en un S7-300

En los modelos S7-300 el patrón de direccionamiento, basado en agrupaciones de *bytes*, es el mismo que el que se ha estudiado en el ejemplo anterior para los modelos S7-1200.

No obstante, la diferencia radica en que, en estos dispositivos, la asignación de bytes está condicionada a la ubicación del módulo en el conjunto del equipo. Cada módulo de expansión tiene reservados cuatro bytes, que se pueden ocupar por completo o no, con entradas, salidas o módulos mixtos.

Así, el primer módulo direcciona en los bytes del 0 al 3, el segundo del 4 al 7 y así sucesivamente.

Si el autómata es de tipo compacto, es decir, si la CPU dispone de entradas y salidas integradas, el número de bytes asignados para estas son los últimos que aceptaría ese modelo de CPU. Así, no encontramos direccionamientos a *bytes* muy altos, como 124, 125..., y en algunos casos 136, 137, etc.

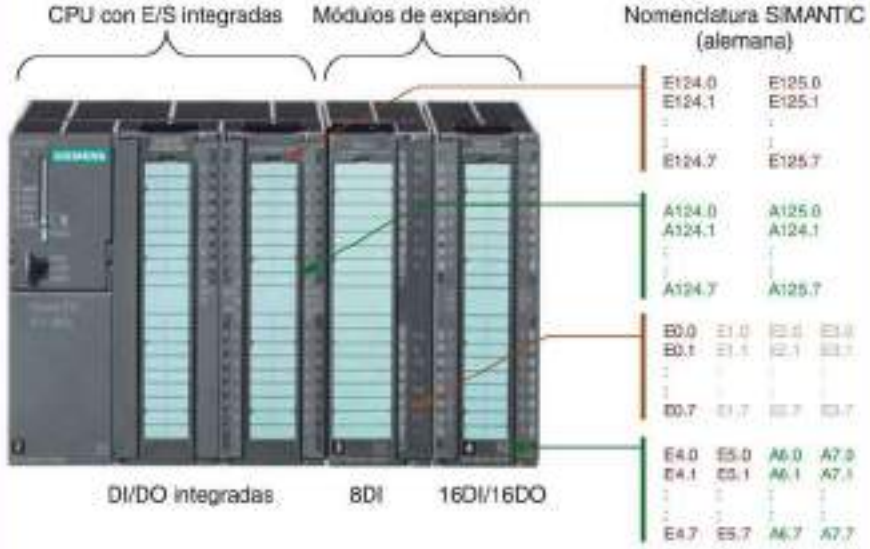


Figura 4.7. Direccionamiento de entradas y salidas en un S7-300 compacto y con módulos de expansión.

4. Programación estructurada

Es una forma de organizar los programas en bloques (subrutinas, funciones...) que son llamados y ejecutados cuando se cumplen las condiciones diseñadas por el usuario.

Esta forma de programación aporta claridad y calidad tanto en el desarrollo del código como en su posterior modificación en tareas de mantenimiento y ampliación de proyectos.

En STEP 7 los programas se estructuran en diferentes tipos de bloques.

- **OB1**: es el bloque principal (*main*). Se ejecuta cíclicamente y desde él se puede llamar a otros bloques. En todos los programas debe haber un bloque OB1.
- **FC**: también denominados *funciones*, contienen programa y pueden ser llamados o llamar otros bloques. Se numeran correlativamente.
- **FB**: son los denominados *bloques de función*. Contienen programa y, a diferencia de los FC, siempre deben tener un bloque de datos (DB) asociado. Se numeran correlativamente.
- **DB**: no contienen funciones de programación. Tienen datos en formato de tabla que se pueden leer o escribir desde otros bloques del programa. Pueden ser de dos tipos: DB globales o DB instancia. Los primeros pueden procesarse desde cualquier parte del programa. Los segundos están asociados a algún bloque FB del programa. Se numeran correlativamente.
- **OB**: son bloques especiales con funciones de interrupción, de alarmas o tratamiento de errores que se ejecutan cuando se produce el evento para el que han sido diseñados. No es necesario llamarlos desde otros bloques, simplemente se crean con el programa de usuario y se almacenan en la memoria del PLC. Los OB tienen asignado un número que indica cuál es su funcionamiento. Este debe ser consultado en el manual de usuario.

Figura 4.8. Ventana de TIA PORTAL para crear los bloques de programación en STEP 7

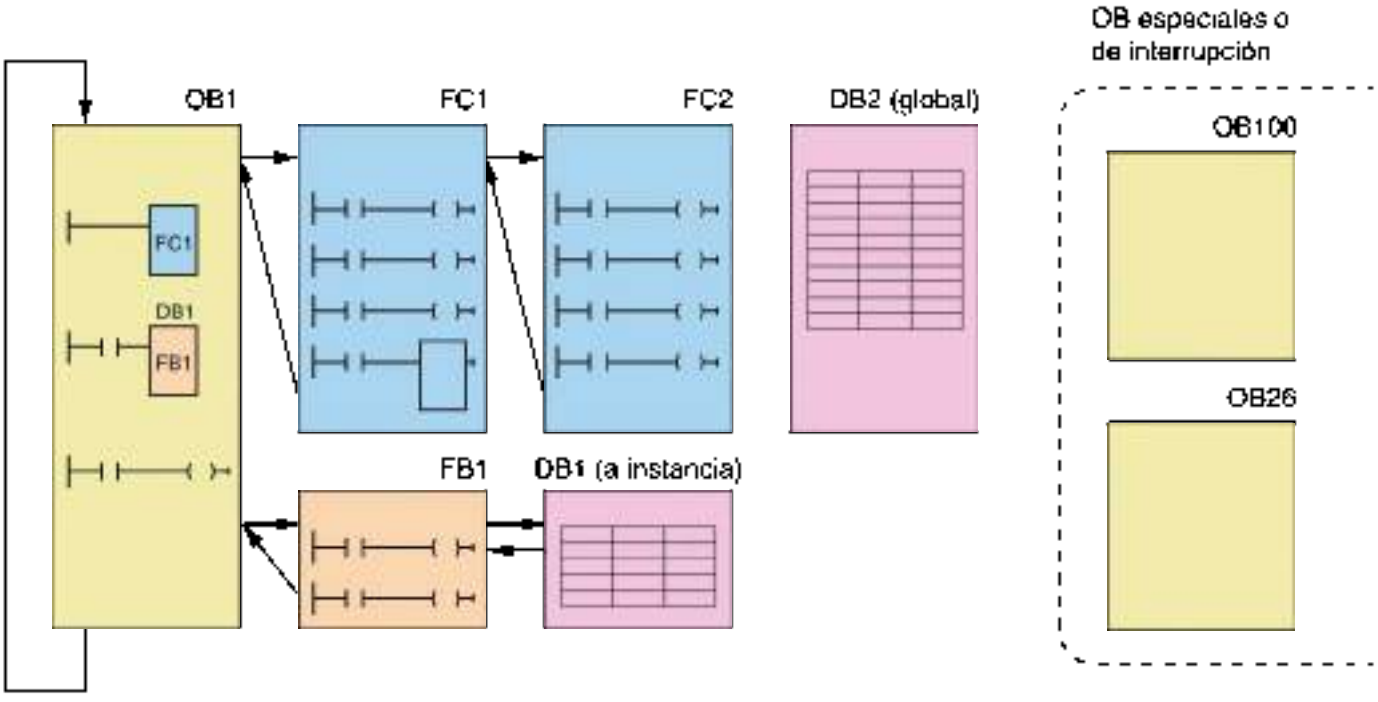


Figura 4.9 Programación estructurada en STEP 7

5. Ciclo de SCAN

Es la secuencia de operaciones que se ejecuta de forma repetitiva en la memoria del autómata. Este tiene un tiempo predefinido por defecto (por ejemplo, 150 ms), que en algunas CPU puede ser ajustado por el usuario.

En los autómatas de Siemens el ciclo de SCAN se ejecuta de la siguiente manera:

1. El PLC pasa de STOP a RUN.

   Se ejecuta el bit o bloque de arranque. En el caso de los PLC de Siemens se trata del bloque OB100, que es un bloque no cíclico que solamente se ejecuta cuando el autómata pasa de STOP a RUN.
2. Se lee la imagen del proceso en la que se encuentra el estado de las señales de entrada (PAE).
3. Se ejecutan los bloques cíclicos, que en este caso es el OB1, y la llamada a otros bloques desde él, si es que existen.
4. Se escribe en la imagen del proceso sobre las señales de las salidas (PAA).
5. Vuelve a repetirse el ciclo desde la lectura de señales de entrada de la imagen del proceso.

Ejecución secuencial

Debido a la ejecución secuencial del ciclo de SCAN, hay que tener en cuenta que las líneas de programación se ejecutan de arriba hacia abajo, por lo que, ante una contradicción entre líneas, siempre tendrá preferencia la que más abajo esté escrita.

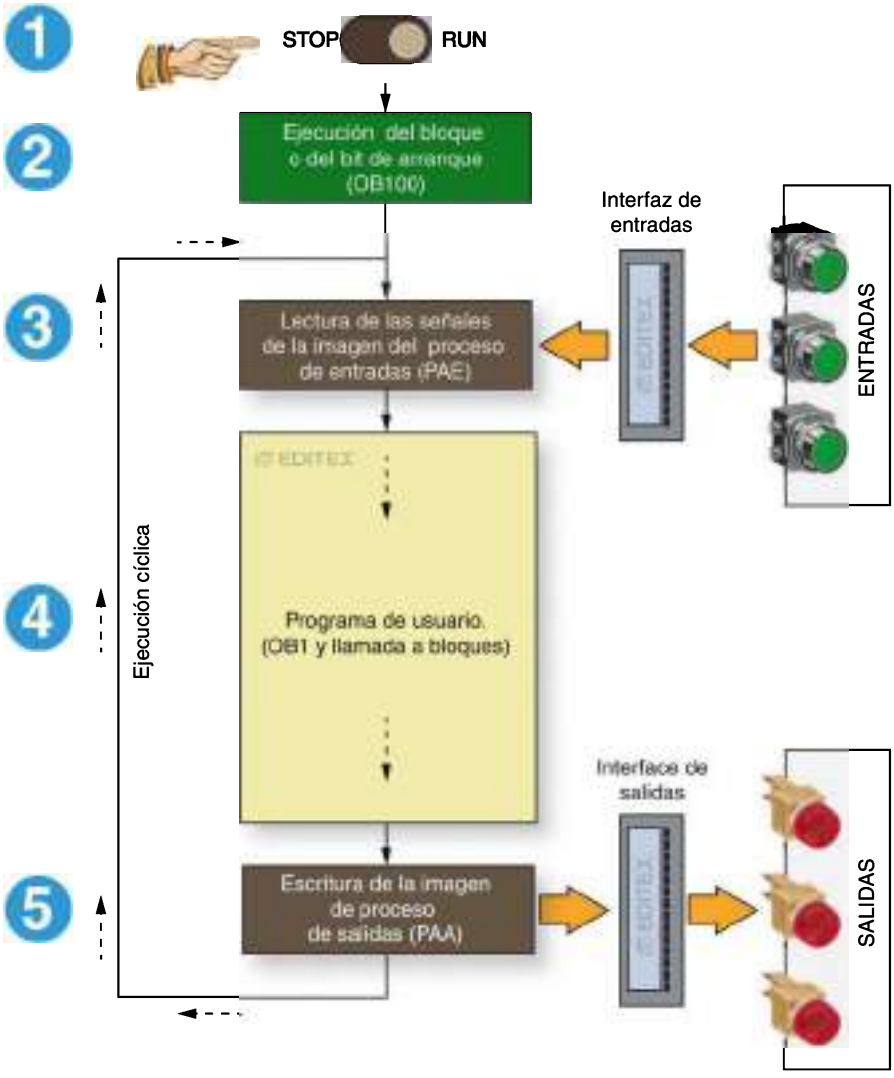


Figura 4.10. Ciclo de SCAN.

6. Variables en STEP 7

Las variables pueden ser:

- **Absolutas:** como se ha visto anteriormente, son aquellas que apuntan a zonas físicas de la memoria. El operando está precedido por el símbolo %. Por ejemplo: **%Q5.4**
- **Simbólicas:** se representan con un nombre simbólico creado por el usuario. Pueden ser de dos tipos:
  - **Globales:** cuyo alcance afecta a todos los bloques del proyecto. Se representan entre comillas. Por ejemplo: **"Mi_Variable"**
  - **Locales:** su alcance está limitado a un solo bloque. Se representan con un nombre precedido del símbolo #. Por ejemplo: **#Mi_Variable_Local**

Para facilitar la programación, es aconsejable crear la tabla de variables con los nombres simbólicos asignados por el usuario. Dicha tabla se creará al principio con las variables conocidas, pero se podrá editar y completar en el proceso de desarrollo del proyecto añadiendo todas las variables que sean necesarias.

Variables simbólicas

La programación en STEP 7, a través de TIA PORTAL, está pensada para trabajar con variables simbólicas. Si el usuario no las crea previamente, TIA PORTAL les asigna nombres temporales basados en el siguiente patrón: Tag_1, Tag_2, etc.

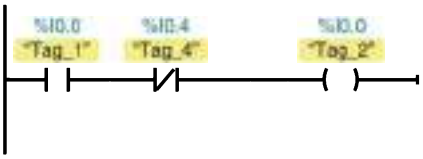


Figura 4.11. Asignación automática de nombres a variables.

VARIABLES PLC			
Nombre	Tipo de datos	Dirección	Comentario
Marcha_1	Bool	%I0.0	Pulsador de marcha motor 1
Marcha_2	Bool	%I0.2	Pulsador de marcha motor 2
Paro	Bool	%I0.3	Pulsador de paro general
Motor_1	Bool	%Q0.0	Motor 1
Motor_2	Bool	%Q0.1	Motor 2

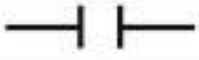
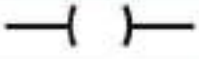
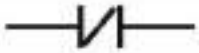
7. Programación en lenguaje de contactos KOP (LD)

El lenguaje a contactos permite crear programas para PLC de forma gráfica como si de un circuito eléctrico se tratase.

7.1. Símbolos básicos

Los símbolos básicos de programación para este lenguaje son los siguientes:

- **Contactos:** con un comportamiento similar al de los circuitos eléctricos cableados. Pueden ser abiertos y cerrados.
- **Asignaciones:** similares a las bobinas de los circuitos cableados. Reciben el valor de una combinación lógica previa. Pueden ser directas o negadas.

Contactos		Asignaciones (bobinas)	
Abierto		Directa	
Cerrado		Negada	

7.2. Asociación de contactos

Utilizando los símbolos básicos vistos en el apartado anterior, la programación en lenguaje KOP se hace asociando contactos entre sí y asignado su resultado lógico a las bobinas.

Los contactos se pueden asociar en serie, en paralelo o en circuitos mixtos de ambas conexiones.

Los programas en KOP se organizan en segmentos. Cada segmento puede disponer de una o más redes de contactos con sus correspondientes asignaciones.

Hay que tener en cuenta que en la programación destinada a los modelos S7-300 y S7-400 no es posible poner redes de contactos que no estén interconectadas entre sí en un mismo segmento, lo que sí es posible para los modelos S7-1200 y S7-1500.

Segmentación para S7-300 y S7-400

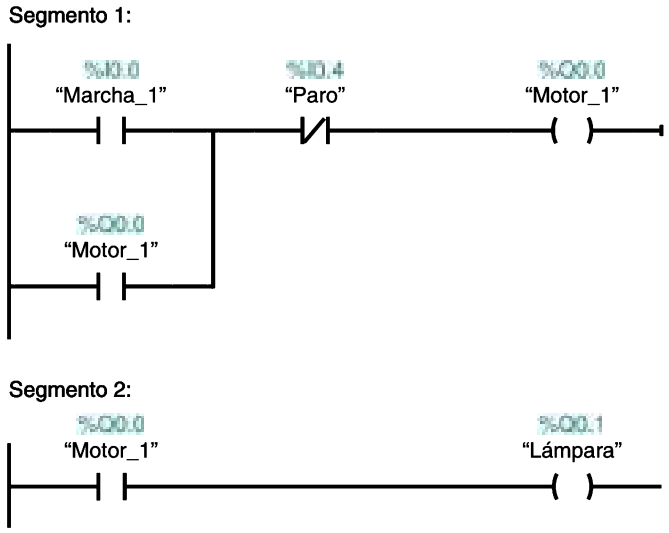


Figura 4.12. Forma de separar redes en segmentos para cualquier modelo de CPU.

Segmentación para S7-1200 y S7-1500

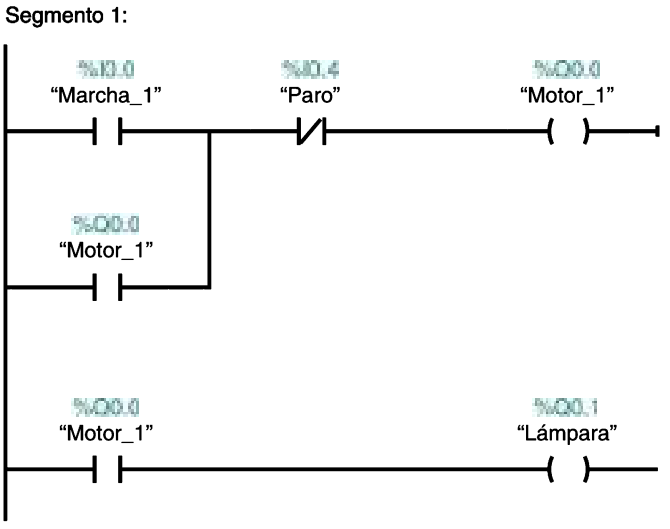


Figura 4.13. Forma de separar redes en segmentos válida solo para los S7-200 y S7-1500.

7.3. Funciones lógicas con contactos

A continuación se describen las operaciones básicas de asociación de contactos que se pueden implementar en el lenguaje KOP.

7.3.1. Conexión serie (AND)

Tiene su correspondencia con el circuito eléctrico combinacional de contactos en serie.

El resultado de la operación solamente se escribe en la bobina cuando los dos contactos están a «1» simultáneamente.

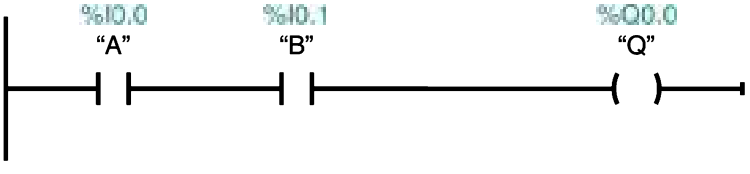


Figura 4.14. Asociación de contactos en serie.

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Figura 4.15. Tabla de la verdad.

$$Q = A \cdot B$$

Figura 4.16. Ecuación lógica.

7.3.2. Conexión paralela (OR)

Tiene su correspondencia con el circuito eléctrico combinacional de contactos en paralelo. El resultado de la operación es «1» cuando cualquiera de los contactos está activado. En la única situación en la que la bobina está desactivada es cuando los dos contactos están a «0».

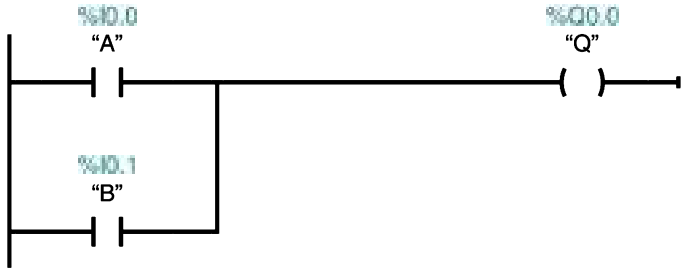


Figura 4.17. Esquema.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Figura 4.18. Tabla de la verdad.

$$Q = A + B$$

Figura 4.19. Ecuación lógica.

7.3.3. Operación negación (NOT)

La operación NOT invierte el valor de la operación lógica previa. Algunos PLC tienen un contacto específico para esta función denominado *NOT*. No obstante, como se muestra en la figura, existen otras formas para conseguir el mismo resultado. En este caso, si la variable del contacto está a «0», el resultado en la bobina es «1», y viceversa

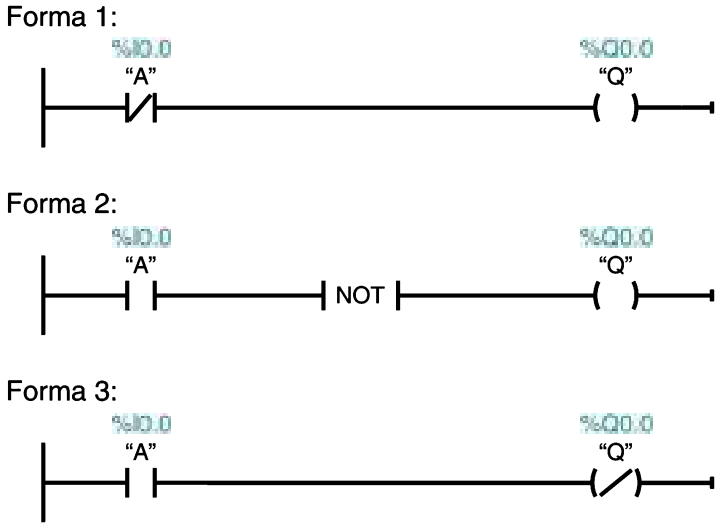


Figura 4.20. Esquema.

A	Q
0	1
1	0

Figura 4.21. Tabla de la verdad.

$$Q = \overline{A}$$

Figura 4.22. Ecuación lógica.

7.3.4. Operación serie negada (NAND)

El resultado de esta operación es inverso al de la operación AND. Se consigue negando el resultado lógico de dos contactos conectados en serie. El resultado siempre es «1», excepto cuando los dos contactos están a «1».

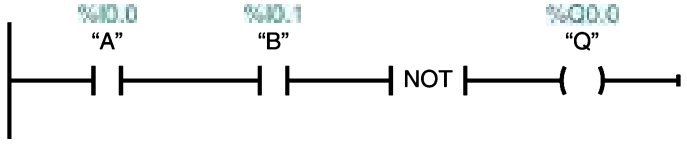


Figura 4.23. Esquema.

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

Figura 4.24. Tabla de la verdad.

$$Q = \overline{A \cdot B}$$

Figura 4.25. Ecuación lógica.

7.3.5. Operación paralelo negada (NOR)

El resultado de esta operación es inverso al de la operación OR. Se consigue negando el resultado lógico de dos contactos conectados en paralelo.

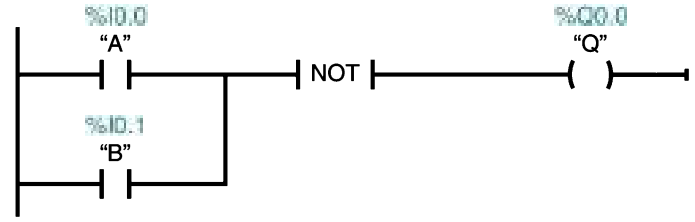


Figura 4.26. Asociación de contactos en serie.

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

Figura 4.27. Tabla de la verdad.

$$Q = \overline{A + B}$$

Figura 4.28. Ecuación lógica.

7.3.6. Agrupaciones de contactos

El lenguaje KOP permite realizar de forma gráfica grupos de contactos que a su vez están agrupados entre sí.

Agrupación AND

Conecta bloques de contactos con la función serie (AND).

En la figura se muestra una agrupación serie de dos bloques de contactos en paralelo.

Agrupación OR

Conecta grupos de contactos con la función paralelo (OR).

La figura muestra la agrupación en paralelo de dos bloques de contactos conectados entre sí en serie.

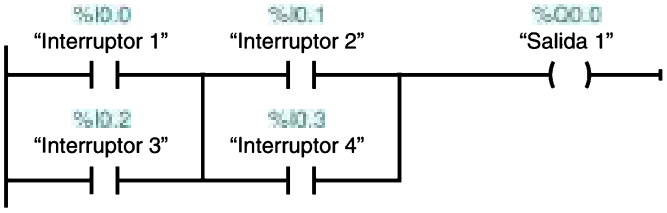


Figura 4.29. Dos grupos de contactos en OR conectados entre sí en AND.

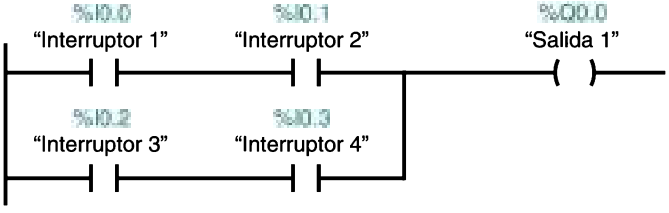


Figura 4.30. Dos grupos de contactos en AND conectados entre sí en OR.

7.3.7. Más de una salida en una red de contactos

Es posible utilizar más de una salida en una red de contactos que tiene agrupaciones en común.

En este caso, el estado de los interruptores del 1 al 4 afecta al funcionamiento de todas las salidas.

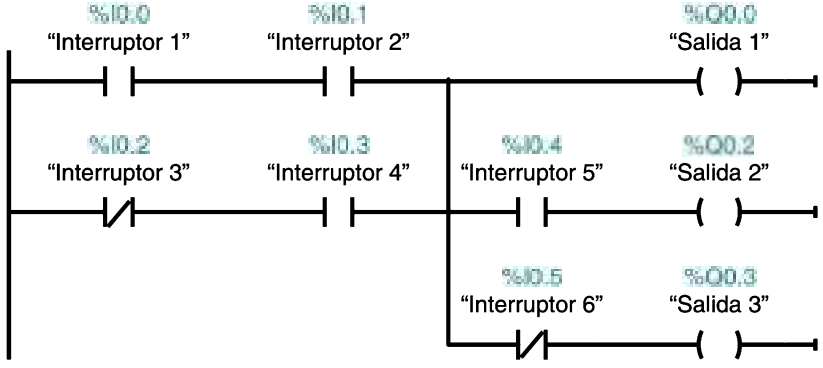


Figura 4.31. Uso de varias salidas en una misma red de contactos.

7.4. Circuitos realimentados

Se aplica de forma similar a lo ya estudiado para los bloques de funciones lógicas. En este caso, el circuito de realimentación se realiza poniendo un contacto de la salida en paralelo con el contacto, o contactos, que la activan. Un contacto negado en serie con el conjunto hace la función de RESET o desactivación.

$$Salida = (Marcha + Salida) \cdot \overline{Paro}$$

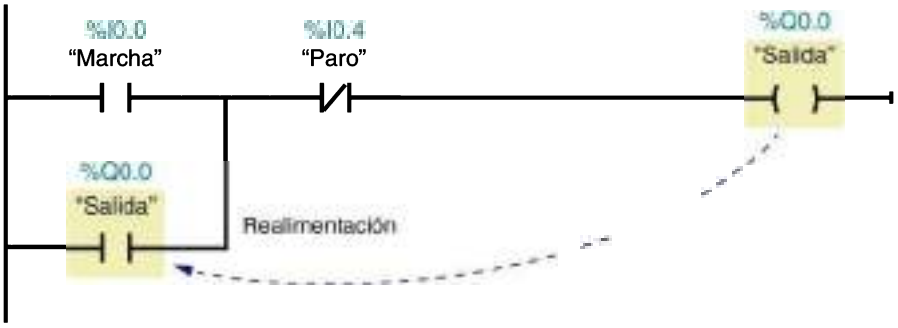


Figura 4.32. Ejemplo de programa con realimentación en KOP.

Par obtener los circuitos con realimentación con métodos intuitivos se pueden aplicar las reglas estudiadas en unidades anteriores.

7.4.1. Activación y desactivación de bobinas

- **Activación:** los contactos normalmente abiertos de las señales de activación se ponen en paralelo al contacto de la realimentación.
- **Desactivación:** los contactos normalmente cerrados de las señales de desactivación se conectan en serie con el bloque de la realimentación.

Ejemplo

La bobina Q1 se activa desde las entradas A, B o C y se desactiva desde las entradas D, E y F. La ecuación lógica y su correspondiente programa en KOP es el siguiente:

$$Q1 = (A + B + C + Q1) \cdot \overline{D} \cdot \overline{E} \cdot \overline{F}$$

Segmento 1:

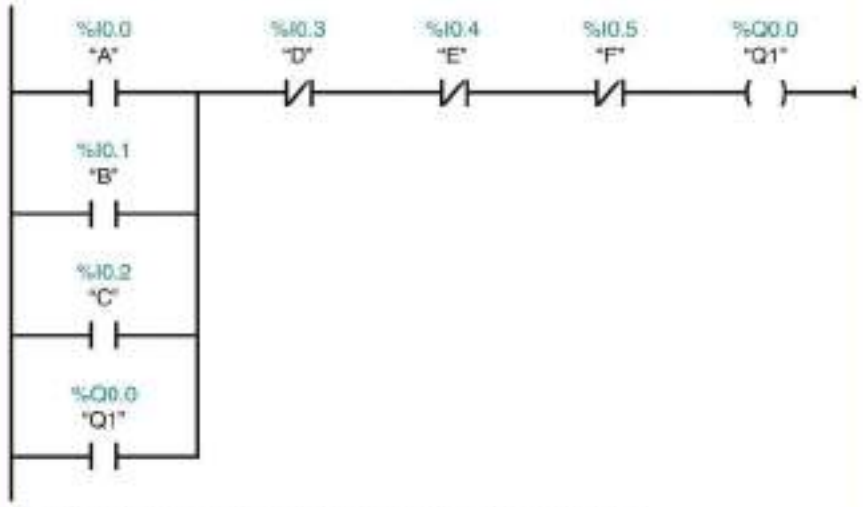


Figura 4.34. Activación y desactivación de una salida desde varios puntos.

Asignación de contactos

El lenguaje KOP permite poner la misma entrada en más de un contacto del programa, algo que es realmente útil. Sin embargo, no se debe asignar la misma variable, por ejemplo, de una salida a más de una bobina, ya que, por la secuencia de ejecución de instrucciones por el ciclo de SCAN, solamente funcionará la última que está escrita.

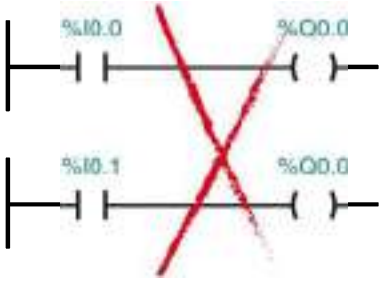


Figura 4.33. Forma incorrecta de programar una misma salida.

7.4.2. Condiciones entre asignaciones (bobinas)

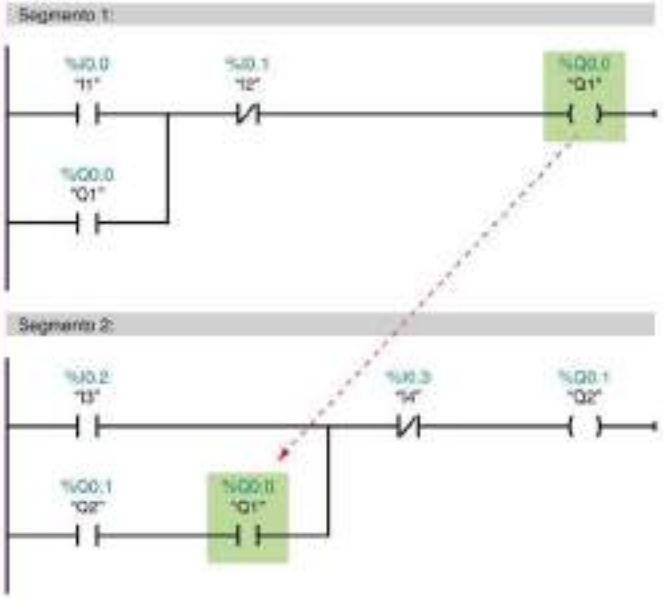
De igual forma a como ya se estudió para el lenguaje de funciones lógicas FBD, se pueden dar varios casos para condicionar las asignaciones entre sí en circuitos con realimentación. A continuación se muestran varios ejemplos de programación pero sin entrar en detalle sobre su funcionamiento, ya que las explicaciones son las mismas que las estudiadas en la unidad anterior.

Ejemplos

Condicionar la activación de una salida a otra

Caso 1:

La activación de Q2 está condicionada a la activación de Q1. La desactivación de Q2 no se ve afectada por Q1.

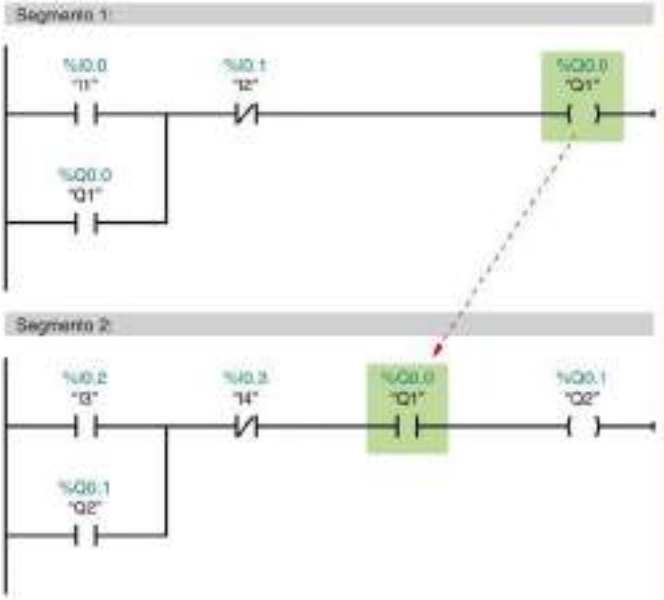


$Q1 = (I1 + Q1) \cdot \overline{I2}$

$Q2 = (I3 \cdot Q1 + Q2) \cdot \overline{I4}$

Caso 2:

La activación de Q2 está condicionada a la activación de Q1. Al contrario que en el caso anterior, Q2 se desactiva si lo hace Q1.



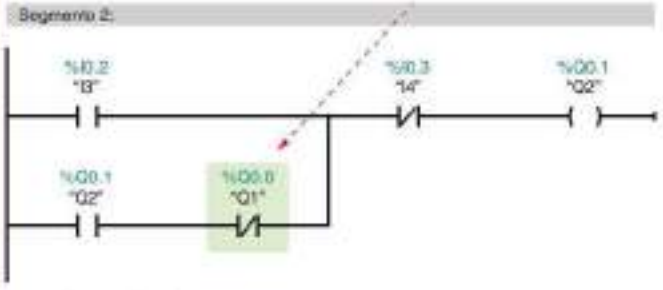
$Q1 = (I1 + Q1) \cdot \overline{I2}$

$Q2 = (I3 + Q2) \cdot \overline{I4} \cdot Q1$

Condicionar la activación de una salida a la NO activación de otra

Caso 1:

Q2 se activa aunque Q1 no lo esté previamente. La desactivación de Q2 no se ve afectada por Q1.

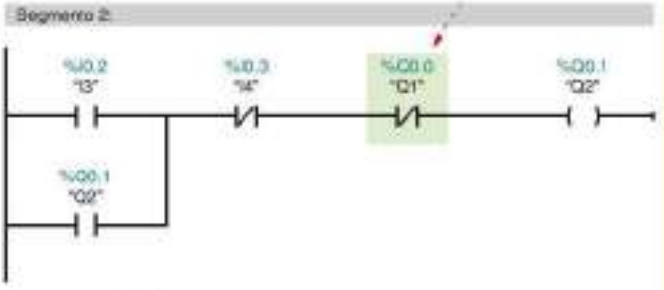


$Q1 = (I1 + Q1) \cdot \overline{I2}$

$Q2 = (I3 \cdot \overline{Q1} + Q2) \cdot \overline{I4}$

Caso 2:

La activación de Q2 está condicionada a que Q1 no esté activada. En este caso, Q2 se desactiva si Q1 se activa.



$Q1 = (I1 + Q1) \cdot \overline{I2}$

$Q2 = (I3 + Q2) \cdot \overline{I4} \cdot \overline{Q1}$

Figura 4.35. Ejemplos de cómo condicionar una salida de diferentes formas.

7.5. Marcas internas *(bytes memories)*

En STEP 7, las marcas internas se direccionan siguiendo el mismo patrón *byte.bit* utilizado para las entradas y salidas. En este caso, el identificador del operador es la letra M, precedido del símbolo %.

En todos los autómatas programables, el área de marcas está limitado a un determinado número de *bytes*. Así, para conocer la cantidad de *bytes* que se pueden direccionar, es necesario consultar la hoja de características de la CPU que facilita el fabricante.

A modo de ejemplo, se muestra la cantidad de *bytes* de algunos modelos de CPU de Siemens:

CPU	Área de marcas	Rango
S7-312C	256 bytes	%M0.0 a %M255.7
S7-318	8192 bytes	%M0.0 a %M8191.7
S7-1212C	4096 bytes	%M0.0 a %M4095.7
S7-1214C	8192 bytes	%M0.0 a %M8191.7
S7-1512C	16384 bytes	%M0.0 a %M16383.7

El uso de las marcas en formato de bit permite asociarlas a bobinas y contactos de la misma manera que otros operandos.

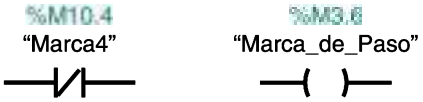


Figura 4.36. Uso de las marcas en formato de bit.

Ejemplo

El siguiente es un sencillo ejemplo de uso de las marcas. La salida Q1 no se activa hasta que se han puesto a «1» lógico las marcas M10.0 y M10.1.

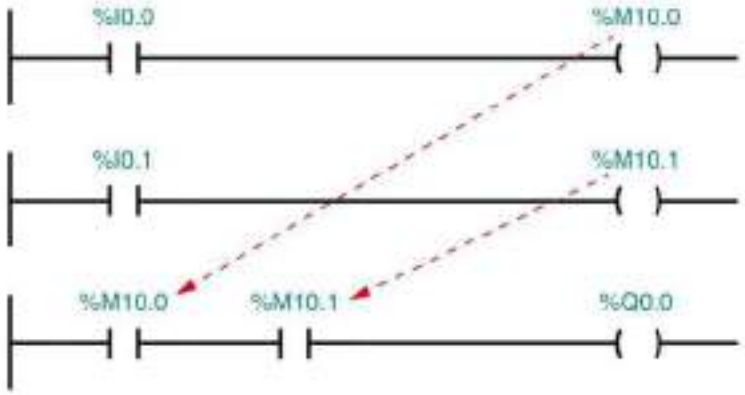


Figura 4.37. Ejemplo de uso de las marcas.

Direccionar marcas

De igual forma que las entradas y salidas, las marcas se pueden direccionar de forma absoluta (%) o mediante nombres simbólicos (""), los cuales se gestionan desde la tabla de variables del proyecto.

7.5.1. Remanencia

El concepto de «remanencia» está presente en todos los autómatas programables. Consiste en configurar determinadas zonas de memoria del PLC (marcas, temporizadores, etc.) para que mantengan su estado ante un arranque en caliente del dispositivo.

En el caso de las marcas, es necesario configurar el área deseada en cantidad de *bytes* como remanente.

Una vez configurado en el dispositivo, las marcas configuradas de esta manera mantendrán el estado en el que se encontraban antes de la parada del programa.

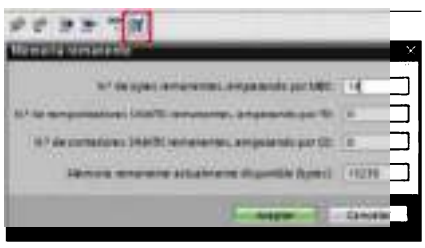
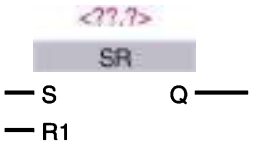
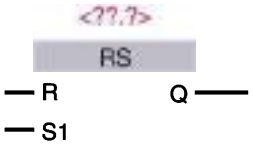


Figura 4.38. Ventana de configuración de zonas remanentes en la tabla de variables.



Bloque SR con preferencia al RESET.



Bloque RS con preferencia al SET.

Figura 4.39. Bloques SR y RS.

7.6. Biestables *(flip flop)*

De igual forma que en otros lenguajes, el uso de los biestables en KOP tiene como objetivo activar con memoria un bit mediante una instrucción SET y desactivarlo mediante una instrucción RESET.

En STEP 7 el uso de los biestables se puede realizar de diferentes formas.

7.6.1. Biestables de bloque

Se presentan como un bloque único de forma rectangular que tiene dos entradas (SET y RESET), una salida y un operando asociado en su parte superior.

Estos bloques pueden ser de dos tipos: SR y RS. Se debe utilizar uno u otro en función de la preferencia que se le quiera dar a una de las entradas, en el caso de que ambas estén a «1» lógico al mismo tiempo.

La entrada que tiene preferencia es la que está más abajo representada en el bloque. Así, tiene preferencia el RESET en el SR y el SET en el RS.

Ejemplo

A continuación se muestra un ejemplo de un biestable de bloque.

Los dos segmentos representan el mismo modo de funcionamiento. En el de la izquierda no se utiliza la salida Q del bloque, ya que la salida que hay que controlar se representa en la parte superior del propio bloque. En el de la derecha, la asignación de la salida se ha conectado directamente a la Q del bloque, y el operando, que no puede estar vacío, se ha configurado con una marca.

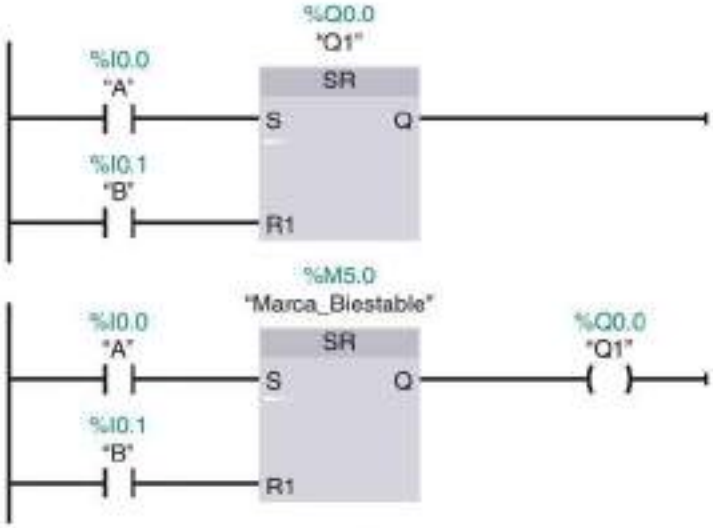


Figura 4.40. Dos formas de usar los biestables de bloque.

7.6.2. Bobinas de activación (S) y de desactivación (R)

El funcionamiento es el mismo que cualquier otro biestable; la diferencia está en que las funciones de activación (SET) y desactivación (RESET) se encuentran separadas como bobinas de asignación individuales. Esto aporta gran flexibilidad en el momento de programar el control de una variable desde diferentes zonas del programa.

Estas bobinas se pueden utilizar con cualquier operando en formato de bit (salidas, marcas, etc.).

La preferencia al SET o al RESET dependerá del orden en el que se ejecuta cada bobina. La última que se ejecuta es la que tiene preferencia.

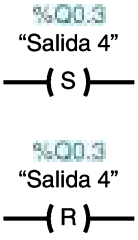


Figura 4.41. Bobinas de activación y desactivación.

Ejemplo

Siguiendo la secuencia del ciclo de SCAN, la ejecución de programa se realiza de arriba a abajo y de izquierda a derecha. Por tanto, si hay instrucciones que se contradicen, siempre tendrá preferencia la que se ha ejecutado en última posición. Así, con bobinas SET y RESET, que afectan a una misma variable, siempre tiene preferencia la que está escrita más abajo en el programa.

En el ejemplo de la figura, si los pulsadores se mantienen activados al mismo tiempo, la salida se mantendrá desactivada, ya que la bobina RESET es la que tiene preferencia.

Figura 4.42. Ejemplo de secuencia de ejecución de instrucciones.

De igual forma a como se ha realizado con los circuitos con realimentación, también es posible aplicar las reglas para el desarrollo de programas por métodos intuitivos que se han estudiado en unidades anteriores.

Ejemplo

A continuación se muestran ejemplos de cómo activar, desactivar y condicionar salidas para el desarrollo de programa por métodos intuitivos, usando las bobinas de activación y desactivación.

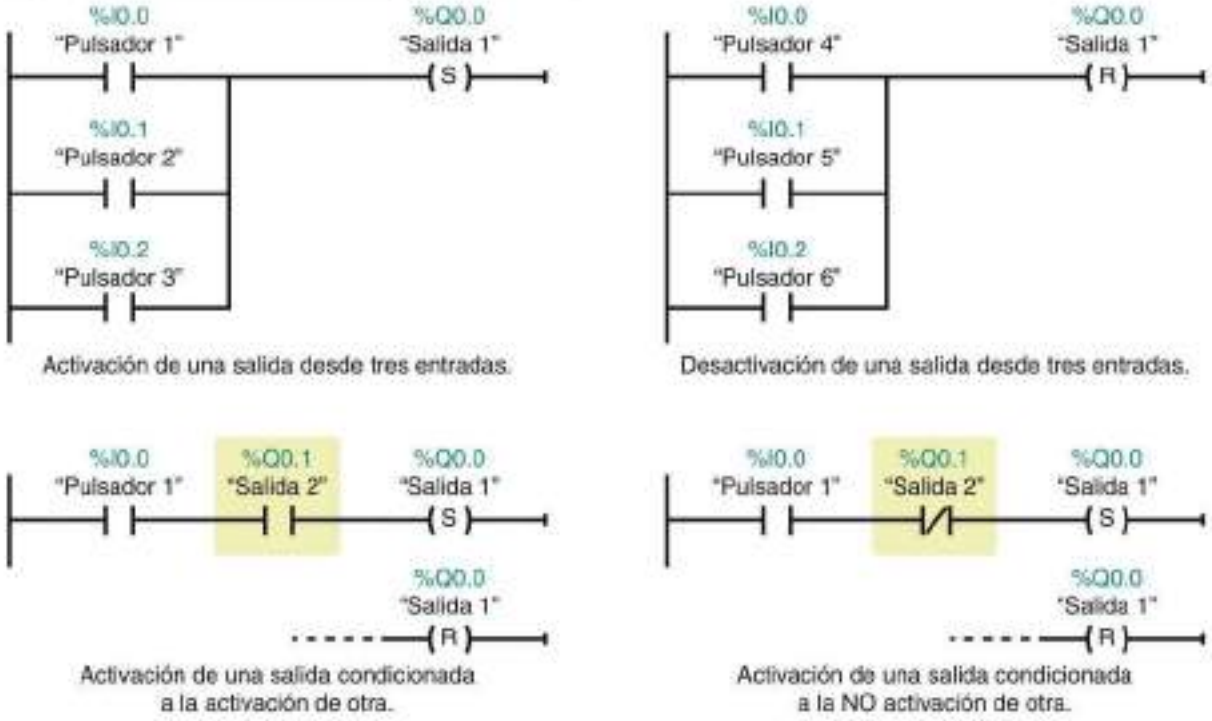


Figura 4.43. Ejemplos de cómo activar, desactivar y condicionar salidas.

7.6.3. Activar y desactivar un mapa de bits

Son instrucciones que permiten activar o desactivar varios bits a partir de una dirección específica.

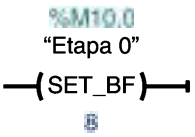


Figura 4.44. Bloque para activar mapa de bits.

Figura 4.45. Bloque para desactivar mapa de bits.

Disponen de dos operandos. El superior indica la dirección del bit y el inferior la cantidad de bits que hay que activar o desactivar a partir de él.

Actividades

2. Escribe el siguiente ejemplo de programación y comprueba su funcionamiento. Acciona en orden los pulsadores conectados a las entradas, es decir: pulsador 1, pulsador 2 y pulsador 3. ¿Qué ocurre si se mantienen accionados todos los pulsadores al mismo tiempo? ¿Por qué?

Figura 4.46. Ejemplo de uso de los bloques de activación y desactivación del mapa de bits.

3. Realiza el siguiente programa y comprueba su funcionamiento. Los segmentos del lado izquierdo aseguran que solamente se activa una marca cada vez que se acciona un pulsador. Los del lado derecho son las acciones sobre las salidas que se ejecutan cada vez que se activan las marcas.

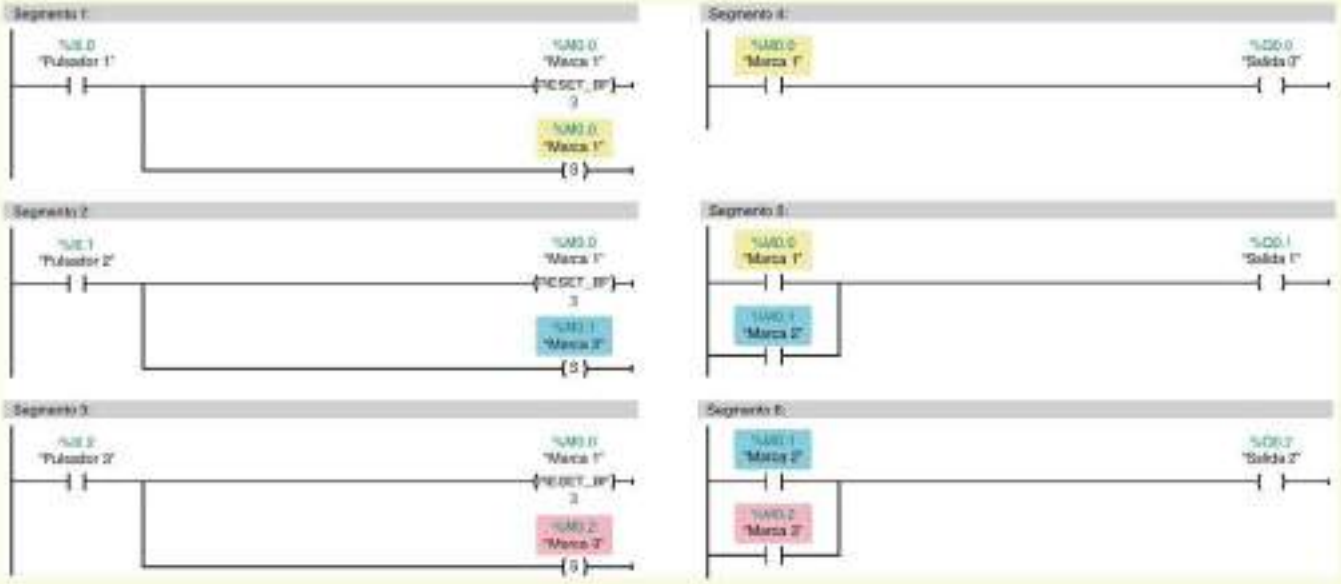


Figura 4.47. Programa de activación de marcas y acciones sobre salidas.

7.7. Detectores de flancos

Las funciones de flanco permiten detectar cuándo una variable digital cambia de valor lógico. Cuando esto ocurre, se genera un impulso de una duración de tiempo muy breve, que puede ser utilizada en el programa de usuario para evitar señales permanentes.

Los flancos pueden ser de dos tipos: positivos (P) o negativos (N), también llamados *ascendentes* o *descendientes*.

En el lenguaje KOP de STEP 7, los flancos se representan mediante un contacto especial con dos operandos. El de la parte superior es el que se corresponde con la señal de la que se desea detectar el flanco, por ejemplo, una entrada, y el de la parte inferior es una marca en formato de bit que debe ser univoca y no debe utilizarse más que para esta función en el programa.

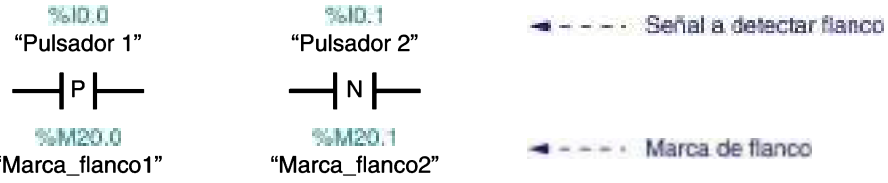


Figura 4.48. Detectores de flanco en STEP 7.

Señales con y sin flancos

Los flancos tienen gran utilidad para evitar señales permanentes. Con ellos es posible detectar el momento en el que se accede o se abandona el captador, pero no cuando se mantiene la acción sobre él.

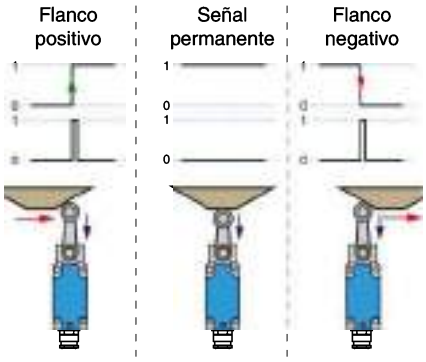


Figura 4.49. Señales con y sin flanco.

Ejemplo

El siguiente ejemplo muestra cómo se detecta el flanco de dos pulsadores. La salida se activa con el flanco negativo del pulsador 1 y se desactiva con el flanco negativo del pulsador 2.

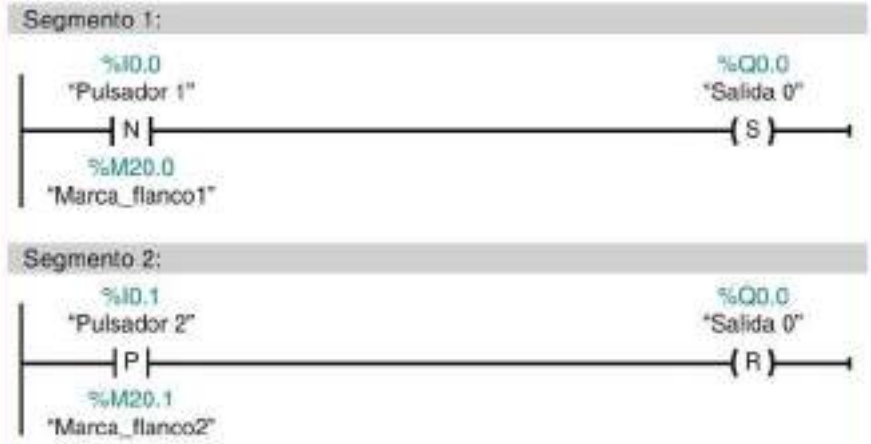


Figura 4.50. Ejemplo de uso de los flancos.

Actividades

4. Realiza y comprueba el programa para activar varias salidas mediante flancos asociados a entradas del PLC.
   - La salida Q1 se activa con el flanco positivo del pulsador 1.
   - La salida Q2 se activa con el flanco negativo del pulsador 1.
   - Las salidas de la Q3 a la Q7 se activan con el flanco positivo del pulsador 2.
   - Todas las salidas se desactivan con el flanco negativo del pulsador 3.

Vocabulary

- Bobina: coil.
- Contacto: contact.
- Biestable: flip-flop.
- Paso (Etapa): step.
- Administrador: manager.
- Dato: data.
- Direccionamiento: addressing.
- Dirección: address.
- Programación estructurada: *structured programming*.
- Variable local: *local variable*.

PRÁCTICA PROFESIONAL RESUELTA

Herramientas

- Herramientas básicas del electricista
- PC con TIA PORTAL preinstalado

Material

- Un PLC (S7-300 o S7-1200)
- Cable PC-Adapter (si el PLC es un S7-300)
- Latiguillo Ethernet (si el PLC es un S7-1200)

Comunicación de PLC mediante TIA PORTAL

Objetivo

Establecer la comunicación de un PLC con TIA PORTAL para poder probar todas las actividades propuestas en esta unidad.

Precauciones

- TIA PORTAL detecta automáticamente la configuración *hardware* de los dispositivos S7-1200 y S7-1500. Si embargo, los equipos S7-300 y S7-400 deben ser configurados manualmente.
- En el caso de la conexión Ethernet, es aconsejable desconectar el wifi del equipo y configurar y cambiar las opciones del adaptador de red de cable a modo de adquisición de IP automática. Aunque el equipo no disponga de una IP fija, TIA PORTAL se encargará de asignar una IP temporal para hacer posible la conexión.

Desarrollo

A continuación se describen los pasos que hay que seguir para establecer la conexión entre un PLC de Siemens y TIA PORTAL. Inicialmente se describe cómo hacerlo para un S7-1200, cuyo proceso también es válido para los S7-1500, y posteriormente se describen los detalles de cómo hacer la configuración para un S7-300, que es prácticamente la misma que para los anteriores pero con la diferencia de que el reconocimiento del dispositivo no se hace de forma automática.

CASO 1: puesta en servicio de un S7-1200

1. Conectar el latiguillo Ethernet entre el S7-1200 y el ordenador que tiene instalado TIA PORTAL.

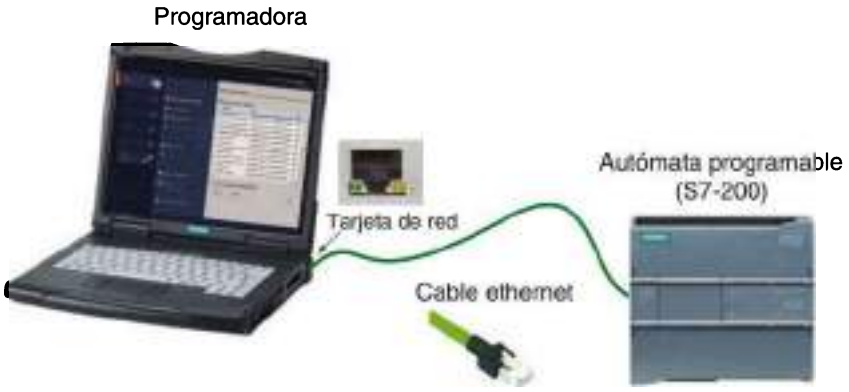


Figura 4.51. Conexión Ethernet entre la programadora y el PLC.

2. Una vez arrancado TIA PORTAL, hacer clic en **Vista del proyecto.**

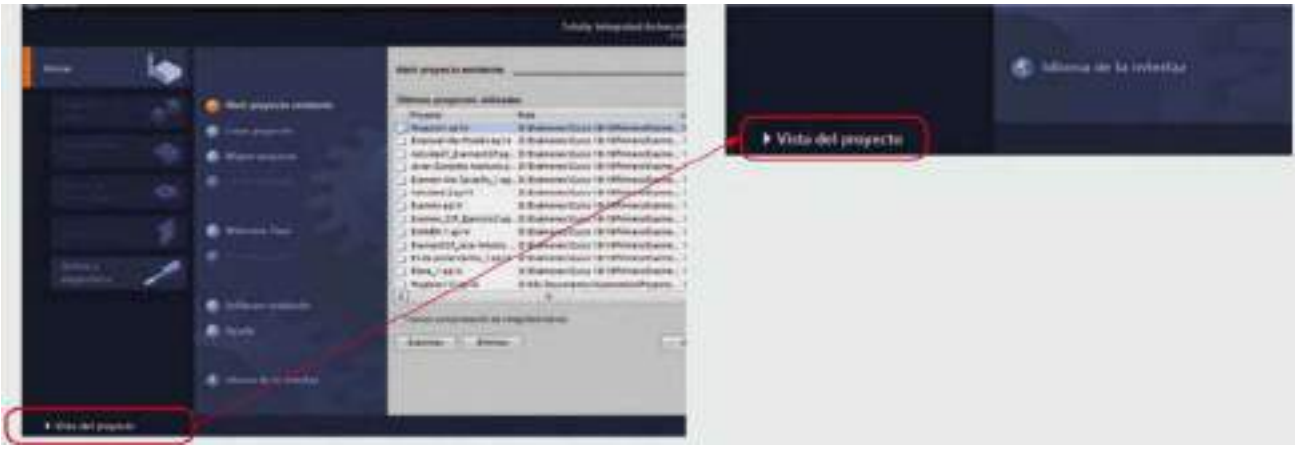


Figura 4.52. Ir a la vista de proyecto.

3. En la vista de proyecto, crear uno nuevo desde **Proyecto>Nuevo.**
4. En el árbol de proyecto, hacer doble clic en **Agregar dispositivos.**
5. En la ventana **Agregar dispositivos,** seleccionar **Controladores** y, dentro de esta categoría, elegir la carpeta de equipos SIMATIC S7-1200.
6. Dentro de la carpeta CPU, elegir CPU sin especificar y seleccionar el único equipo que aparece en su interior.
7. Elegir la versión de *firmware* de la CPU con la que se va a establecer la conexión y aceptar la configuración.

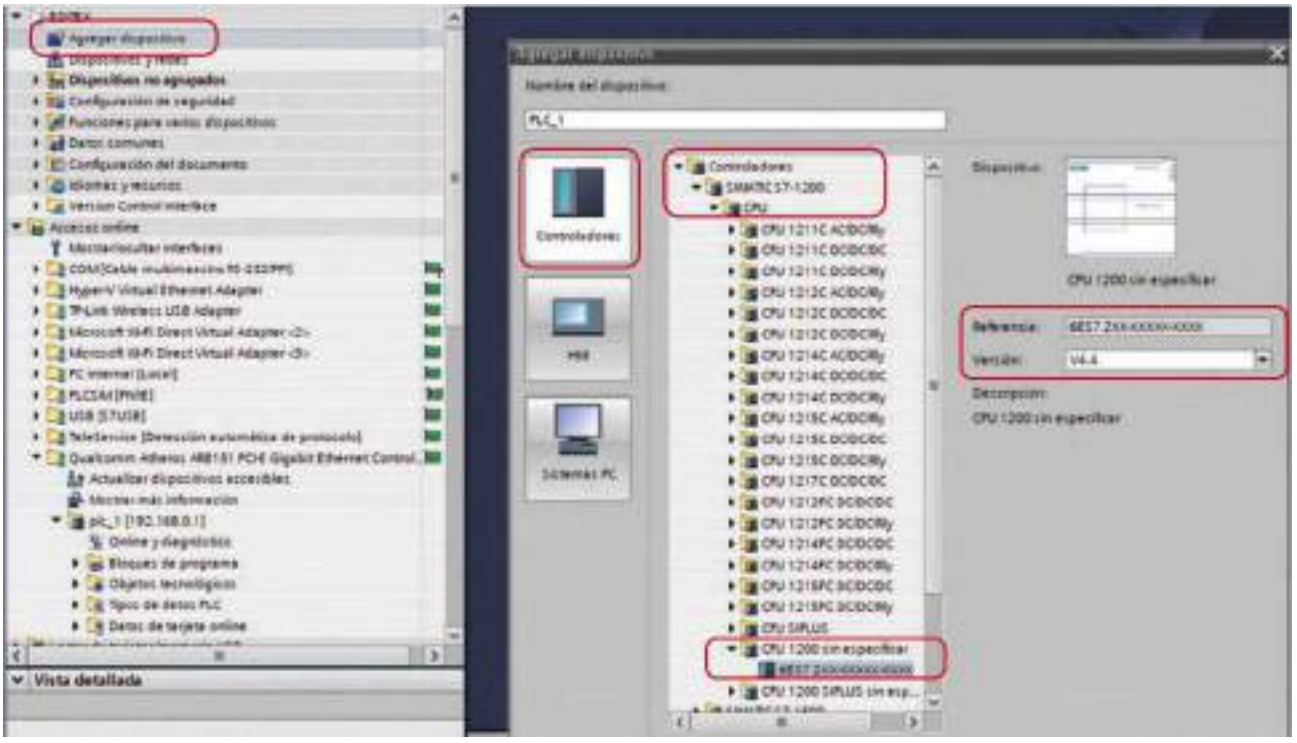


Figura 4.53. Agregar dispositivo.

8. Aparece una nueva ventana con un PLC S7-1200 sin determinar.
9. En el mensaje con fondo de color amarillo que aparece debajo de la CPU, hacer clic en **Determinar.**

PRÁCTICA PROFESIONAL RESUELTA

continuación

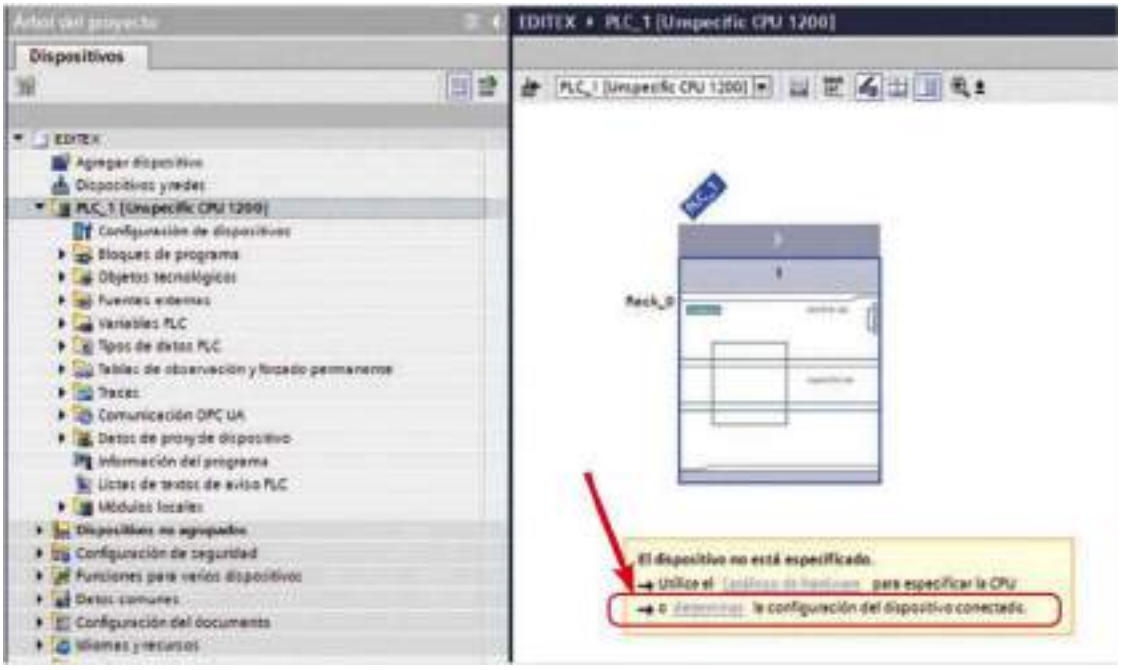


Figura 4.54. Determinar dispositivo

10. En la ventana emergente, seleccionar conexión PN (Profinet) en el menú **Tipo de interfaz PG/PC**, la tarjeta de red de ordenador por la que se hace la conexión en interfaz PG/PC.
11. Hacer clic en el botón **Iniciar búsqueda**. Si el dispositivo es reconocido, aparecerá en la tabla de **Dispositivos accesibles** de esta ventana.
12. Hacer clic en el botón **Detección**.

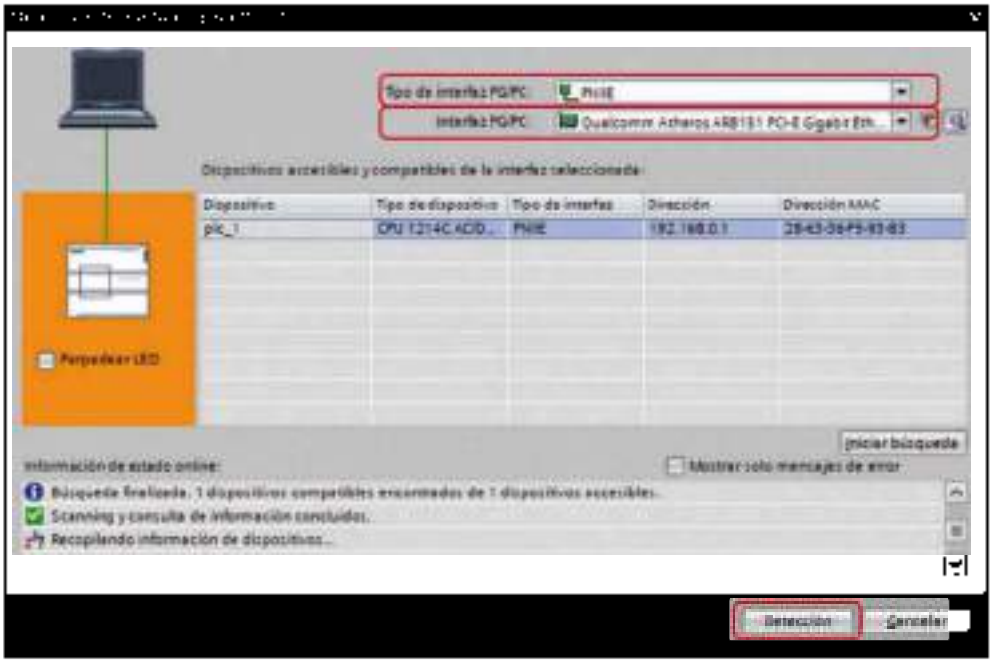


Figura 4.55. Determinar dispositivo.

13. El equipo será insertado de forma automática en el proyecto y ya estará listo para ser programado.

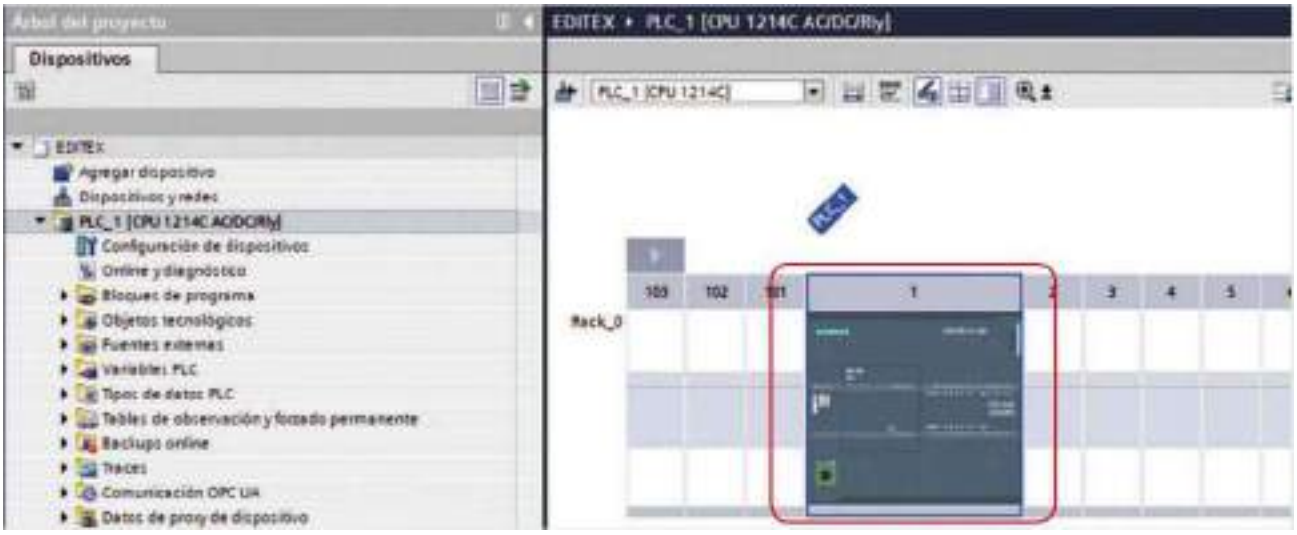


Figura 4.56. S7-1200 agregado al proyecto

CASO 2: puesta en servicio de un S7-300

1. Realizar la conexión de cable PC-Adapter entre el ordenador y el S7-300. Dependiendo del modelo de CPU que se vaya a utilizar, es posible que el autómata disponga de dos puertos con conectores DB de nueve pines. El de la izquierda es el destinado a la comunicación MPI y es este al que hay que conectar el cable PC-Adapter.

Figura 4.57. Conexión de un S7-300 a la programadora

2. Realizamos los pasos vistos anteriormente para agrergar un S7-1200 al proyecto
3. En este caso, se ha de elegir la carpeta de controladores SIMATIC S7-300.
4. Elegimos la carpta de la CPU que se va a configurar.
5. Dentro de ella, seleccionamos la referencia del equipo que se desea configurar. Esta referencia se encuentra serigrafiada en el frontal de la tapa que cubre los puertos de comunicación de la CPU.

PRÁCTICA PROFESIONAL RESUELTA

continuación

6. Seleccionamos la version de *firmware* de la CPU. Este dato se encontrará una vez levantada la tapa de los puertos de comunicación.
7. Después de aceptar la ventana **Agregar dispositivo**, el equipo se integrará en el proyecto.

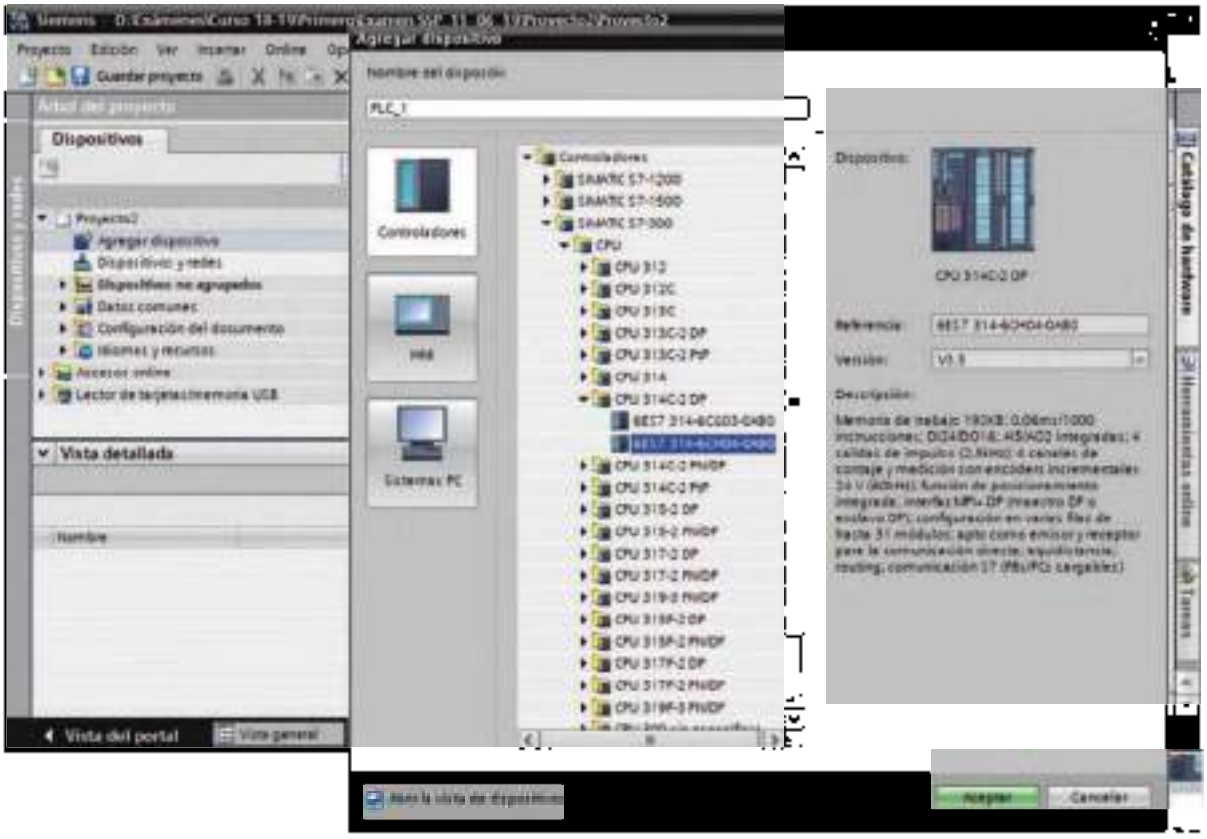


Figura 4.58. Selección de un controlador S7-300

8. En la vista de dispositivos se añadirán todas las tarjetas de ampliación de que disponga nuestro PLC físico. Se deberán seleccionar en el catálogo de productos y agregarlas manualmente según sus referencias.

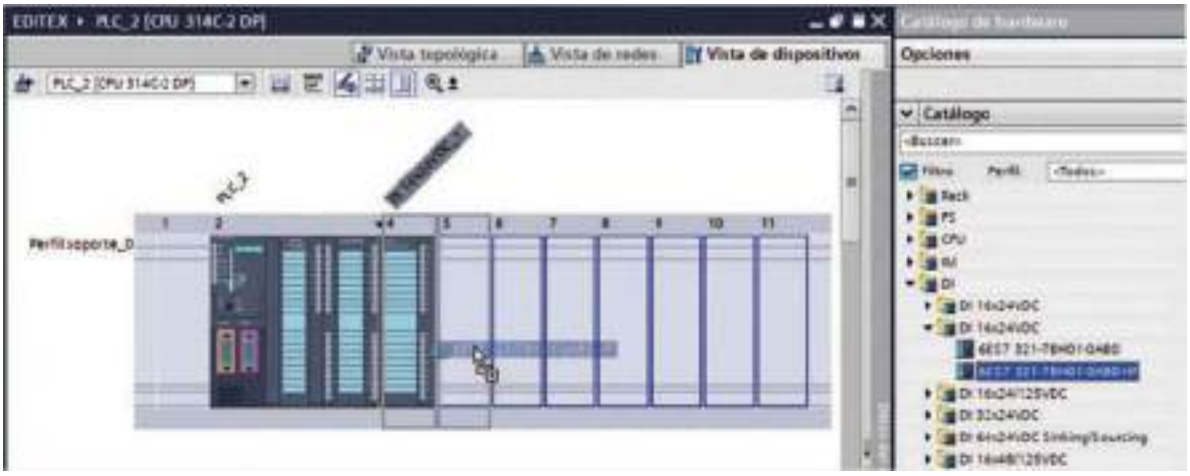


Figura 4.59. Configuración de tarjetas de expansión en un S7-300

9. En el caso del S7 300, se debe enviar la configuracion realizada en TIA PORTAL al PLC fisico para que reconozca la configuración de las tarjetas de expansión añadidas a través de la vista del dispositivo.

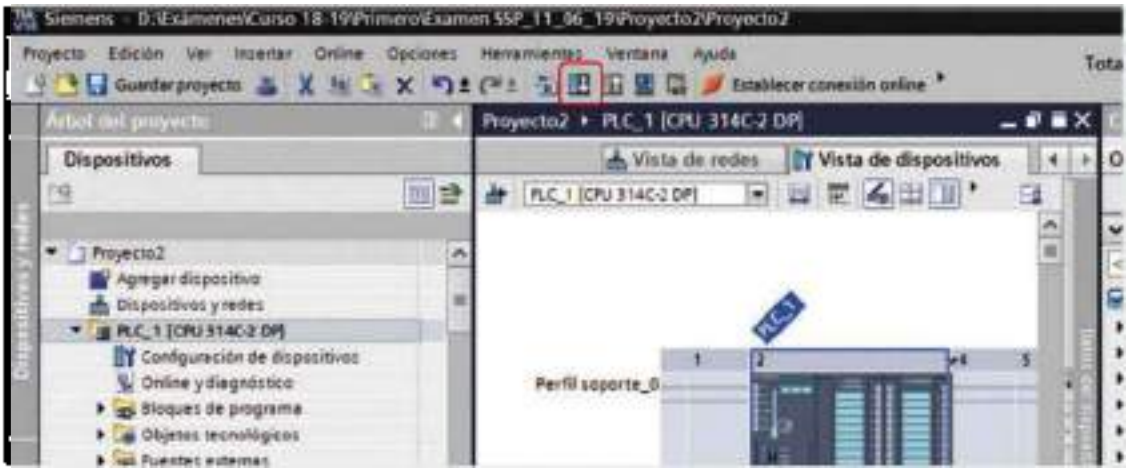


Figura 4.60. Icono para enviar configuración hardware desde TIA PORTAL al equipo físico

10. Se mostrará la ventana de comunicatcón, similar a la vista para el S7 1200. En este caso, en la opcion **Tipo de interfaz PG/PC**, hay que elegir la conexión MPI y en el interfaz PG/PC el dispositivo con el que se va a comunicar.
11. Una vez configurado el equipo, la vista de proyecto será similar a la siguiente:

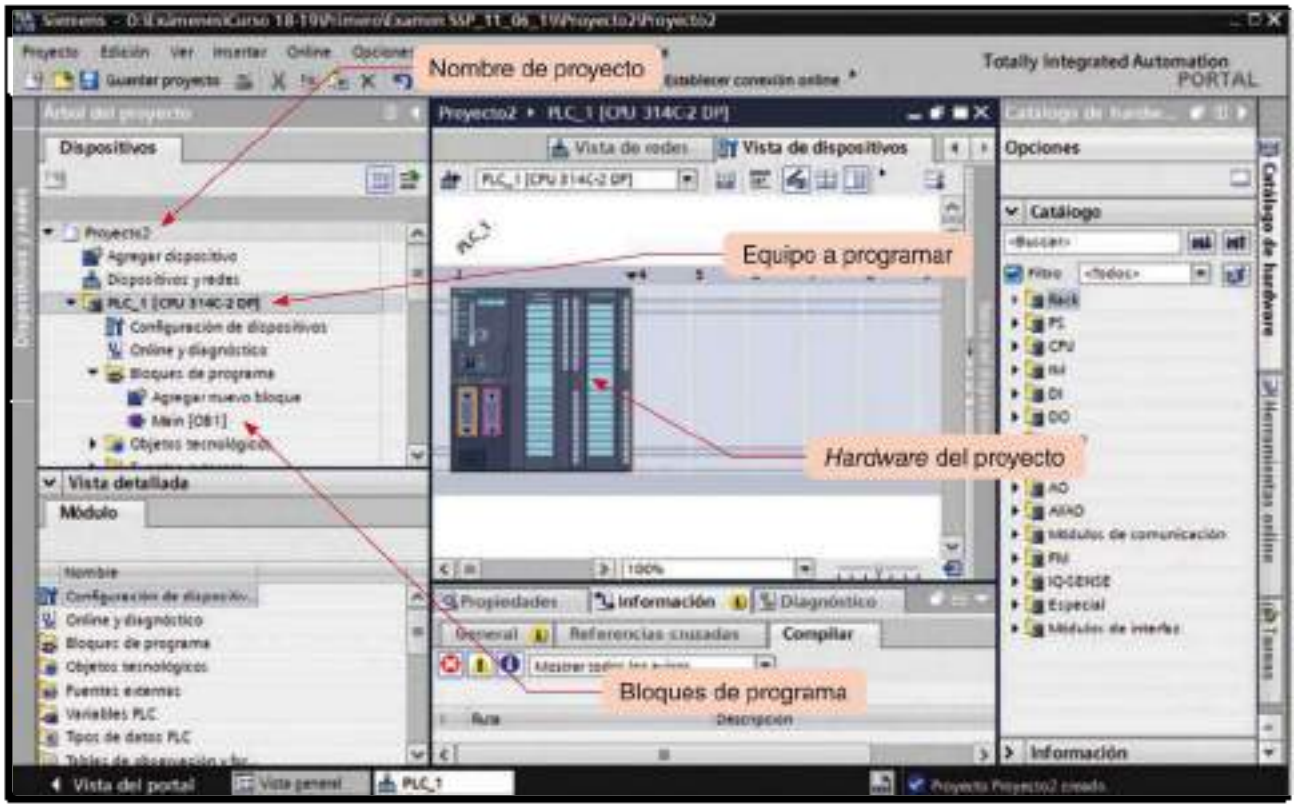


Figura 4.61. Vista de proyecto completamente configurado para un S7-300

TEST DE EVALUACIÓN

RESUELVE EN TU CUADERNO O BLOC DE NOTAS

1. ¿Cuál de estos lenguajes de programación no es aceptado por los S7-1200?

a) KOP.
b) FUP.
c) AWL.
d) SCL.

2. ¿Cuántos bits hay en tres palabras?

a) 24.
b) 16.
c) 32.
d) 8.

3. ¿Cuántos *bytes* son tres dobles palabras?

a) Cinco.
b) Seis.
c) Ocho.
d) Cuatro.

4. Si un pulsador está asociado a un flanco positivo, el pulso se detecta:

a) Mientras el pulsador está accionado.
b) Cuando el pulsador no está accionado.
c) Justo en el momento de accionar el pulsador.
d) Justo en el momento de soltar el pulsador.

5. El símbolo % en una variable indica que:

a) Es simbólica.
b) Es un número.
c) Es un temporizador.
d) Es de tipo absoluto.

6. ¿Cuáles de estas direcciones de entradas y salidas son incorrectas?

a) %I9.6.
b) %Q0.8.
c) %Q3.7.
d) %i10.

7. ¿Cuál de estas respuestas no es correcta en relación con los bloques DB?

a) Contienen programa.
b) Contienen datos.
c) Es un bloque de organización.
d) Los bloques FC siempre lo necesitan.

8. Si en un programa en KOP la bobina RESET de una salida está programada más abajo que la bobina SET de la misma salida:

a) El programa es incorrecto.
b) Solo funcionará el SET.
c) Tiene preferencia el SET.
d) Tiene preferencia el RESET.

9. Para evitar señales permanentes se utilizan:

a) Marcas.
b) Contadores.
c) Flancos.
d) Biestables.

10. Las marcas asociadas a flancos:

a) Se pueden utilizar todas las veces que se desee en el programa.
b) Se pueden usar siempre con todos los flancos asociados a una misma variable.
c) Solo se pueden utilizar una vez en el programa.
d) Deben ser remanentes.

ACTIVIDADES FINALES

1. Responde en relación al tamaño de los tipos de datos en STEP 7:
   a) ¿Cuántos bits hay en tres *bytes*?
   b) ¿Cuántos *bytes* hay en tres *words*? ¿Cuál es el número de bits?
   c) ¿Cuántas dobles palabras son 64 bits?
   d) ¿Cuántos *bytes* son tres palabras?

2. Fíjate en el equipo de la figura. Anota en tu cuaderno el modelo del PLC y los módulos de expansión. Configura un conjunto similar en TIA PORTAL y contesta a lo siguiente:
   a) ¿ Cuántas entradas y salidas digitales hay en total?
   b) ¿Cuántas de las salidas son a relé?
   c) ¿Cuántos bytes ocupa el direccionamiento de entradas?
   d) ¿Y el de salidas?
   e) ¿Cuántas entradas y salidas digitales hay en el módulo de la CPU?
   f) ¿Cuál es el direccionamiento de las entradas y salidas del tercer módulo de expansión?

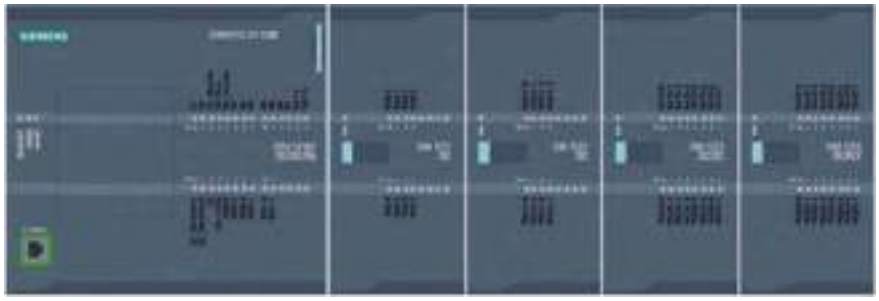


Figura 4.62. PLC y módulos de expansión.

3. Crea un proyecto en TIA PORTAL para el PLC que tengas en tu aula (S7-300, S7-1200 o S7-1500).
   Comprueba cuál es el direccionamiento de entradas y salidas de tu dispositivo.
   Crea tantos bloques FC en la carpeta de programa como actividades se muestran a continuación.
   Para las pruebas, llama solamente al FC de la actividad correspondiente en el OB1 y cárgalo en el dispositivo.
   Crea una tabla de variables con nombres similares a los que se muestran debajo de cada actividad. Estas variables deben estar asociadas a entradas y salidas físicas del autómata.

Figura 4.63. Ejemplo de llamada a FC.

4. Activa una salida mediante tres entradas, usando las funciones lógicas con contactos estudiadas en la unidad. Comprueba el funcionamiento de las funciones AND, NAND, OR y NOR.

   Nombres para las variables: Entrada1, Entrada2, Entrada3, Salida1, etc.

5. Programa y prueba en S7 el programa necesario para arrancar un motor con pulsadores de marcha y paro.

   Variables que utilizar: macha, paro, motor.

ACTIVIDADES FINALES

continuación

6. Programa y prueba un programa para invertir el sentido de giro de un motor pasando por paro. Hay que tener en cuenta que no se debe activar la salida de giro en un sentido cuando la contraria está a «1».

   Variables que utilizar: Pul_Izq, Pul_Dch, Pul_Paro, Motor_Izq, Motor_Dch

7. Programa y prueba un programa para invertir el sentido de giro de un motor sin pasar por paro.

   Símbolos que utilizar: Pul_Izq, Pul_Dch, Pul_Paro, Motor_Izq, Motor_Dch

8. Utilizando biestables (bobinas de activación y desactivación), programa y comprueba los circuitos de las dos actividades anteriores.

9. Programa la activación con realimentación de cuatro salidas en cascada de forma que Q1 se active con I1, Q2 con I2 si previamente está activada Q1, Q3, con I3 si previamente está Q2 y Q4 con I4 antes se ha activado Q3. Todas se desactivan con I4.

10. Utilizando circuitos con realimentación, realiza el programa en KOP para el control de tres salidas como se muestra en la tabla y comprueba su funcionamiento en un PLC.

    Tres salidas Q1, Q2 y Q3 se activan y se desactivan según la siguiente tabla:

Salida	Activar con	Desactivar con	Condiciones para que la salida se active
Q1	A o B	F	— —
Q2	E o C	F, D	Q2 no se puede activar si Q1 lo está previamente. Q2 no se debe desactivar si Q1 se pone a 1
Q3	A o G	F	Q3 no se puede activar hasta que lo haga Q2

11. Utilizando bobinas SET y RESET, haz que las siguientes salidas se activen y desactiven como se indica:
    - Q1 se activa con Entrada1.
    - Q2 se activa con Entrada2 y Entrada3.
    - Q3 se activa con Entrada3 si no están activadas ninguna de las otras dos.
    - Q2 y Q1 se desactivan con Entrada4
    - Todas se desactivan con Entrada5.

12. Programa y prueba el arranque de un motor en estrella-triangulo. La conmutación estrella-triángulo se hace de forma manual mediante un pulsador

    Nombres de variables: Pul_Marcha, Pul_triángulo, Pul_Paro, Relé_estrella, Relé_triángulo, Relé_principal

13. Utilizando cuatro salidas y seis pulsadores, realiza el programa con bobinas SET y RESET para que las salidas queden activadas como se muestra en la figura, cada vez que se acciona el pulsador correspondiente.

    Nombres de variables: I1, I2, I3, I4..., Q1,Q2...

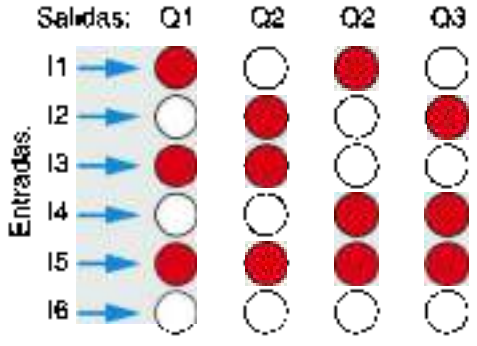


Figura 4.64. Controlar salidas mediante pulsadores.

PRÁCTICA PROFESIONAL PROPUESTA 1

Herramientas

- Ordenador con TIA PORTAL preinstalado

Material

- PLC (S7-1200 o S7-300)
- Cable de programación

Precauciones

Es aconsejable el uso de marcas para programar los dos modos de funcionamiento.

Tratamiento de piezas con rociadores

Objetivo

- Identificar los diferentes elementos de entrada y de salida del proceso y asociarlos a variables del PLC.
- Programar un circuito secuencial con dos opciones de funcionamiento.

Desarrollo

1. Realiza y comprueba el programa en TIA PORTAL para controlar el el siguiente proceso industrial:

Cinta transportadora.

- La cinta transportadora solamente se mueve en un sentido de giro.
- La cinta se pone en marcha con cualquier modo de producción y se detiene cuando el detector B6 detecta, por flanco negativo, que la pieza ha caído a la caja

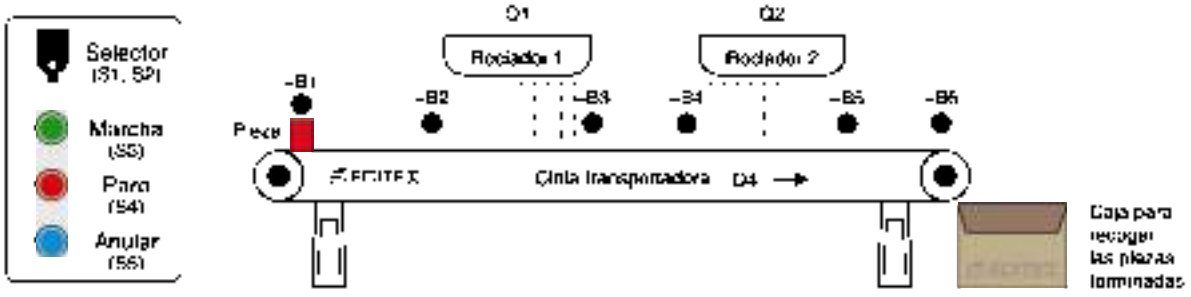


Figura 4.65. Tratamiento de piezas con rociadores.

Funcionamiento de los rociadores:

- El detector de entrada activa el rociador
- El detector de salida desactiva el rociador cuando la pieza lo ha abandonado (flanco negativo)

Funcionamiento de producción:

- Las piezas pueden ser tratadas de dos formas: un solo rociado o dos rociados.
- Los modos de producción se ejecutan de la siguiente manera:

	Selector	Marcha	Presencia de pieza	Rociado 1	Rociado 2
Modo 1	S1	S3	B1	Q1	— —
Modo 2	S2	S3	B1	Q1	Q2

Parada:

- Cuando se acciona el pulsador de parada, los rociadores dejan de funcionar y el motor de la cinta transportadora se detiene.

Modo anular:

- Si se acciona el pulsador anular, los rociadores dejan de funcionar pero la cinta transportadora sigue moviendo la pieza hasta que cae en la caja. Esta cinta se detiene mediante flanco negativo en B6

PRÁCTICA PROFESIONAL PROPUESTA 2

Herramientas

- Ordenador con TIA PORTAL preinstalado

Material

- PLC (S7-1200 o S7-300)
- Cable de programación

Precauciones

- Realizar y comprobar el programa del automatismo de un taladro semiautomático
- Utilizar marcas para diferenciar los distintos movimientos del dispositivo.

Taladro con cargador de piezas en KOP

Objetivo

- Identificar los diferentes elementos de entrada y de salida del proceso y asociarlos a variables del PLC.
- Utilizar flancos para evitar señales permanentes.

Funcionamiento

- Deben utilizarse señales de flanco en aquellos finales de carrera que producen señales permanentes e impiden el correcto funcionamiento secuencial del proceso.

Desarrollo

Desarrolla y prueba el programa del taladro de la figura cuyo funcionamiento es el siguiente:

1. El taladro debe estar en la posición de reposo, que es la que se muestra en la figura.
2. Al accionar el pulsador de marcha (S1), el cargador (Q1) desplaza la pieza hasta ubicarla debajo del taladro (S5).
3. Cuando se acciona el final de carrera S5, el taladro baja (Q2) a la vez que gira la broca (Q5).
4. Una vez taladrada la pieza por completo, se acciona el final de carrera S4.
5. El taladro se eleva (Q3) hasta tocar el final de carrera superior (S3). El movimiento de subida tiene que hacerse con la broca girando (Q5).
6. Una vez que el taladro está en la parte superior, el cargador retira la piza hasta que la posiciona en el punto inicial (S6).

Es importante utilizar señales de flanco para evitar las señales permanentes de algunos finales de carrera, ya que de lo contrario el proceso funcionará de forma anómala.

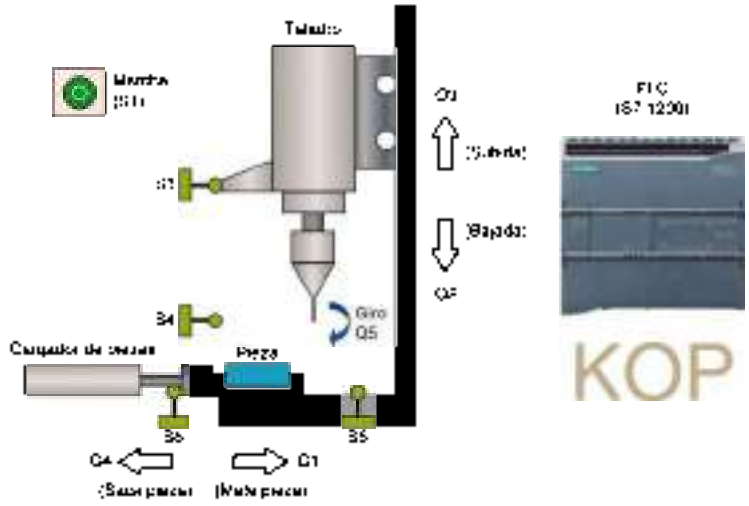
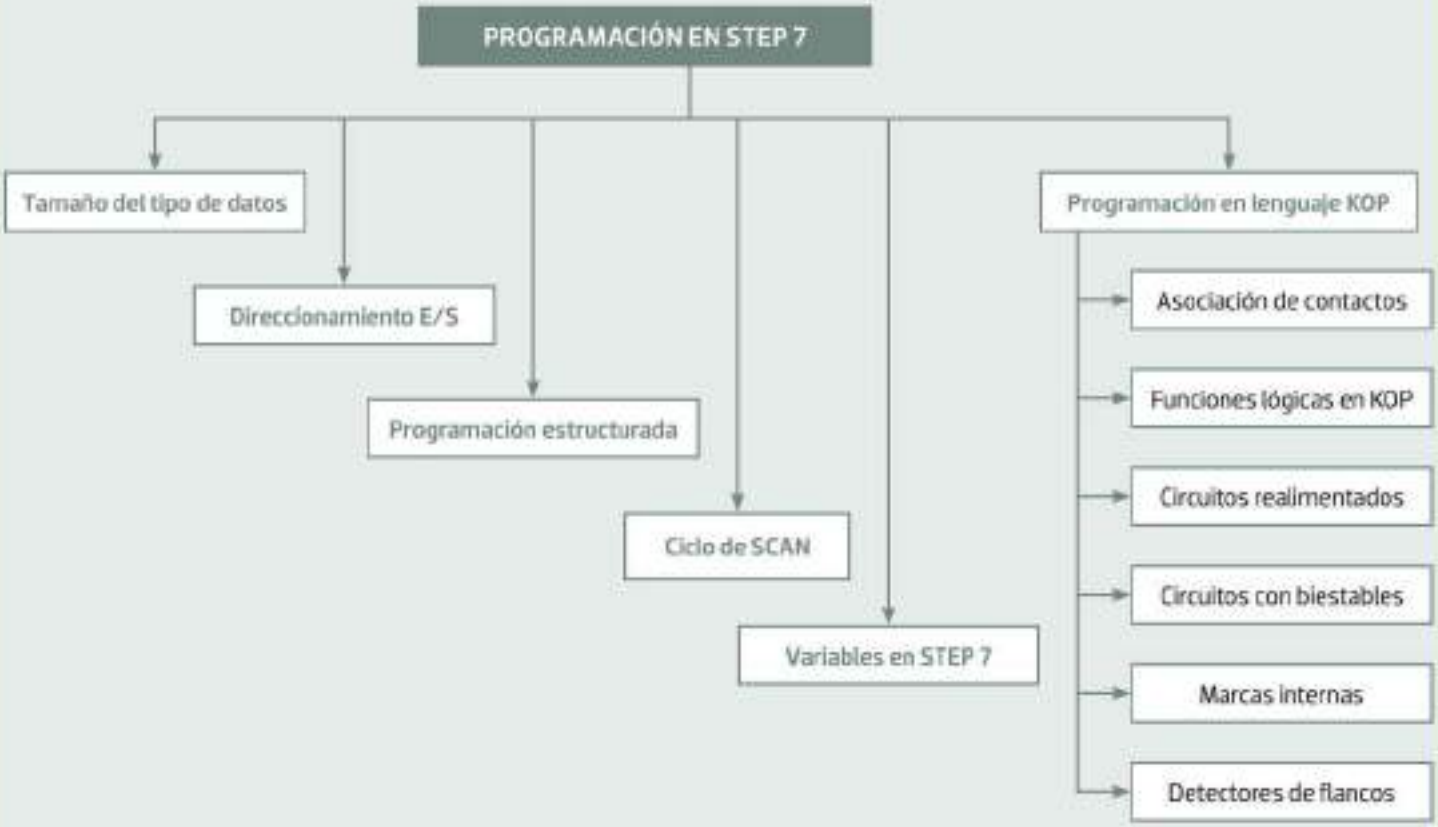


Figura 4.66. Taladro con cargador de piezas.

EN RESUMEN


PROGRAMACIÓN EN STEP 7
Tamaño del tipo de datos
Direccionamiento E/S
Programación estructurada
Ciclo de SCAN
Variables en STEP 7
Programación en lenguaje KOP
Asociación de contactos
Funciones lógicas en KOP
Circuitos realimentados
Circuitos con biestables
Marcas internas
Detectores de flancos

5 Programación en STEP 7 (II)

Vamos a conocer...

1. Temporizadores
2. Contadores
3. Operaciones de comparación
4. Marca de ciclo (*clock memory*)
5. OB de arranque o *STARTUP* (OB100)

PRÁCTICA PROFESIONAL RESUELTA

Programación de una empaquetadora de piezas

PRÁCTICAS PROFESIONALES PROPUESTAS

1. Llenado de recipientes mediante dosificadores
2. Agrupación de productos mediante cintas de transferencia

Y al finalizar esta unidad...

- Conocerás las diferentes formas de utilizar los temporizadores y contadores en STEP 7.
- Compararás datos de tiempo y de cómputo para realizar asignaciones lógicas.
- Utilizarás bloques de programación con DB a instancia.
- Configurarás la marca de ciclo para realizar operaciones de generación de pulsos.
- Crearás el OB de arranque para ejecutar acciones cuando arranque el programa del PLC.
- Resolverás varios procesos industriales utilizando todos los elementos de programación estudiados en la unidad.

1. Temporizadores

En STEP 7 se pueden utilizar dos tipos de temporizadores:

- Temporizadores *hardware* (SIMATIC).
- Temporizadores *software* (IEC).

1.1. Temporizadores *hardware* (SIMATIC)

También denominados *temporizadores S7* o *SIMATIC*, se han utilizado desde los primeros PLC que han empleado el STEP 7 como forma de programación. Están disponibles en todos los modelos de PLC de Siemens (S7-300, S7-400 y S7-1500) excepto en los S7-1200, que solo aceptan temporizadores IEC.

Son temporizadores que tienen reservada un área de memoria determinada y su tamaño depende del modelo de CPU. Así, por ejemplo, un S7-313C tiene 256 temporizadores y una CPU S7-319 tiene 2048.

En su direccionamiento absoluto, estos temporizadores se identifican con la letra T seguida del número del área de memoria. Por ejemplo: %T1, %T2, %T3, etc.

Hay cinco tipos de temporizadores SIMATIC:

- S_PULSE: parametrizar y arrancar temporizador como impulso.
- S_PEXT: parametrizar y arrancar temporizador como impulso prolongado.
- S_ODT: parametrizar y arrancar temporizador como retardo a la conexión sin memoria.
- S_ODTS: parametrizar y arrancar temporizador como retardo a la conexión con memoria.
- S_OFFDT: parametrizar y arrancar temporizador como retardo a la desconexión.

Todos ellos se representan de forma gráficacon el mismo tipo de bloque, el cual está formado por los siguientes parámetros:

- %: operando absoluto del temporizador.
- "": nombre simbólico.
- S: entrada de habilitación o activación.
- R: entrada de RESET o puesta a cero.
- TV: valor de tiempo predeterminado.
- Q: salida.
- BI: valor de tiempo actual codificado en binario.
- BCD: valor de tiempo actual codificado en BCD.

El valor de preselección de tiempo (TV) se escribe en el formato denominado *S5TIME* (en abreviatura, S5T) y se puede dar en horas, minutos, segundos y milisegundos, cuya forma es la siguiente: S5T#xHxMxSxMS.

En la siguiente tabla, podemos ver algunos ejemplos de diferentes valores de preselección de tiempo en formato *S5TIME*:

3 segundos	S5T#3S
1 minuto 10 segundos	S5T#1M10S
1 hora 6 minutos	S5T#1H6M
100 milisegundos	S5T#100MS

Figura 5.1. Carpeta de instrucciones de tiempo para un S7-300 en TIA PORTAL.

IEC frente a SIMATIC

Aunque la tendencia actual es a utilizar temporizadores y contadores IEC, por las ventajas que estos presentan frente a los de tipo SIMATIC, es necesario conocer los segundos, ya que hay cientos de instalaciones que los tienen implementados.

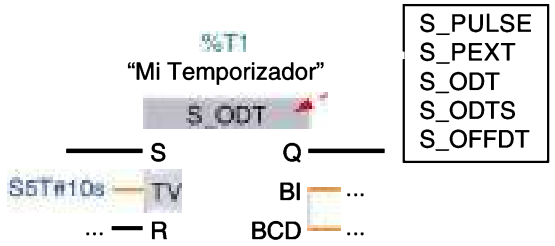


Figura 5.2. Bloque de temporizador SIMATIC.

Saber más

Los temporizadores SIMATIC comienzan a temporizar desde el valor máximo configurado en PV hasta que se ponen a 0. Se puede decir que su valor es descendente.

Temporizadores con retardo

Aquí solamente se utilizarán los temporizadores con retardo a la conexión (S_ODT) y con retardo a la desconexión (S_OFFDT), cuyo funcionamiento es el mismo que el ya estudiado para el relé programable LOGO.

La salida de los temporizadores SIMATIC se puede utilizar de dos formas:

1. Conexión directa de la asignación a la salida Q del bloque.
2. Asociando el operando del temporizador a contactos abiertos, cerrados u otros bloques con entrada en formato de bit.

Aunque la primera forma puede aportar inmediatez por su sencillez, la segunda es mucho más operativa y flexible.

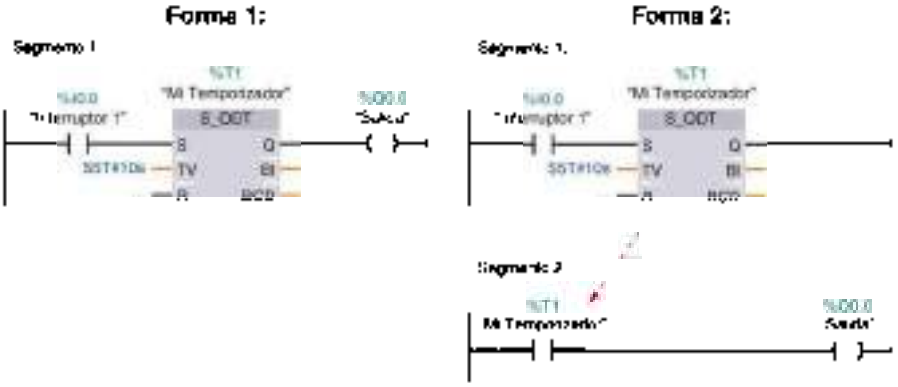


Figura 5.3. Dos formas de usar las salidas de los temporizadores SIMATIC.

Temporización intermitente

En muchas ocasiones es necesario activar una salida de forma intermitente mediante un tren de pulsos. Una forma de hacerlo consiste en utilizar dos temporizadores conectados en cascada, de tal manera que uno asigne el pulso de activación y el otro el de desactivación. Si bien, como se verá más adelante, esta no es la mejor solución para conseguir temporizaciones intermitentes, está justificado su uso cuando se requieren trenes de pulsos personalizados.

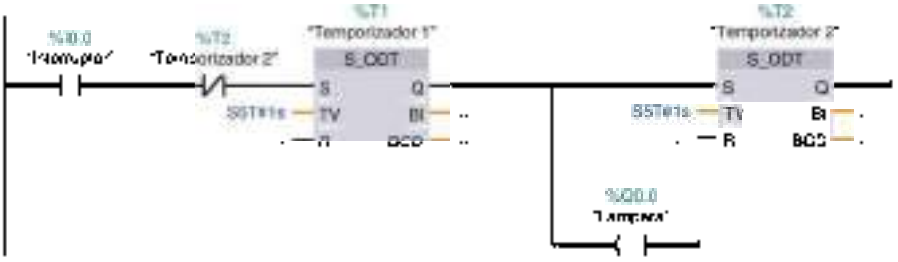


Figura 5.4. Temporizadores para activar salida intermitente.

Actividades

1. Programa el circuito para la maniobra de arranque estrella-triángulo de un motor trifásico, teniendo en cuenta que la conmutación estrella y triángulo se realiza de forma temporizada.
2. Haz que cuatro salidas (Q1, Q2, Q3 y Q4) se activen secuencialmente cada 0,5 segundos. La secuencia debe repetirse cíclicamente siempre que un interruptor, conectado a una entrada del PLC, se encuentre activado.

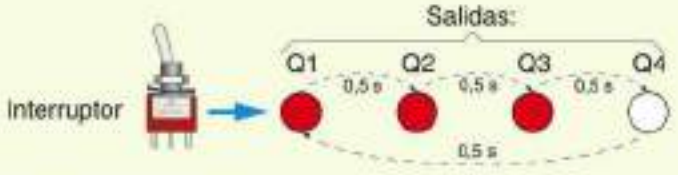


Figura 5.5. Luces en cascada.

1.2. Temporizadores *software* (IEC)

Son temporizadores que no necesitan un direccionamiento físico a una zona de memoria del PLC, por lo que no requieren un direccionamiento absoluto como el realizado para los de tipo SIMATIC.

Están basados en bloques de programación FB, por lo que siempre requieren un bloque de datos a instancia (DB). El nombre asignado a este bloque de datos es el identificador del temporizador. TIA PORTAL asigna automáticamente un nombre simbólico al temporizador basándose en el patrón: «IEC_Counter_0_DB». Si bien el uso de esa denominación es adecuada para la mayoría de las aplicaciones, es aconsejable que el usuario asigne nombres representativos que se puedan reconocer fácilmente entre otros elementos de programación (por ejemplo: «DB_Temp1», «DB_Temp_Motor», etc.).

Los temporizadores IEC pueden ser de cuatro tipos:

- TP: temporizador de impulsos.
- TON: temporizador con retardo a la conexión.
- TOF: temporizador con retardo a la desconexión.
- TONR: temporizador con retardo a la conexión con memoria.

Todos los bloques presentan el mismo aspecto, con las mismas entradas y salidas, excepto el TONR, que tiene una entrada adicional para el RESET.

- %: direccionamiento del DB a instancia.
- "": nombre simbólico para el DF a instancia.
- IN: entrada de activación.
- R: RESET (solo en TONR).
- PT: tiempo de preselección en formato TIME.
- Q: salida.
- ET: dato de salida del tiempo transcurrido.

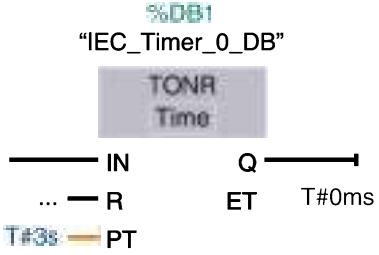


Figura 5.7. Bloque temporizadores IEC.

Los temporizadores IEC utilizan el formato de tiempo TIME (T#), en el que las unidades de tiempo se pueden expresar en días, horas, minutos, segundos y milisegundos, de forma similar a como ya se ha estudiado para los temporizadores SIMATIC.

Ejemplos de configuración de tiempos:

4 segundos	T#4S
2 minutos 10 segundos	T#2M10S
1 hora 19 minutos	T#1H19M
500 milisegundos	T#500MS
5 días 12 horas	T#5D21H

De igual forma que los temporizadores S7, el uso de salida de los temporizadores IEC se puede hacer de dos formas:

1. Conexión directa de la asignación a la salida Q del bloque.
2. Asociando a contactos abiertos, cerrados y, en general, a cualquier función que acepte el dato entrada en formato de bit, el parámetro Q del bloque DB a instancia. Para ello se utiliza la siguiente sintaxis:

«Nombre del DB.Q»

Estándar IEC

Los temporizadores *software* se adaptan al estándar IEC que pretende unificar los lenguajes de programación.

Los autómatas programables S7-1200 solamente admiten este tipo de temporizadores.

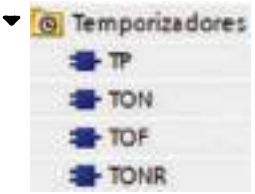


Figura 5.6. Temporizadores IEC de los S7-1200.

Modelos S7-1200

Debes saber que muchas de las funciones denominadas SIMATIC no están disponibles en los autómatas programables S7-1200.

Ambas son compatibles entre sí y pueden coexistir en un mismo programa. Por su flexibilidad y potencia en el momento de programar, aquí se aconseja utilizar la segunda forma.

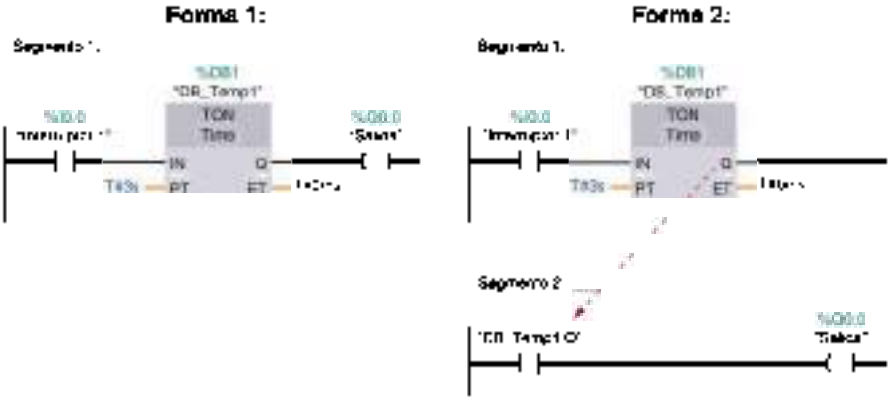


Figura 5.8. Dos formas de usar las salidas de los temporizadores IEC

1.2.1. Temporizador de impulsos (TP)

Activa la salida por un tiempo programado (PT) con solo aplicar un impulso a su entrada (IN) por flanco ascendente. La detección de nuevos impulsos en la entrada, cuando el temporizador está funcionamiento, no afecta al estado de la salida. Permite agregar un impulso con una duración determinada.

Valor ascendente

Los temporizadores IEC, al contrario que los de tipo SIMATIC, comienzan a temporizar desde el valor 0 hasta conseguir el valor máximo asignado a la entrada PT. Se puede decir que su valor es ascendente.

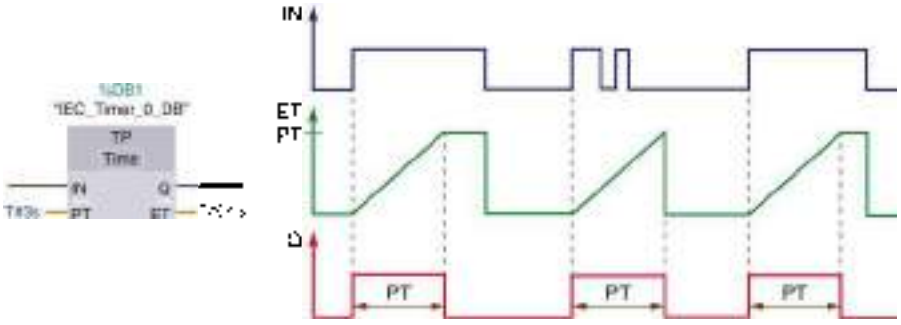


Figura 5.9. Bloque y cronograma de funcionamiento de un temporizador TP

1.2.2. Temporizador con retardo a la conexión (TON)

Una vez activada la entrada del temporizador (IN), la salida (Q) se activa después del tiempo programado (PT). La señal de entrada de activación debe mantenerse para que concluya la activación de la salida. Cuando la señal de dicha entrada se desactiva, el tiempo se resetea y la salida, si estaba activada, se pone a 0.

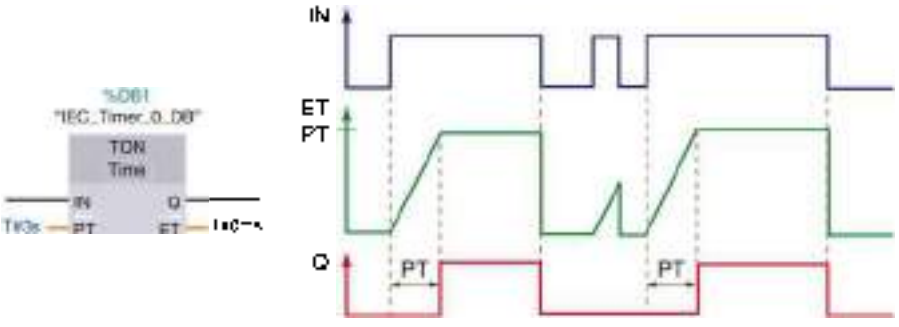


Figura 5.10. Bloque y cronograma de funcionamiento de un temporizador TON.

Actividades

3. Utilizando una entrada y una salida del PLC, comprueba el funcionamiento de los temporizadores TP y TON.

1.2.3. Temporizador con retardo a la desconexión (TOF)

La salida (Q) se activa cuando se habilita la entrada IN del temporizador. Si dicha entrada se mantiene a «1» lógico, la salida también lo hace de forma permanente. La temporización comienza cuando el valor en la entrada IN pasa de «1» a «0», desactivando la salida una vez transcurrido el tiempo preseleccionado (PT).

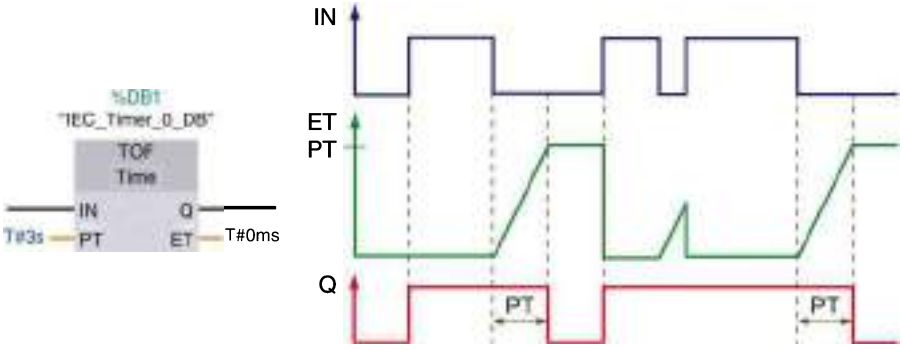


Figura 5.12. Bloque y cronograma de funcionamiento de un temporizador TOF.

Asignación

La asignación [RT] permite resetear cualquier tipo de temporizador IEC. Para ello, solo hay que hacer referencia al DB a instancia asociado al temporizador.

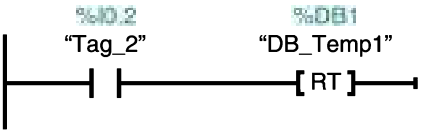


Figura 5.11. Bobina para resetear temporizadores.

1.2.4. Temporizador con retardo a la conexión con memoria (TONR)

El funcionamiento es igual al de retardo a la conexión, pero con una diferencia: si se desactiva la entrada IN, la temporización se detiene pero el tiempo no se pone a cero, sino que se mantiene el valor en el que se quedó. La entrada RESET permite desactivar la temporización una vez ha sido lanzada y desactivar la salida, si es que la temporización había concluido.

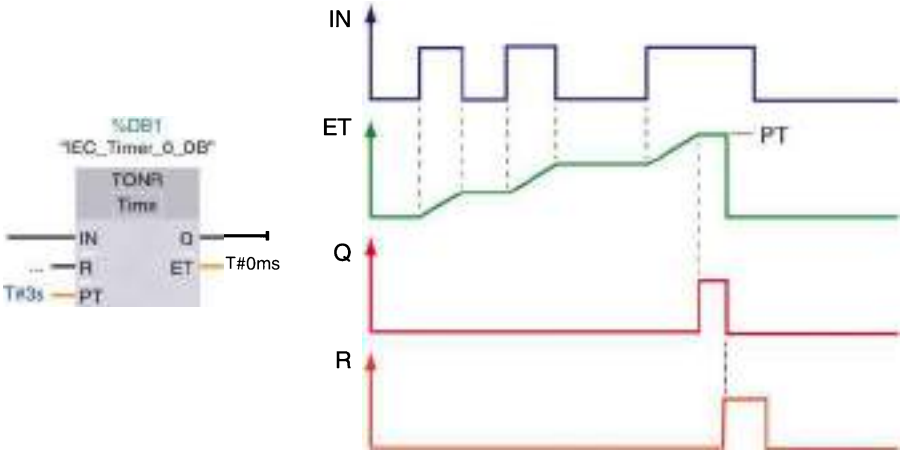


Figura 5.13. Bloque y cronograma de funcionamiento de un temporizador TONR.

Actividades

4. Utilizando una entrada y una salida del PLC, comprueba el funcionamiento del temporizador TOF. Haz lo mismo para el temporizador TONR, utilizando una entrada adicional para el RESET.
5. Programa el control de un semáforo que dispone de lámparas para vehículos y para peatones.
   - En el mismo instante que un peatón acciona el pulsador situado en el semáforo, se activa el amarillo para vehículos, que permanece en esa situación durante 3 segundos.
   - Finalizado este estado, el semáforo de vehículos pasa a rojo y el de peatones a verde, y permanece en esa situación durante 10 segundos.
   - Una vez transcurrido este tiempo, el semáforo debe volver al estado inicial: verde para vehículos y rojo para peatones.

   Durante el tiempo de funcionamiento de la secuencia, debes evitar que cualquier activación sobre el pulsador reinicie el ciclo.

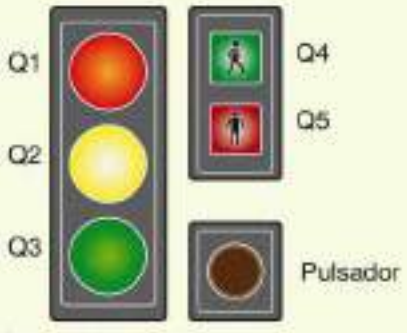


Figura 5.14. Semáforo.

2. Contadores

De igual forma que ocurre con los temporizadores, los contadores pueden ser de dos tipos:

- Contadores *hardware* (SIMATIC).
- Contadores *software* (IEC).

Los primeros se pueden utilizar en todas las gamas de PLC de la firma Siemens, excepto en los S7-1200, que solamente acepta contadores IEC.

2.1. Contadores SIMATIC

Identificador Z

La nemotecnia alemana (SIMATIC) utiliza el identificador Z para los contadores.

Están presentes entre las funciones de programación desde los primeros modelos de S7-300 y S7-400, y en la actualidad también se pueden utilizar en los S7-1500.

Este tipo de contadores tiene reservada un área de memoria determinada y, de igual forma que ocurre con los temporizadores SIMATIC, su tamaño depende del modelo de CPU.

El direccionamiento absoluto de estos temporizadores se hace con el símbolo % más la letra C, seguida del número que el contador tiene en la zona de memoria.

Pueden ser de tres tipos:

- S_CU: contador.
- S_CD: descontador.
- S_CUD: contador/descontador.

Se representan en formato de bloque con una serie de parámetros de entrada y de salida. A continuación, se toma como ejemplo el bloque del contador/descontador (S_CUD), ya que sirve como modelo para los otros dos (S_CU y S_CD).

- %: direccionamiento absoluto.
- "": direccionamiento simbólico.
- CU: entrada para incrementar.
- CD: entrada para decrementar.
- S: entrada para ajustar el contador a valor PV.
- PV: valor de contaje predeterminado (formato C#).
- R: entrada de RESET o puesta a cero.
- Q: salida del contador.
- CV: valor del contaje actual (hexadecimal).
- CV_BCD: valor del contaje actual codificado en BCD.

Figura 5.15. Contador SIMATIC.

El valor del contador se almacena en los parámetros de salida CV y CV_BCD. Estos se pueden utilizar para escribir en variables numéricas y, posteriormente, utilizarse con operaciones de comparación para controlar asignaciones en formato de bit.

La salida Q se pone a 1 cada vez que el valor del contador es diferente de 0.

Actividades

6. Programa el siguiente segmento y comprueba su funcionamiento activando en modo de observación on line de TIA PORTAL.

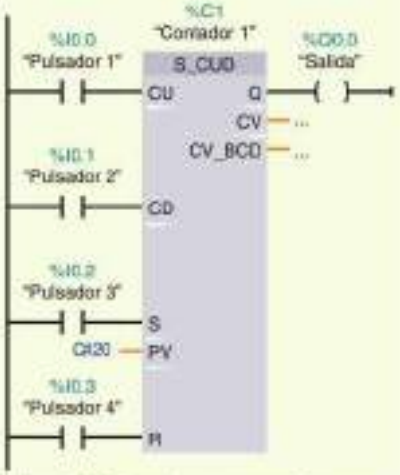


Figura 5.16. Ejemplo para comprobar contador S_CUD.

2.2. Contadores IEC

A diferencia con los contadores SIMATIC, los de tipo IEC no se direccionan a un área de memoria del PLC.

Están basados en bloques de programación FB, por lo que siempre requieren un bloque de datos a instancia (DB). El nombre asignado a este bloque de datos es el identificador del contador. De igual forma que con los temporizadores, TIA PORTAL asigna automáticamente un nombre simbólico al contador basándose en el patrón: «IEC_Counter_0_DB». Si bien el uso de esa denominación es adecuado para la mayoría de las aplicaciones, es aconsejable que el usuario asigne nombres representativos que se puedan reconocer fácilmente entre otros elementos de programación. Por ejemplo: «DB_Contador1», «DB_Cnt_botellas», etc.

De igual forma que en los contadores SIMATIC, hay tres tipos de contadores IEC:

- CTU: contador.
- CTD: descontador.
- CTUD: contador/descontador.

Solamente se describe el CTUD, ya que incluye los otros dos.

- %: direccionamiento absoluto al (DB).
- "": direccionamiento simbólico del DB.
- CU: entrada para incrementar.
- CD: entrada para decrementar.
- R: entrada para poner a cero (resetear).
- LD: cargar el valor de preselección (PV) en el contador.
- PV: valor de preselección al que se activa la salida QU.
- QU: salida que se activa cuando el contador alcanza el valor de preselección PV.
- QD: salida digital que se activa siempre que el contador tengaun valor diferente de «0».
- CF: salida donde se almacena el dato numérico del contador.

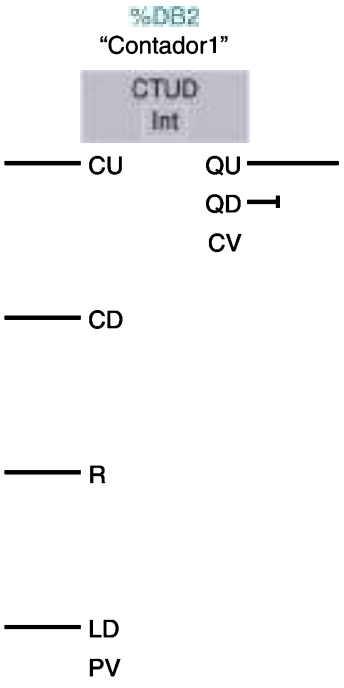


Figura 5.17. Contador CTUD.

En lo básico, este contador funciona de forma similar al de tipo SIMATIC. Sin embargo, presenta una gran diferencia con respecto al anterior, y es que dispone de una salida QU que se activa cuando el valor del contador es igual o superior al número configurado en PV.

De igual forma que los temporizadores IEC, la salida de este tipo de contadores se puede utilizar de dos maneras:

1. Conexionando de forma directa la asignación a la salida QU del bloque.
2. Asociando a contactos abiertos, cerrados y, en general, a cualquier función que acepte el dato entrada en formato de bit el parámetro Q del bloque DB a instancia. Para ello se utiliza la siguiente sintaxis:

«Nombre del DB.QU»

Ambas son compatibles entre sí y pueden coexistir en un mismo programa.

Número de bloques DB

El número máximo de bloques de datos DB que se puede programar en un PLC depende del modelo de CPU utilizada, y está especificado en su hoja de características.

Actividades

7. Comprueba el funcionamiento de un contador CTUD:

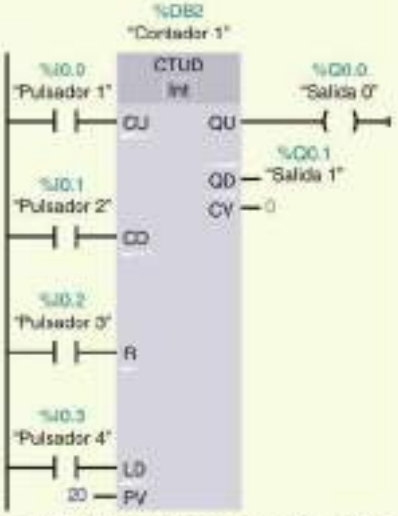


Figura 5.18. Ejemplo para comprobar contador CTUD.

3. Operaciones de comparación

Permiten comparar valores de dos variables, o de una variable y una constante, y, si se cumple la evaluación, obtener un resultado lógico (verdadero o falso).

Las comparaciones en STEP 7 se representan como un contacto en cuyo interior se encuentran la operación de comparación y el tipo de datos de las variables que se van a evaluar. En la parte superior o inferior se escriben los operandos de las variables que se van a comparar.

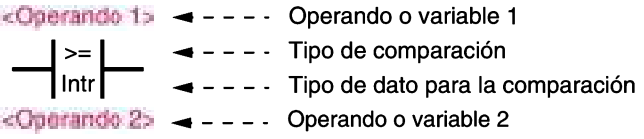


Figura 5.19. Comparación.

Posibles comparaciones en STEP 7

=	Igual que	>=	Mayor o igual que
<=	Menor o igual que	<	Menor que
>	Mayor que		

Las operaciones de comparación se pueden insertar en los segmentos, asociándolos a otros contactos en serie o en paralelo.

3.1. Comparación de contadores

Los contadores, tanto los SIMATIC como los IEC, almacenan su valor en las variables CV de salida de las que dispone el bloque.

En los contadores SIMATIC es necesario crear previamente una variable en formato de entero y posteriormente asignársela a la salida CV. Dicha variable se utilizará posteriormente en las operaciones de comparación.

Sin embargo, los contadores IEC no requieren crear dicha variable, ya que es posible leer el valor de CV directamente en el operando del comparador.

Ejemplos

A continuación se muestra un ejemplo de cómo comparar los valores almacenados en dos contadores (uno SIMATIC y otro IEC) para activar salidas digitales si las evaluaciones se cumplen (en este caso, igual que y mayor o igual que). Prográmalo y comprueba su funcionamiento.

Ejemplo con contador SIMATIC

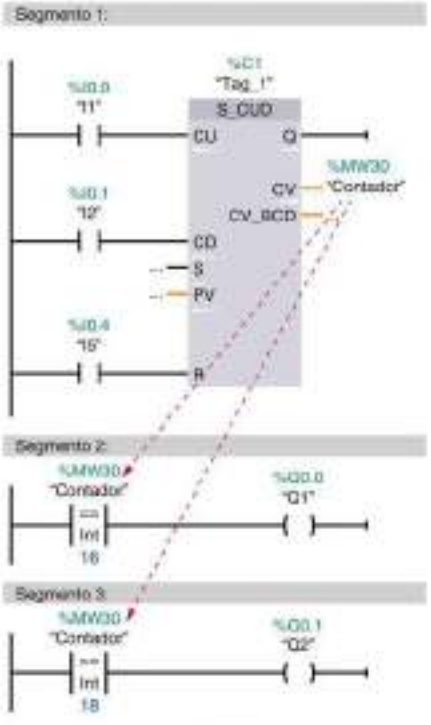


Figura 5.20. Ejemplo con contador SIMATIC.

Ejemplo con contador IEC

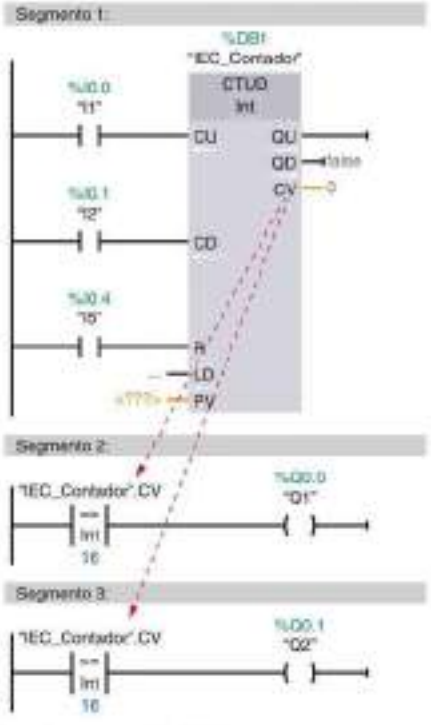


Figura 5.21. Ejemplo con contador IEC.

3.2. Comparación de tiempos

Las operaciones de comparación también se pueden utilizar para comparar los tiempos de evolución de los temporizadores. Esto permite que, con un solo temporizador, se puedan ejecutar asignaciones lógicas a diferentes valores de tiempo.

La operación de comparación con el formato S5TIME no es aceptada por los S7-300. Solamente es posible realizar esta operación con los modelos S7-1500. Sin embargo, la comparación con el formato TIME es aceptada por todos los equipos que trabajan con temporizadores IEC (S7-300. S7-1200 y S7-1500).

Actividades

8. Utilizando un PLC físico, o el simulador correspondiente, comprueba el funcionamiento del programa mostrado en la figura para activar varias salidas a diferentes tiempos utilizando un solo temporizador. Debes tener en cuenta que los temporizadores SIMATIC comienzan en el valor máximo de tiempo (TV) hasta que alcanzan el valor 0, y los IEC lo hacen al contario, comienzan a temporizar desde 0 hasta que alcanzan el valor de preselección (PT). En los temporizadores SIMATIC es necesario crear una variable en formato S5TIME, que en el ejemplo está direccionada a %MW20. En los temporizadores IEC, el valor del tiempo en formato TIME se puede tomar de la salida ET del bloque de datos.

Ejemplo con temporizador SIMATIC (Solo S7-1500)

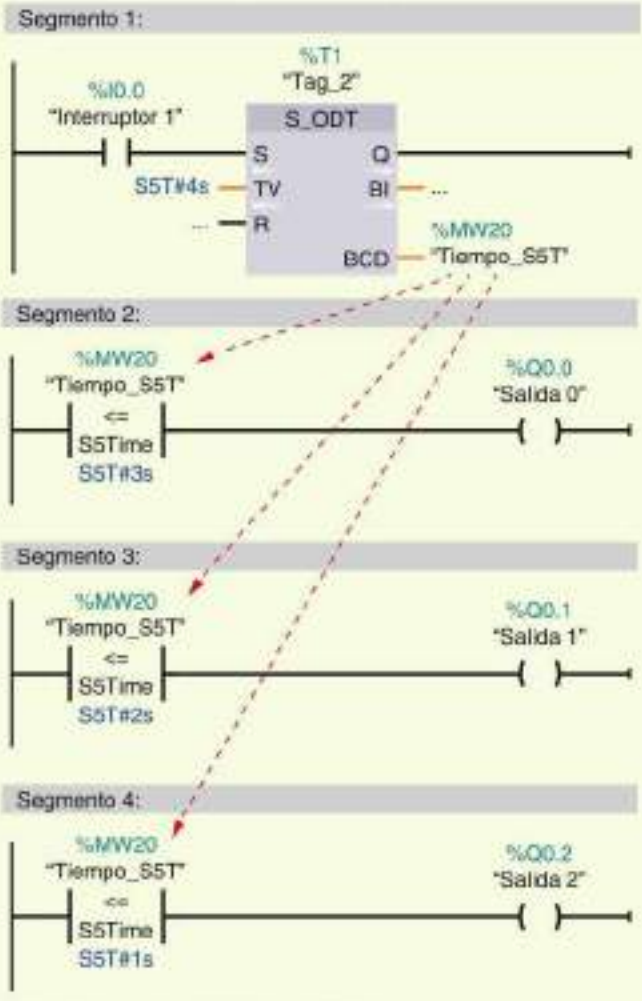


Figura 5.22. Comparación de tiempos S5TIME.

Ejemplo con temporizador IEC (S7-300, S7-400, S7-1200 y S7-1500)

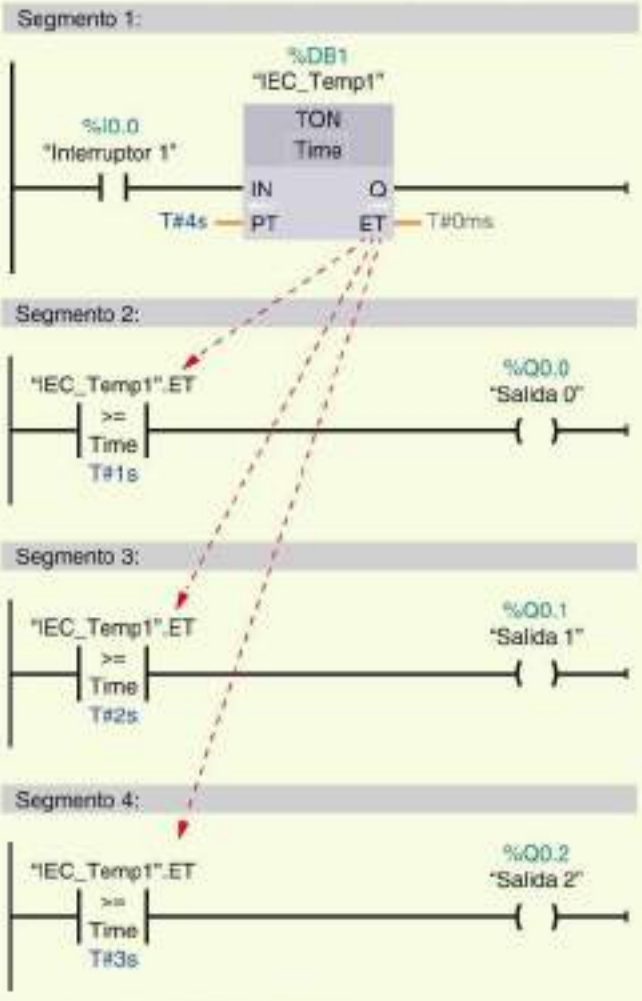


Figura 5.23. Comparación de tiempos TIME.

3.3. Rangos con comparaciones

Permiten controlar asignaciones digitales (booleanas) si un valor numérico está dentro de un rango.

En STEP 7 los rangos se pueden hacer de dos formas:

- Asociando, en serie o en paralelo, operaciones de comparación.
- Utilizando funciones específicas de rango.

3.3.1. Asociación de operaciones de comparación

Se pueden utilizar como si de contactos se tratasen, haciendo agrupaciones en serie o en paralelo.

Evaluar valor dentro de un rango

Se conectan en serie dos comparaciones, como se muestra en la figura. En este caso, si el valor del número (en el ejemplo, un contador) está dentro del rango (entre 5 y 12 en el ejemplo), la salida se activa.

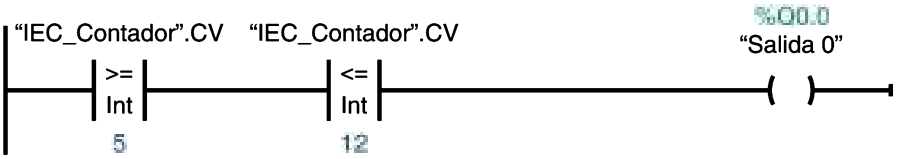


Figura 5.24. Evaluación dentro de un rango.

Evaluar fuera de un rango

Es el caso contrario al anterior. Si el número que se va a evaluar está fuera del rango, la salida se activa.

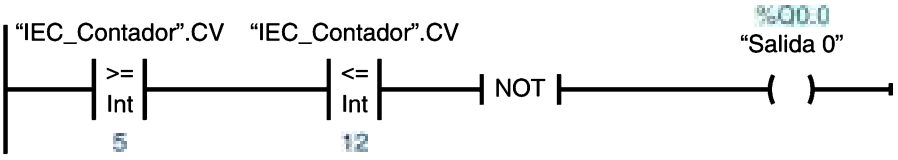


Figura 5.25. Evaluación fuera de un rango.

Evaluación de varios rangos

Es posible evaluar varios rangos al mismo tiempo, para controlar una misma asignación, asociándolos en paralelo, como se muestra en la figura. En este caso, la salida se activa si el valor del contador está entre 5 y 10 o entre 15 y 20.

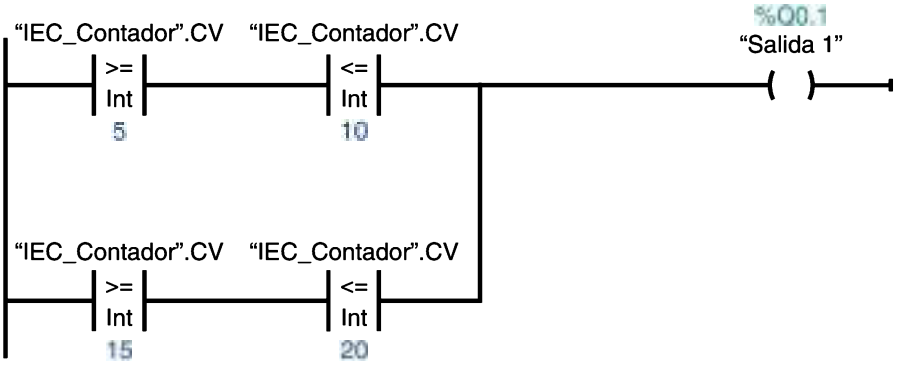


Figura 5.26. Evaluación dentro de varios rangos.

Actividades

9. Comprueba con un PLC todos los ejemplos mostrados en esta página para evaluar valores dentro o fuera de un rango.

3.3.2. Funciones de rango

Son funciones de programación que permiten evaluar números dentro o fuera de un rango utilizando un bloque de programación diseñado para tal fin.

Estos bloques solamente están disponibles para los S7-1200 y S7-1500 y son de dos tipos:

Bloque de evaluación dentro de un rango *(IN_RANGE)*

La salida se activa cuando el valor evaluado en VAL, en este caso el de un contador, está dentro de los límites MIN y MAX del rango.

Figura 5.27. Bloque para evaluar un número dentro de un rango.

Bloque de evaluación fuera de un rango *(OUT_RANGE)*

Funciona al contrario que el ejemplo anterior. En este caso, la salida se activa cuando el valor evaluado en VAL está fuera del rango.

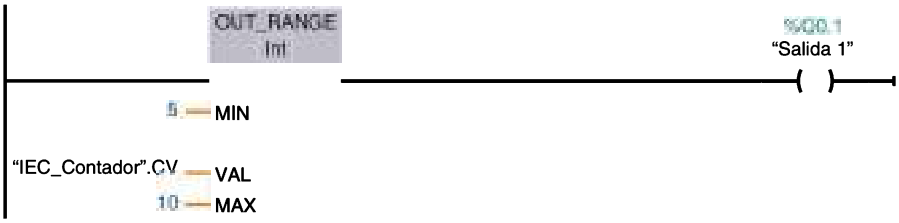


Figura 5.28. Bloque para evaluar un número fuera de un rango.

Ejemplo

De igual forma que las operaciones de comparación, los bloques de evaluación de rangos se pueden asociar en serie y en paralelo entre sí para establecer condiciones de funcionamiento más complejas. En el ejemplo de la figura, la salida se activa si el valor del contador se encuentra entre 5 y 10 o entre 15 y 20.

Figura 5.29. Asociación paralela de dos bloques de evaluación de rangos.

Vocabulary

- Temporizador: timer.
- Tiempo: time.
- Contador: counter.
- Conexión: on.
- Desconexión: off.
- Negación: NOT logic inverter.
- Campo: field.
- Flanco positivo: positive edge.
- Flanco negativo: negative edge.
- Reloj: clock.
- Flujo: flow.
- Etiqueta: tag.
- Retardo: delay.
- Tiempo transcurrido: elapsed time.
- Contar: count-up.
- Descontar: count-down.
- Preestablecido: preset.

Actividades

10. Comprueba con un S7-1200 o un S7-1500 todos los ejemplos mostrados en esta página del uso de los bloques de rango.

4. Marca de ciclo *(clock memory)*

Las marcas de ciclo permiten generar trenes de pulsos simétricos en función de unos patrones preestablecidos en un *byte* diseñado para tal fin.

El período y la frecuencia de los bits de la marca de ciclo son fijos y se corresponden con lo mostrado en la siguiente tabla tabla:

Bit del byte de marca de ciclo	.7	.6	.5	.4	.3	.2	.1	.0
Duración del periodo(s)	2,0	1,6	1,0	0,8	0,5	0,4	0,2	0,1
Frecuencia (Hz)	0,5	0,625	1	1,25	2	2,5	5	10

El *byte* de la marca de ciclo debe ser configurado en la CPU (1). Una vez activado (3), se le debe asignar el número de *byte* que se desea usar para esta función (4). TIA PORTAL asigna nombres simbólicos a cada uno de bits de la marca de ciclo (5), que podrán ser utilizados en los programas como operandos binarios, por ejemplo, en contactos abiertos y cerrados del lenguaje KOP.

Recuerda

Una vez configurada la marca de ciclo, es importante cargar la configuración *hardware* en el dispositivo.

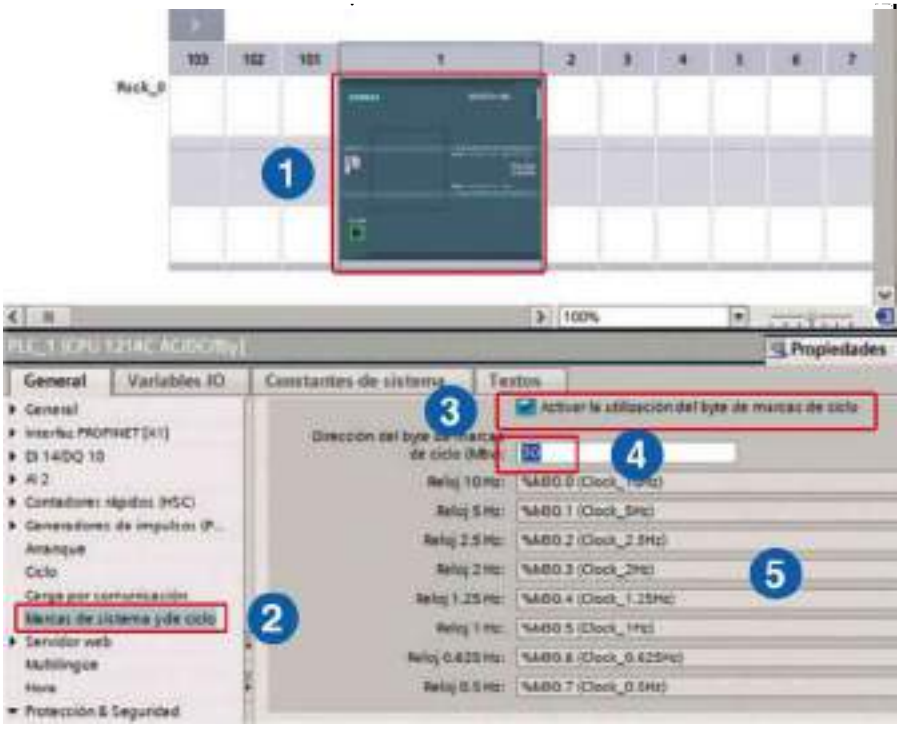


Figura 5.30. Ventana de configuración de la marca de ciclo en una CPU 17-1200.

Actividades

11. Configura la marca de ciclo de un PLC y realiza el siguiente programa para comprobar cómo las salidas digitales se activan de forma intermitente a diferentes frecuencias.

    En el ejemplo de la figura, se ha asignado el byte 30 para la marca de ciclo y se han utilizado los bits M30.3 y M30.5. En el segmento 1, cada vez que se acciona el «Pulsador 1», la «Salida 1» parpadea con una cadencia de 0,5 segundos. En el segmento 2, lo hace la «Salida 2», con el «Pulsador 2», con una cadencia de 1 segundo.

Figura 5.31. Ejemplo de uso de los bits de la marca de ciclo.

5. OB de arranque o *STARTUP* (OB100)

Como ya se estudió en la unidad anterior, en el epígrafe relacionado con el ciclo de SCAN de un autómata programable, el OB de arranque (OB100) se ejecuta una sola vez cuando el autómata pasa de STOP a RUN. Una vez que esto ocurre, se procesa el bloque cíclico OB1 y con él todas las llamadas a otros bloques, si es que se ha programado.

El OB100 se programa de forma similar a otros bloques de programación ya utilizados. Sin embargo, hay que tener en cuenta que, al no ser un bloque cíclico, todo lo que en él se encuentra solamente se ejecuta cuando la CPU pasa de STOP a RUN.

El OB de arranque se puede utilizar para actualizar datos, resetear salidas o marcas, ejecutar acciones de señalización indicando que el PLC ha tenido un arranque en caliente, inicializar secuencias, etc.

Los bloques OB no requieren ser llamados desde otros bloques. Simplemente se crean, se programan y se cargan en la memoria de la CPU.

Recuerda

Se puede decir que el OB de arranque es el equivalente a la marca M8 que utilizamos en el relé programable LOGO.

Figura 5.32. Uso del bloque OB100 en el árbol de programa.

Ejemplo

El siguiente ejemplo muestra cómo funciona el bloque OB100. En él hay un programa que activa con SET la marca M40.0.

En el bloque secuencial OB1 se utiliza un contacto asociado a dicha marca, el cual activa un segmento en el que una salida funciona de forma intermitente con el bit M30.3 de la marca de ciclo.

Así, cuando el autómata pasa de STOP a RUN, la M40.0 se activa y con ella su contacto, y de inmediato la salida parpadea, indicando que el autómata se ha arrancado.

La marca activada en el arranque M40.0 se puede desactivar manualmente mediante el pulsador conectado a la entrada I0.0.

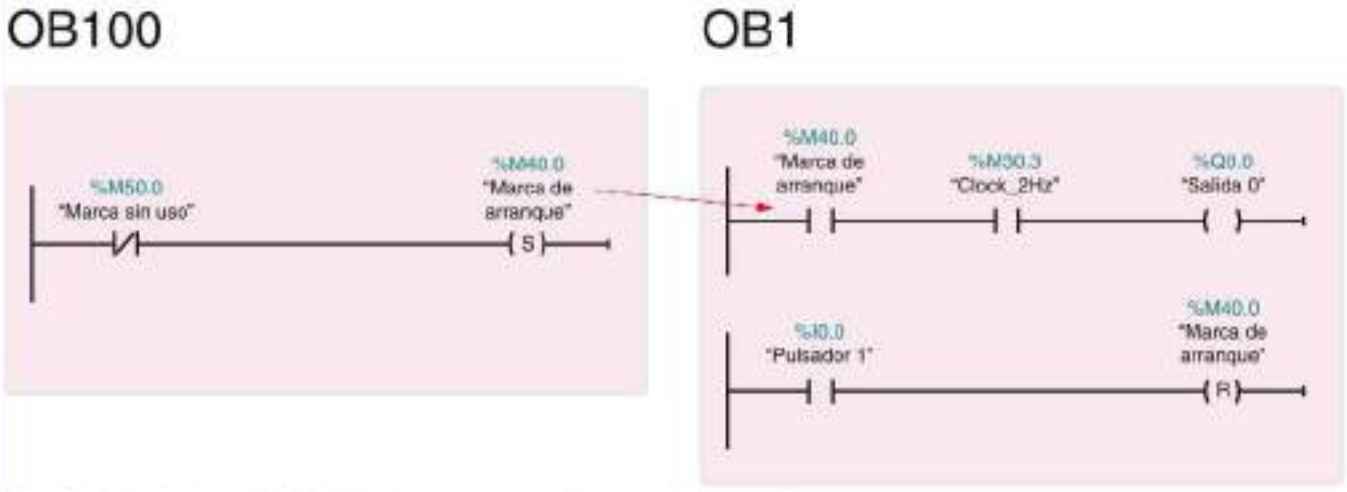


Figura 5.33. Ejemplo de uso del OB100 para activar una marca en el momento del arranque.

Importante:

El contacto de marca M50.0 usado en el OB100 solamente es necesario en los S7-300, ya que no permite conectar salidas sin una combinación lógica previa. En los S7-1200 y S7-1500 se puede, si se desea, prescindir de él.

En este ejemplo se ha utilizado el OB para el programa cíclico, pero se podría haber usado cualquier otro bloque FC o FB para el mismo fin.

PRÁCTICA PROFESIONAL RESUELTA

Herramientas

- Herramientas básicas del electricista
- PC con TIA PORTAL preinstalado

Material

- Un PLC (S7-300 o S7-1200)
- Cable PC-Adapter (si el PLC es un S7-300)
- Latiguillo Ethernet (si el PLC es un S7-1200)

Programación de una empaquetadora de piezas

Objetivo

Utilizar un contador para la toma de decisiones en un proceso de empaquetado.

Precauciones

Hay que tener en cuenta que los movimientos del cilindro se hacen con válvulas monoestables, por lo que solamente se controla eléctricamente el movimiento de extensión del cilindro, ya que el retorno se hace por muelle.

Desarrollo

En esta práctica profesional se muestra el programa para controlar un proceso de empaquetado basado en dos cilindros neumáticos, cuyo funcionamiento es el que se indica a continuación.

El dispositivo cuenta con un cargador de piezas por gravedad donde se encuentran los elementos que hay que empaquetar. El sistema está formado por dos cilindros neumáticos de doble efecto controlados por sendas válvulas monoestables.

La carga de piezas se hace manualmente, de una en una. Cuando se acciona el pulsador S1, el cilindro empujador desplaza una pieza (Q1) hasta el punto de empaquetado. Cuando se han cargado tres piezas, el cilindro empaquetador se extiende (Q2) y las saca del proceso.

Una vez realizado el empaquetado, el proceso comienza de nuevo y permite cargar otras tres piezas mediante el mismo procedimiento.

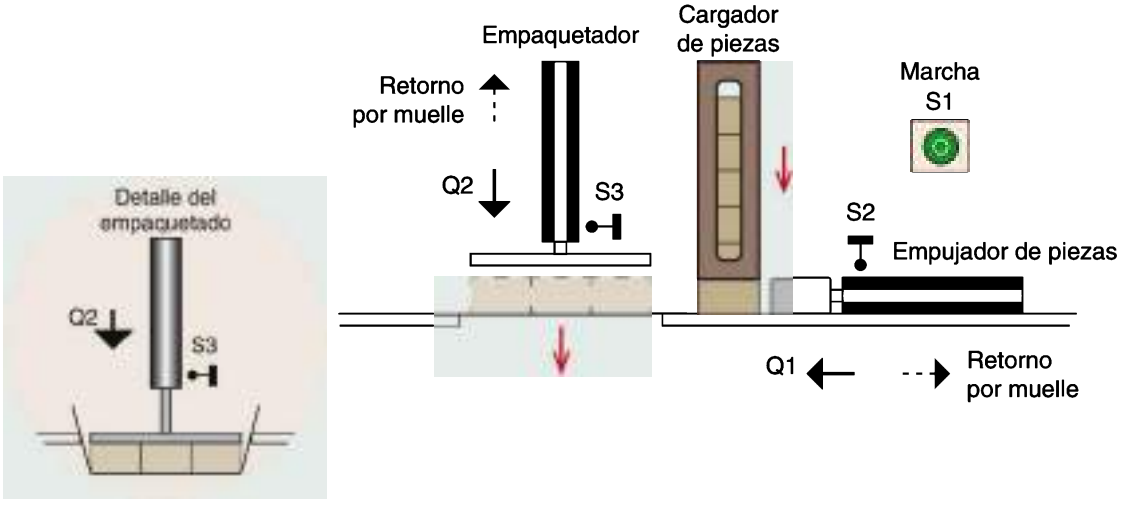


Figura 5.34. Empaquetadora de piezas.

1. Hay que conocer los detalles de funcionamiento del circuito neumático que controla los cilindros del proceso.

   Los dos cilindros están controlados por válvulas monoestables. Es decir, solamente diponen de una bobina que es la que hay que alimentar, a través de las salidas del PLC, para extender el cilindro, ya que el retorno se

hace por un muelle o resorte. Así, el control desde el PLC solamente requiere una salida por cilindro, ya que los movimientos de recogida de ambos se realizan de forma automática al dejar la bobina sin señal.

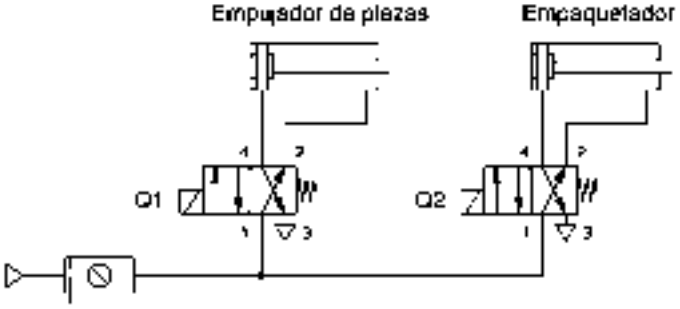


Figura 5.35 Esquema neumático de la empaquetadora de piezas

En la siguiente figura se muestra en detalle el conexionado de uno de los cilindros neumáticos a la electroválvula monoestable

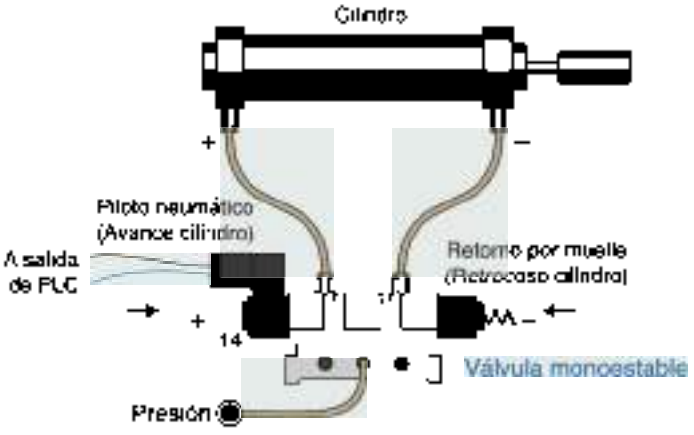


Figura 5.36. Conexión de un cilindro a la válvula monoestable.

2 Una vez entendido el funcionamiento de los actuadores del sistema, hay que elaborar una lista con las variables que se van a utilizar en la programación. Ya que el programa va a necesitar flancos, hay que crear varias marcas para tal fin.

Nombre de variable	Tipo de datos	Dirección	Comentario
Q1	Bool	%Q0.0	Carga pieza
Q2	Bool	%Q0.1	Empaqueta piezas
S1	Bool	%I0.0	Pulsador de marcha
S2	Bool	%I0.1	Final de carrera empujador
S3	Bool	%I0.2	Final de carrera empaquetador
M_Flanco1	Bool	%M0.0	Marca de flanco 1
M_Flanco2	Bool	%M0.1	Marca de flanco 1
M_Flanco3	Bool	%M0.3	Marca de flanco 1

3. Crear un bloque FC para el programa principal.
4. Escribir el programa en el FC que se muestra a continuación, el cual está descrito segmento a segmento.

PRÁCTICA PROFESIONAL RESUELTA

continuación

SEGMENTO 1

Se debe utilizar un contador para conocer el número de piezas que ha desplazado el cilindro empujador. El incremento de este contador se hace cuando se acciona el final de carrera S2, ya que es en ese momento en el que las piezas han sido desplazadas hasta la zona de empaquetado.

El contador se debe poner a cero (RESET) cuando las piezas se han empaquetado. Para ello se utiliza un contacto, con flanco negativo, del final de carrera S3, que es el que detecta que el cilindro se ha extendido.

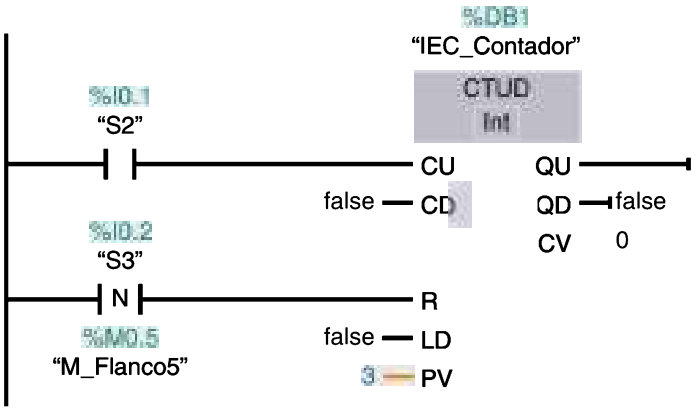


Figura 5.37. Segmento del contador.

SEGMENTO 2

La salida Q1 controla la bobina de la electroválvula del cilindro empujador. Esta se activa con SET cuando se acciona el pulsador de marcha. Se debe evitar que se pueda activar si está en movimiento el cilindro del empaquetador o si el contador ha alcanzado el valor de PV, que en este caso es 3.

SEGMENTO 3

La salida Q1 que extiende el cilindro empujador debe desactivarse cuando este ha depositado la pieza en el tapiz de empaquetado. Para ello, se utiliza el fin de carrea S2, que aquí se ha usado con flanco negativo, aunque no es necesario.

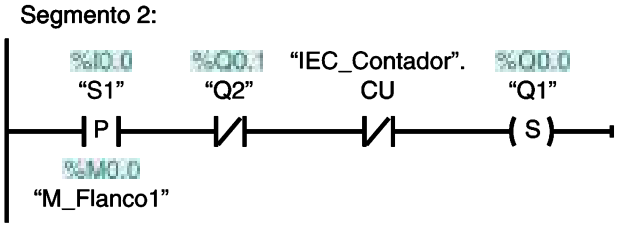


Figura 5.38. Extensión del cilindro empujador.

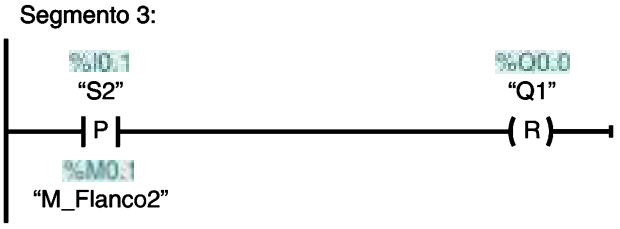


Figura 5.39. Recogida del cilindro empujador.

SEGMENTO 4

En este segmento se evalúa el valor del contador. Si este ha alcanzado el valor de preselección, que en este caso es el número de piezas que se van a empaquetar, el cilindro de empaquetado se extiende con la salida Q2. El uso del flanco en el contacto es opcional.

SEGMENTO 5

Cuando el cilindro de empaquetado está extendido por completo, las piezas pasan a otra zona del proceso y se desactiva Q2. Esto hace que dicho cilindro se recoja por el resorte de la electroválvula. De igual forma que en el segmento anterior, el uso del flanco en el contacto es opcional.

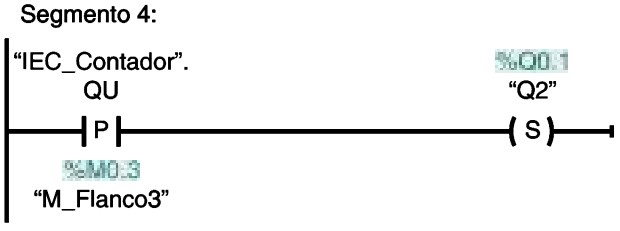


Figura 5.40. Extensión del cilindro de empaquetado.

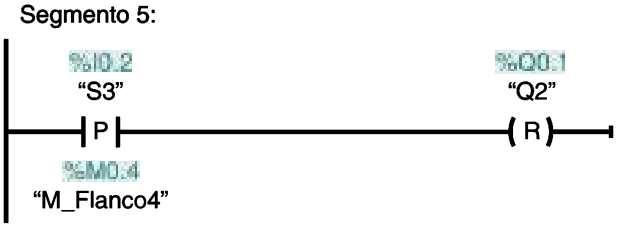


Figura 5.41. Recogida del cilindro de empaquetado.

Mejoras en el funcionamiento

5. Al programa anteriormente comentado se le pueden añadir dos funcionalidades para mejorar su funcionamiento:
   - Parada para detener el proceso en cualquier momento.
   - Vaciado manual de las piezas que se han quedado cargadas después de una parada.

Parada:

Consiste en detener el funcionamiento del proceso. En este caso, como las electroválvulas que controlan los cilindros son monoestables, la parada consiste en parar el movimiento de ambos cilindros reseteando las salidas que controlan las bobinas de dichas electroválvulas. Así, a los segmentos 3 y 5, anteriormente mostrados, hay que añadirles el pulsador que actúa como parada.

Vaciado manual:

Si se hace una parada con el método descrito anteriormente, se pueden quedar algunas piezas sin procesar en el tapiz de empaquetado. En este caso, se ha previsto su vaciado mediante la acción sobre un pulsador conectado a una entrada digital. La acción sobre dicho pulsador debe extender el cilindro del empaquetado y resetear el contador para que el proceso comience desde cero.

6. Añadir las nuevas variables al programa:

Nombre de variable	Tipo de datos	Dirección	Comentario
S4_Vaciar	Bool	%I0.3	Pulsador de vaciado manual
S5_Parada	Bool	%I0.4	Pulsador de parada

7. Añadir los contactos de estas variables a los segmentos correspondientes del programa.

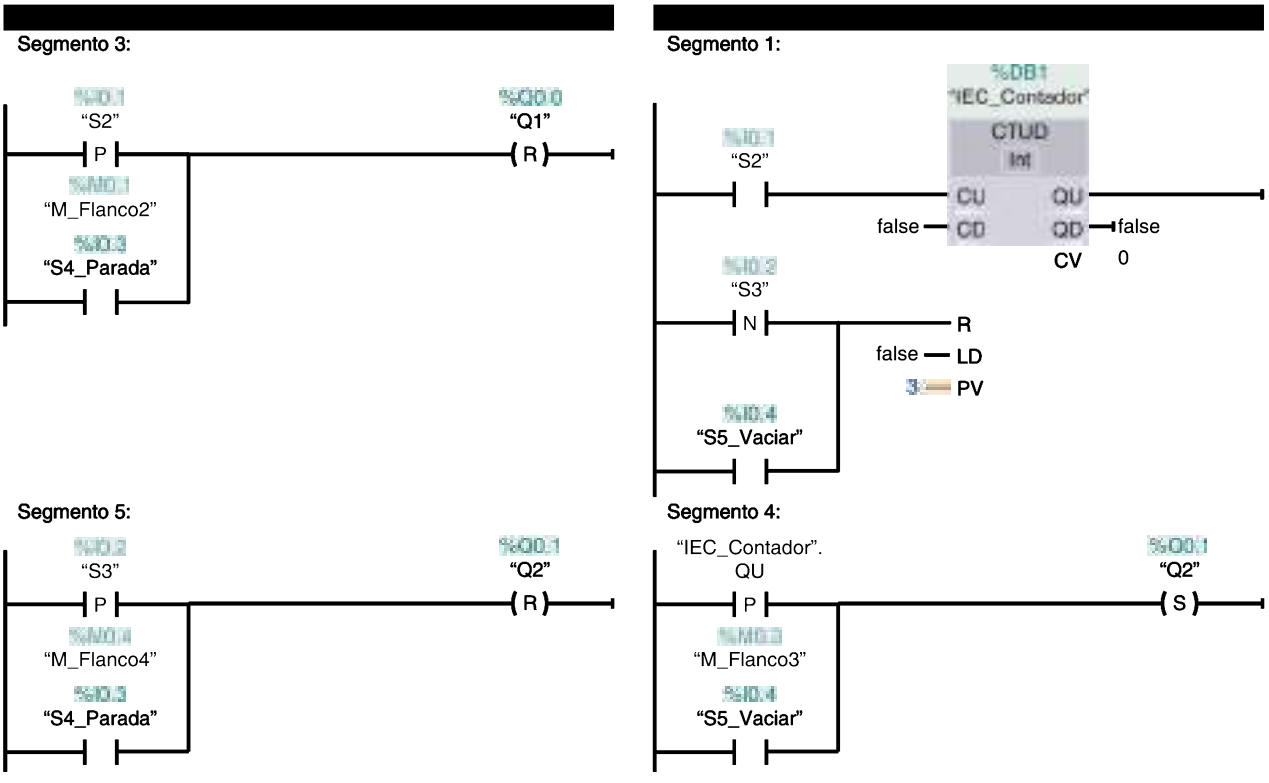


Figura 5.42. Programa del empaquetador de piezas: parada y vaciado manual.

TEST DE EVALUACIÓN

RESUELVE EN TU CUADERNO O BLOC DE NOTAS

1. **Los S7-1200 de Siemens utilizan los temporizadores:**
   a) SIMATIC.
   b) IEC.
   c) Los dos tipos.
   d) Ninguno de ellos.

2. **El formato S5Time se utiliza en:**
   a) Temporizadores SIMATIC.
   b) Temporizadores IEC.
   c) Contadores.
   d) Bloques OB.

3. **Un temporizador TON es un temporizador:**
   a) De impulsos.
   b) Con memoria.
   c) Con retardo a la desconexión.
   d) Con retardo a la conexión.

4. **Si un pulsador está asociado a un flanco positivo, el pulso se detecta:**
   a) Mientras el pulsador está accionado.
   b) Cuando el pulsador no está accionado.
   c) Justo en el momento de accionar el pulsador.
   d) Justo en el momento de soltar el pulsador.

5. **Un contador CTU se utiliza para:**
   a) Contar.
   b) Descontar.
   c) Contar y descontar.
   d) No es un tipo de contador.

6. **La salida Q de un contador SIMATIC:**
   a) Está siempre a 1.
   b) Se activa cuando se alcanza el valor de PV.
   c) Se pone a 1 cuando el contador es diferente de 0.
   d) Se pone a 1 cuando se resetea el contador.

7. **¿Cuál de estas respuestas no es correcta en relación con los bloques DB?**
   a) Contienen programa.
   b) Contienen datos.
   c) Es un bloque de organización.
   d) Los bloques FC siempre lo necesitan.

8. **Si se desea comparar el tiempo de un temporizador IEC, ¿qué salida habrá que utilizar en la función de comparación?**
   a) No es posible.
   b) QU.
   c) QD.
   d) CV.

9. **Si en un programa se utiliza la función IN_RANG y sus valores MAX y MIN son 20 y 10, respectivamente, la salida Q se activará cuando el valor VAL sea:**
   a) 12.
   b) 22.
   c) 30.
   d) 5.

10. **Si se desea que un contacto asociado a un bit de la marca de ciclo del *byte* 22 parpadee con una cadencia de 0,8 segundos, se debe direccionar a:**
    a) M22.3.
    b) M23.4.
    c) M22.0.
    d) M22.4.

ACTIVIDADES FINALES

1. Utilizando un S7-300 o un S7-1500, prueba el comportamiento de los diferentes tipos de temporizadores SIMATIC. Si no dispones de ningún modelo de estos PLC, puedes utilizar el PLCSim correspondiente para simular su funcionamiento. Busca en la ayuda de TIA PORTAL los cronogramas de cada uno de los temporizadores y comprueba que el funcionamiento de lo que has programado es correcto.

2. Haz lo mismo que se pide en la actividad anterior con los temporizadores IEC.

3. Realiza el programa del proceso industrial de la figura, sabiendo que funciona de la siguiente manera:

   Al accionar el pulsador de marcha, la cinta transportadora gira hacia la derecha y desplaza la pieza en dirección a la caja de recogida. Cuando la pieza pasa por el detector B1 se activa el rociador 1 durante 3 segundos, cuando pasa por B2 lo hace el rociador 2 durante 10 segundos y cuando pasa por B3 el tercer rociador funciona durante 4 segundos. La cinta trasportadora se para a los 7 segundos de cerrarse el rociador número 3.

   Los rociadores deben ser controlados por detectores a la conexión y el paro de la cinta transportadora por un detector a la conexión.

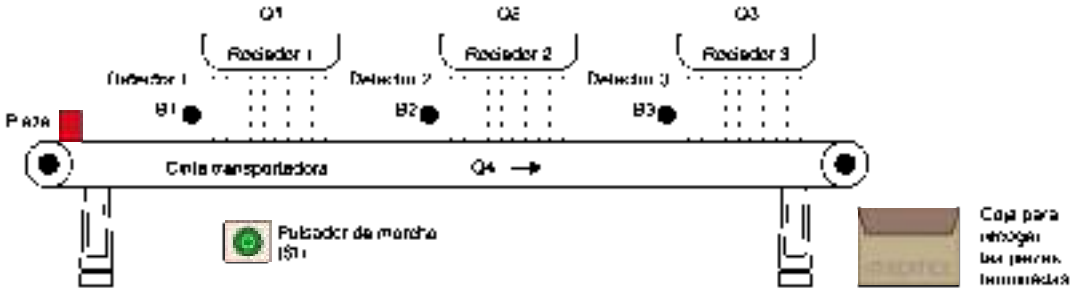


Figura 5.43. Proceso para el tratamiento de piezas

4. Comprueba el funcionamiento del siguiente programa para hacer que una salida se active de forma intermitente. La figura muestra el ejemplo tanto en su versión SIMATIC como en su versión IEC. Prueba a poner diferentes tiempos en cada uno de los temporizadores y observa cómo la lámpara parpadea de forma asimétrica.

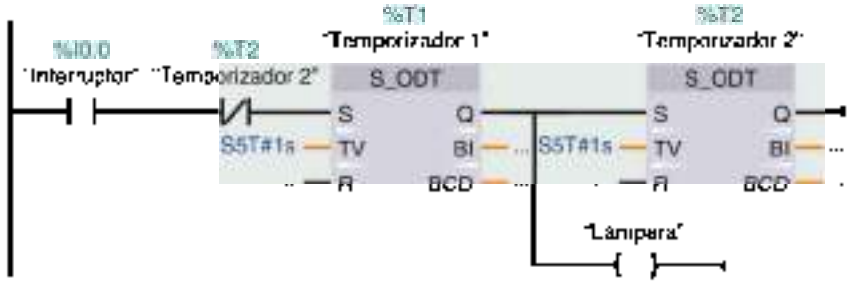


Figura 5.44. Versión SIMATIC

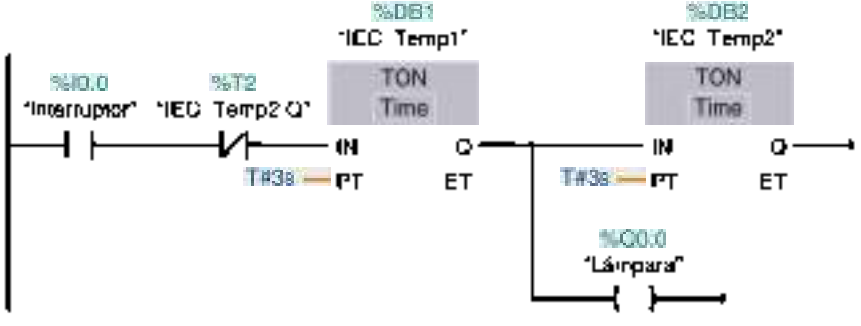


Figura 5.45. Versión IEC

ACTIVIDADES FINALES

continuación

5. Utilizando bobinas SET y RESET, haz el programa para el siguiente funcionamiento:

a) Q1 se activa con S1.

b) Q2 se activa con S2.

c) Q3 se activa con S3.

d) Todas se desactivan con S4 o si se mantiene S5 pulsador durante más de 4 segundos.

6. En el programa anterior añade una salida Q4 que se active mediante S6. La salida Q4 debe funcionar de forma intermitente. Si dicha salida está más de 6 segundos activa se deben resetear las demás salidas.

7. Utilizando un solo temporizador, haz que cinco salidas (Q1, Q2, Q3, Q4 y Q5) se activen consecutivamente cada segundo. Cuando todas estén encendidas se deben apagar todas y comenzar de nuevo la secuencia con la misma pauta de tiempo.

8. Utilizando un solo contador, haz que cuando su valor sea igual a 3 se active una salida, cuando sea igual a 6 se active otra y cuando sea igual a 9 una tercera. Al llegar a 12 se desactivan todas y el contador se pone a cero.

9. Utilizando un contador con una entrada para contar y otra para descontar, e instrucciones de comparación, haz que las salidas se activen de la siguiente forma en función del valor del contador.

a) Q1 cuando es mayor de 5.

b) Q2 cuando es menor de 10.

c) Q3 cuando el valor está entre 12 y 16.

d) Q4 cuando el valor está entre 0 y 5 o entre 15 y 20.

10. Utilizando una solo entrada y una sola salida, haz un programa que funcione en modo telerruptor, de forma que cada vez que se accione la entrada con un pulsador el valor de la salida cambie. Es decir, si estaba a 1 pasa a 0 y si estaba a 0 pasa a 1.

11. Haz un programa con cuatro salidas: el encendido de cada una de ellas se hace de forma individual en función del valor de un contador; así, si el contador es 1 se activa la salida 1, si el contador es 2 se activa la salida 2 y así sucesivamente. Se debe prever que el contador no pueda bajar de 1 y no pueda subir de 4.

12. Utilizado un contador/descontador, haz que una salida funcione de forma intermitente con una cadencia diferente en función del valor del contador.

a) Entre 0 y 10: sin intermitencia.

b) Entre 11 y 20: 2 segundos.

c) Entre 20 y 25: 1 segundo.

d) Valor superior a 26: 0,4 segundos.

13. Crea un programa que encienda dos lámparas intermitentes con la marca de ciclo, de forma que estén activadas de manera alterna. Es decir, cuando una esté a 1 la otra debe estar a 0 y viceversa.

14. Programa un circuito para arrancar un motor con pulsadores de marcha y paro. Se debe señalizar con el parpadeo de una salida cuando el motor está parado o en funcionamiento. La cadencia en parado debe ser de 1,6 segundos y la de marcha de 0,4 segundos.

15. Realiza un programa para controlar cuatro salidas (Q1, Q2, Q3, Q4) con sus respectivos pulsadores de marcha y paro (un grupo marcha/paro por cada salida). Si el autómata pasa de STOP a RUN, una salida (Q5) debe parpadear con una cadencia de 0,5 segundos. Mientras dure esta situación, ninguna de las otras salidas podrá activarse. Para que pueda hacerse, es necesario accionar un pulsador de acuse que elimine el parpadeo de Q5 y permita el funcionamiento normal de los circuitos que controlan las demás partes del programa.

16. Realiza el programa para que al pasar de STOP a RUN el PLC se activen los bits pares del primer *byte* de salidas. Después de 5 segundos del arranque se deben activar los bits impares, y desactivarse los pares. Después de 3 segundos de esto se deben desactivar todas las salidas.

PRÁCTICA PROFESIONAL PROPUESTA 1

Herramientas

- Herramientas básicas del electricista
- PC con TIA PORTAL preinstalado

Material

- Un PLC (S7-300 o S7-1200)
- Cable PC-Adapter (si el PLC es un S7-300)
- Latiguillo Ethernet (si el PLC es un S7-1200)

Llenado de recipientes mediante dosificadores

Objetivo

- Identificar los diferentes elementos de entrada y de salida del proceso y asociarlos a variables del PLC.
- Programar un proceso automatizado con movimientos repetidos para la misma salida.

Precauciones

Es aconsejable el uso de marcas para programar el avance de la gaveta hacia los dosificadores.

Desarrollo

1. Desarrolla y prueba el programa para el llenado de recipientes de la figura, cuyo funcionamiento es el siguiente:
   - El tipo de mezcla se selecciona con el conmutador.
   - Al accionar el pulsador de marcha, solo si hay presencia de gaveta (S5), y en función de la posición del selector, la cinta transportadora mueve la gaveta hasta el dosificador correspondiente.
   - La gaveta se detiene debajo de cada dosificador cuando es detectado por el sensor correspondiente (B1, B2 o B3).
   - El material se deposita en la gaveta durante el tiempo marcado en la tabla de mezclas.
   - Una vez transcurrido dicho tiempo, la gaveta vuelve al punto de carga (S5).
   - Tres lámparas señalizan cada una de las mezclas. En ellas, la luz será fija cuando se desplacen hacia los dosificadores e intermitente cuando el material se vierta en la gaveta.

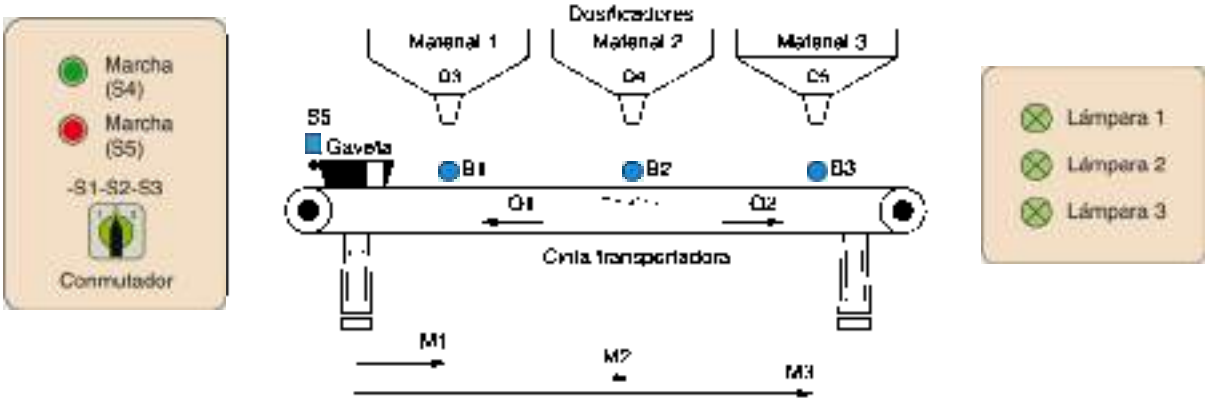


Figura 5.46. Llenado de recipientes mediante dosificadores.

	Selector	Marcha	Detector	Tiempo
Mezcla 1	S1	S4	B1	5 s
Mezcla 2	S2	S4	B2	10 s
Mezcla 3	S3	S4	B3	15 s

PRÁCTICA PROFESIONAL PROPUESTA 2

Herramientas

- Herramientas básicas del electricista
- PC con TIA PORTAL preinstalado

Material

- Un PLC (S7-300 o S7-1200)
- Cable PC-Adapter (si el PLC es un S7-300)
- Latiguillo Ethernet (si el PLC es un S7-1200)

Agrupación de productos mediante cintas de transferencia

Objetivo

- Identificar los diferentes elementos de entrada y de salida del proceso y asociarlos a variables del PLC.
- Uso de contadores para toma de decisiones

Precauciones

- No es necesario programar el sistema de alimentación de piezas. Eso pertenecería a otro proceso que no está aquí descrito. La llegada de piezas a la cinta I se hace de forma aleatoria y siempre con una separación entre ellas
- Debes tener en cuenta algunas señales de los detectores: deben estar asociadas a flancos
- En el programa no se ha previsto un pulsador de puesta en marcha ni de parada.

Desarrollo

Desarrolla y prueba el programa para controlar el sistema de transferencia y agrupación de piezas de la figura, cuyo funcionamiento es el siguiente.

- Los productos llegan a la cinta transportadora por la zona de alimentación. Las cajas entran de una en una y separadas entre sí.
- La cinta transportadora 1 gira y desplaza las cajas hacia la derecha. En este estado, la cinta transportadora 2 permanece parada.
- B1 detecta las cajas cada vez que pasan delante de él. Si el número es igual a tres, la cinta transportadora 2 se pone en marcha. Una vez que las cajas han abandonado la cinta transportadora 1 (B3), esta deja de funcionar.
- La cinta transportadora 2 lleva el paquete de tres cajas hasta la zona de descarga y se para cuando se detecta que el conjunto ha abandonado de la cinta.
- En ese momento, la cinta transportadora 1 se pone de nuevo en marcha y se repite todo el proceso como se ha indicado anteriormente.

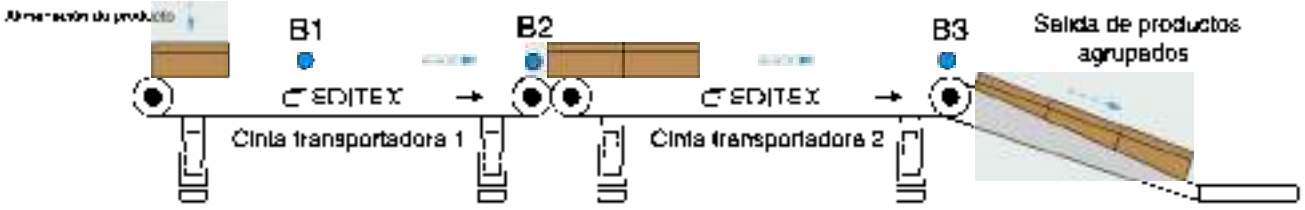
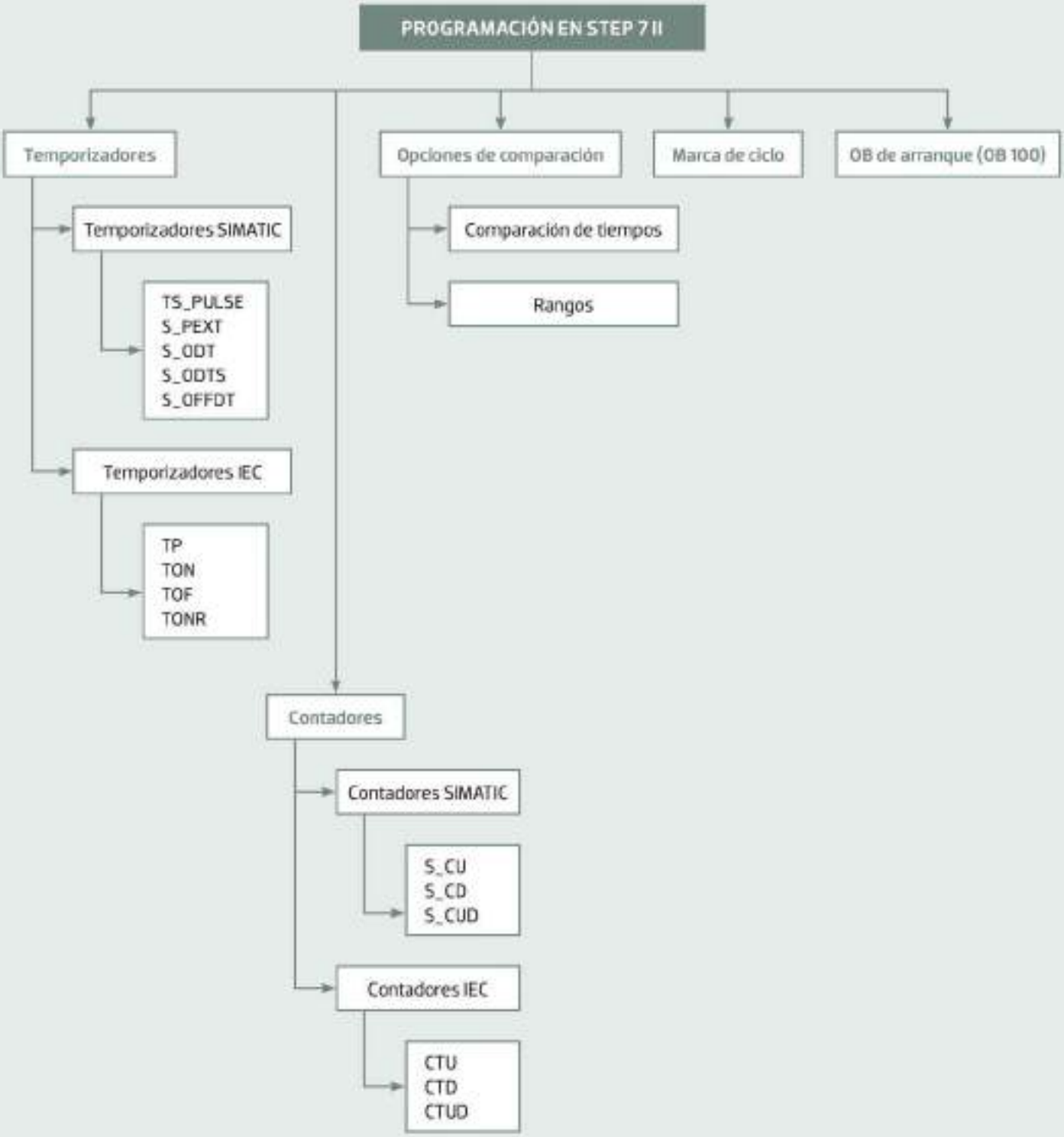


Figura 5.47. Sistema de transferencia y agrupación de productos

EN RESUMEN


PROGRAMACIÓN EN STEP 7 II
Temporizadores
Temporizadores SIMATIC
TS_PULSE
S_PEXT
S_ODT
S_ODTS
S_OFFDT
Temporizadores IEC
TP
TON
TOF
TONR
Opciones de comparación
Comparación de tiempos
Rangos
Marca de ciclo
OB de arranque (OB 100)
Contadores
Contadores SIMATIC
S_CU
S_CD
S_CUD
Contadores IEC
CTU
CTD
CTUD

6 GRAFCET

Vamos a conocer...

1. Introducción al GRAFCET
2. Necesidad de un gráfico secuencial
3. Partes del GRAFCET
4. Sintaxis del GRAFCET
5. Tipos de GRAFCET
6. Modos de funcionamiento en el GRAFCET
7. Estructuración del GRAFCET

PRÁCTICA PROFESIONAL RESUELTA

GRAFCET de taladro semiautomático con cargador de piezas

PRÁCTICAS PROFESIONALES PROPUESTAS

1. GRAFCET de un taladro con cargador de piezas y desahogo
2. Sistema automático de tratamiento de piezas

Y al finalizar esta unidad...

- Sabrás qué es un GRAFCET y para qué se utiliza.
- Diseñarás secuencias con los diferentes tipos de GRAFCET.
- Conocerás las partes que constituyen los GRAFCET y las reglas para utilizarlos.
- Emplearás los GRAFCET para representar los distintos modos de funcionamiento de sistemas secuenciales.
- Crearás secuencias usando la representación estructurada basada en GRAFCET.
- Resolverás varios procesos industriales utilizando gráficos secuenciales.

1. Introducción al GRAFCET

El GRAFCET es un diagrama que permite representar de forma gráfica los diferentes estados y transiciones de un automatismo secuencial.

Fue desarrollado por la AFCET (*Association Française pour la Cybernétique Economique e Technique*) a finales de los años setenta del siglo xx para dar solución a la intensiva implementación, en ese momento, de todo tipo de automatismos secuenciales.

Nunca estuvo ligado a una tecnología en concreto aunque, con los años, su relación con la programación de los autómatas ha sido más que evidente.

En el año 1988, el GRAFCET se estandarizo con la publicación de la norma IEC-848, en la que se establecieron su sintaxis y su representación gráfica. En el año 1992, en un intento de estandarizar los lenguajes de programación en los autómatas programables, la norma IEC 1131-3, aún vigente, lo incluyó como un posible lenguaje que adoptar según el criterio de los fabricantes de PLC. Este lenguaje se ha llamado SFC.

El SFC como lenguaje de programación es, en esencia, lo mismo que el GRAFCET original, aunque ha sido adaptado para facilitar su implementación en el entorno de los autómatas programables.

Así, podemos afirmar que cuando se hace referencia al GRACET se está hablando de un método gráfico de desarrollo de sistemas secuenciales y cuando se hace referencia al SFC se habla de un lenguaje de programación destinado a los autómatas programables, lo que no quita que el GRAFCET original pueda ser implementado en cualquier lenguaje de programación de PLC o de propósito general.

Aquí se va a utilizar el GRAFCET como método de diseño gráfico y en la próxima unidad se implementará en autómatas programables a través del lenguaje a contactos (KOP)

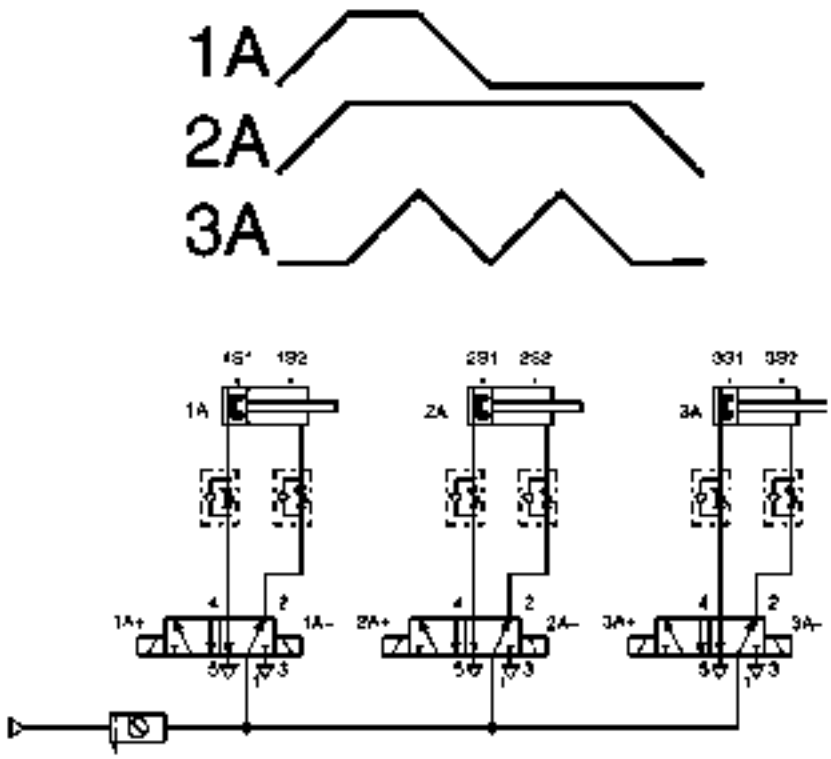
Vocabulario

GRAFCET es el acrónimo del francés *graphe fonctionnel de commande etape transition*, que, traducido al español, significa «gráfico funcional de comando etapa-transición».

SFC: es el acrónimo del inglés *sequential function chart*, cuyo significado es «grafico de funciones secuenciales»

Normas IEC

En este libro se utilizarán las especificaciones que dictan las normas IEC-848 e IEC1131-3, además de algunas adaptaciones personalizadas del autor.



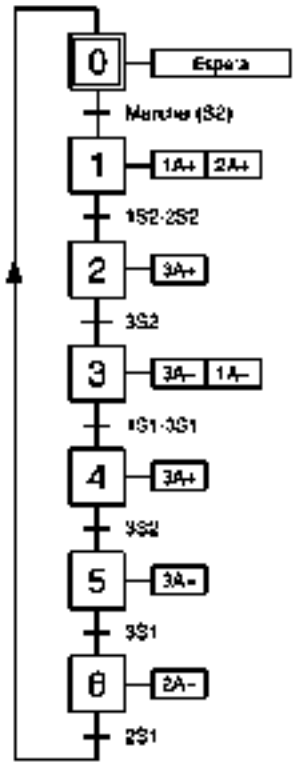


Figura 6.1 Representación de un GRAFCET para el control de un circuito electroneumático.

2. Necesidad de un gráfico secuencial

Saber más

El GRAFCET, aplicado a la programación de PLC, no debe entenderse como un método de simplificación, como ocurre con otros métodos más intuitivos. El GRAFCET facilita el diseño y la estructuración de los programas, pero a costa de un mayor número de líneas o segmentos de programación, lo cual no supone un problema en los controladores lógicos actuales.

El uso de GRAFCET, como método gráfico de diseño, se hace imprescindible en aquellos desarrollos de sistemas secuenciales que tienen gran cantidad de estados, controlados también por numerosos dispositivos de detección.

Con el GRAFCET es posible obtener secuencias «blindadas» de los procesos industriales que hay que automatizar, de tal manera que cuanto mayor sea la complejidad de sistema que hay que controlar, más justificado está su uso.

Ejemplo de aplicación de un GRAFCET

Un GRAFCET se puede entender como una «descripción» por pasos del funcionamiento de una máquina secuencial.

Así, la secuencia de trabajo del taladro semiautomático, que ya se ha utilizado como ejemplo en anteriores ocasiones, es la siguiente:

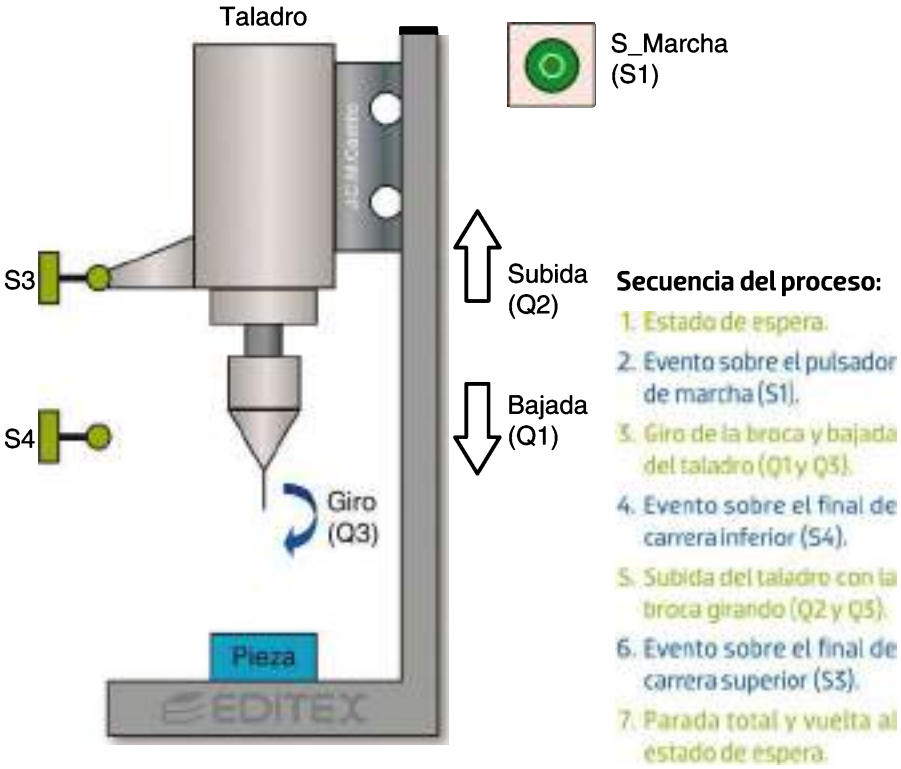


Figura 6.2. Ejemplo de secuencia del proceso de un taladro.

En ella se puede observar que cuando en los captadores se generan eventos, se ejecutan acciones sobre los motores.

Si tanto los eventos como las acciones se representan gráficamente, se obtiene un esquema de la secuencia que se denomina *GRAFCET*.

Los eventos (de color azul) se representarán en las denominadas *transiciones* y las acciones (de color verde) en las etapas.

Una vez descrita la secuencia, es posible diseñar dos tipos de GRAFCET:

- GRAFCET descriptivo o de primer nivel, en el que simplemente se describe cómo es la secuencia del proceso, sin estar asociada a una tecnología en particular.
- GRAFCET tecnológico o de segundo nivel, en el que se describen las acciones y transiciones con los operandos de la tecnología que se va a utilizar (cableada, programada, etc.).

Así, ambos GRAFCET para la secuencia del taladro son

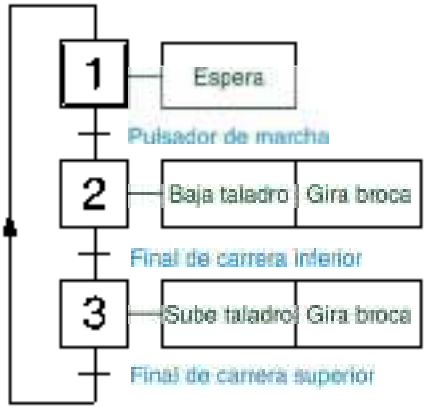


Figura 6.3. GRAFCET descriptivo o de primer nivel

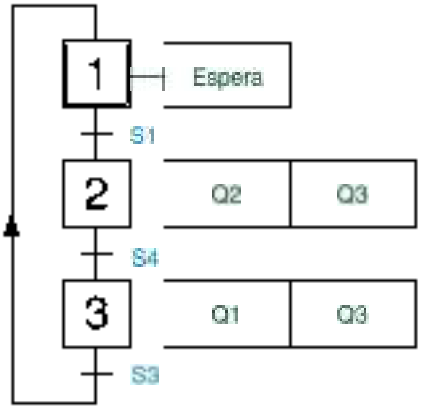
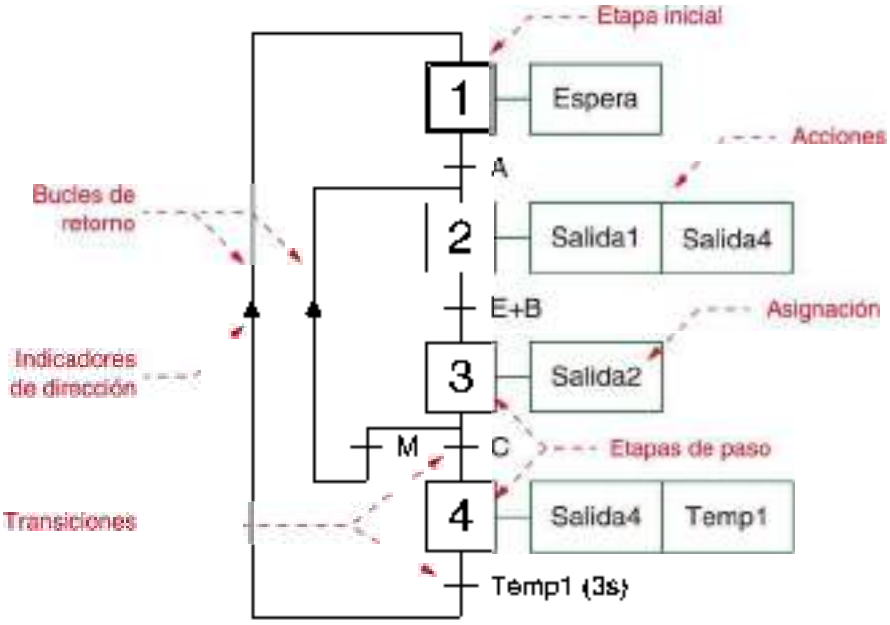


Figura 6.4. GRAFCET tecnológico o de segundo nivel.

3. Partes del GRAFCET

Está formado por un conjunto de símbolos asociados entre sí: etapas, transiciones, etiquetas y líneas de dirección.



6.5. Partes de un GRAFCET.

Recuerda

Si el proceso lo requiere, un GRAFCET puede disponer de varias etapas iniciales.

3.1. Etapas

Representan los diferentes estados del proceso secuencial. Su símbolo es un cuadrado con un número en su interior y pueden ser de dos tipos: de paso o iniciales.

Las etapas se asocian a variables del tipo X1, X2, etc., en función del número con el que estén representadas.

El número de una etapa se puede repetir si dicho número se utiliza en cadenas diferentes. Por ejemplo, en un GRAFCET maestro y en uno o más GRAFCET parciales podemos encontrar etapas con el mismo número, pero siempre en diferentes GRAFCET.

Acciones en las etapas

En las etapas se representan las acciones sobre los dispositivos de actuación del proceso.

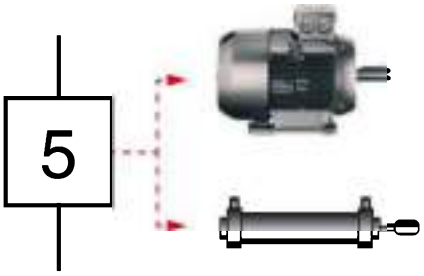


Figura 6.7. Acciones sobre actuadores en etapa.

La etapa inicial muestra el estado en el que comienza el proceso y está representada por dos cuadrados concéntricos. Las etapas de paso solamente se representan con un cuadrado.

Figura 6.6. Etapa inicial y etapa de paso número 5.

La secuencia solamente puede tener una etapa activa, excepto si la secuencia se ha diseñado con más de una etapa inicial o se ha producido una convergencia a etapas simultáneas.

3.1.1. Acciones en etapas

Las etapas tienen las denominadas *etiquetas,* en las que se indican las acciones que realizar. Se representan gráficamente mediante un rectángulo que «cuelga» de la etapa. Cuando la secuencia llega a una etapa determinada, se ejecutan las acciones que en ella se indican.

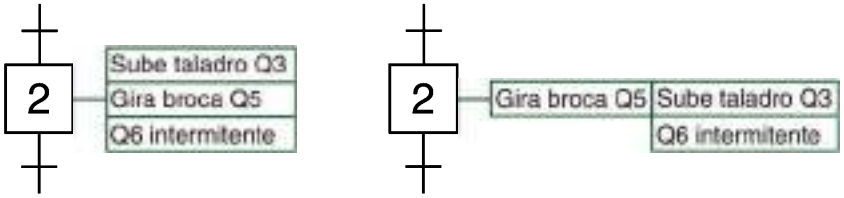


Figura 6.8. Formas de representar las acciones en las etapas.

El orden de llamada de las acciones mediante las etiquetas no influye en su comportamiento. Se pueden representar en forma de columna, de fila o mixta.

3.1.2. Asignaciones en las acciones

Las acciones se pueden asignar de diferentes formas:

- **Asignación directa:** es aquella que activa la variable siempre que el flujo del programa esté en la etapa. No se representa de ninguna forma en especial. Simplemente, se escribe el nombre de la variable en la etiqueta de la acción. En el lenguaje SFC suele estar precedida por la letra N.
- **Asignación de activación:** asigna el valor lógico «1» a la variable. Es similar a una función SET. Se representa con: **variable :=1.**
- **Asignación de desactivación:** asigna el valor lógico «0» a la variable. Es similar a una función RESET. Se representa con: **variable :=0.**
- **Asignación de un valor a una variable:** carga un valor, en este caso numérico, a una variable.

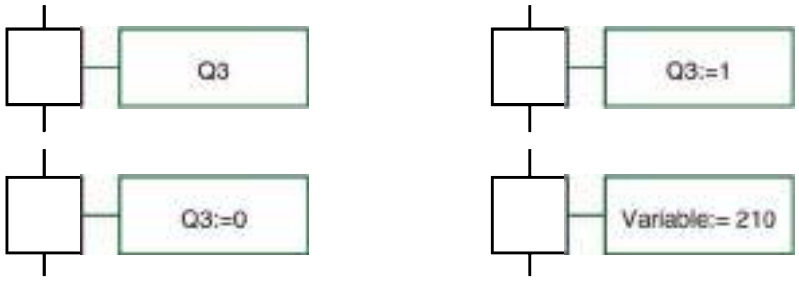


Figura 6.9. Diferentes formas de asignaciones: directa, activación, desactivación y asignación de valor a una variable.

3.1.3. Tipos de acciones

Podemos clasificar las acciones en función de cómo se ejecutan:

- De forma continua.
- Condicionadas.
- Basadas en eventos.
- Activadas por flancos.

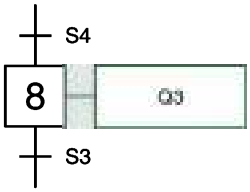
A continuación se describe cada una de ellas.

Acción continua

En este caso, la acción se ejecuta siempre que la secuencia se encuentra en la etapa activa y deja de hacerlo cuando se sale de ella.

Acción continua

La salida Q3 está activa siempre que el proceso se encuentre en la etapa 8.

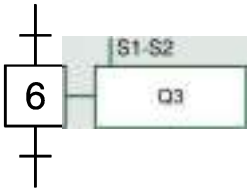


Acción condicionada

Se ejecuta siempre que la secuencia se encuentre en la etapa donde está la acción y, además, se cumpla la condición lógica de una o más señales. Gráficamente se representa con una línea vertical en la parte superior de la etiqueta, en la que se indica la señal o la ecuación lógica de la condición.

Acción condicionada

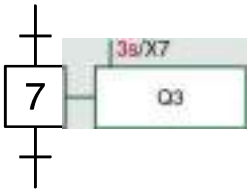
La salida Q3 se activa siempre que el proceso se encuentre en la etapa 6 y se cumpla que las dos señales de S1 y S2 estén activadas a la vez.



Este tipo de acciones también pueden estar condicionadas a acciones de tiempo, tanto con retardo a la conexión como a la desconexión.

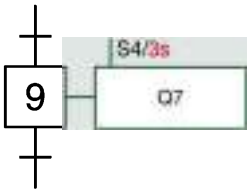
Acción condicionada con retardo a la conexión

La salida Q3 se activa a los 3 segundos de que la secuencia llegue a la etapa X7.



Acción condicionada con retardo a la desconexión

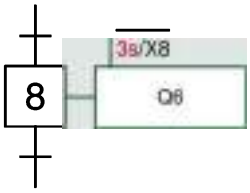
La salida Q7 se activa cuando la secuencia se encuentra en la etapa X9 y está activa la señal de S4. Una vez que esta última pasa de 1 a 0, se inicia la temporización hasta que Q7 se desactiva.



También se puede condicionar la activación de una acción a un límite de tiempo.

Acción condicionada a un límite de tiempo

La salida Q3 se activa al entrar en la etapa X8 y permanece así durante 3 segundos. Es el funcionamiento inverso al retardo a la conexión.



Etapa activa

Un punto en el interior de una etapa representa la etapa activa.

Figura 6.10. Etapa activa.

Nomenclatura

Las acciones condicionadas a tiempo se expresan de la siguiente manera:

Tmp_Conex/Señal/Tmp_desconex

Ejemplos:

Condición a la conexión 2s/S3

Condición a la desconexión: S3/6s

Condición a la conexión/desconexión: 2s/S3/6s

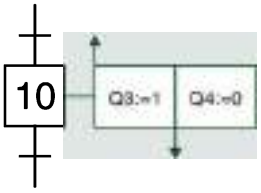
En todos los casos, la señal S3 es la que condiciona la acción.

Acciones por flancos

Solamente se ejecutan cuando se entra o se sale de la etapa en la que se encuentra la acción. Son acciones asociadas a detectores de flanco. El flanco positivo es para detectar cuándo se entra en la etapa y el flanco negativo para hacerlo cuándo se sale.

Acciones por flancos

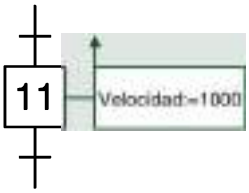
Aquí se ejecutan dos acciones: Q3 se activa justo en el momento en el que se entra en la etapa 8 y Q4 se desactiva cuando se sale de la misma etapa.



Este tipo de acciones son especialmente útiles para activar y desactivar señales mediante funciones SET y RESET, así como para asignar valores numéricos a variables.

Acción para asignación de valor a variable

Al entrar en la etapa X11 se le asigna un valor de 1000 a la variable denominada *Velocidad*.



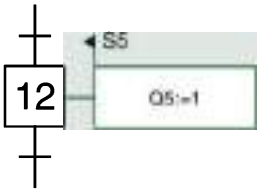
Acciones por eventos

La acción es verdadera si se cumple que la etapa está activa y que se produce un evento en la señal de la condición.

Esta acción se representa de forma similar a la acción condicionada, en la que la línea vertical tiene forma de banderola.

Acción por evento

Q5 se activa cuando la secuencia se encuentra en la etapa 12 y se produce un evento en S5.



3.2. Transiciones

Una transición es una condición que permite el paso de una etapa a otra. Se representa con una línea horizontal en forma de cruz sobre la línea de dirección que une dos etapas.

Para que una transición sea receptiva debe estar activada la etapa que le precede, es decir, la que está por encima de ella.

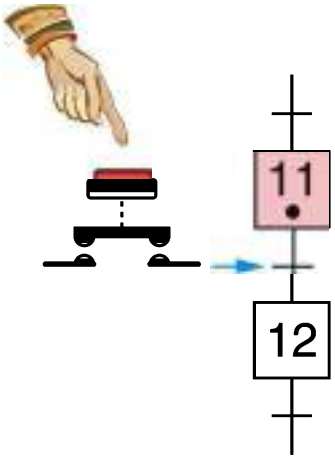


Figura 6.11. Transición.

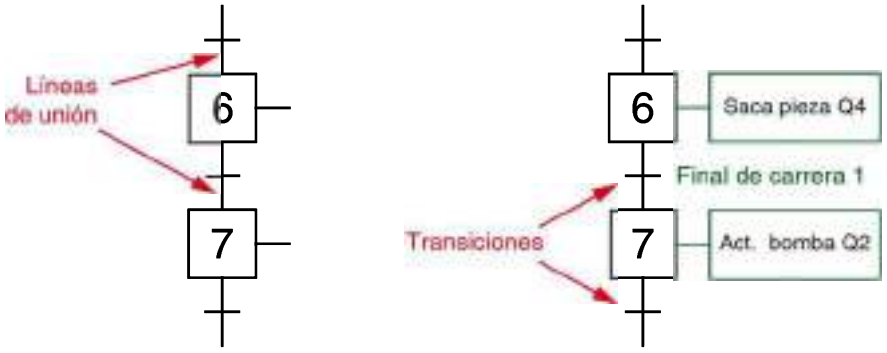


Figura 6.12. Transiciones en el GRAFCET.

3.2.1. Condiciones lógicas en las transiciones

En las transiciones se pueden representar los identificadores de las señales o sus combinaciones lógicas, si es que existen, tanto de forma textual como de forma gráfica.

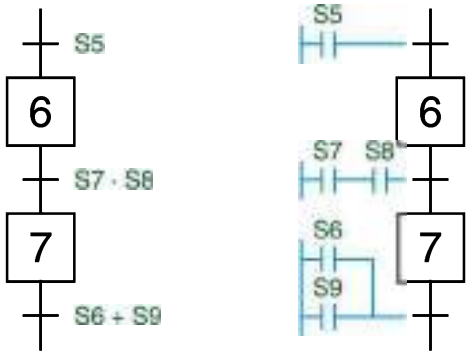


Figura 6.13. Representación de las transiciones: textual y gráfica.

3.2.2. Transiciones temporizadas

En numerosas ocasiones las transiciones necesitan ser receptivas a señales temporizadas, de forma que sea posible medir el tiempo necesario para flanquearlas.

La sintaxis de esta operación es similar a la ya descrita anteriormente para las acciones temporizadas con retardo a la conexión.

Si lo que se desea es controlar el tiempo de paso de una etapa a otra:

Tiempo/Etapa anterior, por ejemplo: 4s/X3

Si el tiempo de paso entre etapas está condicionado a la acción sobre una señal:

Tiempo/Señal, por ejemplo: 4s/Pulsador

A continuación se muestran ambos ejemplos de forma gráfica. En el caso de la izquierda, el paso de la etapa 3 a la 4 se realiza después de 4 segundos. En el caso de la derecha, el paso entre etapas solamente se realiza si el pulsador está accionado durante 4 segundos.

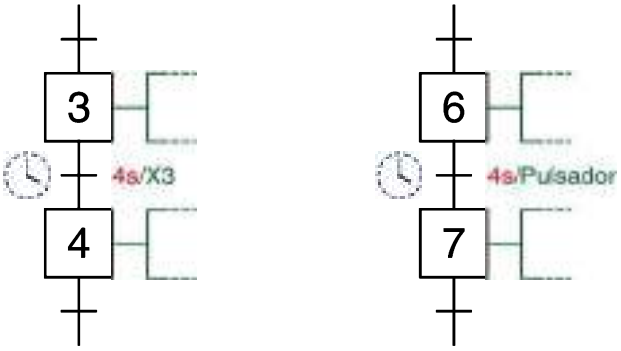


Figura 6.15. Representación de las transiciones: textual y gráfica.

3.2.3. Transición en función del valor de una variable

Permite establecer condiciones de paso entre etapas en función del valor almacenado en una variable. La receptividad se establece mediante operaciones de comparación.

Identificadores de las transiciones

Opcionalmente, y para facilitar la documentación de los proyectos, las transiciones pueden disponer de un nombre identificador. Este se escribe entre paréntesis al lado izquierdo de las mismas.

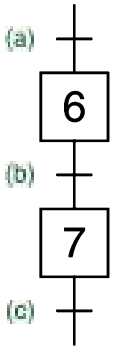


Figura 6.14. Identificadores de las transiciones.

Operaciones de comparación

- == Igual que
- >= Mayor o igual que
- <= Menor o igual que
- > Mayor que
- < Menor que

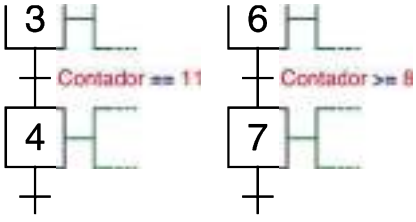


Figura 6.16. Transiciones basadas en valores de variables.

4. Sintaxis del GRAFCET

Es de suma importancia que los gráficos secuenciales basados en GRAFCET estén correctamente diseñados, ya que esto facilitará su interpretación y, si es necesario, una posterior ampliación.

Para ello solamente es necesario aplicar unas reglas de diseño básicas:

- La lectura de la secuencia debe realizarse de arriba hacia abajo.
- Hay que utilizar indicadores de dirección, en formato de flecha, para representar el retorno del flujo del programa hacia etapas superiores. Sin embargo, no es necesario realizar esta representación hacia abajo, salvo que pueda generar errores de interpretación.
- Nunca se podrán representar dos etapas o dos transiciones consecutivas. La lectura del flujo del programa debe realizarse siempre «etapa-transición-etapa-transición».
- Se debe representar, al menos, una etapa inicial.
- Aunque está permitida la representación de transiciones en horizontal, siempre que sea posible se deben representar en vertical siguiendo el flujo del programa.
- El número de las etapas debe ser unívoco o, lo que es lo mismo, no puede haber dos etapas con el mismo número salvo cuando se utilicen GRAFCET parciales a modo de subrutina.
- La numeración de etapas no necesariamente tiene que ser consecutiva; no obstante, se aconseja que sea así, dentro de lo posible, para su mejor entendimiento e implementación.

Etapa actual

Si es necesario representar la etapa por la que se encuentra la secuencia, esta se marca con un punto en su interior.

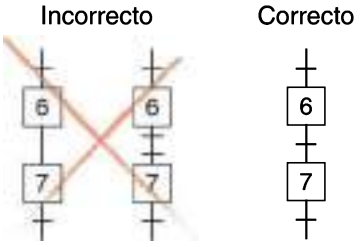


Figura 6.17. Errores que hay que evitar al representar los GRAFCET.

5. Tipos de GRAFCET

La representación básica de los GRAFCET se puede hacer de tres formas:

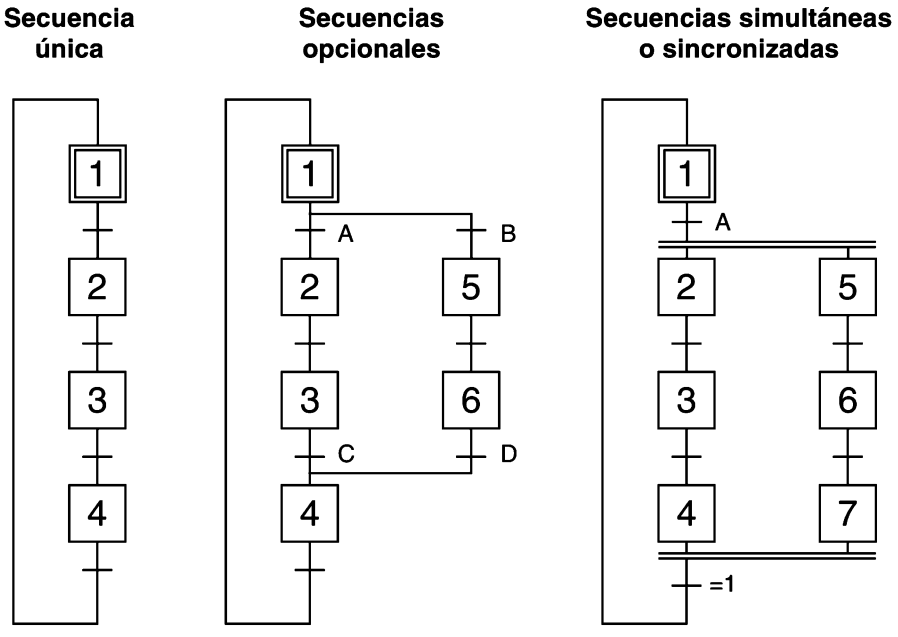


Figura 6.18. Tipos de GRAFCET.

A continuación, vamos a ver las principales características de las tres formas de representación básica de los GRAFCET.

5.1. GRAFCET de secuencia única

Es un conjunto de etapas y transiciones conectadas en cascada y sin bifurcaciones. En este caso, la evolución de la secuencia solamente sigue un camino.

Es la forma más sencilla de representar la secuencia de un proceso, pero también la más incompleta, ya que cualquier proceso industrial requerirá la toma de decisiones para saltar o retornar a otros estados.

5.2. GRAFCET de secuencias opcionales

En este tipo de GRAFCET la secuencia puede optar por seguir dos o más caminos.

Las secuencias opcionales siempre parten desde una etapa y convergen en otra. Su representación se hace mediante líneas de dirección horizontal.

Se denomina *divergencia en O* a la parte del GRAFCET en la que se bifurca la secuencia y, por el contrario, *convergencia en O* a la parte donde confluyen de nuevo.

Ejemplo

En la figura 6.18, se muestra cómo estando en la etapa inicial el proceso puede evolucionar por la 2, a través de la transición A, o por la etapa 5, a través de la transición B.

5.3. GRAFCET de secuencias simultáneas

En este tipo de GRAFCET se ejecutan dos o más caminos al mismo tiempo.

Las secuencias simultáneas siempre parten de una transición de origen y convergen en otra de destino. Su representación se realiza mediante una línea doble de dirección horizontal.

Se denomina *divergencia en Y* a la parte del GRAFCET donde comienzan las secuencias simultáneas y, por el contrario, *convergencia en Y* a la parte donde confluyen de nuevo.

Un GRAFCET de este tipo no evoluciona hasta la secuencia principal, si no han finalizado todos los caminos opcionales. Por este motivo, también se suele denominar *GRAFCET de secuencias sincronizadas*.

Ejemplo

En la figura 6.18, se muestra cómo al cumplirse la transición A se ejecutan simultáneamente los caminos que comienzan por las etapas 2 y 5. Hasta que no han finalizado ambos, con las etapas 4 y 6 respectivamente, no evoluciona por la cadena principal.

5.4. Otras formas de representación

A continuación se muestran algunas variantes para facilitar la representación de los GRAFCET.

5.4.1. Saltos y retornos

Se basan en el concepto de secuencias opcionales y se utilizan para direccionar el flujo del programa a una etapa anterior, en el caso de los retornos, o una posterior, en el caso de los saltos.

Salto y retorno

La siguiente figura representa cómo estando en etapa 2 se puede saltar a la 9 por la transición J y retornar a la etapa 2 desde la 5 por la transición B.

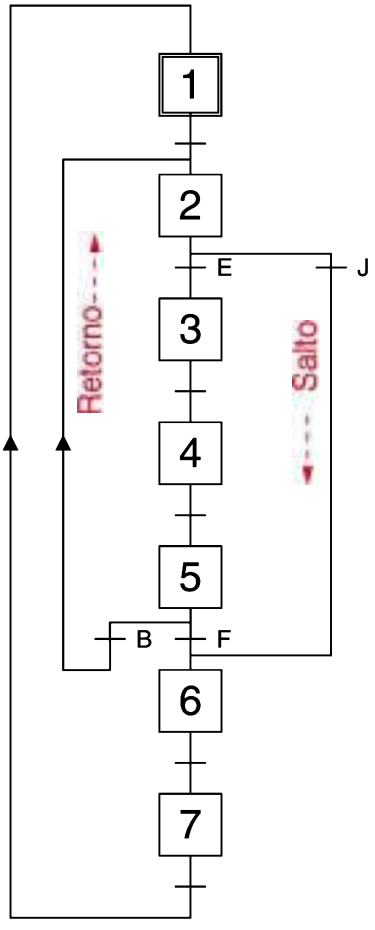


Figura 6.19. GRAFCET con saltos y retornos.

5.5. Bucles con marcas de conexión o METAS

Los saltos y retornos entre etapas se pueden representar mediante marcas de conexión, también denominadas *METAS*. Su uso simplifica los gráficos secuenciales, los hace más claros y legibles y, por lo tanto, más fáciles de implementar.

Las llamadas se representan con flechas cuya dirección indica el flujo de la secuencia. Las etiquetas indican los nombres de las variables de las etapas desde las que se llega o desde las que se procede.

La siguiente figura muestra un mismo GRAFCET de tres maneras diferentes en el que los saltos se representan con o sin METAS.

En todos ellos se representan tres saltos desde las etapas X2, X3 y X4 a la etapa X6 si se cumple la transición Trans1.

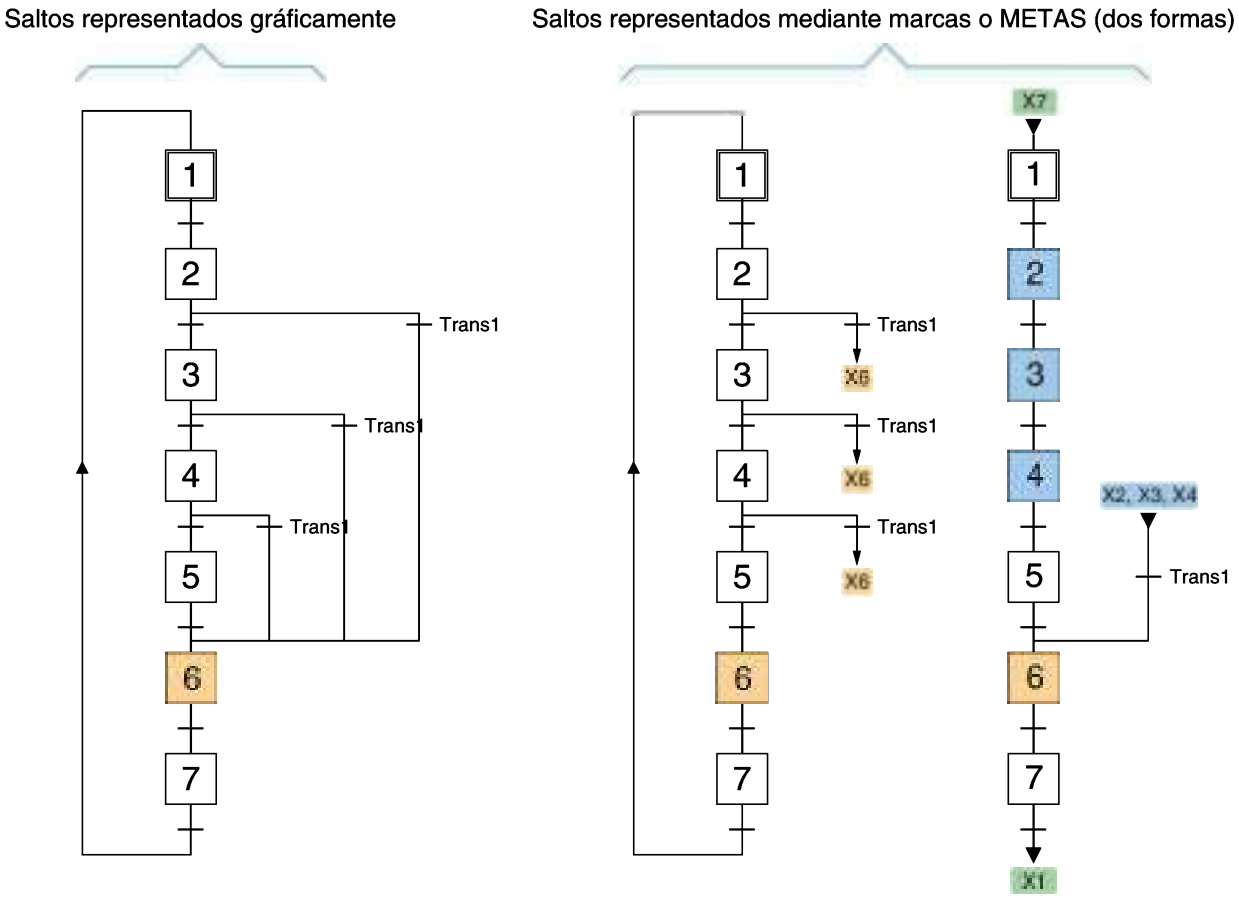


Figura 6.20. Diferentes formas de representar saltos en el GRAFCET.

En el GRAFCET de la derecha se puede comprobar que el uso de marcas de conexión facilita la representación gráfica y mejora la compresión del funcionamiento de la secuencia.

Actividades

1. Dibuja un GRAFCET de 15 etapas con los siguientes saltos y retornos:
   - Saltos de las etapas 3, 4 y 5 a la 12 con S1.
   - Saltos de las etapas 3, 6, 7 a la 14 con S2·S3.
   - Retornos de la etapa 2 desde las etapas 12, 13 y 14 con S4+S5.

5.6. Secuencias con varios GRAFCET

Un proceso secuencial puede utilizar dos o más cadenas GRAFCET. Estas se pueden ejecutar de forma independiente o coordinada entre sí. Cada cadena GRAFCET debe tener su etapa inicial, que se activa cuando se pone en marcha el sistema.

Las etapas de cada cadena se deben etiquetar en diferentes rangos de numeración. Esto facilitará su identificación y una posible renumeración en el caso de que alguna de ellas deba ampliarse con más etapas.

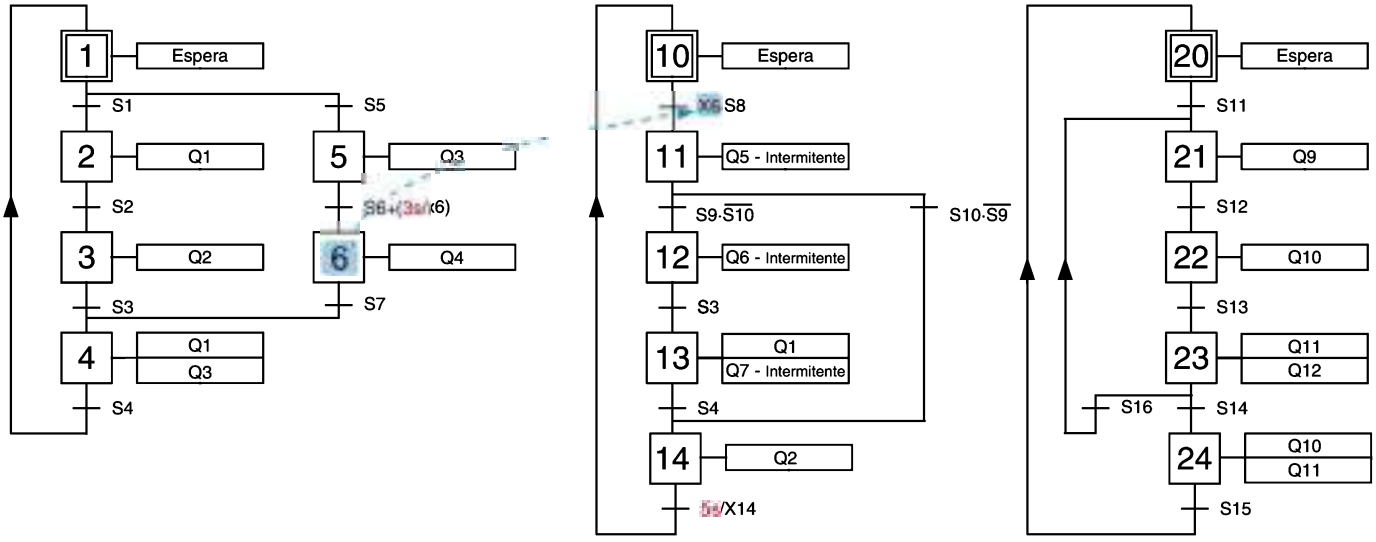


Figura 6.21. Ejecución de varios GRAFCET simultáneos.

Es posible condicionar el funcionamiento entre GRAFCET usando los bits de las etapas de unas cadenas en las transiciones de las otras.

En el ejemplo de la figura se muestran tres cadenas GRAFCET independientes. El funcionamiento del representado en la parte central está condicionado a la activación de la etapa X6 en el GRAFCET de la izquierda. Sin embargo, la ejecución de la cadena de la derecha no depende de ninguno de los otros dos.

5.7. Etapas fuente y etapas pozo

Una **etapa fuente** es aquella que no tiene transición para llegar a ella y, por el contrario, una **etapa pozo** es aquella que no tiene transición para salir de ella. Ambas se utilizan para crear secuencias abiertas sin bucle cíclico y son especialmente útiles para emplearse en GRAFCET parciales, llamados como subrutinas desde otras cadenas, organizadas jerárquicamente en un nivel superior.

También, las etapas pozo se pueden usar para llevar la secuencia a una situación de alarma o de emergencia.

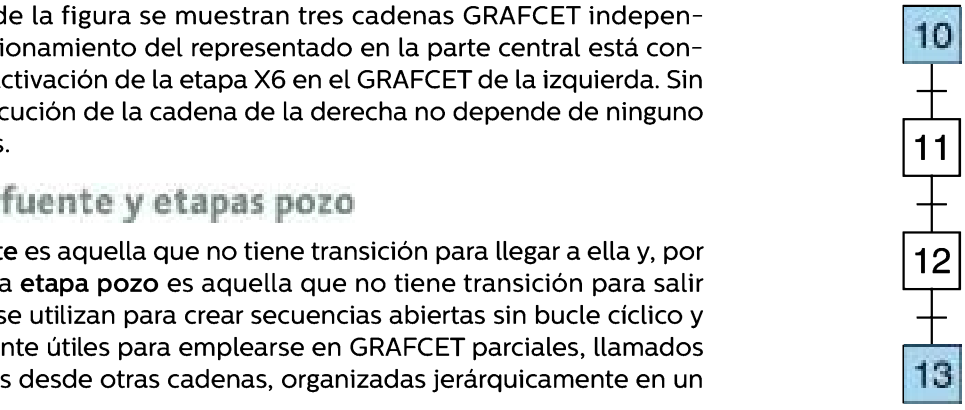


Figura 6.22. Etapas fuente y pozo en una secuencia sin bucle cíclico.

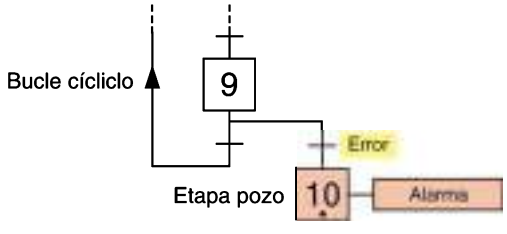


Figura 6.23. Uso de etapa pozo para disparo de una alarma.

6. Modos de funcionamiento en el GRAFCET

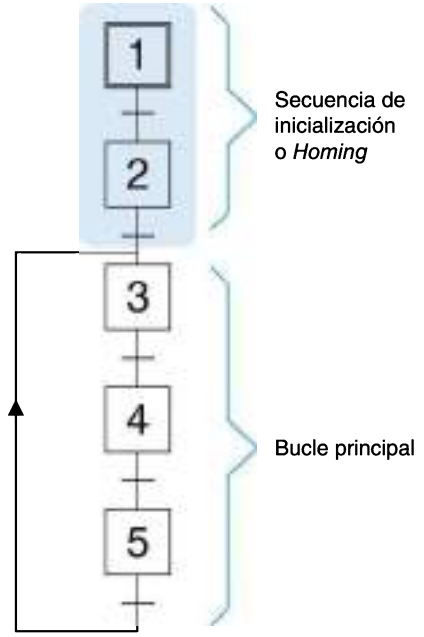


Figura 6.24. *Homing* en GRAFCET.

A continuación se describen diferentes formas para optimizar y adaptar el GRAFCET a cualquier proceso automatizado.

6.1. Secuencia de inicialización o *Homing*

La secuencia de inicialización, también conocida como *Homing,* es una parte del GRAFCET que se ejecuta solamente en el momento de iniciar el proceso. En el caso de los sistemas automatizados mediante autómatas programables, esta situación se produce cuando el PLC pasa de STOP a RUN. Esta acción se denomina habitualmente *arranque en caliente*.

Un arranque en caliente puede producirse por diferentes causas, como: después de un corte inesperado de la energía eléctrica o de una parada del sistema, por ejemplo, debido a tareas de mantenimiento. Cuando esto ocurre, es necesario prever cómo debe responder el proceso cuando el dispositivo de control vuelve a ejecutar el programa. Esto se representa con un conjunto de etapas y transiciones que comienza en una etapa fuente de tipo inicial a la que no le precede ninguna transición. La última transición de este grupo se conecta con la primera etapa del bucle cíclico o principal.

Si es necesario, y se cumplen las condiciones, se puede reutilizar esta secuencia para realizar el referenciado de forma manual mediante el uso de saltos desde cualquiera de las etapas de la secuencia cíclica.

Ejemplos

Volviendo al ejemplo del taladro semiautomático, la zona de Homing podría ser la siguiente:

1. Al ejecutar el GRAFCET por primera vez (etapa X1), el proceso debe señalizar que se ha realizado un arranque en caliente del sistema. Para ello se utiliza una lámpara que parpadea de forma intermitente y que lo hace todo el tiempo que dure el proceso de referenciado.
2. En esa situación, el operario debe accionar el pulsador de marcha u otro asignado para esta función.
3. Si la máquina se había parado cuando el taladro estaba en movimiento, este se sube hasta tocar el final de carrera superior y alcanza la posición de referencia a la espera de comenzar el bucle principal. Si, por el contrario, la máquina estaba ya en su posición de reposo, como el final de carrera superior estaba accionado, la secuencia se situará de forma inmediata en la etapa X3 de espera, una vez accionado por primera vez el pulsador de marcha después del arranque en caliente.

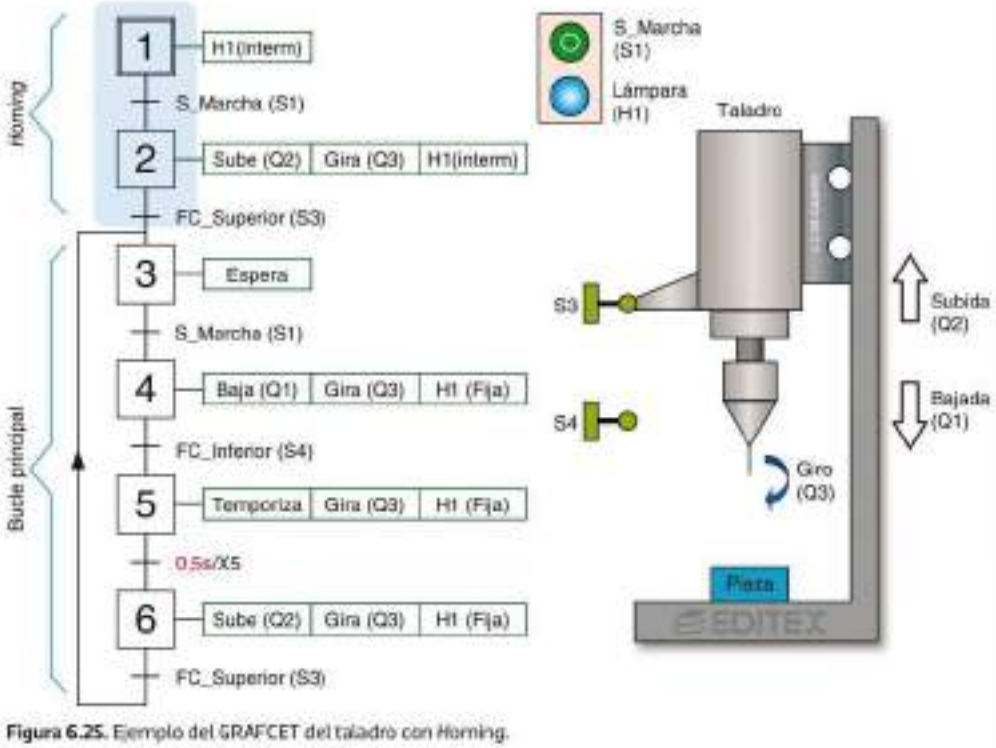


Figura 6.25. Ejemplo del GRAFCET del taladro con *Homing*.

6.2. Parada

Cualquier proceso secuencial debe tener previsto algún método de parada. La parada momentánea consiste en detener la secuencia de forma voluntaria para, posteriormente e igualmente de forma voluntaria, reanudarla desde el estado en el que se quedó.

Hay varios métodos para implementar la función de parada. Aquí se muestra cómo hacerlo mediante un GRAFCET dedicado a ello. En la próxima unidad se mostrará otro método, basado en programación, con el que se consigue el mismo fin de una forma más simple.

El GRAFCET de parada consta de dos etapas, cuya evolución se controla mediante el pulsador de paro, para pasar de la etapa inicial a la de paso, y con el de marcha para hacerlo a la inversa.

Se debe utilizar una variable, que en el ejemplo se ha denominado *Marca de Paro* (M_Paro). Esta se controla con acciones de activación y desactivación en la cadena de parada y los bits negados de ella se utilizan para condicionar todas aquellas acciones del GRAFCET principal que deseen desactivar para detener el proceso.

En la etapa X10 la marca de paro está a 0 (RESET). Cuando se acciona el pulsador de parada, el GRAFCET evoluciona a la etapa X11, la marca de paro se pone a valor lógico 1 y desactiva todas las acciones de la cadena principal que están condicionadas a dicha marca.

El pulsador de parada debe ser normalmente cerrado (NC) por cuestiones de seguridad, por lo que hay que representarlo negado en la transición en la que se referencia.

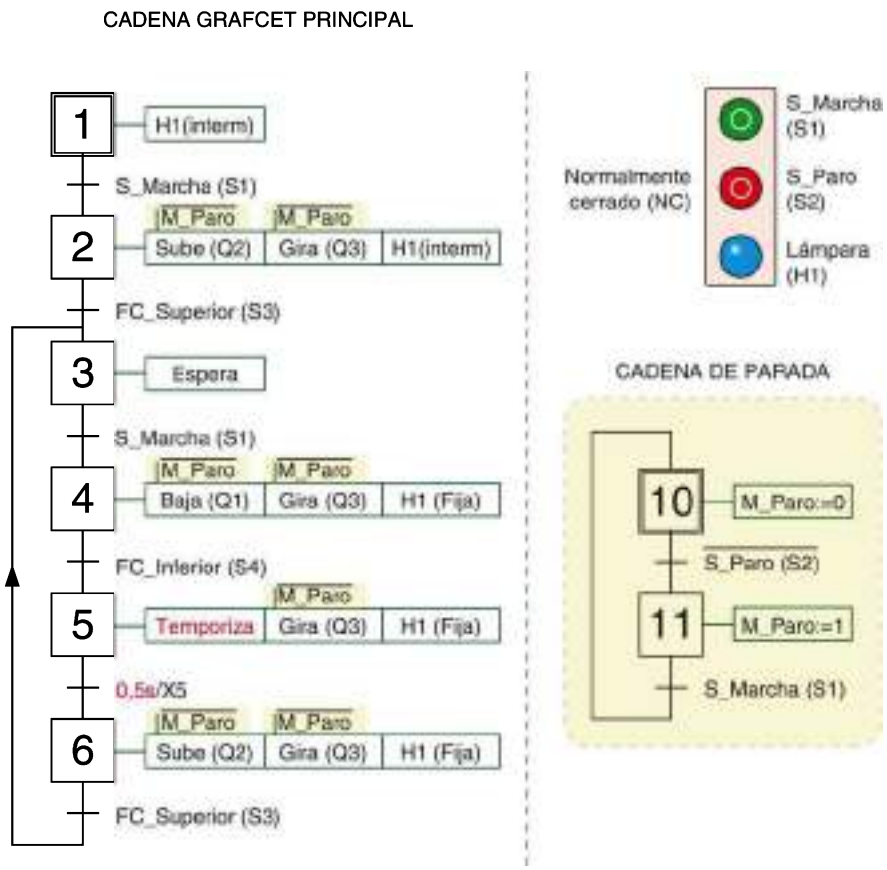


Figura 6.27. Implementación de un GRAFCET de parada.

Parada alternativa

Opcionalmente, también es posible implementar la M_Paro negada en todas las transiciones de la cadena principal del GRAFCET que se desee parar. Esto evita que el GRAFCET evolucione a la siguiente etapa, aunque realice un evento en la transición que es receptiva a la etapa en que se realizó la parada.

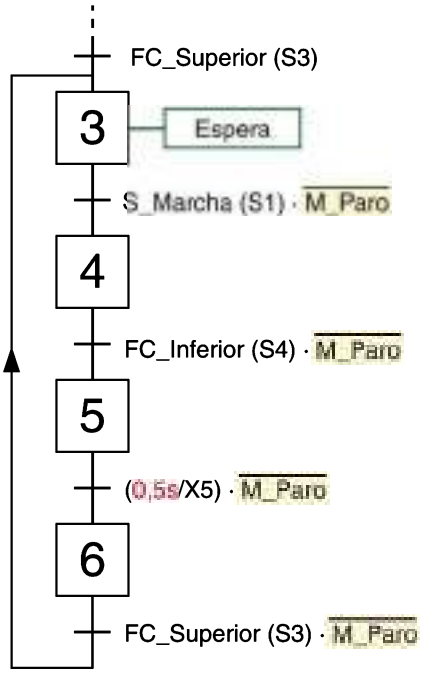


Figura 6.26. Marca de paro en las transiciones.

6.3. Rearme (anular)

La acción de rearme, o anular, es una acción voluntaria del operario que permite detener el proceso que está en marcha para llevarlo a una zona de la secuencia con un funcionamiento específico.

En el caso del ejemplo del taladro, la función anular podría darse, por ejemplo, cuando al accionar un pulsador determinado (S_Anular) se abandone la secuencia principal y se realice una tarea determinada, como puede ser subir el taladro hasta la posición superior.

En el caso del taladro, esta acción se realiza en la etapa X2 de la zona del *Homing,* por lo que se puede aprovechar esta parte del GRAFCET y hacer un salto a dicha etapa desde cualquiera de las del bucle principal.

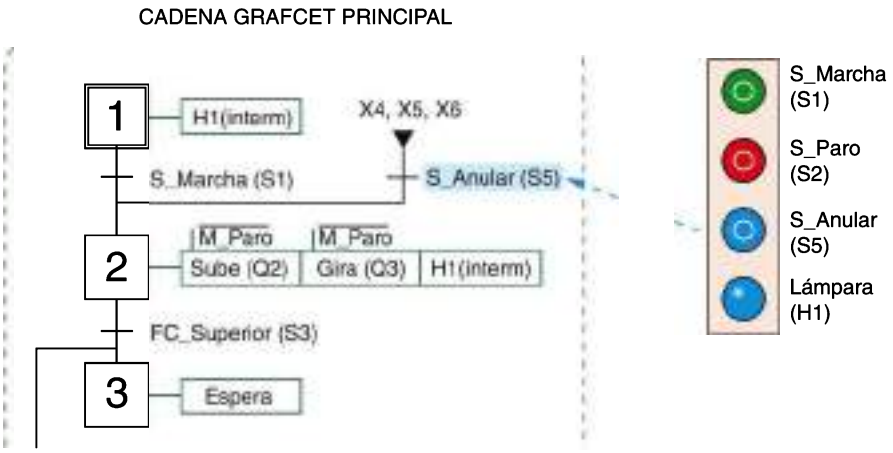


Figura 6.28. Función anular en el GRAFCET del taladro.

En ocasiones no será posible aprovechar las partes del GRAFCET de la zona del *Homing* o del bucle principal, por lo que será necesario representar una secuencia propia para realizar la operación de rearme o anular.

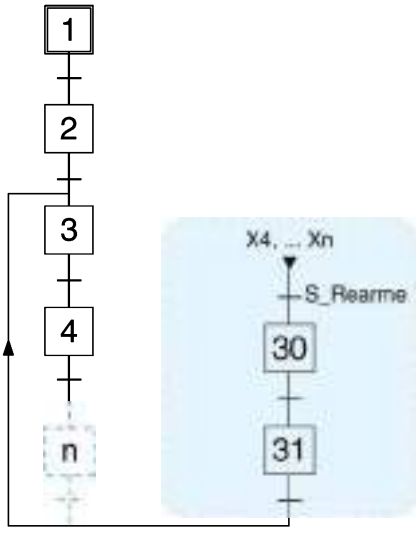


Figura 6.29. Forma de implementar un rearme con su propia secuencia.

En el ejemplo de la figura se muestra una función de rearme con su propia secuencia (etapas X30 y X31). En este caso, si se acciona el pulsador S_Rearme estando en cualquiera de las etapas del bucle cíclico (de la X4 a la Xn), el proceso salta a la etapa X30 y realiza las operaciones en ella indicadas.

Vocabulary

Etapa: *step.*
Transición: *transition.*
Acción: *action.*
Etapa inicial: *initial step.*
Etapa incluyente: *closing step.*
Macroetapa: *macro step.*
Condición: *condition.*
A la activación: *upon activation.*
A la desactivación: *upon desactivation.*
Acción continua: *continuous action.*
Asignación: *allocation.*
Arranque en caliente: *hot start.*
Pulsador de parada: *stop push button.*

6.4. Selección de secuencias

Un proceso industrial puede presentar diferentes modos de funcionamiento seleccionables por el operario antes de ejecutar la secuencia.

Esto puede hacerse mediante un elemento de conmutación, de tal forma que la puesta en marcha esté condicionada al estado de su señal. Así, si se implementa en un GRAFCET de secuencias opcionales, se condicionará la ejecución de una camino u otro.

Figura 6.30. Ejemplo de selección del modo de funcionamiento.

Ejemplos

Continuando con el ejemplo del taladro, en este proceso se pueden prever dos modos de funcionamiento:

1. **Taladrado sin desahogo,** en el que la pieza se taladra por completo de una vez.
2. **Taladrado con desahogo,** en el que la pieza se taladra en dos tiempos, primero hasta la mitad y posteriormente hasta el final. En este caso, cuando la pieza se ha perforado hasta la mitad el taladro sube hasta la parte superior, desahogando y aliviando la broca, para volver a bajar hasta completar la operación.

El uso del selector se hace justo debajo de la etapa de espera del bucle cíclico (X3). Así, en función de cuál sea el modo seleccionado con la señal de conmutación, al accionar el pulsador de marcha (S1) se realizará el taladro con o sin desahogo.

Si el selector S7 está en la posición 1 se ejecuta el modo con desahogo. Si está en la posición 2, se realiza sin desahogo.

XS, X7 y X9 son etapas de paso temporizadas, para que el cambio de sentido de giro del motor que mueve el taladro no se realice de forma instantánea. Esto evita que ambos contactores estén durante un momento activados a la vez y se produzca un cortocircuito a través del circuito de fuerza.

Si el movimiento de subida y bajada del taladro se hiciese con un cilindro neumático se podría prescindir de dichas etapas.

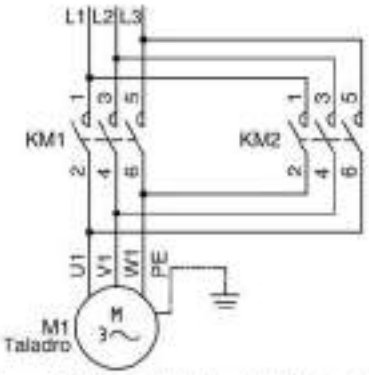


Figura 6.31. Inversión del sentido de giro del motor del taladro.

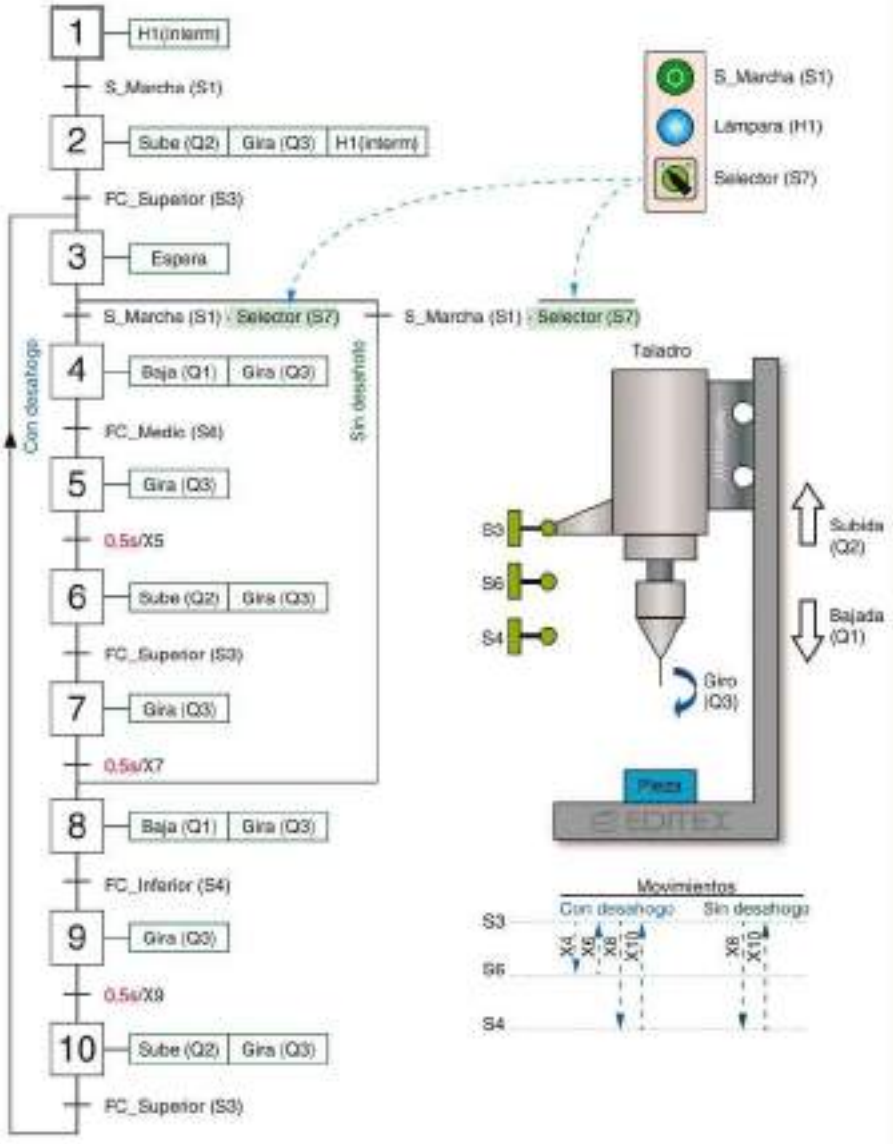


Figura 6.32. GRAFCET del taladro con dos modos de funcionamiento, con y sin desahogo.

6.5. Funcionamiento por pasos

El modo de funcionamiento por pasos, también denominado en ocasiones *semiautomático*, consiste en hacer que un GRAFCET evolucione etapa por etapa mediante la acción manual sobre un pulsador. Este pulsador debe configurarse como una señal de flanco positivo (↑) para evitar que, con una simple pulsación, la secuencia evolucione de forma no deseada varias etapas de una vez.

Lo habitual en un GRAFCET es poder conmutar entre el modo continuo y el modo por pasos. Para ello se utiliza una señal de selección que se asocia en paralelo con la señal del pulsador que realiza el avance por pasos. Así, en función de si dicha señal está o no negada, es posible anular o activar este modo de funcionamiento.

En el modo semiatuomático, para evitar que los actuadores queden activados cuando se cumplen las etapas en las que se ejecuta el movimiento, cada acción debe estar condicionada a la señal negada del final de carrera o detector que lo detiene.

Representación de flancos

La representación literal de flancos se hace mediante una flecha delante del identificador de la señal. El flanco positivo hacia arriba y el negativo hacia abajo.

Flanco positivo: ↑**Pulsador**

Flanco negativo: ↓**Pulsador**

Modo por pasos

Este modo de operación es especialmente útil para comprobar el funcionamiento de una máquina secuencial y realizar los ajustes pertinentes en sus actuadores y sensores.

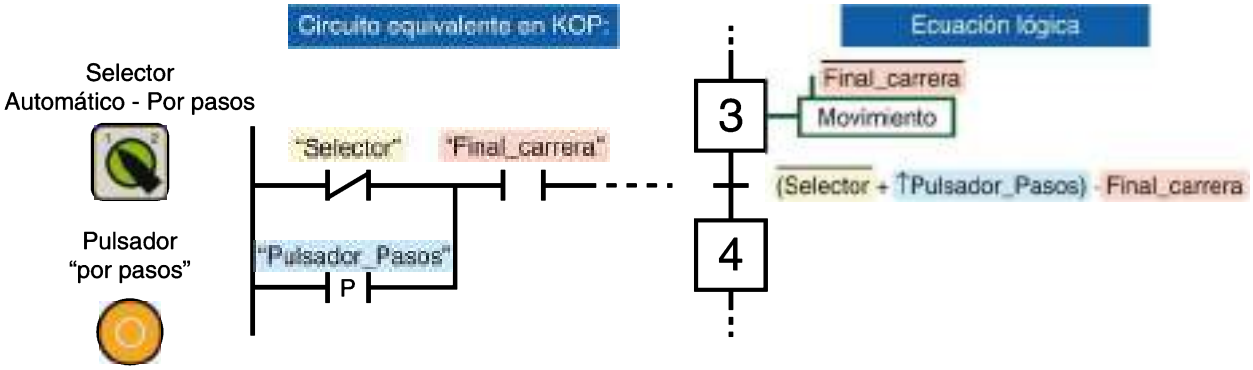


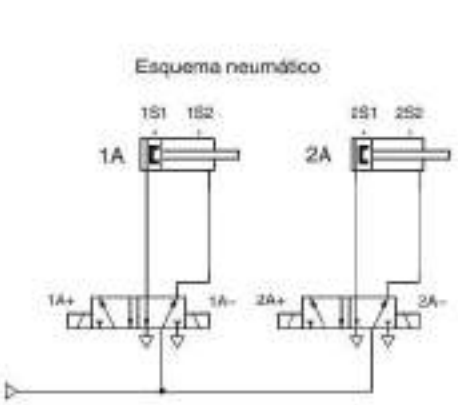
Figura 6.33. Ejemplo de de uso del modo por pasos en las transiciones.

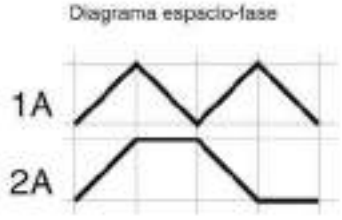
Ejemplos

En el siguiente ejemplo se muestra el control de una secuencia de movimientos de dos cilindros neumáticos.

En todas las transiciones, además de los finales de carrera que se accionan cuando los cilindros han realizado su recorrido (a la extensión o la recogida), se ha conectado en serie el bloque usado para el funcionamiento por pasos.

Para evitar que las electroválvulas se queden activadas cuando se cumplen las etapas en el modo semiautomático, cada acción está condicionada a la señal negada del final de carrera que detiene el movimiento.





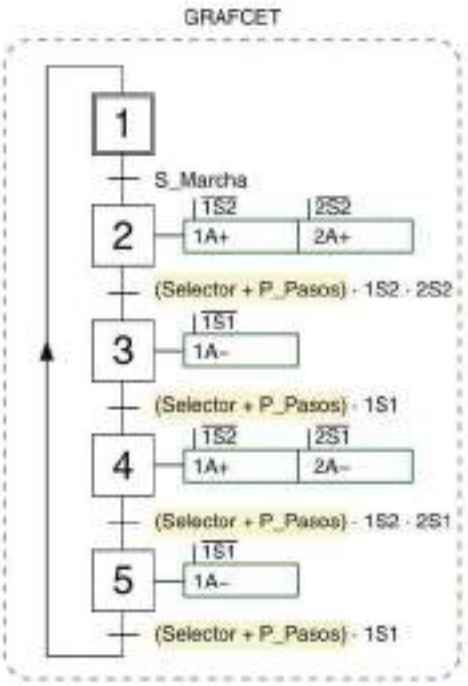


Figura 6.34. Ejemplo de GRAFCET con funcionamiento por pasos.

6.6. Contadores en GRAFCET

Las transiciones no solamente se pueden flanquear mediante señales digitales, también es posible configurarlas para que sean receptivas a la comparación de valores numéricos como, por ejemplo, los almacenados en los contadores.

El uso de contadores es muy habitual en todo tipo de procesos automatizados.

Con los contadores es posible tomar decisiones en función de un número de eventos y, por tanto, también son de gran utilidad para su uso en las secuencias basadas en GRAFCET.

En ellas para implementarlos hay que representar lo siguiente:

- En etapas:
  - Incremento y decremento del valor del contador.
  - Puesta a 0 o RESET del contador.
- En transiciones:
  - Evaluación del valor almacenado en la variable del contador.

De forma genérica, la sintaxis que utilizamos para representar los diferentes usos y operaciones de los contadores en el GRAFCET se ilustra de la siguiente manera:

- Contar: Contador:=Contador+1 o simplemente Contador+1
- Descontar: Contador:=Contador–1 o Contador–1
- Resetear contador: Contador:=0
- Evaluar comparación: Contador==10; Contador>=12, etc.

Ejemplo

En el siguiente ejemplo se muestra cómo usar un contador en un GRAFCET.

Cada vez que se llega a la etapa 9, el contador se incrementa en 1.

Estando en dicha etapa se evalúan dos transiciones que son receptivas a la señal del sensor S6 y a la del valor de la variable del contador Cnt1:

- Si dicho valor es menor o igual a 10, el flujo de la secuencia retorna hacia una etapa superior.
- Si, por el contrario, el valor del contador es mayor de 10, se flanquea la transición representada a la derecha y se alcanza la etapa 10.
- Al entrar en dicha etapa, el contador se pone a 0 y comienza de nuevo su cómputo.

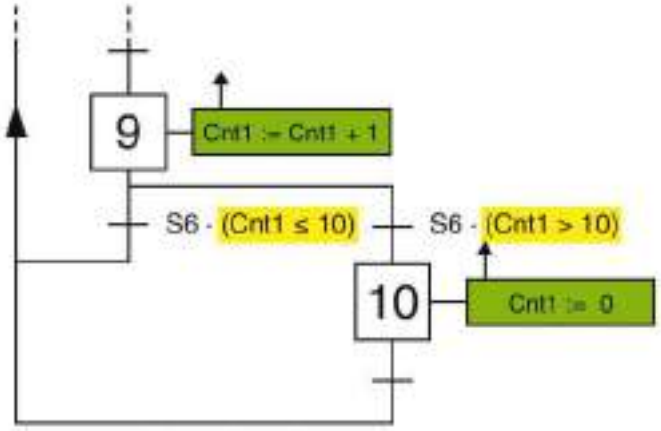


Figura 6.35. Uso del contador en el GRAFCET.

Vocabulary

- Etapa: step.
- Diagrama: chart.
- Diagrama de flujo: flow chart.
- Transición: transition.
- Red: net.
- Enlaces: links.
- Disparo: trigger.
- Acción: action.
- Salto: jump.
- Retorno: return.
- Asignación: assignment.
- Maestro: master.
- Esclavo: slave.
- Home: inicio.
- Macroetapas: macro steps.
- Etapas incluyentes: enclosing steps.
- Ramas alternativas: alternative branches.
- Secuencias paralelas: parallel sequences.

7. Estructuración del GRAFCET

Subrutinas

Los GRAFCET parciales vienen a ser lo que en otros lenguajes se denomina *subrutinas*.

De forma similar a como ocurre con los lenguajes de programación generalistas o para PLC, las secuencias GRAFCET se pueden estructurar por niveles, organizándolas jerárquicamente en cadenas parciales, que se llaman desde otras cadenas de nivel superior.

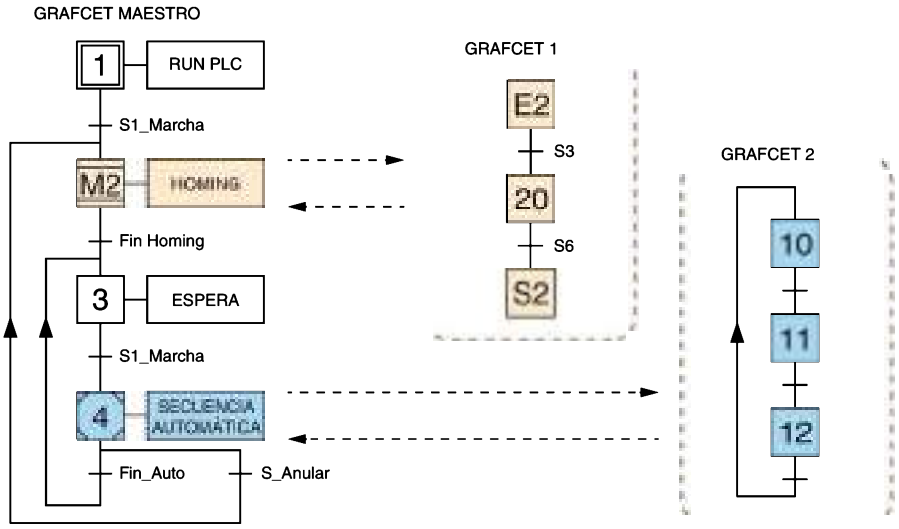


Figura 6.36. Ejemplo de un GRAFCET estructurado.

Para la estructuración del GRAFCET, se puede recurrir a los siguientes recursos:

- GRAFCET parciales o expansiones.
- Etapas macro (macroetapas).
- Etapas incluyentes.
- Acciones de llamada a GRAFCET parciales.

Los símbolos relacionados con estos conceptos son:

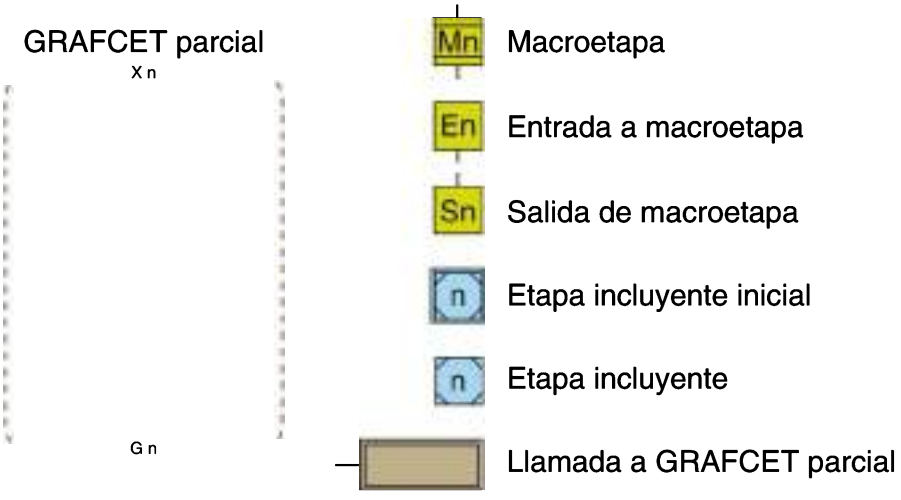


Figura 6.37. Símbolos para estructuración del GRAFCET.

El uso de estos elementos gráficos, permite diseñar las secuencias GRAFCET de forma estructurada.

A continuación se describe como utilizarlos.

7.1. GRAFCET parciales o expansiones

Los GRAFCET parciales son cadenas secuenciales que dependen de otro GRAFCET, normalmente denominado *maestro* o *global*.

Los GRAFCET parciales se identifican con la letra G seguida del número de orden que hace en el circuito. La numeración de las etapas en estas cadenas puede coincidir con las de otros GRAFCET, pero en la misma no puede haber dos etapas con el mismo número.

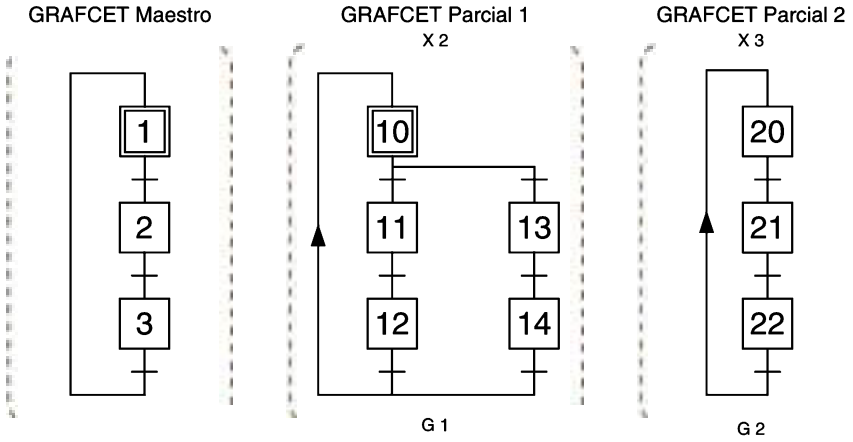


Figura 6.38. GRAFCET parciales.

Bits de etapa

Los bits de etapa de los GRAFCET parciales se identifican con el número del GRAFCET y el de etapa: G1X12.

7.2. Macroetapas (etapas macro)

Las macroetapas son una forma gráfica de estructurar las secuencias por bloques funcionales. Se utilizan para «aligerar» el GRAFCET principal, haciéndolo más entendible y fácil de editar.

Las etapas macro se representan con dos líneas paralelas horizontales en su interior y se identifican con la letra M seguida de un número (M1, M2, M3, etc.).

La secuencia que pertenece a una macroetapa, también denominada *expansión,* se representa mediante una cadena separada del GRAFCET principal. Esta debe disponer de una etapa de entrada y otra de salida, que se deben identificar con las letras E y S respectivamente, seguidas del número de la macroetapa. Así, en la figura del ejemplo las etapas de entrada y de salida de la macroetapa M2 se identifican con E2 y S2.

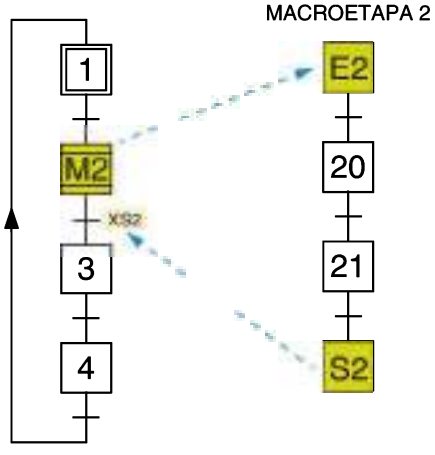


Figura 6.40. Ejemplo de macroetapa.

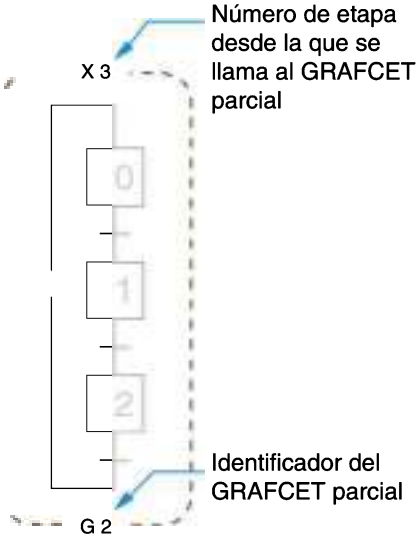


Figura 6.39. Identificación de GRAFCET parcial.

Cadena de la expansión

La cadena de la expansión de una macroetapa puede estar cerrada o no por el bucle cíclico.

Características de la macroetapa:

- No es posible salir de ella hasta que finalice la secuencia asociada.
- Las macroetapas no son reutilizables. Una misma macro no se puede usar dos veces en el mismo programa.

7.3. Etapas incluyentes

Son etapas asociadas a GRAFCET parciales. A diferencia de las macroetapas, las secuencias asociadas a ellas no disponen de etapa de entrada y de salida.

Características de las etapas incluyentes:

- Las etapas incluyentes se pueden abandonar en cualquier momento, interrumpiendo de esa manera la ejecución del GRAFCET parcial que se llama desde ella. En este sentido, se trata como una etapa estándar y, si la transición o transiciones que permiten salir de ella se flanquea, la etapa se desactiva.
- De igual forma que las macroetapas, las etapas incluyentes no son reutilizables.
- Los GRAFCET parciales asociados a ellas puede tener secuencias con bucle cíclico o de tipo abierto con etapa fuente y etapa pozo.
- La etapa por la que comienza la cadena parcial debe marcarse con un asterisco (*).
- Las etapas incluyentes pueden ser también de tipo inicial.
- El GRAFCET parcial asociado a una etapa de este tipo deja de ejecutarse con un bit de la última etapa del mismo, usado en la transición que permite la salida de la etapa incluyente.

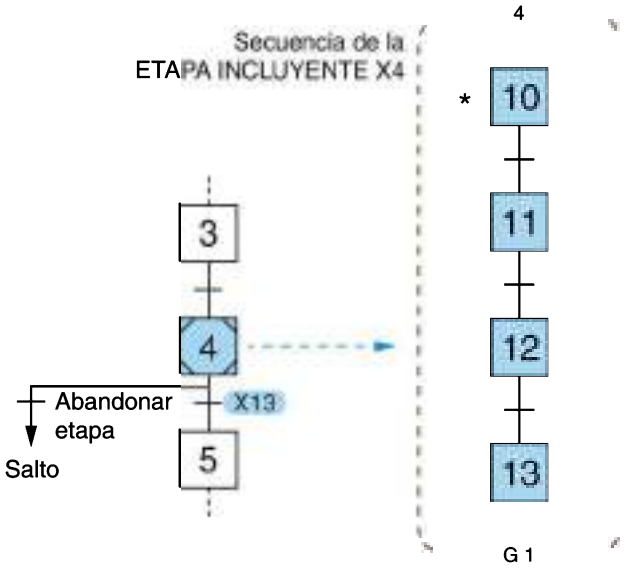


Figura 6.41. Ejemplo de etapa incluyente.

Actividades

2. Diseña un GRAFCET con las siguientes características:
   - El GRAFCET maestro debe tener cuatro etapas, de las cuales dos de ellas serán incluyentes.
   - Los GRAFCET parciales asociados a las etapas incluyentes deben ser de secuencia única y tener 4 y 5 etapas respectivamente.
   - Se debe prever salir de las etapas incluyentes para volver a la etapa inicial mediante un pulsador denominado S_Anular.

Ejemplo

A continuación se muestra un ejemplo de uso de las macroetapas y las etapas incluyentes. Para ello se ha utilizado como modelo una máquina plegadora de chapa.

Descripción del funcionamiento

Las láminas de chapa se insertan manualmente. Cuando se acciona el pulsador de marcha, si el detector inductivo detecta que hay una chapa en la base, el cilindro fijador baja para presionar la chapa. Cuando este acciona el final de carrera S4, el cilindro doblador baja hasta tocar S6. Una vez que la chapa ha sido doblada, ambos cilindros suben al mismo tiempo hasta que cada uno alcanza su correspondiente final de carrera (S3 y S5).

Anular

Estando la máquina en marcha, si se acciona el pulsador S_Anular, los dos cilindros deben recogerse hasta alcanzar la parte superior. Los movimientos deben hacerse de forma independiente. Primero debe subir el doblador y luego el fijador. Todo el tiempo que se esté realizando esta operación se debe señalizar con la lámpara intermitente.

Arranque en caliente

Si se ejecuta el programa después de un arranque en caliente, la lámpara debe ponerse intermitente. En esa situación se debe accionar el pulsador de marcha para que ambos cilindros suban hasta la posición de reposo (la de la figura), de la misma forma que se ha descrito en el proceso Anular.

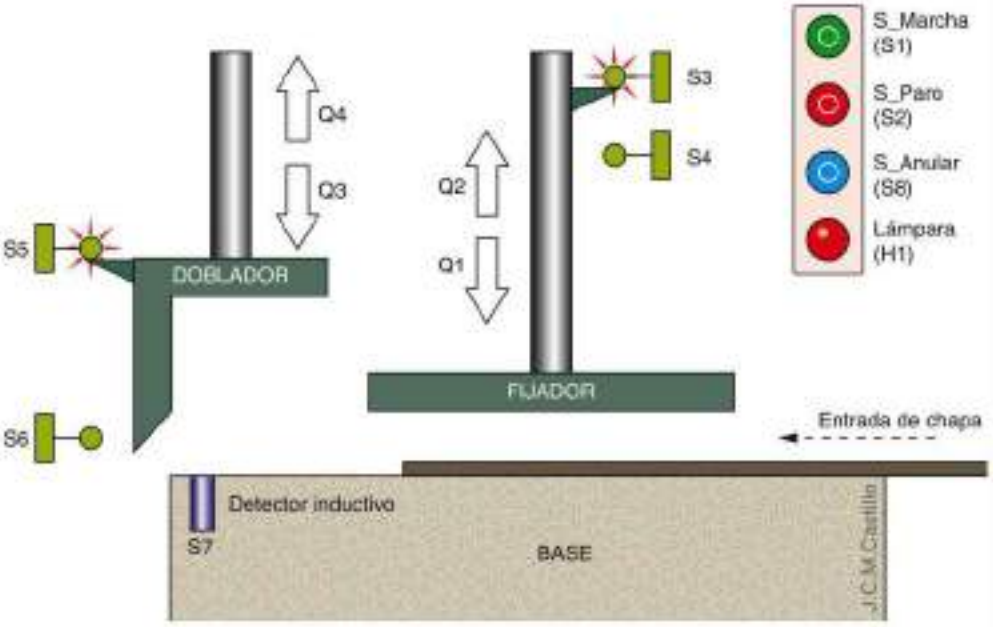


Figura 6.42. Solución al ejemplo de la plegadora.

Solución

Para el proceso de arranque en caliente o Homing se ha optado por usar una macroetapa, ya que no es necesario salir de ella antes de que este finalice.

Para la secuencia principal se ha usado una etapa incluyente, de la cual se puede salir mediante el pulsador S_Anular, aunque no haya finalizado el GRAFCET asociado.

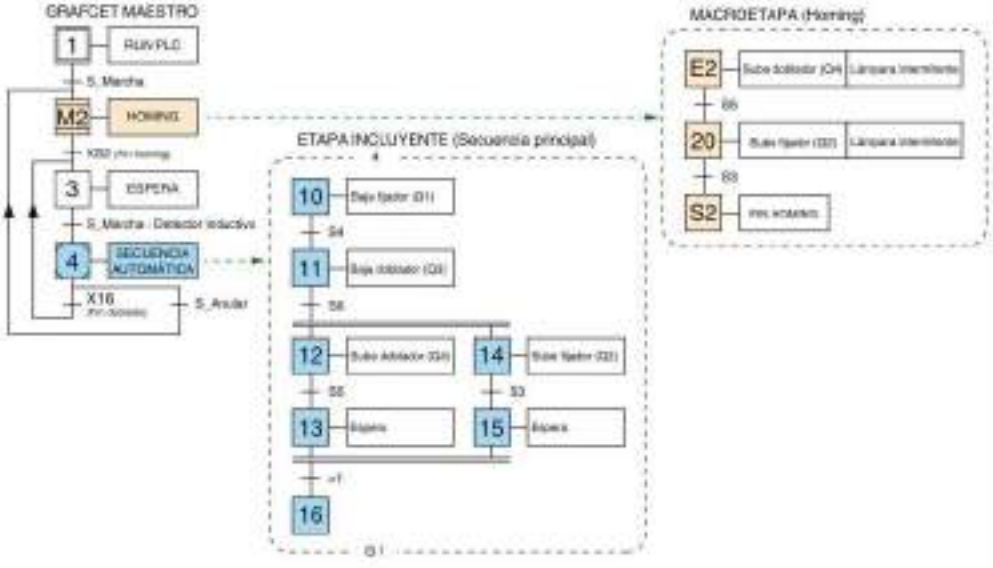


Figura 6.43. Plegadora de chapa.

7.4. Acciones de llamada a GRAFCET parciales

Acciones de llamada

Las etiquetas para gestionar este tipo de acciones son las siguientes:

Gn	Identificador de GRAFCET parcial
GnXn	Bit de etapa de variable parcial

Son acciones asociadas a etiquetas del GRAFCET que, en lugar de controlar señales o variables, hacen llamadas a otros GRAFCET. Se pueden usar individualmente en las etapas o en conjunto con otras acciones y se representan como las etiquetas de acciones pero con un doble rectángulo. De igual forma que ocurre con las etapas incluyentes, la etapa donde se realiza la llama a GRAFCET parciales puede abandonarse sin necesidad de finalizar la secuencia.

La forma de llamar a los GRAFCET parciales se realiza con las acciones mostradas en los siguientes ejemplos:

Ejemplos

Gn {*}:

Congelación de un GRAFCET parcial. La secuencia de este GRAFCET se queda en la etapa en que se encontraba pero no evoluciona. Es importante tener en cuenta que las acciones no se desactivan.

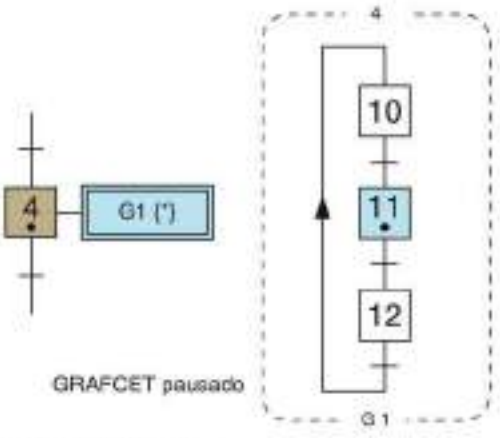


Figura 6.44. Pausar un GRAFCET.

Gn {...}:

Llamada al GRAFECT parcial con activación de la etapa o etapas iniciales que este contenga.

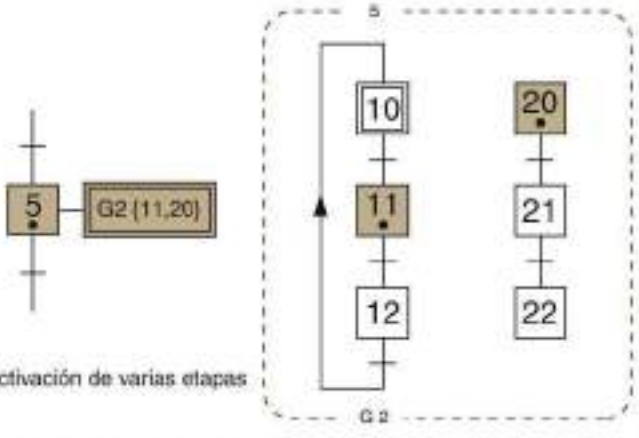


Figura 6.45. Activar una o más etapas de un GRAFCET parcial.

Gn {}:

Desactivación de todas las etapas de GRAFCET parcial.

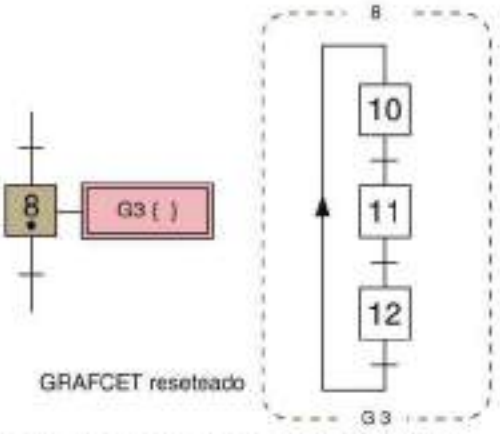


Figura 6.46. Reseteo o desactivación de un GRAFCET parcial.

Gn {INIT}:

Llamada a un GRAFCET parcial con la activación a varias de sus etapas.

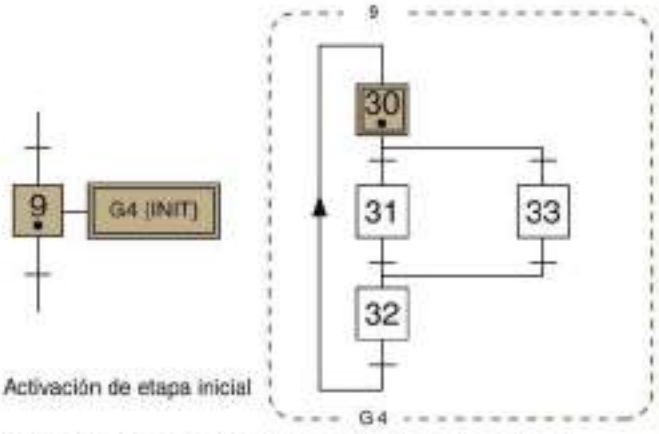


Figura 6.47. Llamar a un GRAFCET parcial y activar su etapa inicial.

Ejemplo

En el siguiente ejemplo se muestra el uso de GRAFCET parciales a modo de subrutinas. Para ello se ha utilizado el proceso automático del taladro, en su funcionamiento con y sin desahogo, visto en la figura 6.32 de un ejemplo anterior.

Las secuencias con y sin desahogo se han separado en dos GRAFCET parciales (G1 y G2). En la etapa X4 se hace una llamada al GRAFCET parcial G1, que corresponde al proceso de taladrado con desahogo. En la etapa X5 se hace la llamada al GRAFCET parcial G2, cuyo funcionamiento corresponde a taladrado sin desahogo.

Cuando el selector se encuentra en la posición 1 y se acciona el pulsador S_Marcha se ejecutan las etapas X4 y X5 de forma consecutiva y se realiza así la operación de taladrado con desahogo. Si, por el contrario, el selector se encuentra en la posición 2, la ejecución de la etapa X4 se omite y se realiza solamente el taladrado sin desahogo, correspondiente al GRAFCET G2.

Los bits de etapa X14 y X23 se utilizan para flanquear las transiciones que están por debajo de las etapas en las que se hacen las llamadas a los GRAFCET parciales y así poder continuar con la evolución del GRAFCET maestro.

Las transiciones con retardo a la conexión representadas en el esquema se usan para evitar que las acciones de subida y bajada del taladro produzcan un cortocircuito, si está controlado mediante un motor trifásico.

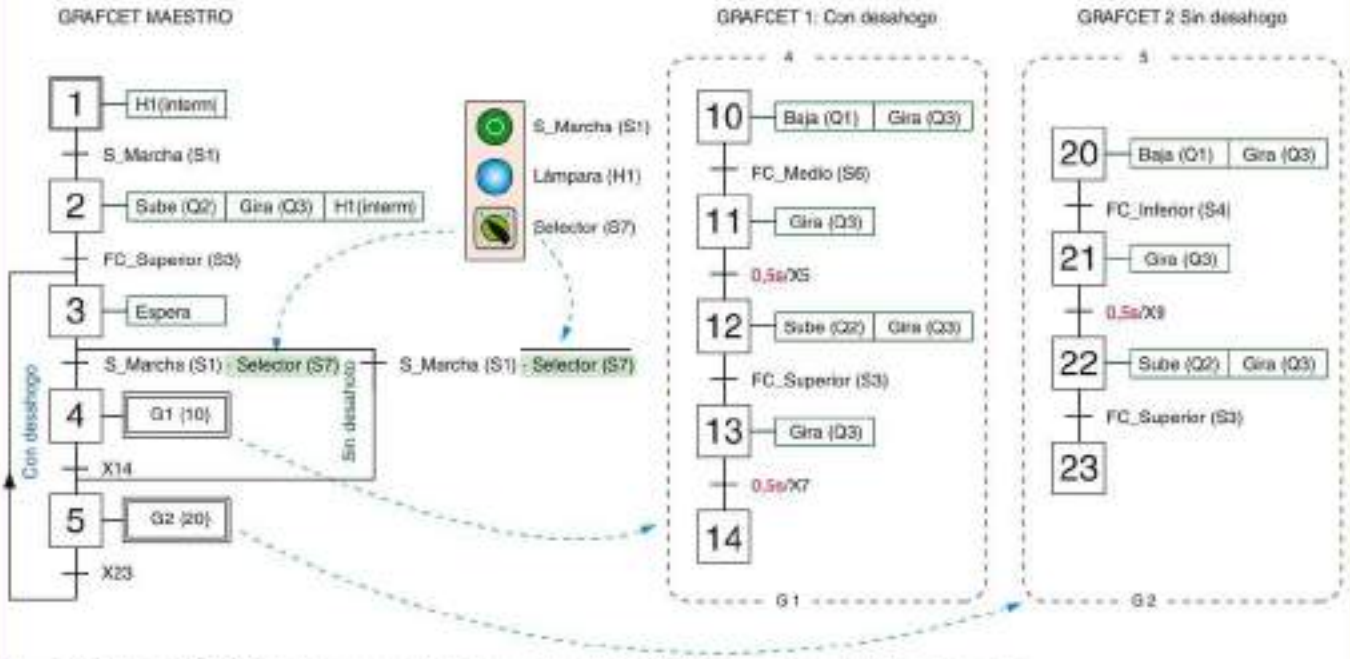


Figura 6.48. Uso de GRAFCET parciales como subrutinas para dar solución al taladro con dos modos de funcionamiento.

Actividades

3. En el ejemplo anterior para el control del taladro con o sin desahogo, implementa los siguientes modos de funcionamiento:
   - **Parada:** detiene el proceso cuando se considere necesario. La reanudación del movimiento se debe hacer accionando nuevamente el pulsador de marcha.
   - **Anular:** permite llevar el taladro a la posición de reposo cuando se acciona el pulsador S_Anular. Esta operación se debe hacer con la lámpara H1 funcionando de forma intermitente y girando la broca, para evitar que se rompa cuando sale de la pieza.

PRÁCTICA PROFESIONAL RESUELTA

Herramientas

- PC con software para el diseño del GRAFCET (ejemplo: Fluidsim)

Material

- Cuaderno de trabajo y herramientas de dibujo

GRAFCET de taladro semiautomático con cargador de piezas

Objetivo

- Diseñar el GRAFCET de un automatismo secuencial
- Utilizar un contador para la toma de decisiones en el flujo de la secuencia.
- Diseñar un sistema de rearme que permita anular la secuencia principal
- Diseñar la cadena GRAFCET para llevar el automatismo a la posición de origen tras un arranque en caliente.

Precauciones

- Hay que tener en cuenta que la subida y bajada del taladro se hace con un motor trifásico, por lo que no es posible invertir el sentido de giro de manera instantánea, ya que produciría un cortocircuito en el circuito de fuerza que controla el motor.
- Para poder cargar y descargar pieza, el taladro siempre debe estar en la posición superior, ya que la broca se rompe si se hace de otra manera.
- La broca siempre debe girar en el mismo sentido, tanto al entrar como al salir de la pieza.

Desarrollo

El proceso que se va a automatizar se basa en el taladro mostrado en los ejemplos del libro, al cual se le ha añadido un sistema para la carga automática de piezas

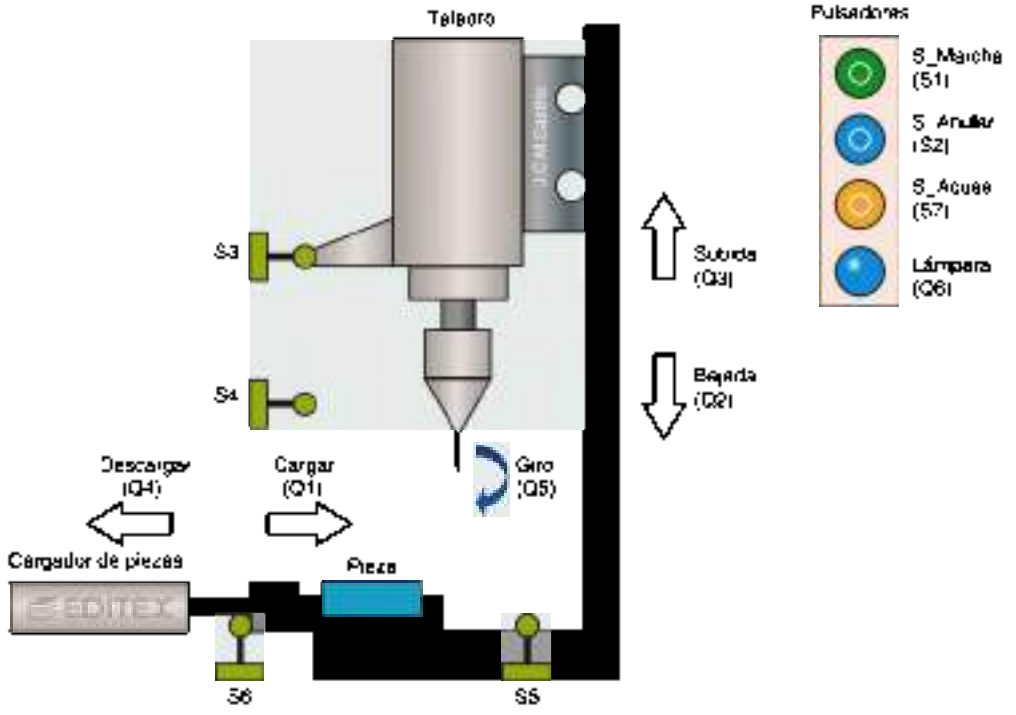


Figura 6.49. Taladro semiautomático con cargador de piezas

Funcionamiento

El GRAFCET se debe diseñar basándose en el siguiente funcionamiento:

Ciclo de funcionamiento estándar

1. Después de accionar el pulsador de marcha, el cargador sitúa la pieza debajo del taladro hasta alcanzar el fin de carrera S5.
2. El taladro baja girando la broca.
3. Cuando se encuentra en la parte inferior, en la que supuestamente se ha taladrado por completo la pieza, se acciona el final de carrera S4.
4. Antes de invertir el sentido de giro para subir el taladro se debe producir una pequeña temporización de aproximadamente 0,5 segundos. Esto evita el cortocircuito en el circuito de fuerza del motor del taladro.
5. El taladro sube hasta accionar el fin de carrera superior S3.
6. El cargador retira la pieza recién taladrada hasta alcanzar el final de carrera S6.
7. Después de un arranque en caliente, el proceso debe informar al operario de esa situación mediante la activación de la lámpara Q6 de forma intermitente.
8. Se acciona el pulsador de marcha.
9. Los elementos del taladro se llevan a la posición de reposo mostrada en la figura. Primero debe subir el taladro y, solamente cuando se encuentre en la parte superior, se debe recoger el cargador de piezas.

Anular

10. Si la máquina está ejecutando la secuencia estándar y se acciona el pulsador S_Anular, los elementos deben llevarse al origen de la misma forma que se ha descrito para la cadena del *Homing*.

Aviso de cambio de pieza

11. Cuando se han realizado diez operaciones de taladrado, el sistema debe avisar al operario para que cambie la broca.
12. Cuando esto ocurra, la secuencia debe pasar a un estado de parada en el que la lámpara avise de tal situación mediante su encendido intermitente.
13. Una vez cambiada la broca, la secuencia debe salir de este estado mediante un pulsador de acuse de cambio de broca (S7) y volver al estado en el que se puede comenzar de nuevo la ejecución del ciclo de taladrado estándar.

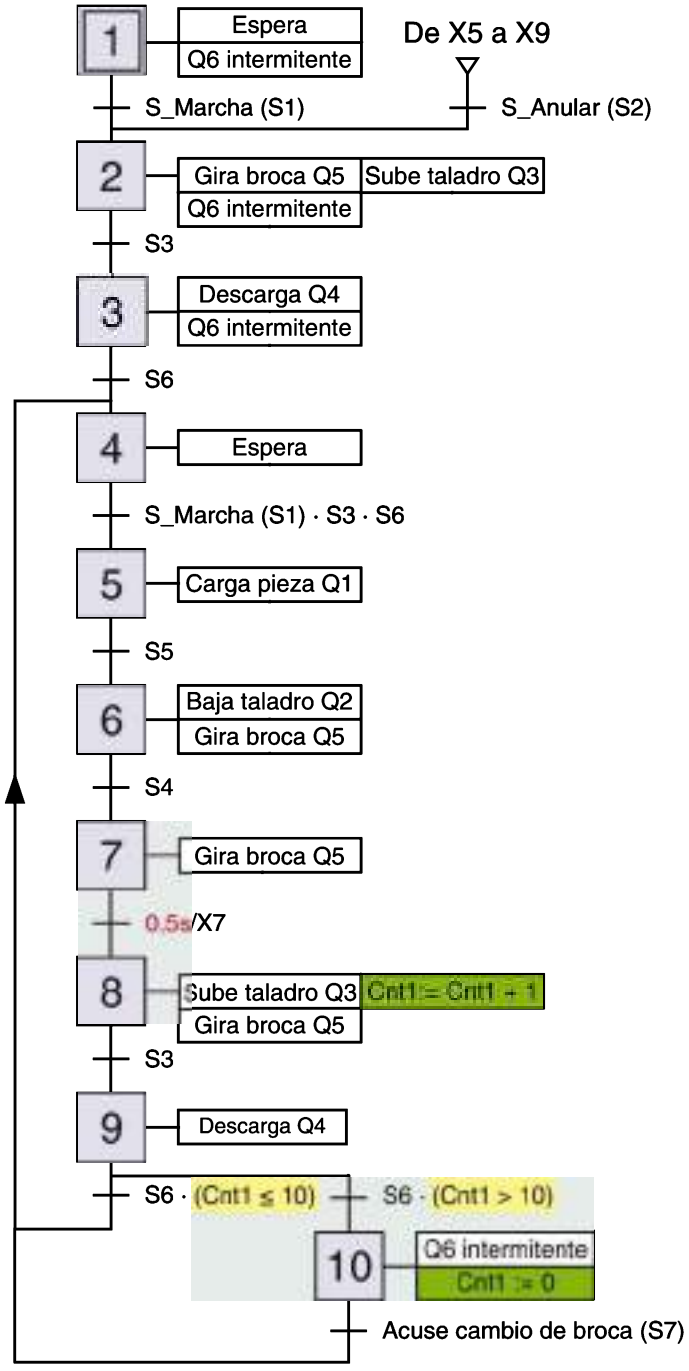


Figura 6.50. GRAFCET solución.

TEST DE EVALUACIÓN

RESUELVE EN TU CUADERNO O BLOC DE NOTAS

1. **GRAFCET es:**
   a) Un diagrama.
   b) Un esquema eléctrico.
   c) Un lenguaje de programación.
   d) Una asociación técnica.

2. **SFC es:**
   a) Un lenguaje de programación.
   b) Una asociación técnica.
   c) Un esquema eléctrico.
   d) Un tipo de datos usado en las secuencias.

3. **Las variables asociadas a las etapas se representan con la letra:**
   a) E.
   b) M.
   c) X.
   d) S.

4. **Si en un GRAFCET hay que indicar que una lámpara señalizadora se debe activar, ¿dónde hay que hacerlo?**
   a) En los bucles continuos.
   b) Cuando el pulsador no está accionando.
   c) En los indicadores de dirección.
   d) En las etapas.

5. **Si en una etapa hay una acción que dice «Salida:=1», ¿qué significa?**
   a) Es una salida activada con flanco.
   b) Es lo mismo que activar la salida con SET.
   c) Es una acción condicionada.
   d) Se incrementa en uno el valor de un contador.

6. **La sintaxis 3s/X10 en una transición significa que:**
   a) La etapa X10 se abandorará 3 segundos después de entrar en ella.
   b) Se llegará a la etapa X10 en 3 segundos.
   c) Las acciones de las transiciones se activarán después de 3 segundos.
   d) Es un retardo a la desconexión de 3 segundos.

7. **En un GRAFCET de secuencias opcionales de dos caminos con una única etapa inicial...:**
   a) En algún momento puede haber dos etapas activadas a la vez.
   b) Nunca podrá haber dos etapas activadas a la vez.
   c) Esa configuración no es posible.
   d) Solamente se podrá pasar por el segundo camino si se diseña con dos etapas iniciales.

8. **Si después de una etapa nos encontramos con dos líneas paralelas horizontales, significa que:**
   a) Se comienzan secuencias opcionales.
   b) Se comienzan secuencias simultáneas.
   c) Es un salto o retorno a dos etapas a la vez.
   d) Está mal representado el GRAFCET.

9. **Señala la afirmación que es correcta en relación a las macroetapas:**
   a) La misma macroetapa se puede utilizar todas las veces que se desee en un GRAFCET.
   b) No se puede salir de ella hasta que no finalice.
   c) Se puede salir de ella en cualquier momento.
   d) Se utiliza para hacer saltos en el GRAFCET.

10. **Una etapa pozo es:**
    a) Una etapa inicial.
    b) Una etapa que no tienen transición previa.
    c) Una etapa con llamada a un GRAFCET parcial.
    d) Una etapa que no tiene transición de salida.

ACTIVIDADES FINALES

En todas las actividades se debe diseñar el GRAFCET de los procesos descritos. Siempre que sea posible, se usarán GRAFCET parciales para estructurar las secuencias. Estos podrán ser llamados desde macroetapas, etapas incluyentes o como acciones basadas en subrutinas.

1. Sistema de llenado de contenedores en función de su tamaño.

   El sistema consta de: una cinta transportadora que permite el desplazamiento de cajas contenedoras en ambos sentidos y tres surtidores de material que se abren y cierran con cilindros neumáticos. Debajo de cada surtidor se ha instalado un cilindro, en posición normalmente extendida, que detiene las cajas para que sean cargadas con el material correspondiente.

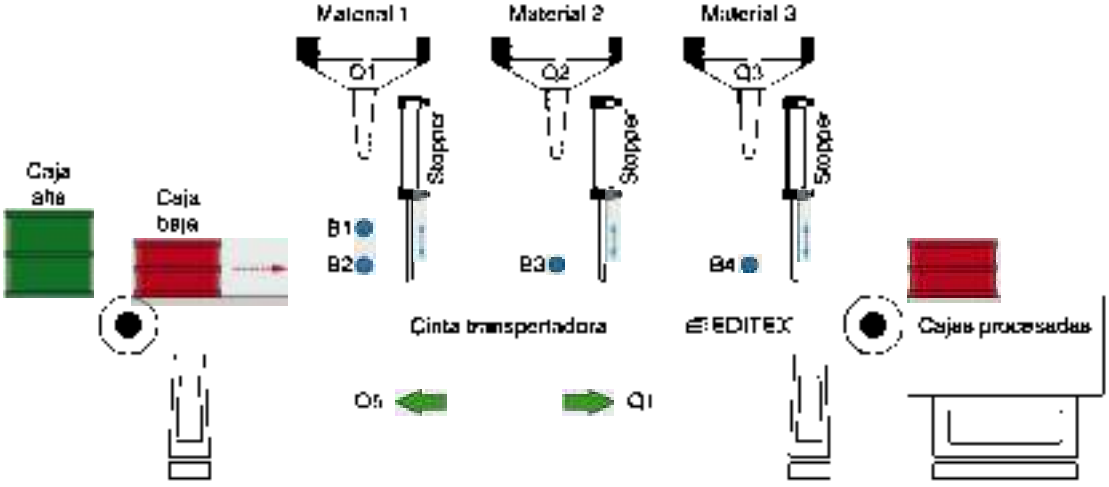


Figura 6.51. Sistema de llenado de cajas contenedoras

Funcionamiento:

- Después de accionar el pulsador de marcha, la cinta transportadora gira a derechas (Q1).
- Las cajas se cargan manualmente por el lado izquierdo de la cinta y se desplazan hasta chocar con los cilindros que funcionan como stopper.
- La cinta transportadora no debe detenerse mientras las piezas están siendo llenadas.
- Los detectores B1 y B2 comprueban la altura de las cajas, que se llenarán según su altura.
- Los stoppers retienen las cajas debajo de cada surtidor el tiempo indicado en la tabla de mezclas. Estos se elevan una vez que ha transcurrido el tiempo programado y desplazan la carga hasta el siguiente punto del proceso.

	Material 1	Material 2	Material 2
Caja alta	4 s	5 s	3 s
Caja baja	--	2 s	4 s

Parada:

- El proceso se detiene al accionar el pulsador de paro. Se reanuda el movimiento si se acciona nuevamente el pulsador de marcha.

Anular:

- Si se acciona el pulsador «Anular», los surtidores se cierran, los stopper se elevan y la caja se lleva al punto de carga en el lado izquierdo de la cinta, que se para después de un tiempo.

Homing:

- Después de un arranque en caliente se debe encender un piloto de forma intermitente, que avisa de que debe ser accionado el pulsador de marcha. Después de esto, el comportamiento de la máquina debe ser similar al descrito en la función Anular.

ACTIVIDADES FINALES
continuación

2. Robot de soldadura.

El sistema consiste en un sencillo robot giratorio que aplica puntos de soldadura a tres piezas que se encuentran en posiciones fijas. El movimiento izquierda-derecha del sistema mecánico se controla con un motor de corriente continua. El conjunto dispone de dos finales de carrera: uno para controlar la posición de origen o de referenciado (*home*) y otro para comprobar cuándo el sistema mecánico reductor acoplado al eje del robot, que a su vez mueve el brazo de soldadura, da una vuelta.

Robot de soldadura

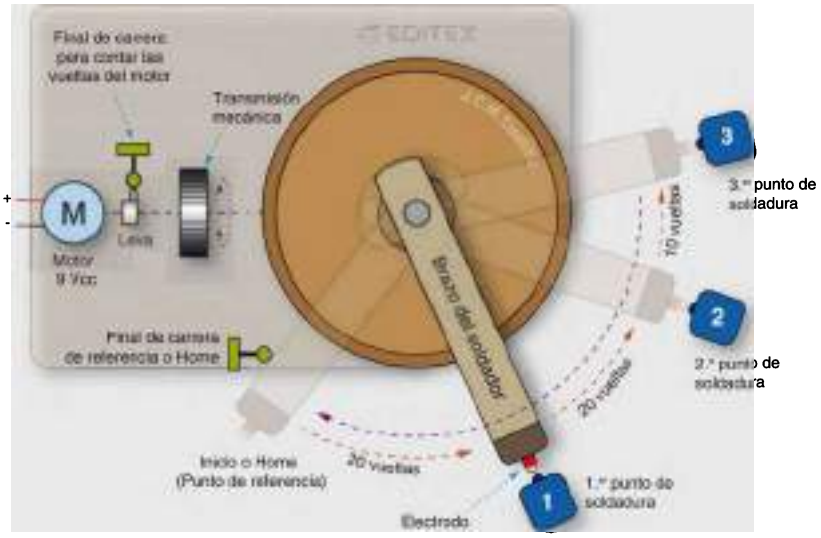


Figura 6.52. Robot de soldadura.

Funcionamiento:

- El brazo parte de una posición de referencia, a la cual se llega después de un arranque en caliente, una operación de cancelación o un fin de ciclo de producción.
- La posición del robot se controla en función del número de pulsos detectados en el interruptor que controla el número de vueltas del motor. Así, se ha estimado que, para alcanzar los diferentes puntos de soldadura, tomando como referencia el origen, el motor debe dar 20, 40 y 50 vueltas, respectivamente.
- Es sistema dispone de dos modos de funcionamiento: soldadura por puntos individuales y soldadura continua de todos los puntos. En el primer caso, cada vez que se acciona un pulsador del cuadro de mando el robot se desplaza al punto correspondiente. En el segundo caso, al accionar un pulsador específico para tal fin el robot realiza la soldadura de todas las piezas de forma consecutiva.
- Después de cada operación de soldadura el robot vuelve a la posición de origen.
- La operación de soldadura se simula parando el brazo en el punto correspondiente durante 5 segundos y activando la lámpara que simula el electrodo de forma intermitente.

Parada:

- La secuencia se detiene cuando se acciona el correspondiente pulsador de paro y se reanuda mediante cualquiera de los pulsadores de marcha de que dispone el sistema.

***Homing*:**

- Después de un arranque en caliente el sistema debe avisar al operario de dicha situación, activando una lámpara de señalización de forma intermitente. En ese estado, si se acciona el pulsador de marcha el brazo del robot busca el punto de referencia.

Anular:

- Si se acciona el pulsador «Anular» se cancela cualquier operación de soldadura y el brazo retorna al punto de origen.

PRÁCTICA PROFESIONAL PROPUESTA 1

Herramientas

- PC con software para el diseño del GRAFCET (ejemplo: Fluidsim)

Material

- Cuaderno de trabajo y herramientas de dibujo

GRAFCET de un taladro con cargador de piezas y desahogo

Objetivos

- Diseñar un GRAFCET de secuencias opcionales.
- Implementar un sistema de parada y de rearme de la máquina.
- Prever el comportamiento de la máquina ante un arranque en caliente.
- Macroetapas, etapas incluyentes o subrutinas.

Precauciones

Las mismas que las descritas en la Práctica Profesional Resuelta de esta unidad.

Desarrollo

1. El proceso que se va a automatizar toma como base el automatismo descrito en la Práctica Profesional Resuelta de esta unidad, al que se le ha incorporado un tercer final de carrera en el recorrido en el que sube y baja el taladro. Este final de carrera va a permitir perforar las piezas en dos tiempos.
2. Así, el GRAFCET se debe diseñar para dos modos de funcionamiento: uno con desahogo y otro sin desahogo, que se seleccionan mediante un interruptor rotativo de dos posiciones. Una vez seleccionado el modo de funcionamiento, se debe accionar el pulsador de marcha.
3. El botón de parada debe detener el proceso en cualquier momento, pudiéndose reanudar de nuevo mediante la acción sobre el pulsador de marcha.
4. El uso del pulsador A_Anular y S_Acuse debe ser el mismo que el planteado en la Práctica Profesional Resuelta.
5. El GRAFCET se debe resolver intentando utilizar secuencias parciales basadas en macroetapas, etapas incluyentes o subrutinas.

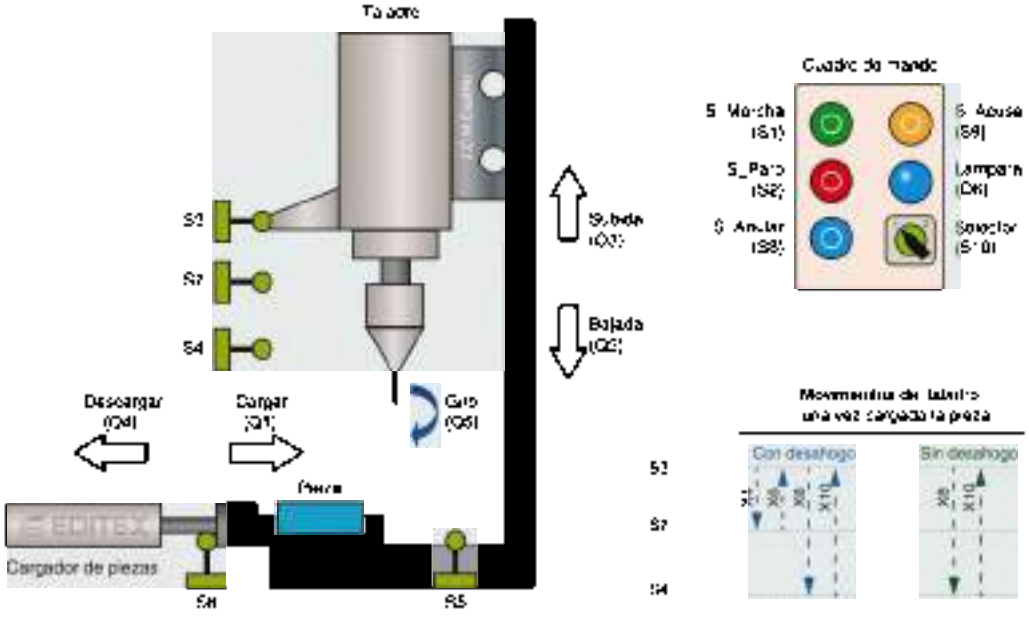


Figura 6.53. Taladro con cargador de piezas con y sin desahogo.

PRÁCTICA PROFESIONAL PROPUESTA 2

Herramientas

- PC con software para el diseño del GRAFCET (ejemplo: Fluidsim)

Material

- Cuaderno de trabajo y herramientas de dibujo

Sistema automático de tratamiento de piezas

Objetivo

- Diseñar un GRAFCET de secuencias opcionales.
- Implementar un sistema de parada y de rearme de la máquina.
- Prever el comportamiento de la máquina ante un arranque en caliente.
- Diseño de GRAFCET parciales para la mejor estructuración de las secuencias.

Precauciones

- El electroimán no se desactiva cuando se detiene la secuencia.
- La pieza debe ser transportada siempre con el carro elevado.
- Los finales de carrera del carro se mueven con él.

Desarrollo

1. El proceso debe comenzar siempre con el electroimán colocado en la parte superior izquierda de la máquina. Es decir, con él elevado y en la posición del final de carrera S1.
2. Se debe prever la secuencia del *Homing* para que el electroimán se coloque en dicha posición después de la intervención del operario sobre el pulsador de marcha.
3. Se pueden tratar dos tipos de piezas, en función de si se acciona el pulsador de marcha 1 o el pulsador de marcha 2:
   - Las del primer tipo se sumergirán durante 5 segundos en la cuba del producto 1 y durante 7 segundos en la cuba del producto 2.
   - Las del segundo tipo solamente pasarán por la segunda cuba durante 10 segundos.
4. Todas las piezas tratadas se depositarán en el punto de descarga, de donde se retirarán manualmente.
5. El desplazamiento de las piezas se realizará con el electroimán elevado.

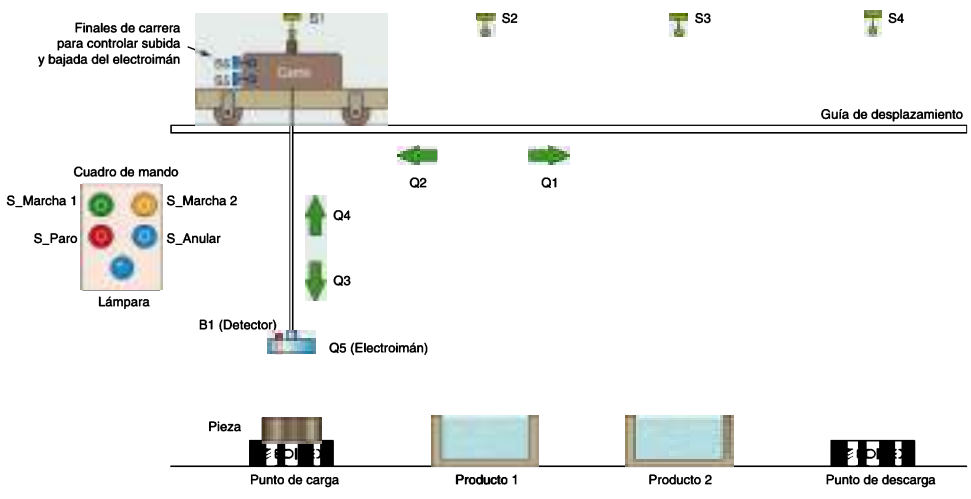
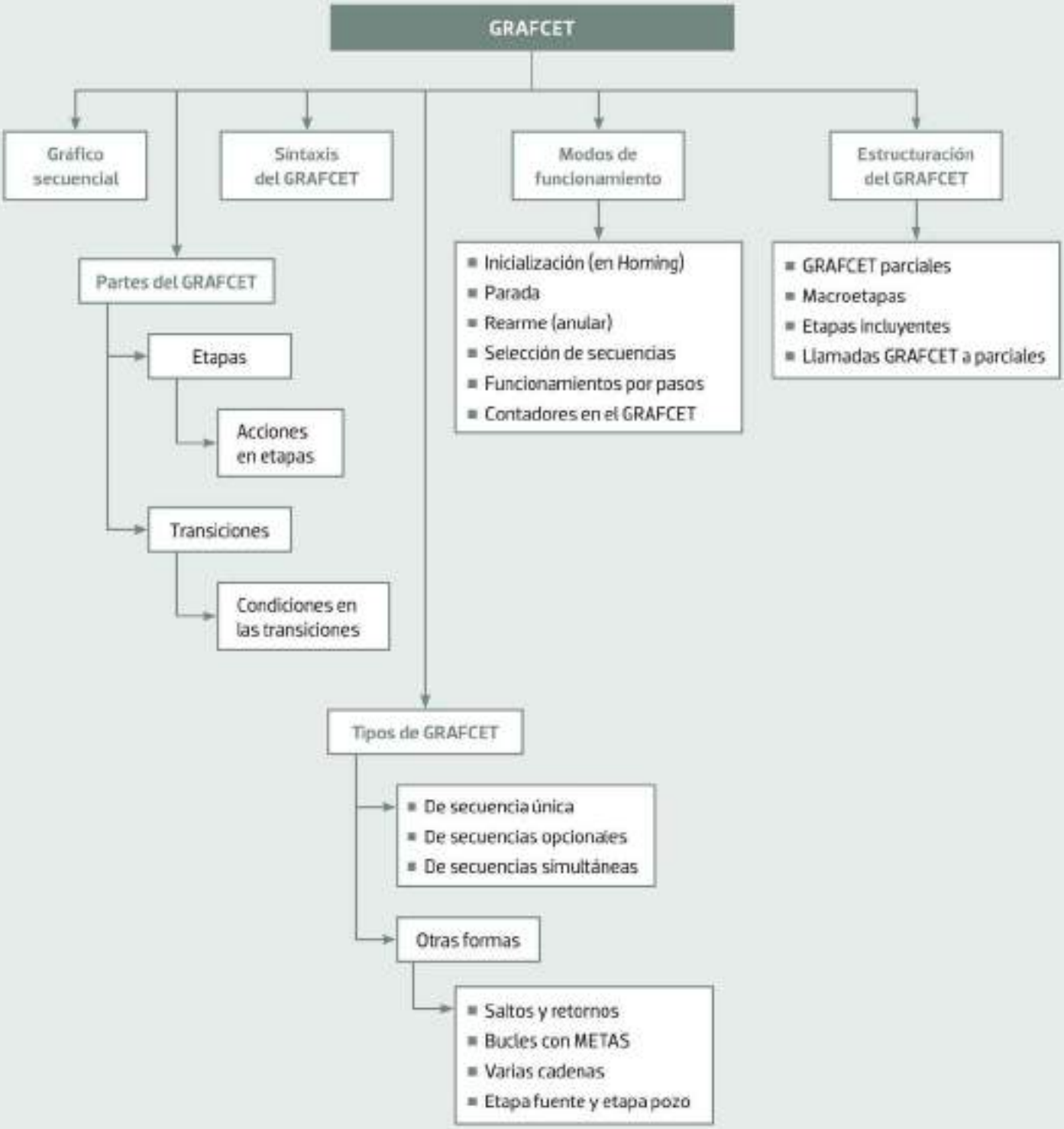


Figura 6.54. Tratamiento de piezas.

6. La acción sobre el pulsador de parada debe detener el desplazamiento de las piezas. En este caso, nunca se debe desactivar el electroimán.
7. La acción sobre el pulsador de anular debe cancelar el tratamiento de cualquier tipo de pieza y depositarla siempre en el punto de descarga.

EN RESUMEN


GRAFCET
Gráfico secuencial
Sintaxis del GRAFCET
Modos de funcionamiento
Estructuración del GRAFCET
Partes del GRAFCET
Etapas
Acciones en etapas
Transiciones
Condiciones en las transiciones
Inicialización (en Homing)
Parada
Rearme (anular)
Selección de secuencias
Funcionamientos por pasos
Contadores en el GRAFCET
GRAFCET parciales
Macroetapas
Etapas incluyentes
Llamadas GRAFCET a parciales
Tipos de GRAFCET
De secuencia única
De secuencias opcionales
De secuencias simultáneas
Otras formas
Saltos y retornos
Bucles con METAS
Varias cadenas
Etapa fuente y etapa pozo

7 GRAFCET en lenguaje de contactos (KOP)

Vamos a conocer...

1. Introducción
2. Administración del GRAFCET
3. Programación de la zona secuencial
4. Programación de la zona de inicialización
5. Programación de la zona de acciones
6. Zona no secuencial

PRÁCTICA PROFESIONAL RESUELTA

Programación del GRAFCET del taladro con desahogo en lenguaje de contactos

PRÁCTICAS PROFESIONALES PROPUESTAS

1. GRAFCET simultáneos
2. GRAFCET para el control de un semáforo

Y al finalizar esta unidad...

- Conocerás cómo se administran las diferentes zonas de un GRAFCET en bloques de programación.
- Asociarás los elementos de los GRAFCET a variables del lenguaje de programación.
- Programarás diferentes cadenas GRAFCET en lenguaje de contactos.
- Utilizarás un sistema metódico para la programación de los GRAFCET.
- Programarás las diferentes acciones que se pueden utilizar en las secuencias basadas en GRAFCET.
- Resolverás, programarás y comprobarás varios casos prácticos basados en GRAFCET.

1. Introducción

Las secuencias basadas en GRAFCET se pueden implementar en cualquier lenguaje de programación, bien de tipo generalista o bien cualquiera de los destinados al control de procesos industriales basados en PLC.

Aquí se utiliza el lenguaje de contactos KOP para implementarlo en los dispositivos de Siemens que utilizan STEP 7. Todos los ejemplos mostrados son válidos para cualquiera de los autómatas de Siemens, S7-300, S7-1200 y S7-1500, aunque podría haber pequeñas diferencias en los primeros.

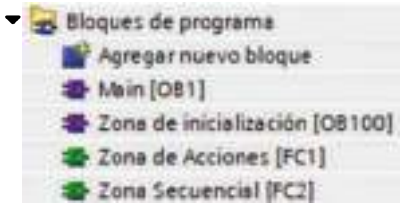


Figura 7.1. Bloques de programación.

2. Administración del GRAFCET

La programación de un GRAFCET se estructura en diferentes zonas:

- Zona de inicialización.
- Zona secuencial.
- Zona de acciones.

La siguiente figura muestra de forma gráfica las diferentes zonas de administración.

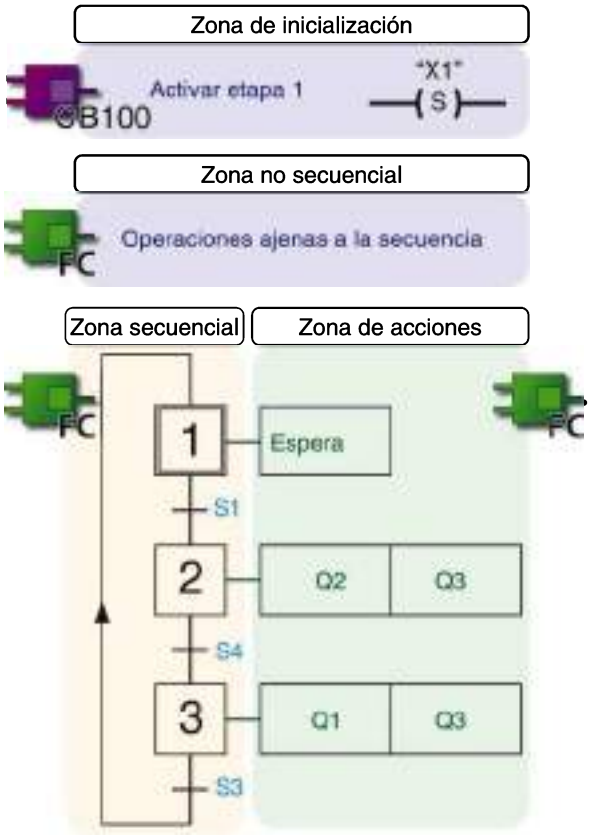


Figura 7.2. Zonas para administrar un GRAFCET.

Estas zonas se organizan en bloques de programación: FC para las zonas secuenciales y zona de acciones, y OB100 para la zona de inicialización.

2.1. Zona de inicialización

La zona de inicialización es aquella en la que se representa lo que tiene que ocurrir en el momento de iniciar la ejecución del programa.

Uso del OB100

En los autómatas S7-1200 y S7-1500 se puede prescindir del uso del OB100 si se considera adecuado, ya que la inicialización se puede realizar con un bit de la denominada *marca de arranque* o del *primer ciclo*. Esta marca se debe activar en el dispositivo, ya que está deshabilitada por defecto.

En esta zona de programación se deben activar las etapas iniciales del GRAFCET y desactivar las restantes. Además, si es necesario, se pueden inicializar valores del programa como contadores, valores de consigna, etc.

El programa de la zona preliminar se ejecuta cuando el PLC pasa de STOP a RUN. Para ello, en los autómatas de Siemens se utiliza el bloque de arranque OB100. Así, todo lo programado en él se ejecutará solo cuando se haga un arranque del sistema.

2.2. Zona secuencial

En la zona secuencial se programa el encadenamiento de etapas y transiciones del GRAFCET.

Dicho de otro modo, es la parte del programa en la que se define cómo evoluciona la secuencia del GRAFCET.

La zona secuencial se escribe en un bloque FC (función) que se llama desde el bloque de organización principal OB1.

Si la secuencia dispone de más de una cadena GRAFCET, como podrían ser las correspondientes a GRAFCET parciales, estas se pueden programar en bloques FC independientes.

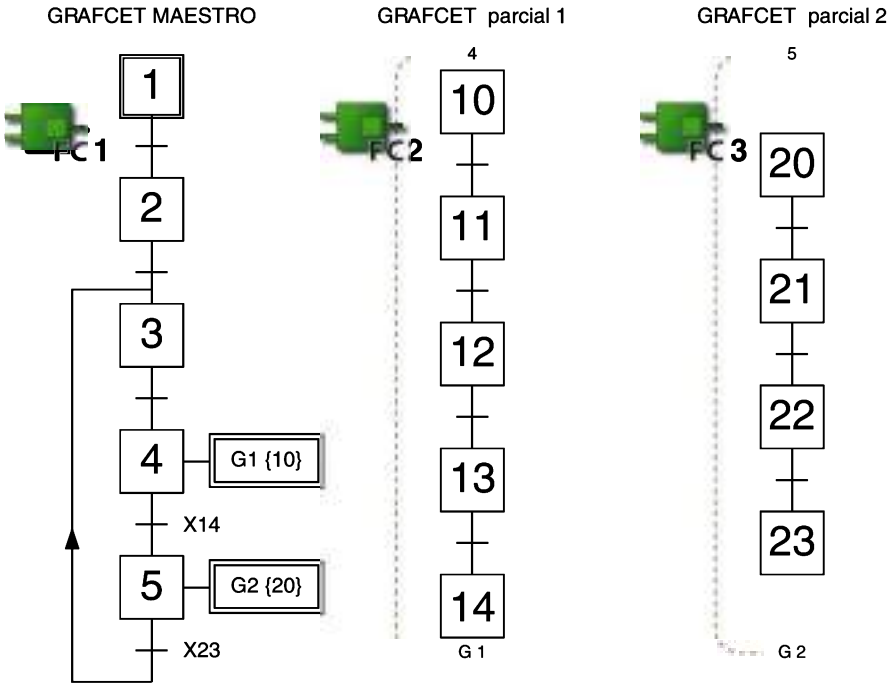


Figura 7.4. Organización de la zona secuencial en varios bloques de programación.

2.3. Zona de acciones

La zona de acciones es aquella en la que se ejecutan las acciones de cada una de las etapas de la secuencia.

Es aconsejable usar un único FC para programar la zona de acciones, especialmente para trabajar con acciones en las salidas. No obstante, si la complejidad del programa así lo requiere, es posible dividir esta zona en varios FC, siempre que no se repitan las mismas asignaciones en varios de ellos.

Al final de esta unidad se estudiará la forma de hacer llamadas a GRAFCET parciales como si se tratasen de acciones. En este caso, es aconsejable crear un FC específico para esta tarea, independiente del FC de la zona de acciones global.

TIA PORTAL

TIA PORTAL permite crear grupos en la carpeta de «Bloques de programa» que facilitan la organización de los bloques de programación del GRAFCET.

Figura 7.3. Ejemplo de organización de bloques en TIA PORTAL.

3. Programación de la zona secuencial

A continuación se describe detalladamente cómo realizar las diferentes operaciones de programación en cada una de las zonas anteriormente descritas, comenzando por la zona secuencial.

La programación del GRAFCET, en lenguaje de contactos, se hace de forma metódica, siguiendo un patrón de representación de la secuencia y las acciones que se producen en cada uno de los estados.

En esta parte se describe cómo programar la evolución de GRAFCET.

3.1. Direccionamiento de las etapas

A cada etapa del GRAFCET se le asocia un bit de la zona «memoria de marcas». El direccionamiento se puede comenzar en cualquier *byte*, pero es importante que todos los bits usados para las etapas sean consecutivos o, al menos, organizados de esa manera para cada cadena GRAFCET. Esto permitirá controlar en bloque todas las marcas pertenecientes a una misma cadena.

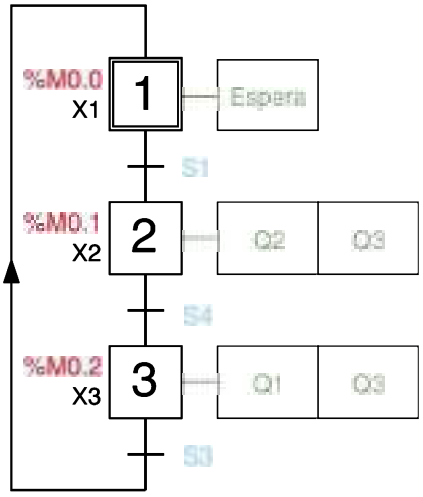


Figura 7.5. Asociación de marcas a etapas.

Organización de las etapas		
Etapa	Bits de marcas	Nombres variables
Etapa 1 (Inicial)	%M0.0	X1
Etapa 2	%M0.1	X2
Etapa 3	%M0.2	X3
...	...	...
Etapa 7	%M0.6	X7
Etapa 8	%M1.0	X8
...	...	...

El direccionamiento de marcas no se debe representar en el gráfico del GRAFCET.

3.2. Segmento modelo de etapa

Cada etapa se representa con un segmento modelo similar al de la figura. En este segmento hay que representar: la etapa a la que se llega (destino), la etapa desde la que se viene (origen) y la transición que debe ser flanqueada para llegar al destino.

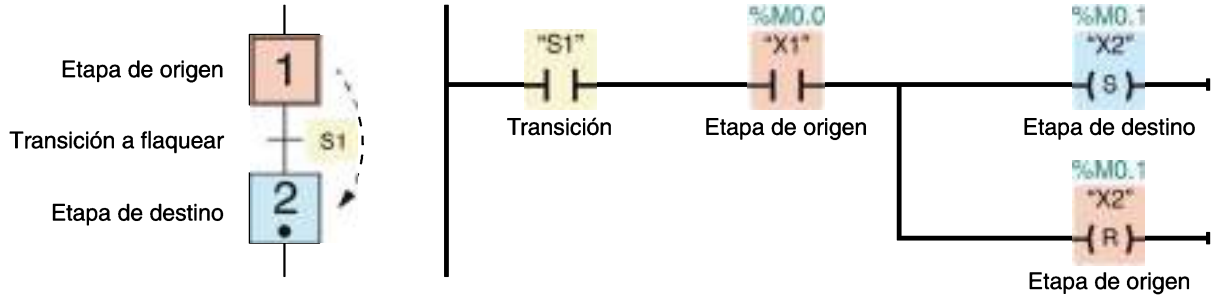


Figura 7.6. Modelo de etapa en KOP.

Para activar el bit de la etapa a la que se desea llegar se utiliza una bobina SET asociada a su marca. Para representar cómo se llega a ella se debe insertar un contacto abierto del bit de la etapa anterior en serie con la condición lógica de la transición. Cuando se activa una etapa siempre es necesario desactivar con RESET la etapa o etapas desde las que se procede.

3.3. Combinaciones lógicas en transiciones

La combinación lógica de las señales de una transición se conecta en serie con el contacto de la etapa desde la que se procede.

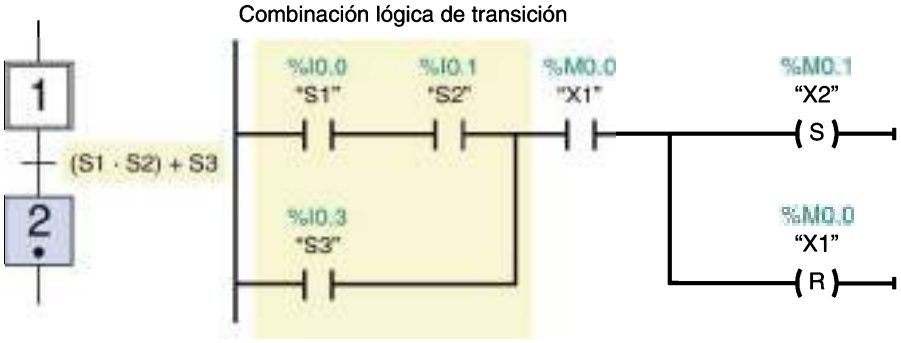


Figura 7.7. Combinaciones lógicas en las transiciones.

3.4. Divergencia de secuencias opcionales (OR)

La programación de las etapas de entrada de los caminos de un GRAFCET basado en secuencias opcionales no es diferente a lo que ya se ha visto para una sola etapa.

Para ello, se sigue el mismo patrón de diseño, es decir: etapa de destino, etapa de origen y transición que se flanquea.

En este caso, todas las etapas de entrada de la divergencia proceden de una etapa común pero con diferentes transiciones.

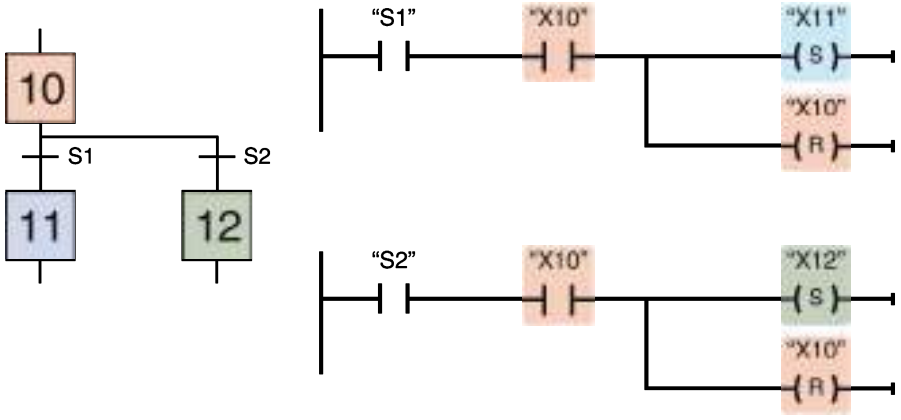


Figura 7.8. Divergencia en OR.

3.5. Convergencia de secuencias opcionales (OR)

Para cerrar una secuencia con varios caminos opcionales es necesario poner en paralelo con la bobina SET de la etapa a la que se llega el RESET de todas las etapas desde las que se procede.

A este bloque de bobinas le preceden las combinaciones lógicas de los contactos que definen los diferentes caminos desde los que se llega a la etapa actual.

Cada uno de estos caminos está formado por una red de contactos en serie, en la que se establece la condición de la etapa y la transición desde la que se viene. Así, debe haber tantas redes en paralelo de este tipo como caminos tenga la convergencia.

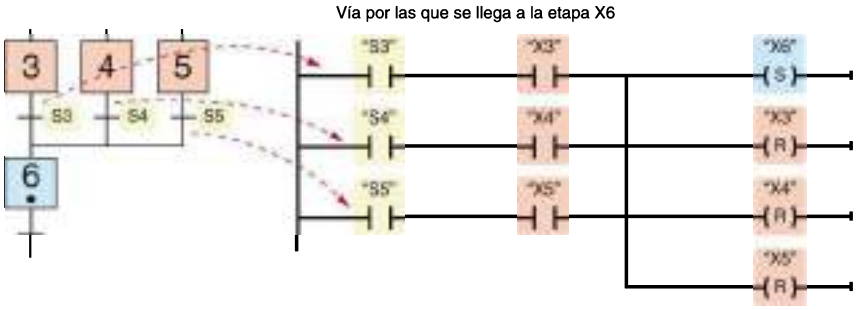


Figura 7.9. Llega a una etapa desde varias secuencias opcionales.

Cuando el número de caminos desde los que se llega a una etapa es elevado, para optimizar el programa, en lugar de poner tantas bobinas RESET como caminos existan, se deben usar funciones para realizar RESET en bloque.

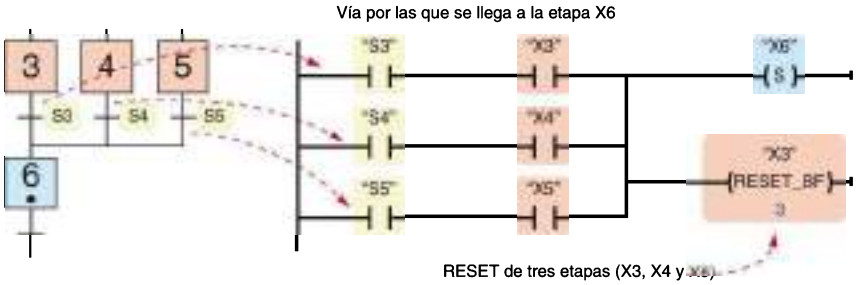


Figura 7.11. RESET de etapas en bloque.

3.6. Divergencia de secuencias simultáneas o sincronizadas (AND)

En una convergencia de secuencias simultáneas se activan tantas etapas como ramas GRAFCET se van a ejecutar a la vez. En el mismo segmento, se deben poner en paralelo todas las bobinas SET de las etapas de destino y la etapa RESET de la de origen.

Es importante no separar las bobinas SET en diferentes segmentos, ya que el programa puede no funcionar.

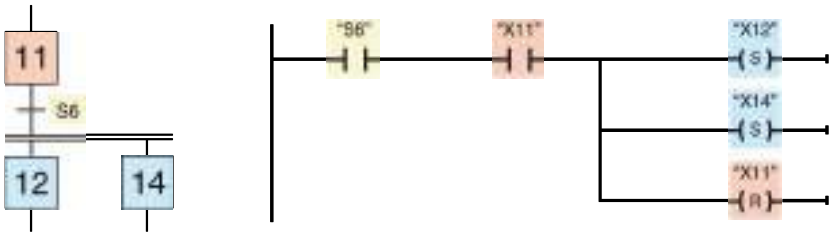


Figura 7.12. Divergencia en Y.

3.7. Convergencia de secuencias simultáneas o sincronizadas (AND)

Las secuencias simultáneas finalizan en etapas de espera de cada uno de los caminos que se quiere sincronizar. Los bits de cada una de estas etapas se combinan en serie para activar la etapa de destino.

En esta situación hay que desactivar con RESET todas las etapas desde las que se procede.

Resetear varios bits

En los S7-300, el bloque para resetear varios bits a la vez tiene el siguiente aspecto:

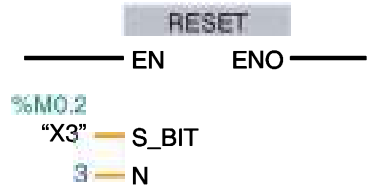


Figura 7.10. RESET en bloque en los S7-300.

Vocabulario

Las ramificaciones de secuencias simultáneas también se suelen denominar *secuencias sincronizadas*.

La transición =1 no dispone de ninguna señal lógica para su flanqueo, pero podría tenerla. Esta representa que «si todos los caminos simultáneos han finalizado, se debe continuar por la secuencia principal».

En el siguiente ejemplo se muestra cómo cuando se han alcanzado todas las etapas finales de los caminos sincronizados (X13 y X15) se activa la etapa X16 de la secuencia principal.

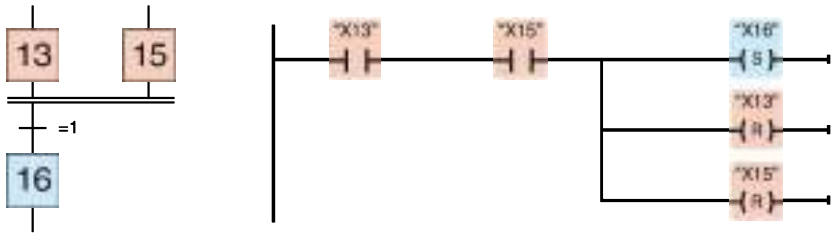


Figura 7.13. Convergencia en Y.

3.8. Temporizaciones en las transiciones

Las transiciones pueden ser flanqueadas después de haber transcurrido un tiempo de permanencia en una etapa. Para ello, se utilizan temporizadores a la conexión, que se activan con la etapa activa, y su señal de salida se emplea en la transición como si de un contacto se tratase.

En la siguiente figura se muestra un ejemplo de transición temporizada:

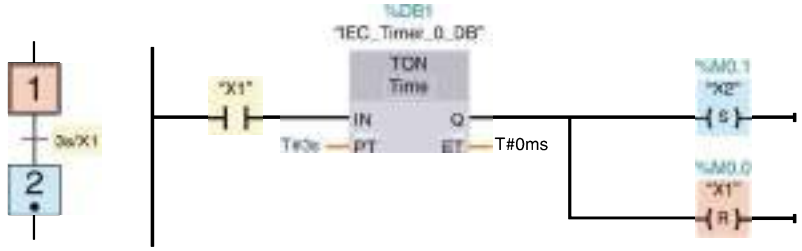


Figura 7.14. Transición temporizada.

Otra forma de realizar esta operación consiste en activar el temporizador en la zona de acciones con el bit de la etapa en la que se desea lanzar la temporización. Luego, en la zona secuencial, en el segmento de etapa, se usa un contacto de este temporizador en la rama de la transición.

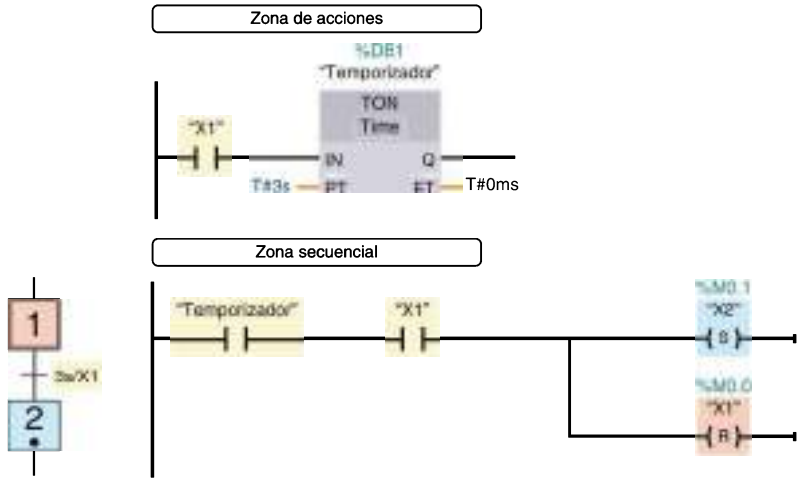


Figura 7.15. Transición temporizada.

3.9. Contadores en las transiciones

El uso de los contadores en el GRAFCET permite direccionar el flujo de la secuencia en función de un valor acumulado.

El cómputo del valor en el contador se gestiona mediante sus entradas (incrementar, decrementar, poner a cero, etc.) desde la zona de acciones.

La evolución de la secuencia en función de este valor se representa en las operaciones lógicas de las transiciones, lo cual se puede hacer de dos formas:

- Con comparaciones.
- Mediante la evaluación del bit de salida del contador.

Uso de comparaciones para direccionar el flujo de un GRAFCET

Para usar este método es necesario utilizar una variable en la que almacenar el valor del contador.

En este caso, en el segmento de la etapa a la que se llega se inserta la comparación de la variable del contador, en serie con el bit de la etapa desde la que se procede.

Variables tipo entero (Int)

El valor de los contadores se almacena en variables tipo entero (Int), como puede ser una marca en formato de palabra MW.

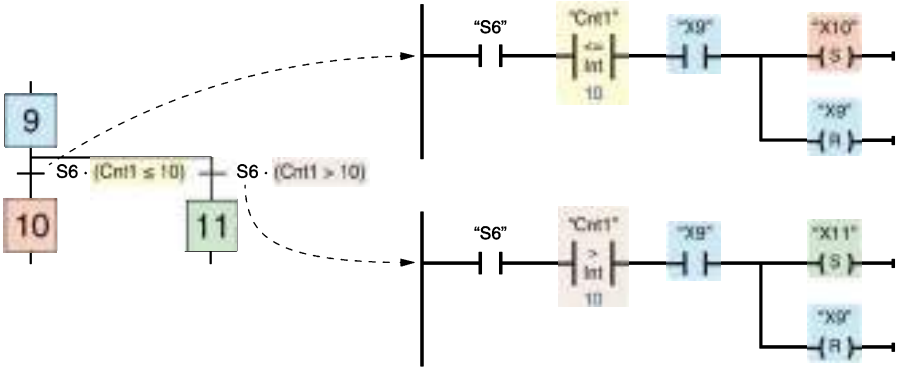


Figura 7.16. Comparaciones de contadores en transiciones.

Uso del bit de salida del contador

Esta opción solo es posible con los contadores IEC, ya que los contadores SIMATIC no disponen de una salida booleana Q que se dispare cuando se alcanza un valor de preselección (PV).

El planteamiento para la programación KOP es similar al anterior, pero las comparaciones se sustituyen por contactos (abiertos o cerrados), asociados a la salida digital del contador, que se insertan en las transiciones, que deben ser receptivas al valor del contador.

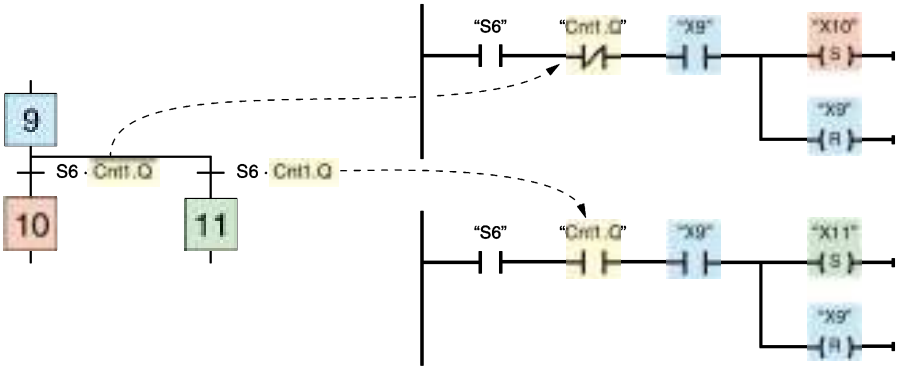


Figura 7.17. Contactos de salida booleana de contadores en transiciones.

3.10. Saltos y retornos desde múltiples etapas

Los saltos y retornos permiten direccionar el flujo del programa a una etapa determinada del GRAFCET.

Esto es especialmente útil para abandonar la secuencia principal y forzarla a ejecutar una parte determinada de la cadena.

Esta operación solamente afecta a la zona secuencial, por lo que no es necesario programar nada al respecto en la zona de acciones.

Para realizar este tipo de saltos o retornos se necesita activar la etapa a la que se desea llegar y desactivar cualquiera desde las que se realiza el salto.

La condición para realizar esta operación se puede hacer de dos formas:

1. Representando las etapas desde las que se desea realizar el salto, en cuyo caso se ponen en paralelo los contactos de los bits de cada una de estas entre sí y el conjunto en serie con el bit de la acción que ejecuta la operación, que en el caso de la figura es S10.
2. Representando las etapas desde las que no se desea realizar el salto, en cuyo caso hay que poner en serie los bits negados de cada una de las etapas, con el contacto de la señal que ejecuta la operación.

Vocabulario

Los **saltos** y **retornos** desde múltiples etapas permiten realizar operaciones de rearme o anulación de las secuencias.

Número de etapas elevado

Cuando el número de etapas desde las que se realiza el salto es elevado, el uso de la 2.ª forma reduce el número de contactos a programar en el segmento en el que se encuentra el bit de la etapa a la se salta.

Recuerda

Esta parte del programa se puede añadir al bloque FC de la zona secuencial ya existente o crear un FC propio para ella.

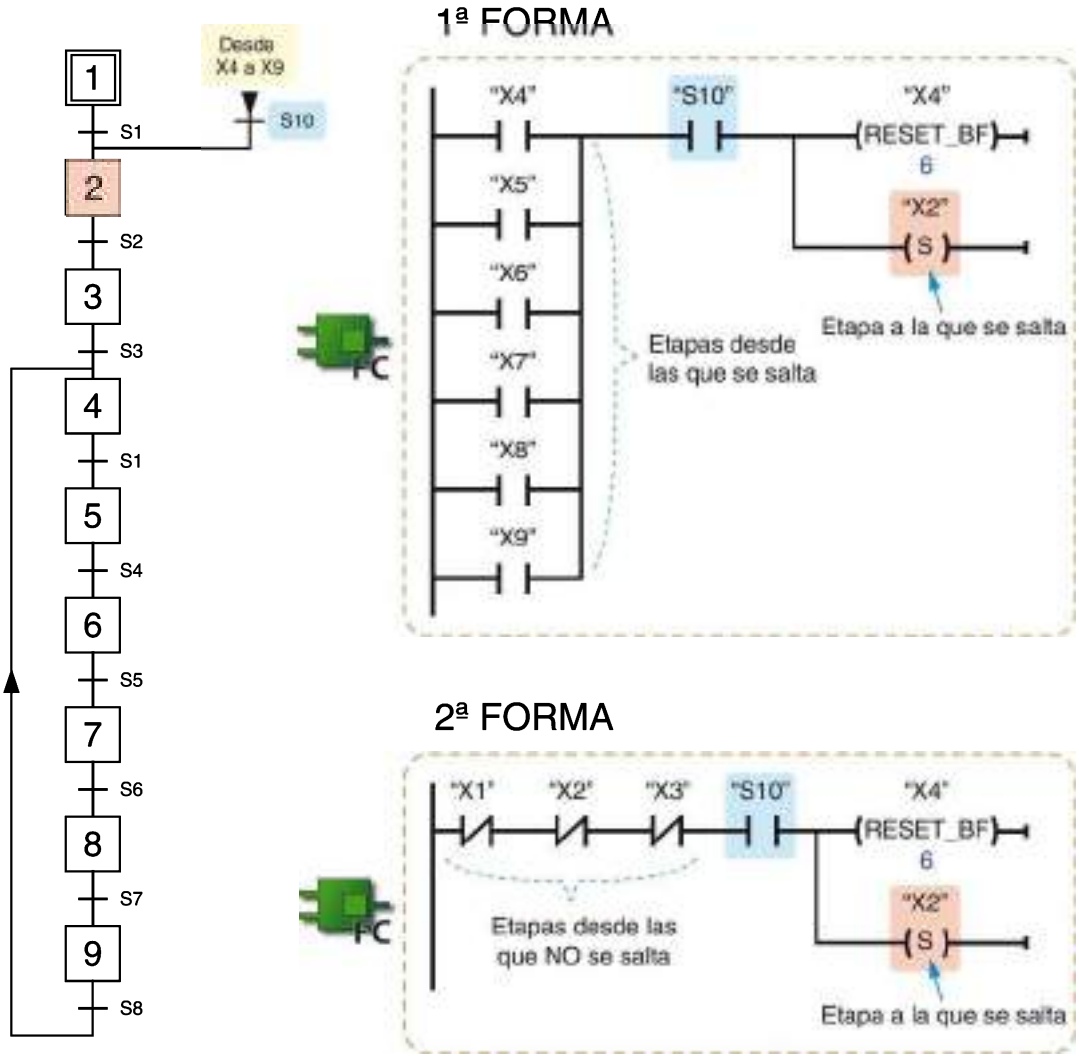


Figura 7.18. Ejemplo de retorno desde múltiples etapas (dos formas).

4. Programación de la zona de inicialización

En la zona de inicialización se escribe el programa que asigna los valores iniciales a las etapas y a los datos del programa que lo requieren.

4.1. Activación de etapas iniciales

Las etapas iniciales se deben activar cuando el PLC pasa de STOP a RUN. En esta situación se deben activar todas las etapas iniciales del GRAFCET, si es que hay más de una, y resetear, si es necesario, las que no lo son.

En todos los modelos de PLC de Siemens (S7-300, S7-1200 y S7-1500) esto se consigue con el bloque de arranque OB100 (STARTUP). En este bloque, se añaden las bobinas SET de las etapas iniciales que se desean activar y de todos los RESET de las etapas que se quieren desactivar. Para realizar esto último, la mejor opción es utilizar instrucciones de programación que faciliten la desactivación de varios bits de marcas en bloque del GRAFCET.

Ejemplo

En el siguiente ejemplo se considera que hay dos cadenas GRAFCET, con un total de 25 etapas. Cuando se ejecuta el OB100 se resetean todas las etapas e inmediatamente se activan la X1 y la X10.

Es aconsejable realizar la desactivación global de etapas usando la operación «desactivar mapa de bits». En este caso, se podría indicar el rango de bits que se desea resetear, sin incluir aquel o aquellos que se van a poner a SET. No obstante, para facilitar la programación, se puede recurrir a desactivar todos los bits de las etapas del GRAFCET e, inmediatamente después, activar con SET aquellos que corresponden a las etapas iniciales.

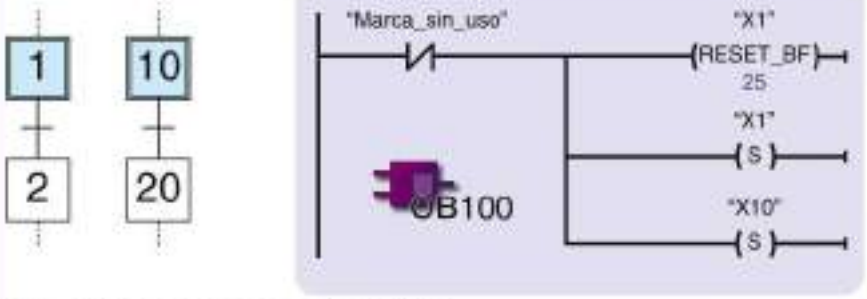


Figura 7.20. Activación de etapas de inicio en OB100.

El contacto cerrado de la denominada marca_sin_uso de la figura es necesario para poder representar la ramificación de las asignaciones. En los S7-300 es obligatorio ponerlo, pero en los S7-1200 y S7-1500 se puede prescindir de él. También es posible configurar las marcas de sistema (solo en los S7-200 y S7-1500) y utilizar el bit denominado AlwaysTrue que, como su nombre indica, mantiene siempre a «1» lógico el contacto asociado.

En los S7-1200 y S7-1500 se puede utilizar el denominado *bit de arranque FirstScan* como sustituto del OB100. Por defecto está deshabilitado y es necesario configurarlo en las denominadas *marcas de sistema* del dispositivo.

Todas las asignaciones que dependen de un contacto asociado a ese bit se ejecutan solamente cuando el PLC pasa de STOP a RUN.

Activación de una etapa

Forma de combinar la activación de una etapa determinada y el RESET de un grupo de etapas, incluida la primera.

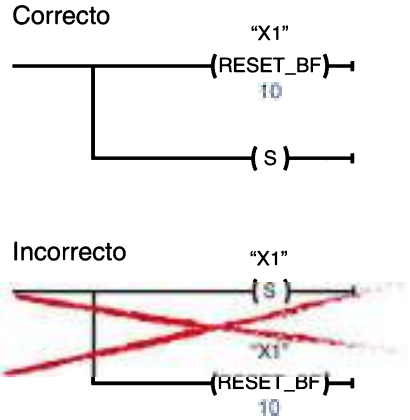


Figura 7.19. Activación de una etapa y RESET del resto.

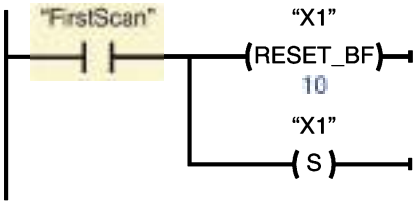


Figura 7.21. Uso del bit de arranque.

5. Programación de la zona de acciones

La zona de acciones es la parte del programa en la que se realizan las asignaciones representadas en las etiquetas de las etapas y pueden ser de diferentes tipos.

5.1. Asignación directa o acción continua

La acción se mantiene siempre que la etapa esté activa y desaparece cuando se abandona dicha etapa.

En el siguiente ejemplo se muestra cómo tres salidas se activan cuando el flujo del programa se encuentra en las etapas 2 y 3 del GRAFCET.

Cuando una salida se activa desde varias etapas se conectan en paralelo tantos contactos de bits de etapa como asignaciones haya en el gráfico secuencial.

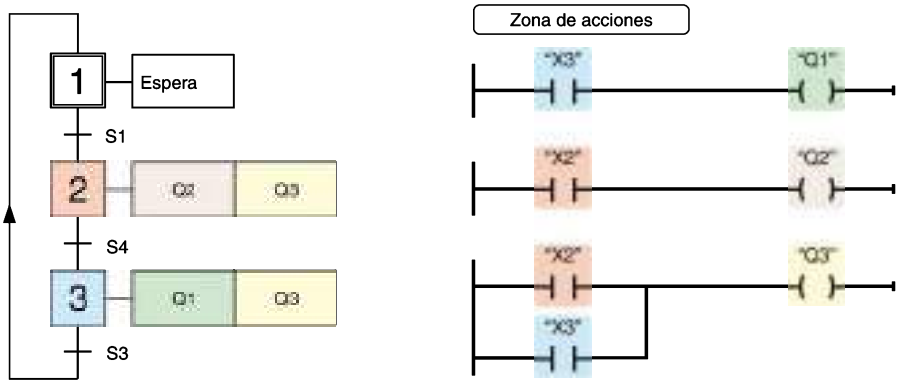


Figura 7.22. Ejemplo de acciones continuas.

5.2. Acción condicionada

La acción se mantiene siempre que la etapa esté activa y se cumpla la condición en la señal.

En el ejemplo de la siguiente figura, la salida Q3 solamente se activa si la secuencia se encuentra en la etapa 11 y está activa la señal de S1.

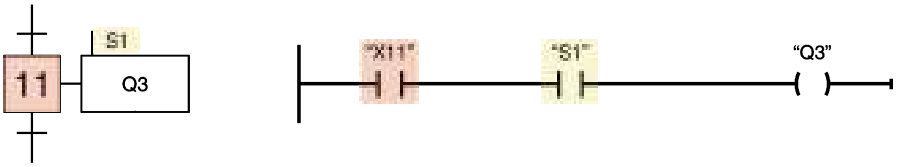


Figura 7.24. Acción condicionada.

5.3. Asignación a la activación (SET)

Pone a «1» lógico una variable booleana. Es el equivalente a poner a SET dicha variable.

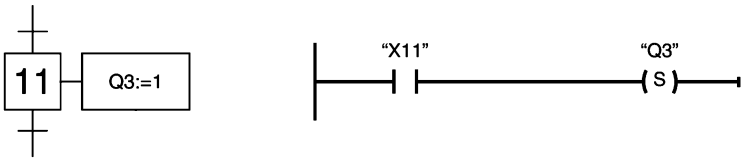


Figura 7.25. Asignación a la activación.

Activación de variables

Si una variable se activa en varias etapas, hay que tener la precaución de no escribir varias veces la asignación a dicha variable en diferentes segmentos. Los programas de PLC se leen de arriba abajo (clico de Scan), por lo que si una asignación a una salida se repite varias veces, solo se ejecutará la última en la que está representada.

Lo correcto es escribir una sola vez la asignación de la salida y poner en paralelo los bits de las etapas desde donde se ejecuta.

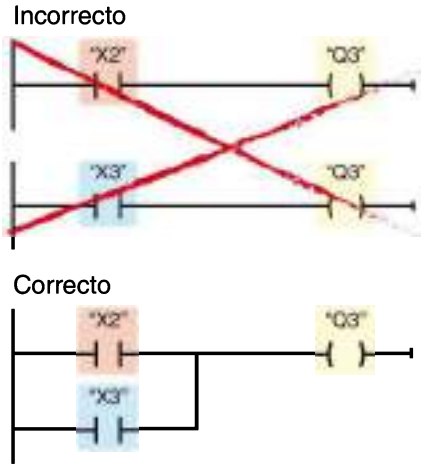


Figura 7.23. Conexión de una salida que se activa en varias etapas.

5.4. Asignación a la desactivación (RESET)

Es el efecto contrario a la acción anterior. Es decir, pone a «0» lógico una variable booleana.

Es el equivalente a poner a RESET una variable que anteriormente estaba a 1.

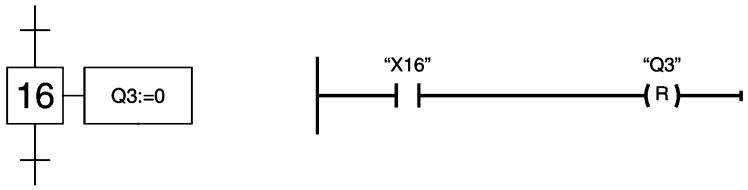


Figura 7.26. Asignación a la desactivación.

Activación/desactivación de salidas

En el lenguaje SFC, la activación y desactivación de salidas se representa de la siguiente manera:

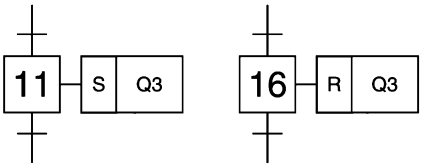


Figura 7.27. Asignación a la activación y desactivación en el lenguaje SFC.

5.5. Acciones condicionadas a temporización

Ejecutan acciones temporizadas una vez alcanzada una etapa. Pueden ser:

Acción con retardo a la conexión

La acción se ejecuta un tiempo después de haber alcanzado la etapa.

En el ejemplo de la figura se representa cómo la salida Q2 se activa a los 3 segundos de que la secuencia llegue a la etapa X2.

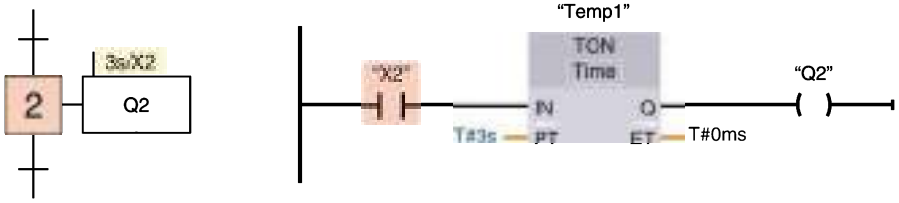


Figura 7.28. Acción con retardo a la conexión.

Acción límite

En este caso la acción se ejecuta nada más entrar en la etapa y deja de hacerlo después de un tiempo considerado como límite.

En el ejemplo de la figura se representa cómo la salida Q3 se activa justo en el instante en que la secuencia entra en la etapa X3 y se desactiva después de un tiempo, que, en este caso, es de 3 segundos.

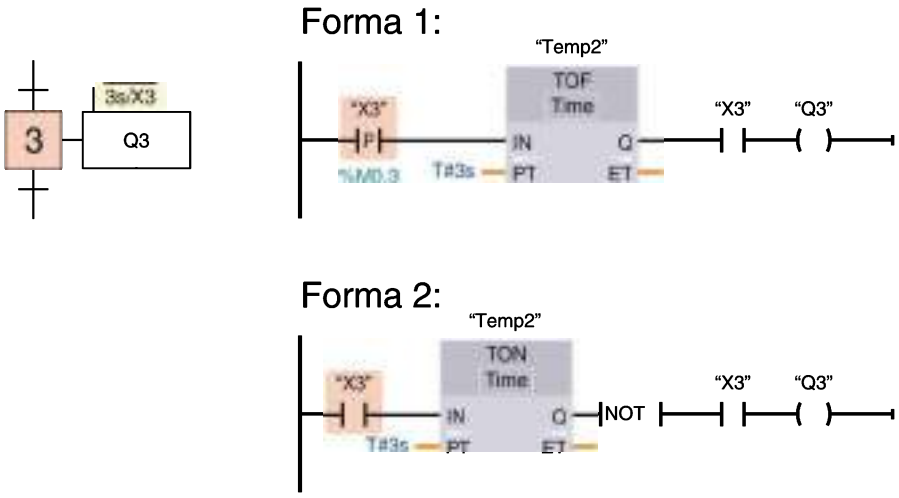


Figura 7.29. Acción límite.

Vocabulary

- Flecha: *arrow*.
- Etapa incluyente: *enclosing step*.
- Flanquear: *flank*.
- Jump: *salto*.
- Etapa de origen: *origin step*.
- Máquinas virtuales: *virtual machines*.
- Etapas opcionales: *branching step*.
- Cadena: *chain*.
- Transición: *transition*.
- Convergencia: *convergence*.
- Divergencia: *divergence*.
- Etapas simultáneas: *simultaneous steps*.
- Asignación: *assignment*.
- Intermitente: *blinker*.

Es importante no transcribir la ecuación lógica de la acción condicionada al lenguaje de contactos, ya que no funcionaría como corresponde. Esa no es más que la forma que tiene la norma para representar la acción límite. No obstante, su implementación puede realizarse de cualquiera de las dos formas representadas en la figura. En ambos casos, el contacto de X3 que está después de la salida del temporizador permite anular la acción límite si se sale de la etapa antes de que haya transcurrido el tiempo.

Acción intermitente o por pulsos

Es una acción que controla una variable booleana mediante pulsos simétricos y consigue el efecto de intermitencia.

En este caso, la mejor opción consiste en utilizar un bit de la marca de ciclo en serie con el bit de etapa que controla la acción.

En el ejemplo de la figura, la salida Q3 parpadea con una cadencia de 2 Hz cuando la secuencia se encuentra en la etapa X11.

Nomenclatura

Forma (no normalizada) que se usará en el libro para representar una acción intermitente.

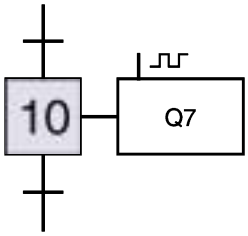


Figura 7.30. Forma de representar una salida intermitente.

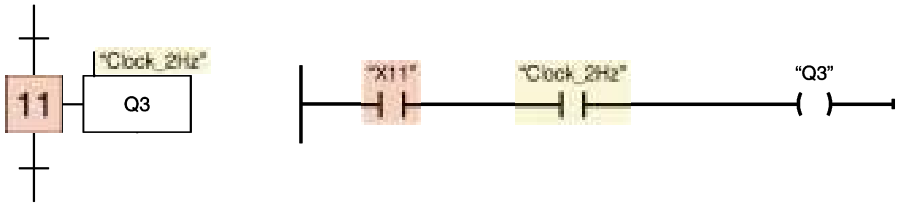


Figura 7.31. Acción intermitente.

5.6. Acciones de asignación por flanco

Son acciones que se ejecutan al entrar o al salir de una etapa, por lo tanto, están gestionadas por detectores de flanco asociados al bit de etapa.

Flanco ascendente

El flanco ascendente, o positivo, se utiliza para detectar la entrada en una etapa. El siguiente ejemplo muestra cómo se activa la salida Q3 al entrar en la etapa X11.

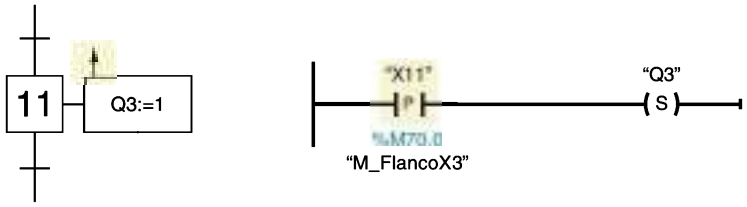


Figura 7.32. Acción con flanco de entrada.

Recuerda

Las marcas de flanco deben ser unívocas y no se deben utilizar para ninguna otra operación en el mismo programa.

Flanco descendente

El flanco descendente, o negativo, se usa para detectar cuándo se sale de una etapa. En el ejemplo de la figura se muestra cómo la salida Q3 se desactiva cuando la secuencia abandona la etapa X16.

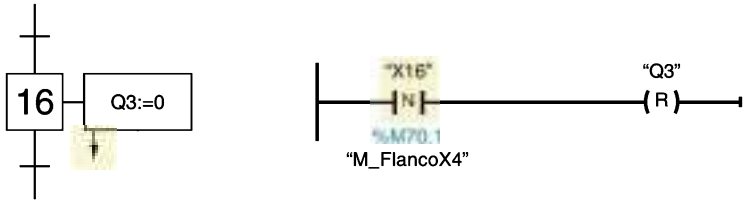


Figura 7.33. Acción con flanco de salida.

5.7. Acciones condicionadas a eventos

Son acciones que permiten activar o desactivar una variable booleana si la secuencia se encuentra en una etapa determinada y, además, se activa la señal que se representa junto a la banderola de la acción.

En la figura 7.34, vemos cómo Q3 se activa con SET cuando el flujo del GRAFCET se encuentra en la etapa X11 y se produce un evento sobre la señal S4.

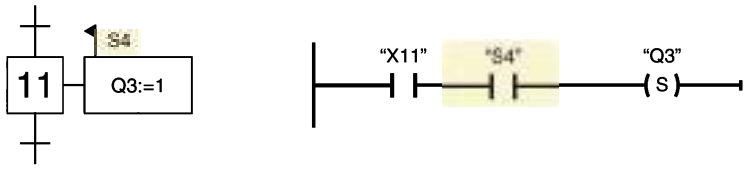


Figura 7.34. Acciones condicionadas eventos.

Caso práctico resuelto

A continuación se muestra la programación en lenguaje de contactos de un GRAFCET de cinco etapas. En él se ejecutan diferentes acciones de las que se han estudiado anteriormente.

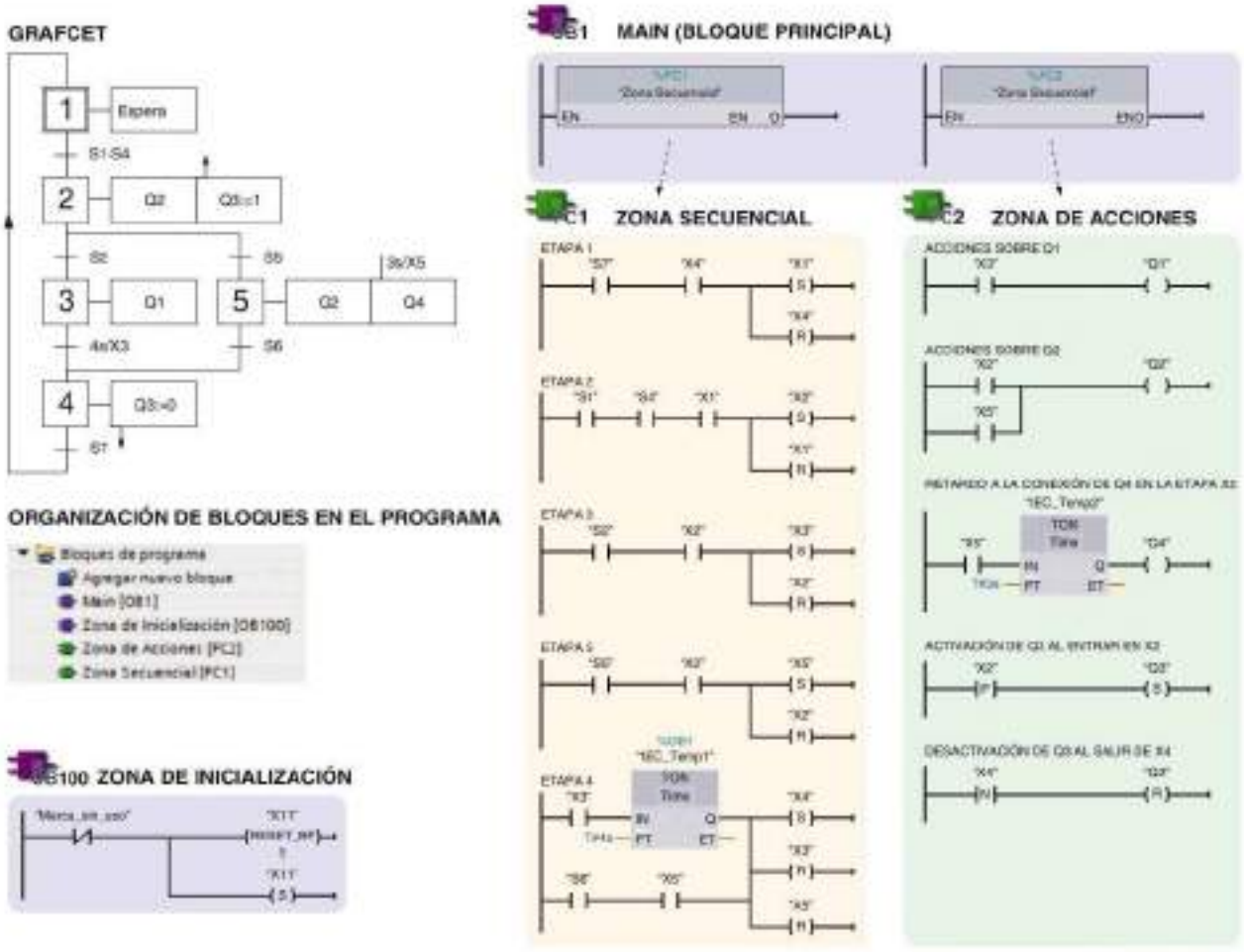


Figura 7.35. Ejemplo de programación de un GRAFCET en KOP.

El contacto cerrado de la denominada marca_sin_uso de la figura es necesario para representar la ramificación de las asignaciones. En los S7-300 es obligatorio ponerlo, pero en los S7-1200 y S7-1500 no. También es posible configurar las marcas de sistema (solo en los S7-1200 y S7-1500) y utilizar el bit denominado *AlwaysTrue* que mantiene siempre a «1» lógico el contacto asociado.

5.8. Acciones con contadores

En la zona de acciones se utilizan las entradas del bloque de contador para realizar las operaciones de contar, descontar, ponerlo a RESET o ajustarlo a un valor determinado.

A estas entradas se conectan los contactos de los bits de etapa desde los que se requiere operar el contador.

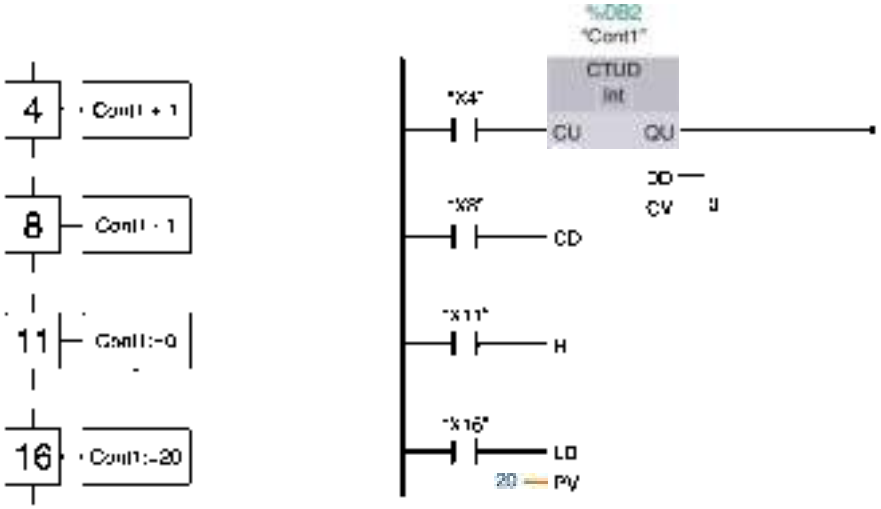


Figura 7.56. Acciones sobre contador.

Actividades

1. Programa en lenguaje de contactos el GRAFCET de la figura y prueba su funcionamiento.

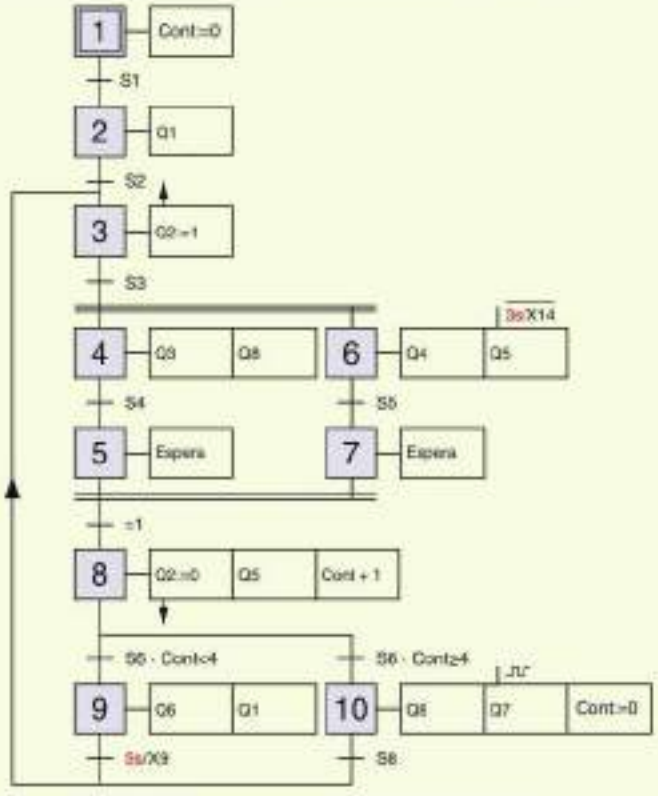


Figura 7.57. GRAFCET para programar en lenguaje de contactos.

5.9. Acciones en SFC

En la unidad anterior quedó claro que el GRAFCET se encuentra estandarizado según la norma IEC-848 y que existe una adaptación de esta representación gráfica para los lenguajes de programación de los autómatas programables denominada *SFC (sequential function chart)*. Conceptualmente ambas son similares, pero disponen de algunas diferencias relacionadas con la representación gráfica, especialmente en las etiquetas de las acciones.

A diferencia de lo establecido en la norma IEC 848, la presentación de las etiquetas de acciones en SFC se hace con dos elementos: la condición o condicionador y la acción sobre la señal o variable. En las etiquetas ambas se encuentran separadas de forma gráfica por dos rectángulos. En el del lado izquierdo se indica la condición y en el lado derecho, la acción.

A continuación se muestra una tabla comparativa entre las acciones contempladas por el estándar IEC 848 y las que regula el lenguaje SFC.

Comparativa entre las acciones del estándar IEC-848 y las del lenguaje SFC

Acciones	Estándar	
	SFC	IEC-848
Acción directa mientras la etapa está activa	N Acción	Acción
Activación (SET)	S Acción	Acción:=1
Desactivación (RESET)	R Acción	Acción:=0
Activación con retardo a la conexión	D Acción	Temp/Xn Acción
Activación por tiempo límite	L Acción	Temp/Xn Acción

6. Zona no secuencial

En un programa puede haber una zona ajena al GRAFCET de forma que en ella se realicen operaciones que no estén condicionadas al funcionamiento de la secuencia. Esta se debe programar en un bloque FC propio, que hay que invocar desde el bloque principal o *MAIN*.

Aunque las acciones que se ejecuten en este bloque del programa pueden ser completamente independientes al GRAFCET, lo que en él suceda puede tener alguna relación con la secuencia como, por ejemplo, establecer una condición de funcionamiento o procesar algún tipo de datos.

Así, en el ejemplo de la figura se muestra cómo un detector, llamado S1, incrementa los pulsos de un contador sin estar condicionado a ningún estado de la secuencia. Sin embargo, cuando el valor del contador es mayor de 10 el GRAFCET pasa de la etapa X9 a X10.

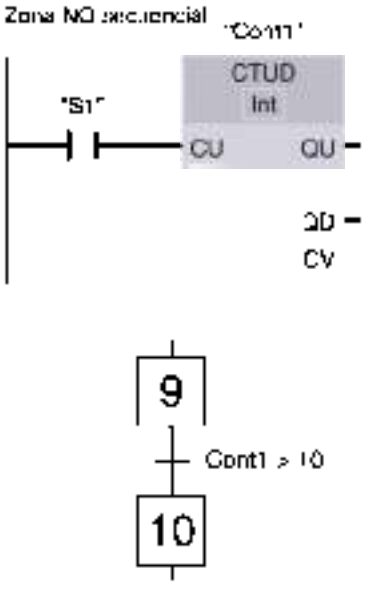


Figura 7.38. Ejemplo de zona no secuencial

PRÁCTICA PROFESIONAL RESUELTA

Herramientas

- PC con TIA PORTAL y PLCSIM

Material

- Un PLC compatible con TIA PORTAL
- Opcional: maqueta que permita la simulación del proceso

Precauciones

- Hay que tener en cuenta que la subida y bajada del taladro se hace con un motor trifásico, por lo que no es posible invertir el sentido de giro de manera instantánea, ya que produciría un cortocircuito en el circuito de fuerza que controla el motor.

Programación del GRAFCET del taladro con desahogo en lenguaje de contactos

Objetivo

- Programación de un GRAFCET en lenguaje de contactos.
- Organizar las diferentes zonas del GRAFCET en bloques de programación.

Desarrollo

1. Se parte del GRAFCET del taladro automático con cargador de piezas de la práctica profesional resuelta de la unidad anterior.

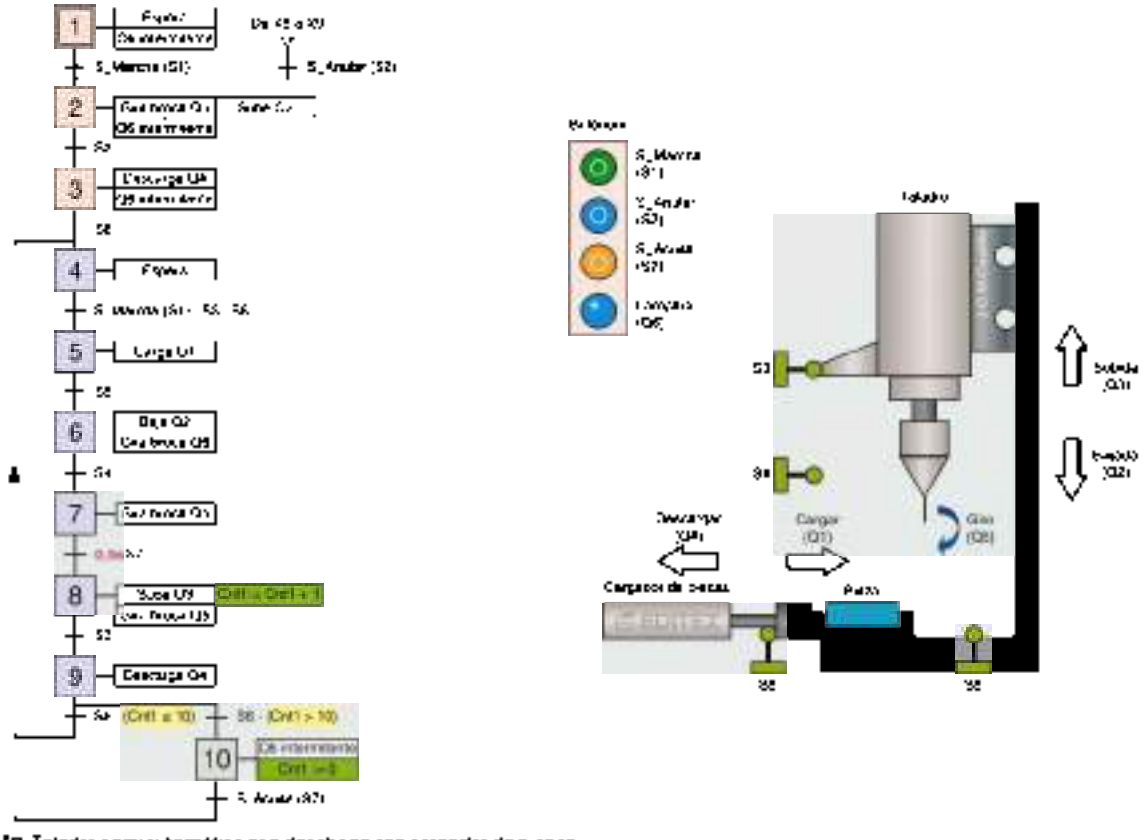


Figura 7.39. Taladro semiautomático con desahogo con cargador de piezas.

2. Hay que crear un proyecto en TIA PORTAL para un S7 1200 o, en su defecto, el PLC del que se disponga en el aula. También es posible utilizar el simulador PLCSim.

3. Activar la marca de ciclo y las variables del sistema. Elegir los bytes de marcas en los que se quiere direccionarlas. Aquí se ha elegido MB100 para las marcas de sistema y MB101 para la marca de ciclo.

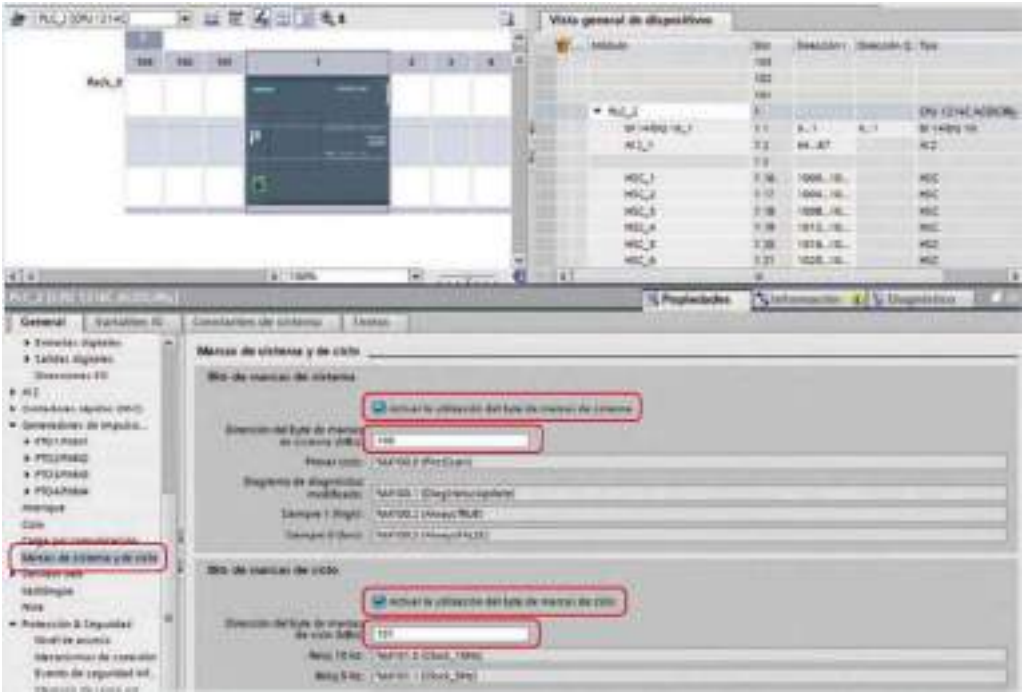


Figura 7.40. Marcas de sistema y de ciclo.

4. Fijándose en la figura, elaborar una tabla de las variables. Hay que tener en cuenta que las variables se pueden editar y crear en cualquier momento, incluso en el proceso de programación.

Nombre	Dirección	Nombre	Dirección
Entradas		Salidas	
S_Marcha (S1)	%I0.0	Carga (Q1)	%Q0.0
S_Anular(S2)	%I0.1	Baja (Q2)	%Q0.1
S3	%I0.3	Sube (Q3)	%Q0.2
S4	%I0.4	Descarga (Q4)	%Q0.3
S5	%I0.5	Giro (Q5)	%Q0.4
S6	%I0.6	Lámpara (Q6)	%Q0.5
S_Acuse (S7)	%I0.7		
Etapas		Variables especiales	
X1	%M0.0	AlwaysTrue	%M100.2
X2	%M0.1	Clock_2Hz	%M101.3
X3	%M0.2		
X4	%M0.3		
X5	%M0.4		
X6	%M0.5		
X7	%M0.6		
X8	%M0.7		
X9	%M1.0		

PRÁCTICA PROFESIONAL RESUELTA

continuación

5. Se deben crear dos FC, uno para la zona secuencial y otro para la zona de acciones, y el OB100 para la inicialización del GRAFCET.
6. En el OB1 se debe llamar de forma incondicional a los dos bloques FC.
7. La programación del bloque de inicialización OB100 debe estar destinada a resetear todas las etapas del GRAFCET y activar la etapa inicial X1. Es importante recordar que el OB100 no requiere ser llamado desde ningún otro bloque.

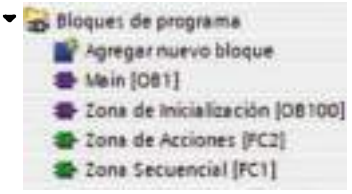


Figura 7.41. Estructura de bloque en el programa.

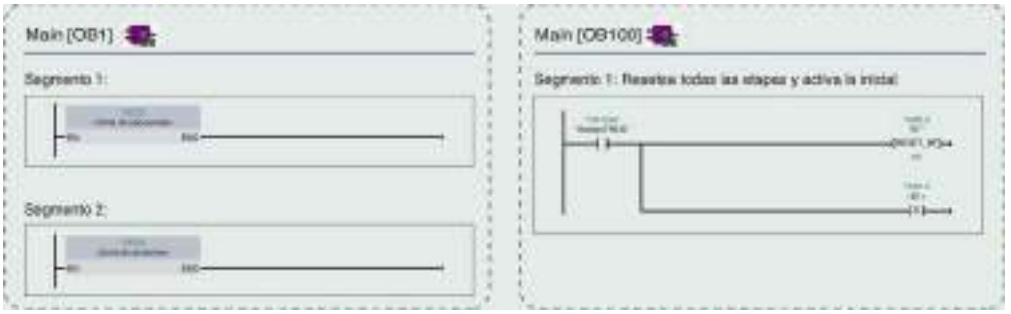


Figura 7.42. Bloques OB1 y OB100.

8. Programar la zona secuencial en el bloque FC1.

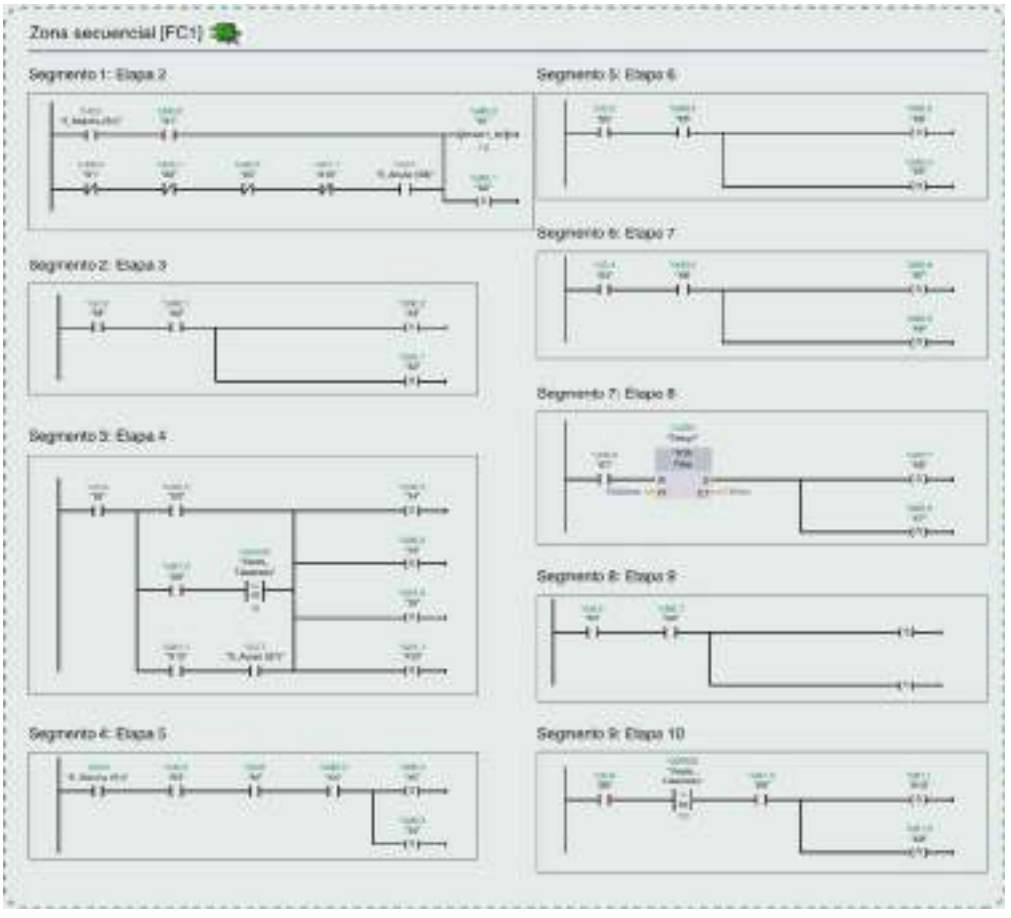


Figura 7.43. Zona secuencial.

9. Hay que tener en cuenta que a la etapa X2 se llega desde la etapa X1, a través del pulsador de marcha, y desde la etapa X4 a la X10 mediante el pulsador de anular. En este último caso se ha optado por representar en el programa, con contactos cerrados en serie, las etapas desde las que no se hace el salto. Como se deben resetear todas aquellas etapas desde las que se llega a X2, la mejor solución es representar una operación para «desactivar mapa de bits» e inmediatamente activar la etapa a la que se llega (X2).
10. A la etapa X4 se llega desde tres caminos: desde X3, X9 y X10. En todos ellos, en la evolución del GRAFCET interviene el final de carrera S6, por lo que solo se representa una vez en el programa. Para la transición que es receptiva en la etapa X9 debe establecerse la condición cuando el valor del contador es igual o inferior a 10. En este caso, se ha optado por utilizar una operación de comparación.
11. La transición en la divergencia hacia la etapa X10 debe tener también una operación de comparación, que será verdadera cuando el contador sea mayor de 10 y esté a «1» el final de carrera S6.
12. Programar la zona de acciones en el bloque FC2. En él se debe incluir el contador que evaluará el número de veces que se ha realizado el taladrado.

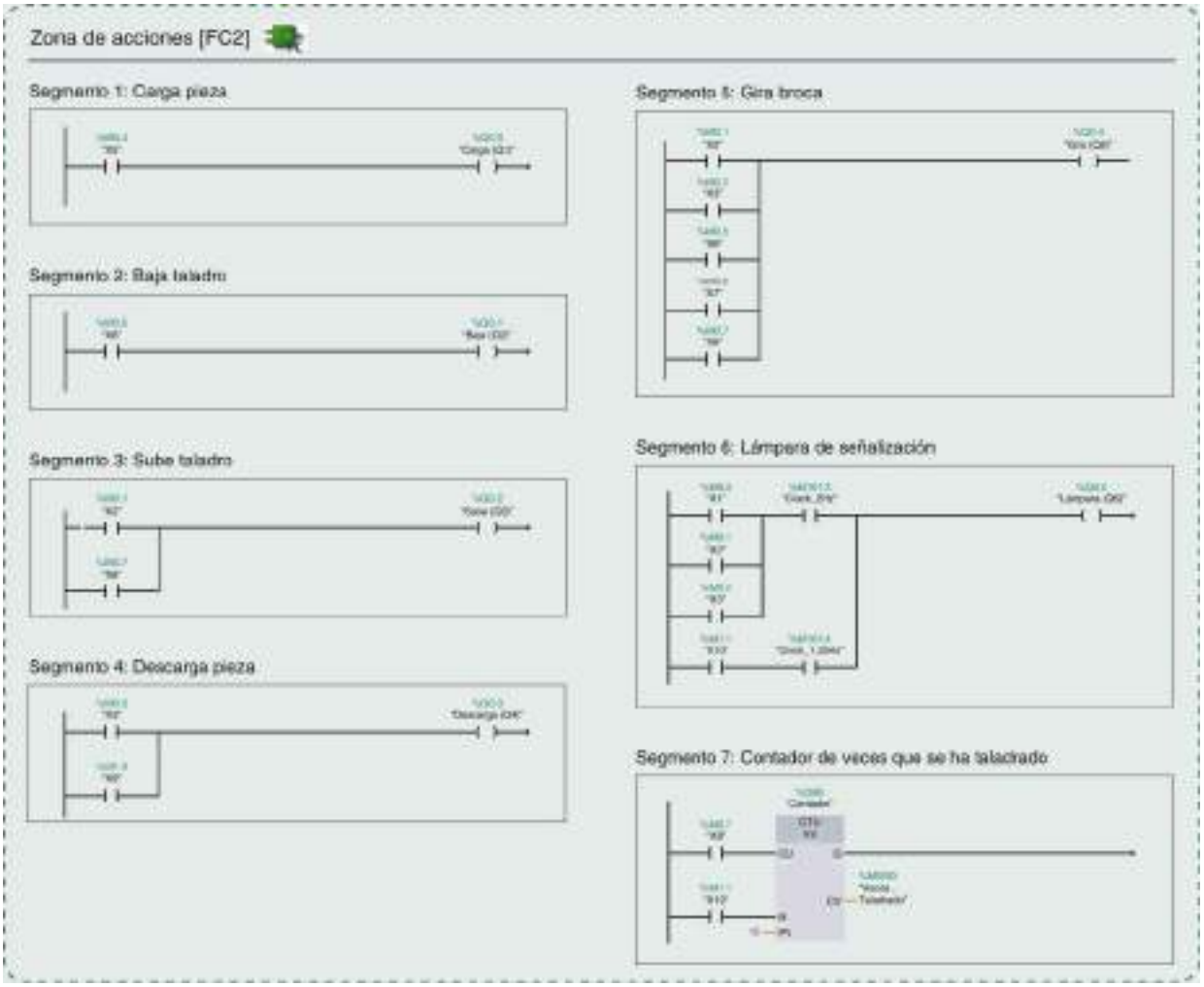


Figura 7.44. Zona de acciones.

13. Para la lámpara Q6 se ha optado por dos tipos de intermitencia, una rápida y otra más lenta. Esto permitirá identificar cuándo la secuencia está ejecutando el *homing* o cuándo está en la etapa X10, solicitando al operario que accione el pulsador de acuse de cambio de broca.

TEST DE EVALUACIÓN

RESUELVE EN TU CUADERNO O BLOC DE NOTAS

1. **El paso entre etapas se representa en:**
   a) La zona de acciones.
   b) La zona inicial.
   c) La zona secuencial.
   d) El bloque *MAIN*.
2. **¿Dónde se activan las salidas?**
   a) Zona de acciones.
   b) Zona inicial.
   c) Zona secuencial.
   d) Bloque *MAIN*.
3. **Las variables asociadas a las etapas son:**
   a) Entradas.
   b) Salidas.
   c) Marcas.
   d) Contadores.
4. **Para que una transición sea receptiva al valor de un contador:**
   a) Se debe usar una comparación en el segmento de etapa.
   b) Se debe usar la «caja» del contador en el segmento de etapa.
   c) Es necesario usar un contacto de un temporizador para realizar esta acción.
   d) No es posible hacerlo.
5. **Si en una etapa hay una acción que dice «Salida:=1», ¿qué significa?**
   a) Es una salida activada con flanco.
   b) Es lo mismo que activar la salida con SET.
   c) Es una acción condicionada.
   d) Se incrementa en uno el valor de un contador.
6. **Si en un GRAFCET hay una acción que representa Cnt1+1 significa que:**
   a) Se incrementa el valor de un contador.
   b) Se resetea un contador.
   c) Se activa la salida de un contador.
   d) Se disminuye el valor de un contador.
7. **¿Qué significa que la entrada CD de un contador esté asociada a un contacto de X4?**
   a) El contador se resetea al llegar a esa etapa.
   b) El contador adquiere el valor de preselección representado en PV.
   c) No se puede hacer. Es un error de programación.
   d) El valor del contador disminuye cuando la secuencia está en la etapa X4.
8. **Si en un GRAFCET hay una acción representada con una flecha en la parte superior de la etiqueta, significa que:**
   a) Es una asignación directa.
   b) Es una asignación basada en evento.
   c) Es una acción por flanco positivo.
   d) Es una acción por flanco negativo.
9. **¿Qué significa que en una acción haya un contacto con la etiqueta «Clock_2Hz» en serie con el bit de etapa?**
   a) Funcionamiento directo.
   b) Funcionamiento temporizado.
   c) Funcionamiento intermitente.
   d) Acción por flanco.
10. **Una acción condicionada con 3s/X4 significa que es una acción:**
    a) Asociada a un contador.
    b) Con retardo a la conexión.
    c) Con retardo a la desconexión.
    d) Límite.

ACTIVIDADES FINALES

1. Diseña y programa un **GRAFCET** de secuencias opcionales para invertir el sentido de giro de un motor trifásico. S1 es para el giro a izquierdas, S3 para el giro a derechas y S2 para la parada. El arranque del motor, tanto en un sentido de giro como en otro, siempre debe hacerse desde parado. Hay que tener en cuenta que el pulsador de parada S2 es normalmente cerrado (NO).

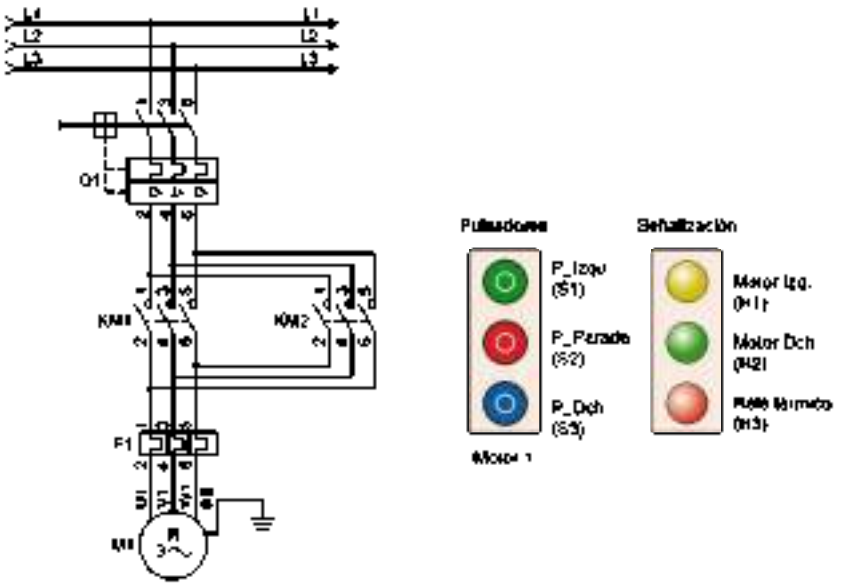


Figura 7.45. Inversión del sentido de giro de un motor trifásico.

2. Basándote en el circuito de la actividad anterior, diseña y programa el GRAFCET con las opciones de funcionamiento siguientes:
   - Partiendo de una posición de reposo o de parada, al accionar S1, el motor gira a izquierdas durante 10 segundos. Después de eso, el motor se para durante 2 segundos y arranca en sentido contrario, funcionando hasta que se acciona el pulsador de parada.
   - Por el contrario, partiendo de la posición inicial, si se acciona S3, el motor funciona de forma similar a lo descrito anteriormente, pero girando en sentido contrario a lo indicado en el párrafo anterior.

   El pulsador de parada permite detener el funcionamiento por completo y llevar la secuencia a la etapa inicial o de reposo.

5. Diseña y programa el GRAFCET para arrancar un motor trifásico en estrella-triángulo. El motor arranca en estrella al accionar S2 y conmuta a triángulo después de 2 segundos. El motor se puede detener en cualquier momento accionando el pulsador de parada S1, que es normalmente cerrado. Recuerda que el contactor KM1 (principal) debe estar activado tanto cuando el motor funciona en estrella como en triángulo.

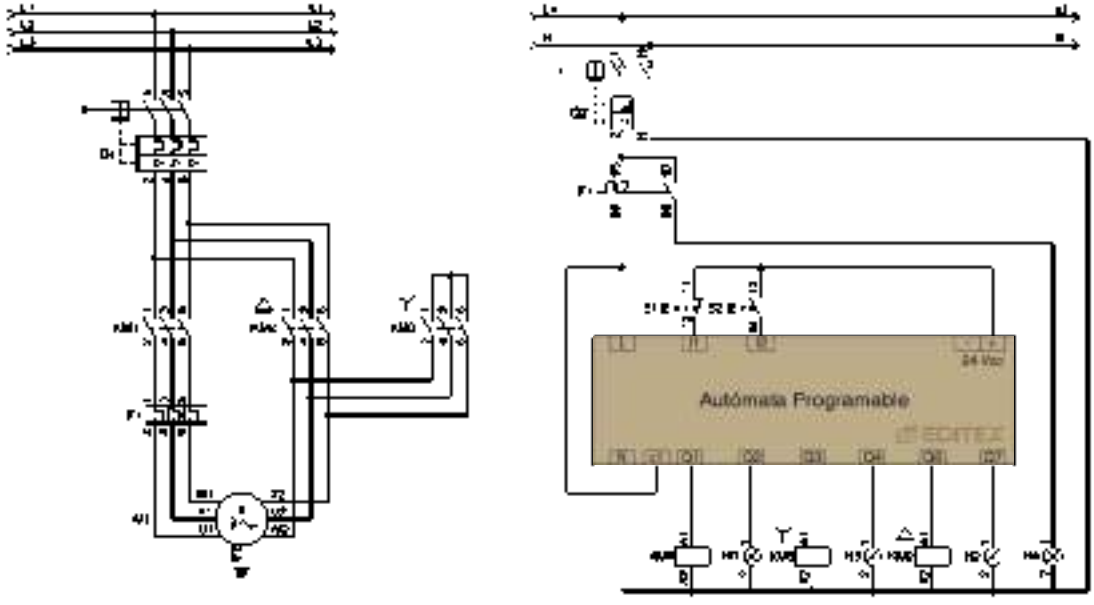


Figura 7.46. Esquema para arrancar un motor trifásico en estrella-triángulo.

ACTIVIDADES FINALES

continuación

4. Implementa el siguiente GRAFCET en lenguaje de contactos y comprueba su funcionamiento en un PLC real o con el simulador

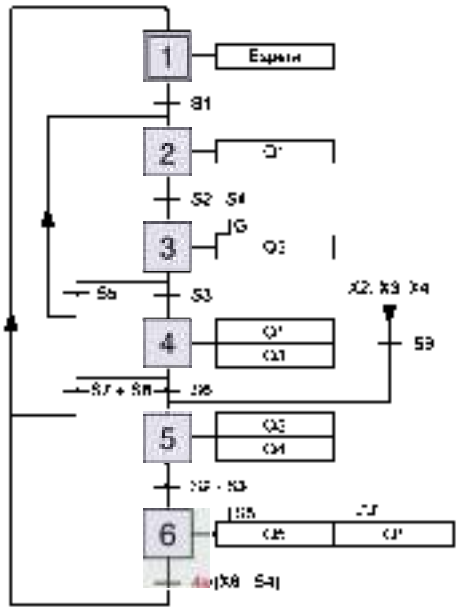


Figura 7.47. GRAFCET actividad 4.

5. Implementa el siguiente GRAFCET en lenguaje de contactos y comprueba su funcionamiento en un PLC real o con el simulador.

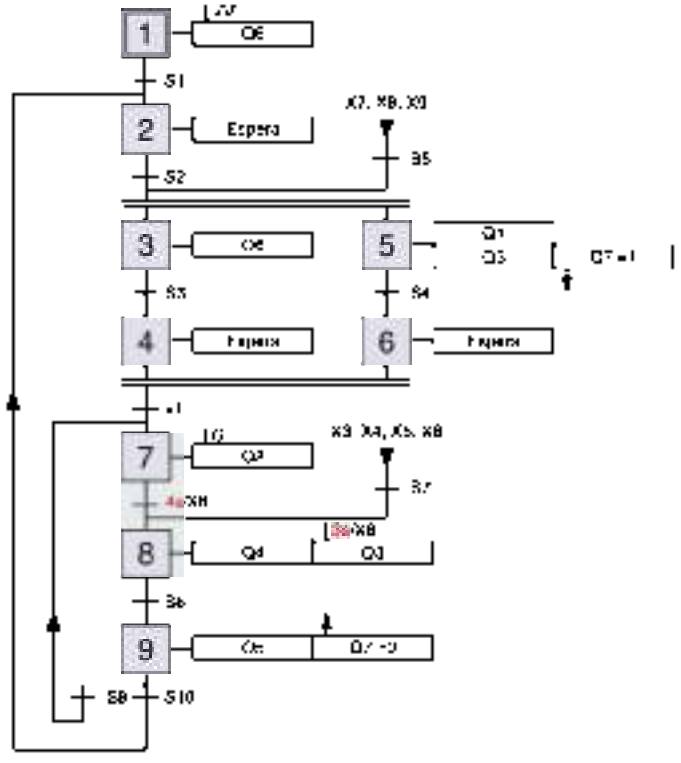


Figura 7.48. GRAFCET actividad 5.

PRÁCTICA PROFESIONAL PROPUESTA 1

Herramientas

- PC con TIA PORTAL y PLCSIM

Material

- Un PLC compatible con TIA PORTAL

GRAFCET simultáneos

Objetivos

- Realizar un programa en lenguaje de contactos KOP para el control de tres GRAFCET simultáneos.
- Sincronizar el funcionamiento de alguna secuencia con los bits de etapas de cualquiera de los otros GRAFCET.

Precauciones

- Las tres etapas iniciales se deben activar en el momento de pasar el PLC de STOP a RUN.
- La zona secuencia se puede separar en diferentes FC, uno por GRAFCET.
- Es aconsejable utilizar una única zona de acciones para el control de todas las salidas.

Desarrollo

1. Crea un nuevo proyecto en TIA PORTAL para el PLC que consideres más adecuado.
2. Configura la marca de ciclo en el dispositivo.
3. Observa los GRAFCET de la figura y crea una lista de variables.
4. Crea tres bloques FC para programar individualmente las zonas secuenciales de cada una de las cadenas del GRAFCET.
5. Crea un solo FC para la zona de acciones. En este bloque deben estar las acciones de todas las cadenas.
6. Crea el bloque OB100. En él se debe programar la activación de las tres etapas iniciales y el reseteo de todas las demás.
7. En el OB1, haz la llamada a todos los demás bloques.
8. Carga el programa en el PLC y comprueba su funcionamiento.

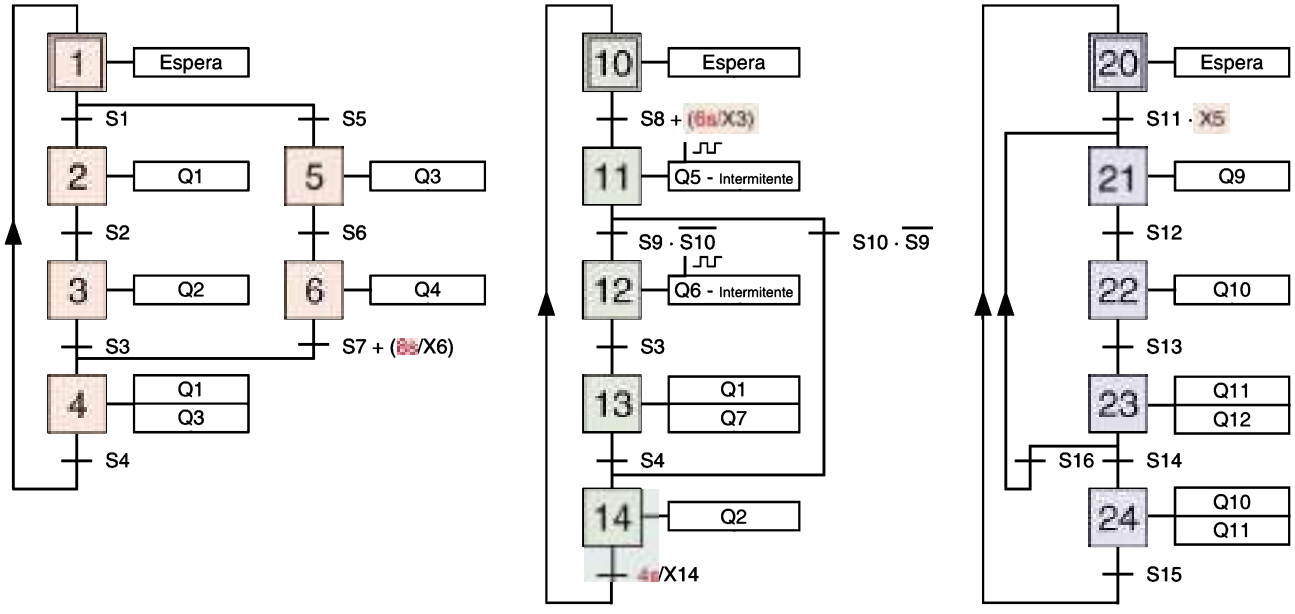


Figura 7.49. GRAFCET simultáneos.

PRÁCTICA PROFESIONAL PROPUESTA 2

Herramientas

- PC con TIA PORTAL y PLCSIM

Material

- Un PLC compatible con TIA PORTAL

GRAFCET para el control de un semáforo

Objetivo

Programar un GRAFCET cuya evolución se basa en el uso de temporizadores en las transiciones.

Precauciones

- Las lámparas con la misma etiqueta se controlan con la misma salida, por lo que no es necesario duplicarlas en el programa. Se da por supuesto que dichas lámparas están conectadas eléctricamente en paralelo entre sí.
- Los tiempos de paso marcados en la tabla son orientativos y están pensados para realizar las pruebas del programa. En un sistema real deberían ser mayores.

Desarrollo

1. Crea un nuevo proyecto en TIA PORTAL para el PLC que se considere más adecuado.
2. Configura la marca de ciclo en el dispositivo.
3. Observa la figura y la tabla de funcionamiento del proceso y representa en tu cuaderno el GRAFCET.
4. Crea una tabla de variables.
5. Crea y programa todos los bloques para administrar el GRAFCET
6. Carga el programa en el PLC y comprueba su funcionamiento.

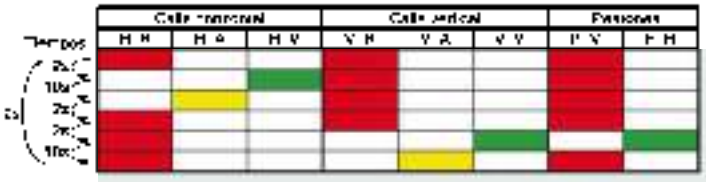


Figura 7.50. Tabla de funcionamiento.

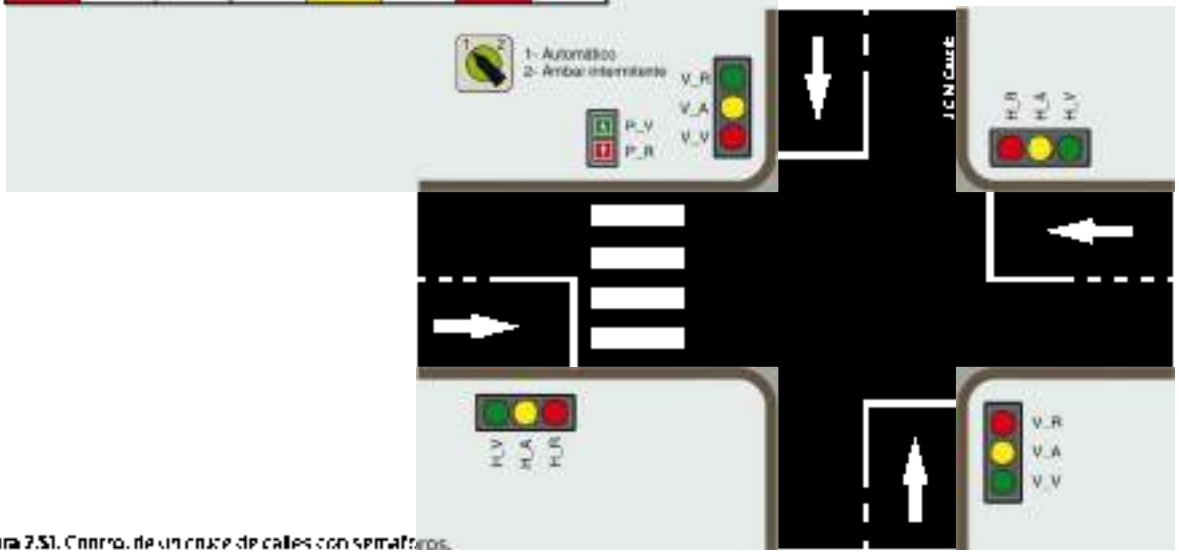
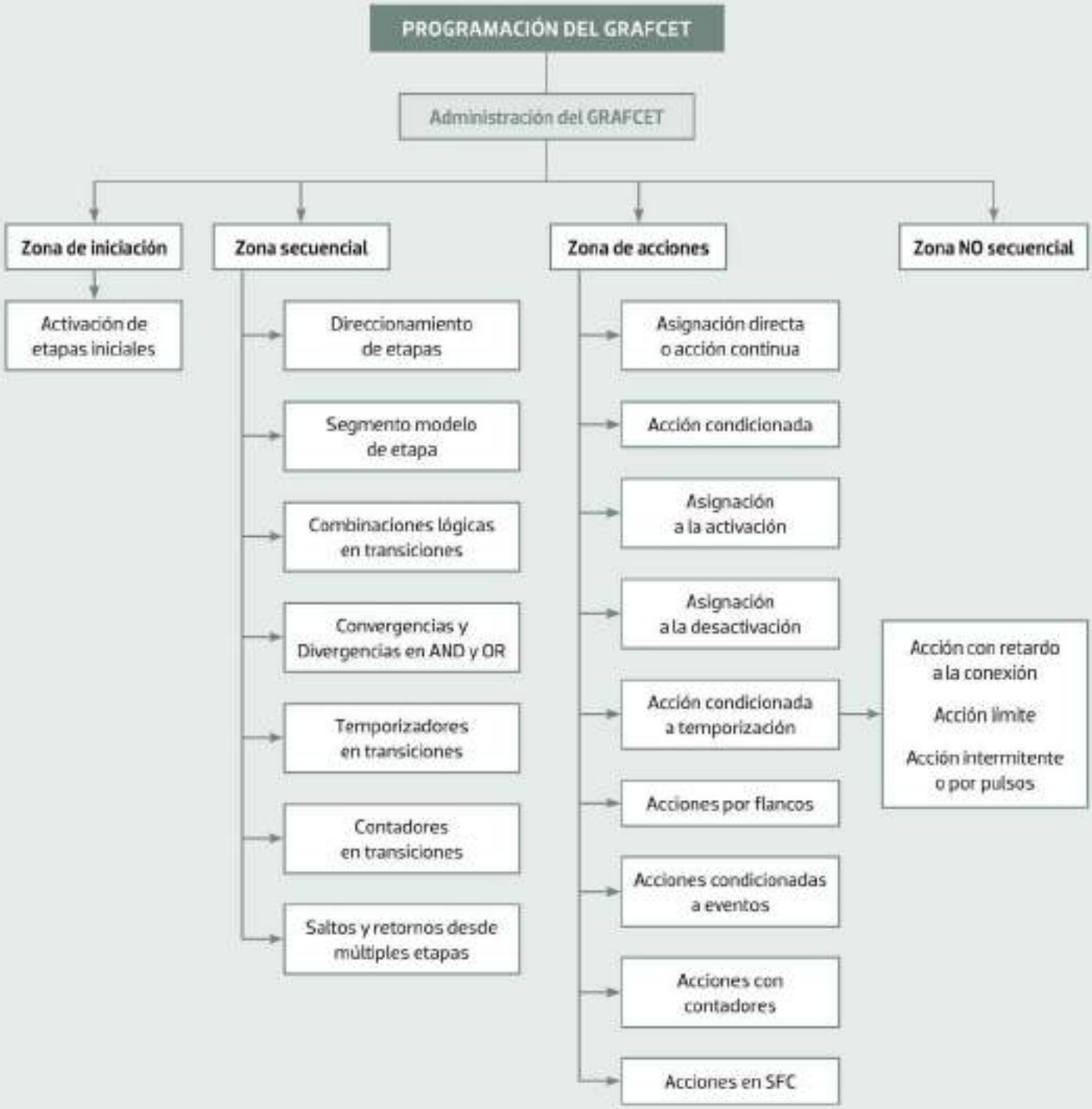


Figura 7.51. Control de un cruce de calles con semáforos.

7. Añade un interruptor al proceso para conmutar entre dos modos de funcionamiento:
   - Modo automático.
   - Lámparas ámbar intermitentes

EN RESUMEN


PROGRAMACIÓN DEL GRAFCET
Administración del GRAFCET
Zona de iniciación
Zona secuencial
Zona de acciones
Zona NO secuencial
Activación de etapas iniciales
Direccionamiento de etapas
Segmento modelo de etapa
Combinaciones lógicas en transiciones
Convergencias y Divergencias en AND y OR
Temporizadores en transiciones
Contadores en transiciones
Saltos y retornos desde múltiples etapas
Asignación directa o acción continua
Acción condicionada
Asignación a la activación
Asignación a la desactivación
Acción condicionada a temporización
Acción con retardo a la conexión
Acción límite
Acción intermitente o por pulsos
Acciones por flancos
Acciones condicionadas a eventos
Acciones con contadores
Acciones en SFC

8 Modos de funcionamiento y estructuración del GRAFCET

Vamos a conocer...

1. Programación de modos de funcionamiento
2. Programación de GRAFCET estructurado

PRÁCTICA PROFESIONAL RESUELTA

Programación del GRAFCET del taladro con desahogo

PRÁCTICAS PROFESIONALES PROPUESTAS

1. Programación de un GRAFCET de un circuito electroneumático
2. Control de un sistema de selección de piezas por alturas

Y al finalizar esta unidad...

- Sabrás cómo se programan los diferentes modos de funcionamiento de secuencias basadas en GRAFCET.
- Programarás una parada en el GRAFCET.
- Diseñarás programas para el funcionamiento de una secuencia por pasos.
- Conocerás cómo se organiza la programación de secuencias basadas en GRAFCET estructurado.
- Programarás etapas incluyentes y macroetapas.
- Diseñarás secuencias con llamadas a GRAFCET parciales.

1. Programación de modos de funcionamiento

1.1. Parada (tipo pausa)

La parada consiste en la interrupción del funcionamiento del proceso, deteniendo los movimientos o acciones que puedan resultar peligrosas para las personas que están en el entorno de la máquina.

La parada aquí descrita permite pausar voluntariamente (y posteriormente reanudar) el funcionamiento de la secuencia mediante dos pulsadores, uno para realizar la parada y otro para volver a ponerlo en marcha. Normalmente, el primero será cerrado (NC) y el segundo abierto (NO).

En esta parada se ejecutan dos operaciones:

1. Desactivar los actuadores controlados por las acciones, especialmente aquellos que generan movimientos en la máquina.
2. Evitar que la secuencia evolucione, aunque las señales de las transiciones que son receptivas estén activas.

La parada en un GRAFCET se puede implementar de diferentes maneras. A continuación se muestran dos de esas formas, una basada en GRAFCET y otra en programación.

Modo parada

En la mayoría de los ejemplos se da por supuesto que debe implementarse el modo de parada, aunque no se represente en el GRAFCET.

1.1.1. Parada con GRAFCET específico

Consiste en utilizar un GRAFCET específico e independiente de la cadena principal para gestionar una marca, que aquí se ha denominado *Marca de Paro* (M_Paro). Esta marca se habilita en la etapa X11, cuando se acciona el pulsador de parada, y se deshabilita en la X10, cuando se acciona el pulsador de marcha.

El bit negado de M_Paro se utiliza para condicionar las acciones del GRAFCET principal que se desea desactivar cuando se produce la parada. En el ejemplo de la figura 8.1 se observa que las salidas Q1 y Q2 están afectadas por la marca de paro; sin embargo, la lámpara intermitente H1 no lo está.

GRAFCET de dos etapas

En STEP 7 se puede implementar un GRAFCET de dos etapas, tal como se muestra en el programa del ejemplo. No obstante, la misma solución para otros dispositivos podría producir que una etapa anulase a la otra, por lo que sería necesario que las señales de las transiciones, «S_Paro» y «S_Marcha», tuviesen que estar asociadas a señales de flanco.

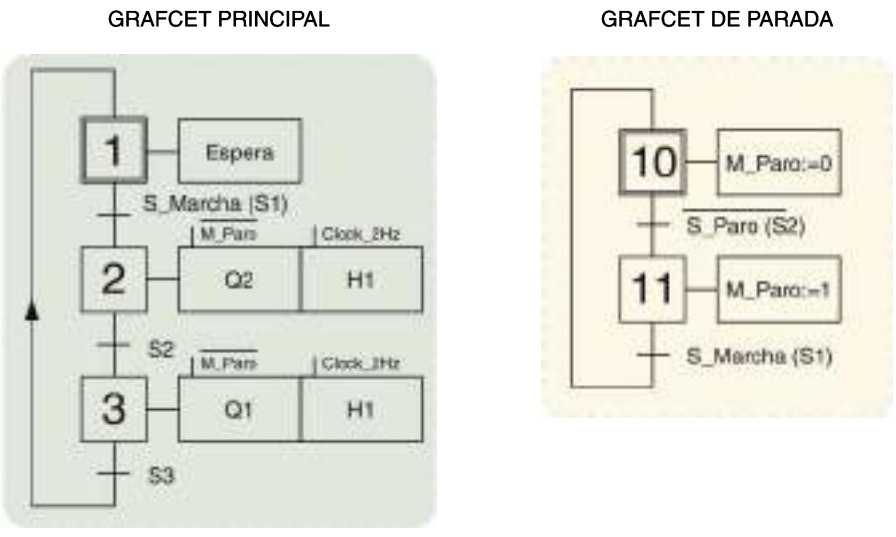


Figura 8.1. Parada con GRAFCET específico.

No todas las salidas de la secuencia principal deben estar afectadas por la marca de paro. Solo será necesario desactivar aquellas cuyo funcionamiento pueda presentar peligro para la propia máquina o para las personas que la utilizan (motores, actuadores neumáticos, sistemas de caldeo, etc.).

Ejemplos

Veamos ahora la implementación en el lenguaje de contactos de esta forma de parada.

Para el GRAFCET principal se han utilizado dos bloques de programación FC, uno para la zona secuencial y otro para la zona de acciones. Para el GRAFCET de parada se ha utilizado un FC independiente en el que se han insertado su zona secuencial y su zona de acciones.

En el bloque de arranque OB100 se ha previsto la activación de las dos etapas iniciales X1 y X10, y la desactivación de todas las demás.

Las acciones Q1 y Q2 están condicionadas a un bit negado de la marca de paro, de forma que se deshabilitan cuando dicha señal está a «1» lógico.

El bloque de la zona secuencial del GRAFCET principal está condicionado a la marca de paro. Si bien esta solución es opcional, puede resultar interesante tenerla en cuenta para evitar que, por error, la secuencia evolucione cuando está activado el paro.

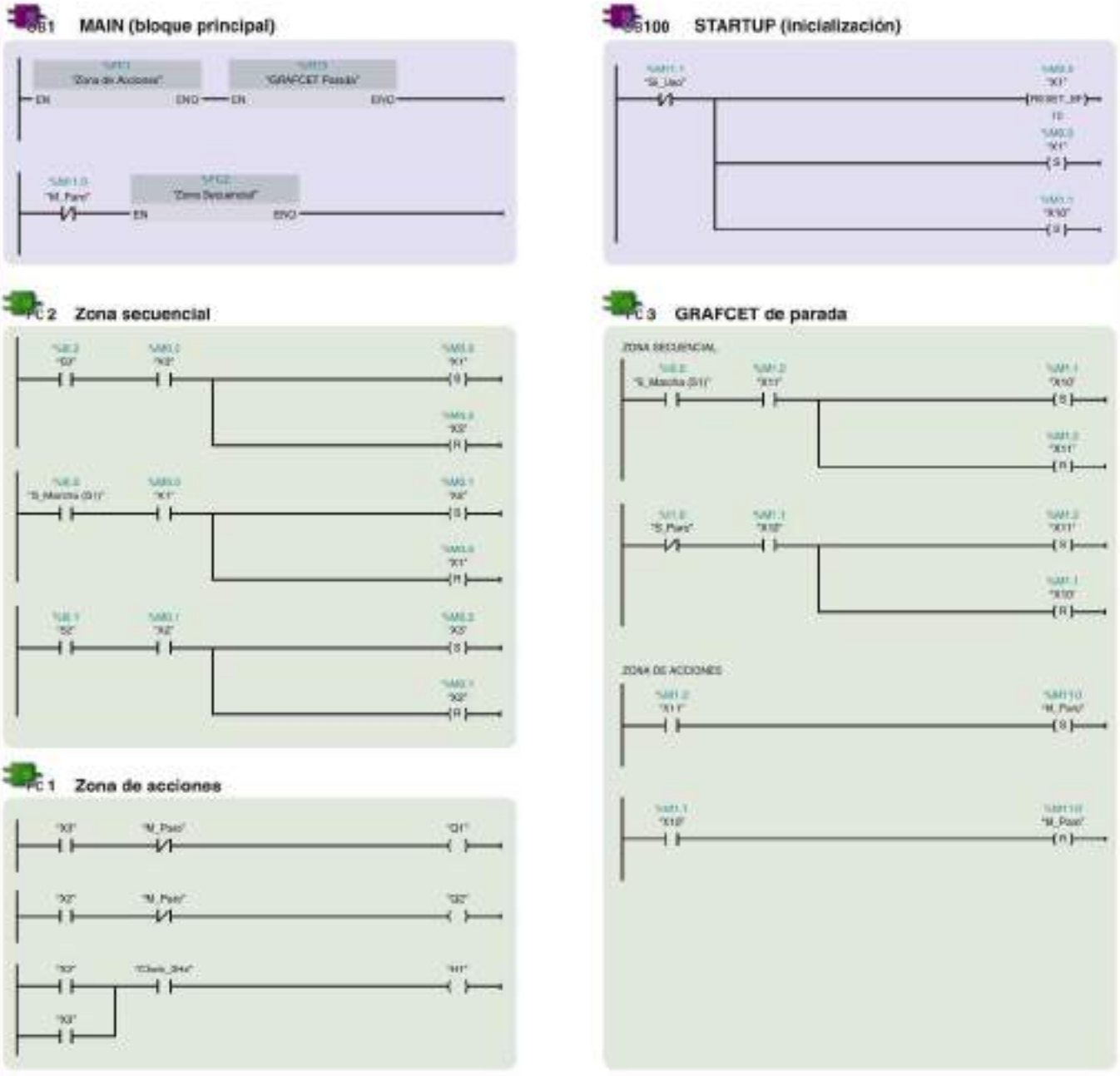


Figura 8.2. Ejemplo de programación de parada con GRAFCET específico para tal fin.

1.2. Parada por programación

El GRAFCET de parada se puede sustituir por un sencillo bloque de programación en el que la gestión de la marca de paro (M_Paro) se hace directamente con los pulsadores de marcha y paro, según se muestra en la figura 8.6.

En este caso, los bits de la marca de paro (M_Paro) se usan de igual forma que en el ejemplo anterior, para desactivar acciones y, si es necesario, deshabilitar el bloque de la zona secuencial.

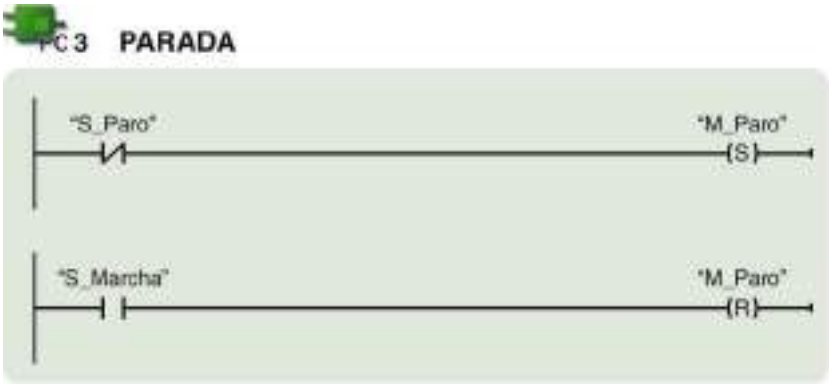


Figura 8.5. Programa para el bloque de parada.

El contacto del pulsador S_Paro se programa normalmente cerrado, ya que el contacto físico de este pulsador también lo debe ser.

1.3. Modo por pasos

En la unidad anterior se explicó en qué consistía este modo de funcionamiento. A continuación se muestra cómo se implementa en el leguaje de contactos.

Para controlar este modo de funcionamiento se requieren dos señales digitales: una para conmutar entre el modo por pasos y el modo secuencial automático y otra que permita ejecutar manualmente la secuencia etapa por etapa. Para la primera se utilizará un conmutador o selector y, para la segunda, un pulsador.

En todas las transiciones se deben conectar contactos de ambas señales en paralelo, en serie con el contacto utilizado para la señal de la transición.

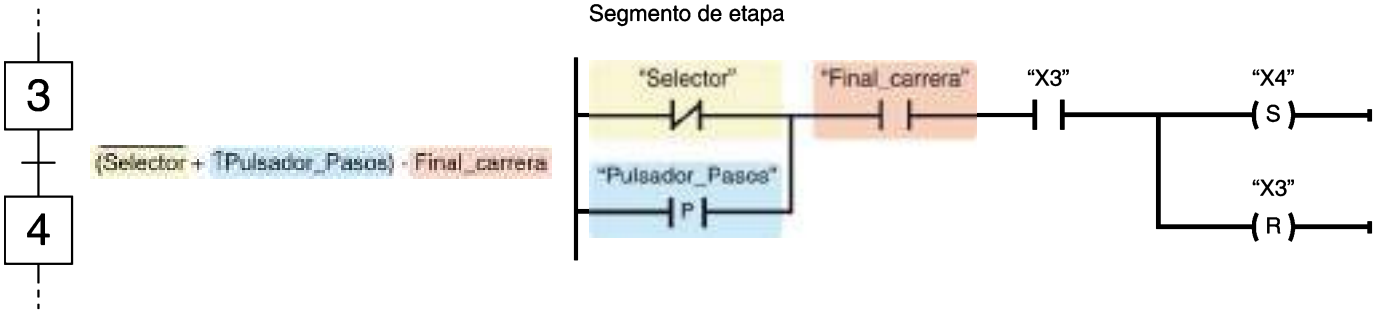


Figura 8.7. Segmento modelo del modo por pasos.

El pulsador por pasos debe asociarse a un contacto con flanco positivo, de lo contrario se puede saltar varias etapas consecutivas en lo que dura la acción sobre el pulsador, especialmente si los actuadores ejecutan movimientos muy rápidos que activan las señales de las transiciones en ese breve instante de tiempo.

Recuerda

El bit de la marca de paro (M_Paro) debe desactivar aquellas acciones que sea necesario parar y, opcionalmente, deshabilitar el bloque que hace la llamada al bloque de la zona secuencial.

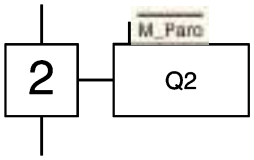


Figura 8.3. Acción con bit de la marca de paro.

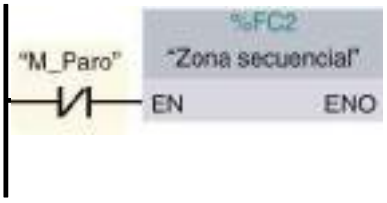


Figura 8.4. Deshabilitación del bloque secuencial con la marca de paro.

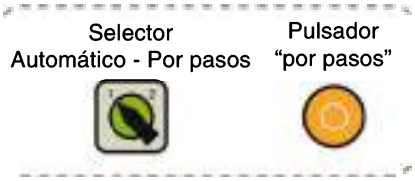


Figura 8.6. Señales para el funcionamiento por pasos.

Ejemplo

Diseña un GRAFCET de cuatro etapas con el modo de funcionamiento por pasos.

Observa que este modo de funcionamiento no afecta a la transición que es receptiva en la etapa 1 para pasar a la 2, por lo tanto, no es necesario poner en él el bloque de contactos.

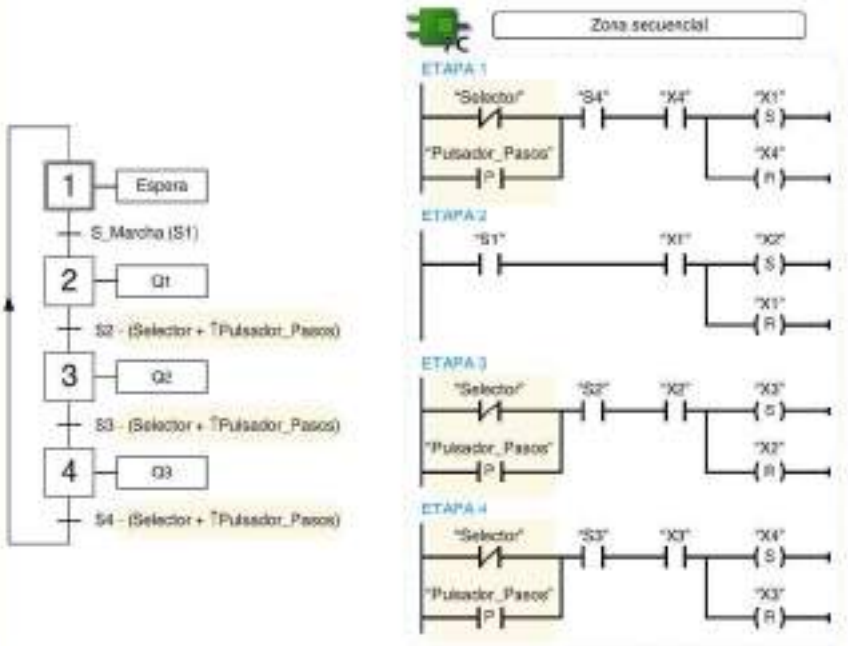


Figura 8.8. GRAFCET de cuatro etapas por pasos.

Para evitar repetir el grupo de contactos en todos los segmentos de etapa se debe recurrir a la creación de un bloque de programación (FC) específico, donde las señales del selector y del pulsador por pasos controlen una marca (M_por_pasos) cuyos contactos se deben usar en aquellas transiciones del GRAFCET donde sea necesario.

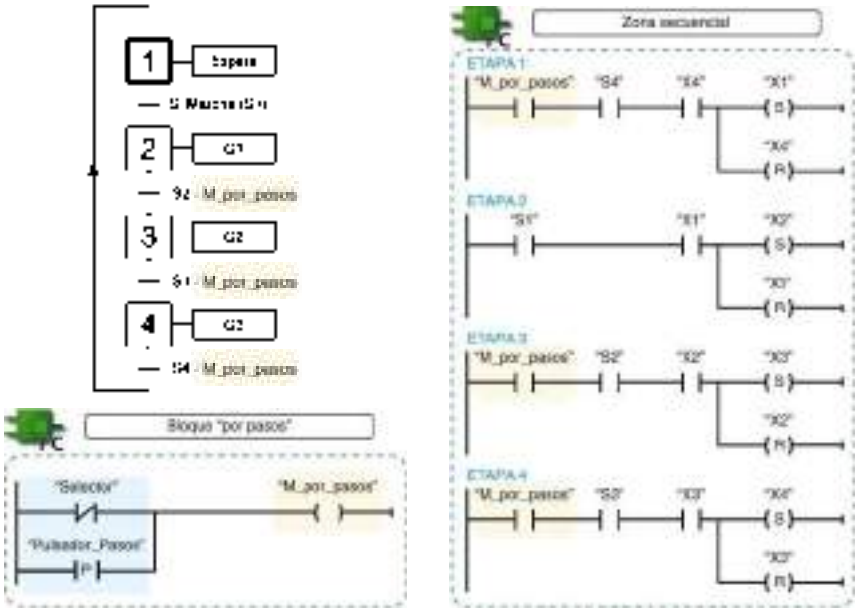


Figura 8.9. Bloque por pasos.

1.4. Acciones en el modo por pasos

El funcionamiento por pasos presenta un inconveniente que afecta a las acciones que se ejecutan en las etapas. Cada vez que se avanza un estado, el proceso queda receptivo a una transición, la cual no será flanqueada hasta que se accione el pulsador por pasos propio del modo, lo que significa que, aunque el detector o final de carrera correspondiente sea accionado cuando el actuador llega al final de su recorrido, no se cambiará de etapa y, por tanto, la acción continuará activa.

Cuando esto sucede es necesario identificar el tipo de actuador que se va a controlar en cada acción, y los mayores problemas se presentan en aquellas situaciones en las que hay que generar movimientos que tienen un principio y un final, como pueden ser los de tipo lineal, en los cuales hay que desactivar el actuador cuando se alcanza alguno de sus extremos.

A continuación se estudian tres casos característicos y se aportan las soluciones más adecuadas a cada uno de ellos.

1.4.1. Acciones sobre válvulas monoestables

Cuando se utilizan válvulas monoestables para controlar un cilindro neumático o hidráulico no hay que establecer ninguna condición en la acción sobre este actuador, ya que, si se retira la alimentación de su solenoide, el actuador volverá a su posición de reposo y no es lo deseable.

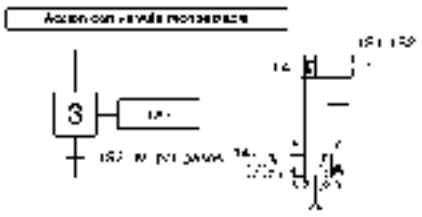


Figura 8.10. Acción con válvula monoestable.

1.4.2. Acciones sobre válvulas biestables

El control de las válvulas biestables se realiza con pulsos de dos señales independientes, una para el avance y otra para el retroceso.

Como no es necesario mantener el nivel activo de dichas señales una vez que la válvula ha realizado la conmutación de la posición del actuador, se podría desactivar la alimentación de los solenoides. Para ello, se condiciona la señal negada del sensor, o final de carrera, que se acciona cuando el movimiento alcanza el final de su recorrido con la acción que lo provoca.

No obstante, como no supone ningún problema mantener la alimentación de los solenoides de las electroválvulas biestables una vez que se ha alcanzado el final del recorrido del actuador, el tratamiento de estas acciones puede ser el mismo que el descrito para las válvulas monoestables.

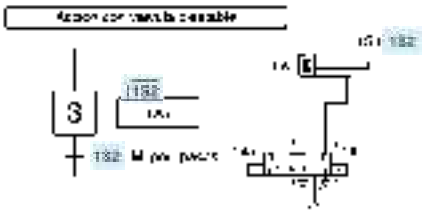


Figura 8.11. Acción con válvula biestable.

1.4.3. Acciones sobre actuadores eléctricos tipo motor

Otra cosa bien diferente se presenta cuando el movimiento lineal se produce a través de un motor eléctrico. En este caso sí que es importante desconectar el actuador una vez que se ha alcanzado el final del recorrido, por lo que siempre se debe condicionar la acción que genera dicho movimiento al detector o final de carrera que controla su desconexión no esté activado.

En el GRAFCET esto se representa como una acción condicionada negada de la señal del sensor alcanzado al final del recorrido en la etiqueta de la acción que genera el movimiento.

En el ejemplo de la figura 8.12, se muestra como un motor trifásico, supuestamente acoplado a un actuador lineal, genera movimientos a izquierdas y a derechas. Para detectar cuando el elemento móvil ha alcanzado sus extremos, se han dispuesto en ellos sendos finales de carrera (S1 y S2). Así, cuando el motor se mueve hacia la derecha en la etapa X3, si la acción está afectada por el modo «por pasos», es necesario desactivar el contactor que genera el movimiento (KM1) con una señal negada del final de carrera que se alcanzará cuando este movimiento haya concluido.

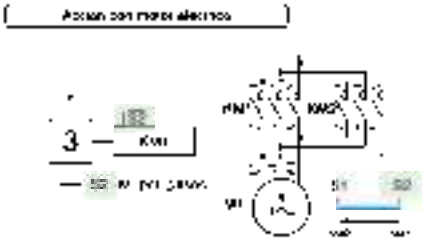


Figura 8.12. Acción con motor eléctrico.

Ejemplos

Diseña y programa el GRAFCET del taladro con cargador de piezas con las siguientes características de funcionamiento:

1. Se debe prever una secuencia Homing que permita llevar el dispositivo a las condiciones de inicio, cuyo funcionamiento es: en el momento en que el sistema pasa de STOP a RUN, la lámpara debe parpadear para indicar al operario que accione el pulsador de marcha. Una vez realizada la acción sobre este pulsador, el taladro debe situarse en la parte superior y el cargador de piezas en la zona de descarga.
2. Se debe diseñar una parada tipo pausa, que se ejecuta cuando se acciona un pulsador. Esta operación también debe implementarse en la secuencia del Homing. Después de la parada, la reanudación de la secuencia debe hacerse mediante el pulsador de marcha.
3. Deben contemplarse dos modos de funcionamiento, automático y por pasos, que se elegirán mediante un selector. El modo por pasos no debe afectar al Homing.

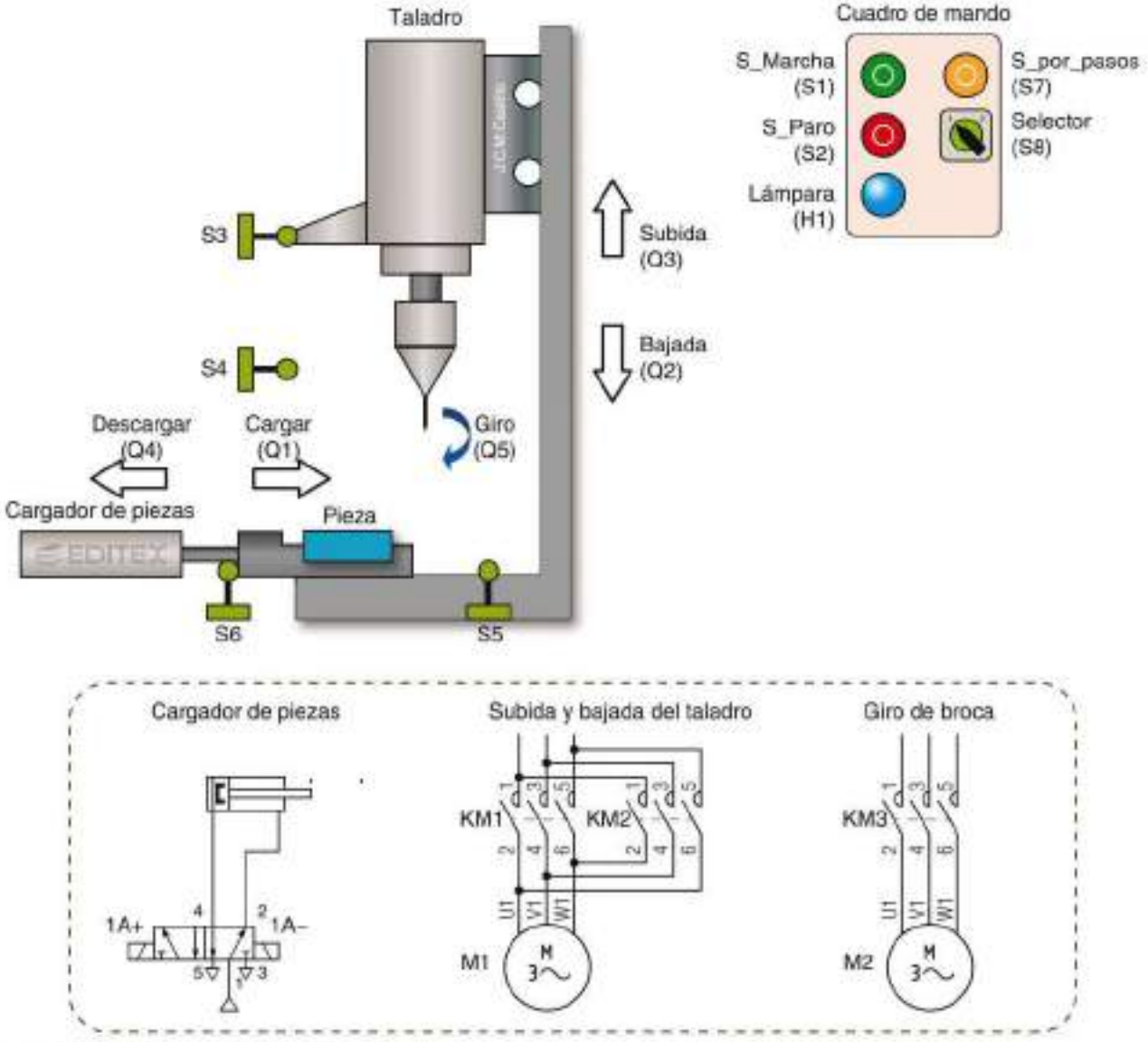


Figura 8.13. Actuadores del taladro con cargador de piezas.

continúa >

» continuación

4. En el modo por pasos hay que tener en cuenta lo siguiente:
   - El cargador de piezas se mueve con un cilindro de doble efecto controlador con una válvula biestable.
   - Los movimientos de subida y bajada del taladro se realizan con un motor eléctrico trifásico, al cual hay que invertir el sentido de giro mediante contactores.
   - La broca gira con un motor eléctrico sin inversión de sentido de giro, que no requiere ser parado cada vez que se avanza una etapa en el modo por pasos.

Así, la solución del GRAFCET de este proceso es la siguiente:

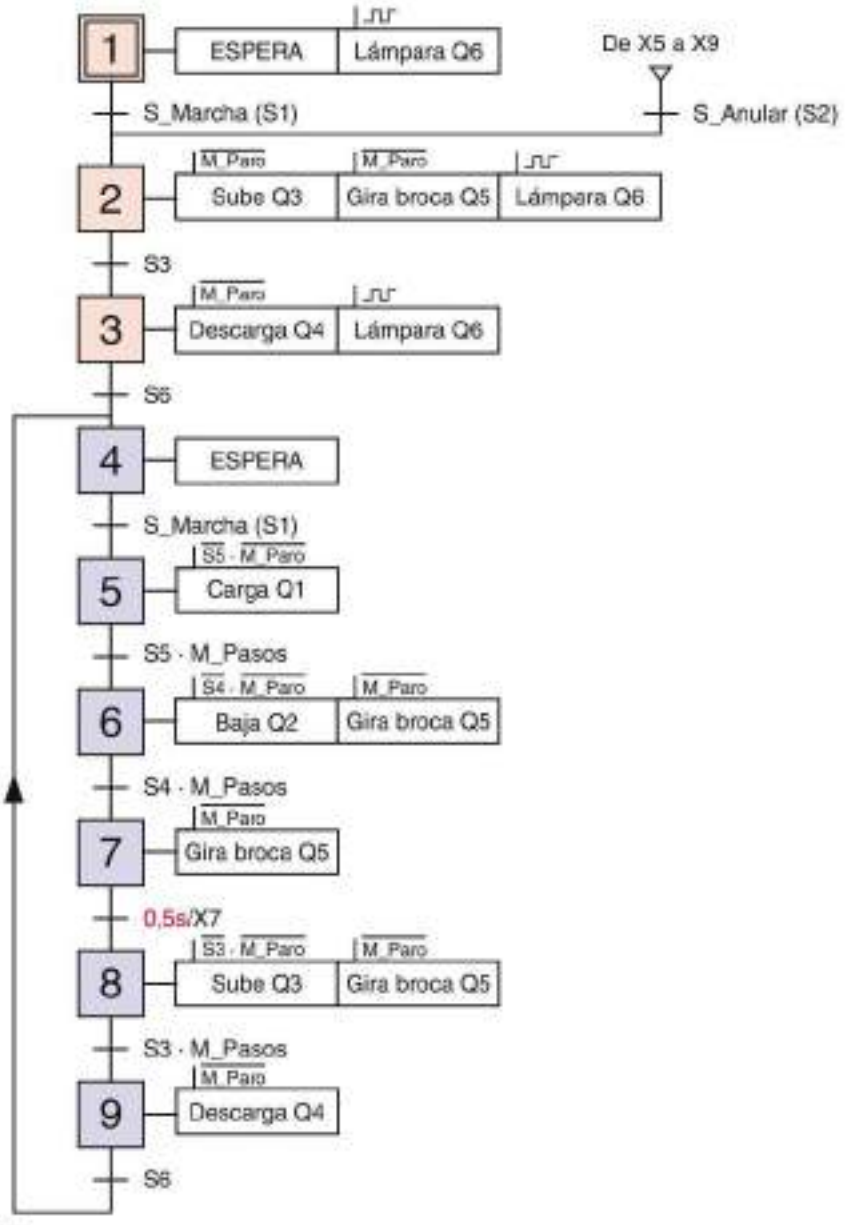


Figura 8.14. GRAFCET completo.

5. Se debe prever la cancelación del proceso con el pulsador S_Anular. Así, cuando la secuencia se encuentra entre las etapas X5 y X9, si se acciona dicho pulsador el sistema debe posicionar, por este orden, el taladro en la parte superior y el cargador de piezas en la posición recogida. Como esta secuencia ya se encuentra reflejada en la parte del Homing, se puede aprovechar realizando un salto a la etapa X2 de dicha zona.

1.5. Modo manual/automático

Muchos procesos automatizados, como pueden ser aquellos que reciben piezas o elementos en una zona de carga, pueden requerir dos formas de funcionamiento:

- Manual, en el que la secuencia se inicia voluntariamente por la acción directa del operario a través de un pulsador destinado para tal fin.
- Automático, en el que la secuencia se inicia de forma autónoma en el momento que la pieza es detectada en el punto de carga.

El sistema debe disponer de un dispositivo de conmutación que facilite la selección de un modo u otro.

Si se toma como ejemplo el taladro semiautomático estudiado anteriormente, al que se le ha añadido un detector B1 en el cargador de piezas, el inicio de la secuencia queda condicionado a la presencia o no de un objeto en dicho punto. Así, en el modo manual, el proceso se inicia siempre que B1 esté activo y se accione voluntariamente un pulsador de puesta en marcha. Sin embargo, en el modo automático la secuencia se inicia de forma autónoma cuando el elemento que se va a procesar se encuentra en el punto de carga (B1).

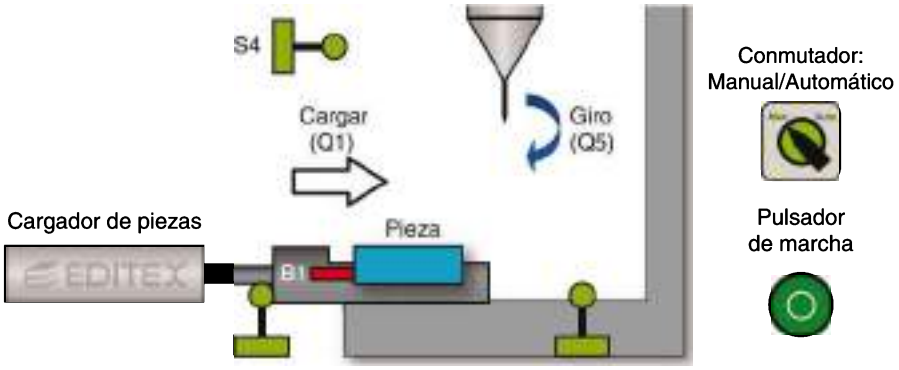


Figura 8.15. Ejemplo de uso del modo manual/automático.

De esta forma, la representación del modo de funcionamiento manual/automático se hace en el segmento de la etapa que inicia la secuencia de producción, inmediatamente después de la etapa de espera.

En el siguiente ejemplo se muestra cómo cuando el proceso se encuentra en la etapa 3 y el detector B1 es «1», en la transición que es receptiva se pueden dar dos situaciones: que el conmutador esté cerrado (modo automático), por lo que la secuencia se ejecutará de forma automática sin necesidad de intervención del operario, o que el conmutador esté abierto (modo manual), donde será necesario accionar el pulsador para ejecutar el proceso.

Figura 8.16. Segmento con el modo de funcionamiento manual/automático.

Vocabulary

- Modo por pasos: step mode.
- Puesta en marcha: startup.
- Principal: main.
- Parada: stop.
- Pulsador de parada: stop button.
- Pulsador: push-button.
- Conmutador: switch.
- Etapa incluyente: enclosing step.
- Jump: salto.
- Máquinas virtuales: virtual machines.
- Derivación: branching.
- Cadena: chain.
- Zona secuencial: sequential zone.
- Zona de acciones: actions zone.
- Etapa inicial: first step.

2. Programación de GRAFCET estructurado

A continuación se describe la forma de programar las cadenas de GRAFCET estructurado descrito en la unidad anterior.

2.1. Macroetapas

El uso de las macroetapas consta de dos partes: la representación de la macroetapa en el GRAFCET maestro o principal y la cadena que se ejecuta cuando se invoca la macroetapa.

La programación de una macroetapa en la zona secuencial del GRAFCET maestro se hace de forma similar a cualquier otra etapa.

Cuando se accede a una macroetapa en el GRAFCET principal se debe activar la etapa de entrada de su cadena auxiliar usando un flanco positivo del bit de la macroetapa. A partir de ahí, la zona secuencial de esta cadena se programa como en cualquier otra secuencia.

Para abandonar una macroetapa se debe usar el bit de la etapa de salida de su cadena asociada. Este hay que emplearlo en la transición que está por debajo de ella.

Es aconsejable separar en diferentes bloques FC las zonas secuenciales del GRAFCET maestro y de la macroetapa; sin embargo, el bloque de la zona de acciones puede ser el mismo.

En el bloque de inicialización (OB100) se debe activar la etapa inicial X1 y resetear todas las demás etapas, incluidas las de la macroetapa.

Pulsador de parada

Las acciones de una macroetapa también deben poderse desactivar con el pulsador de parada.

Recuerda

Una macroetapa debe diseñarse para que no se pueda salir de ella hasta que finalice y, además, no puede ser reutilizada en el mismo programa.

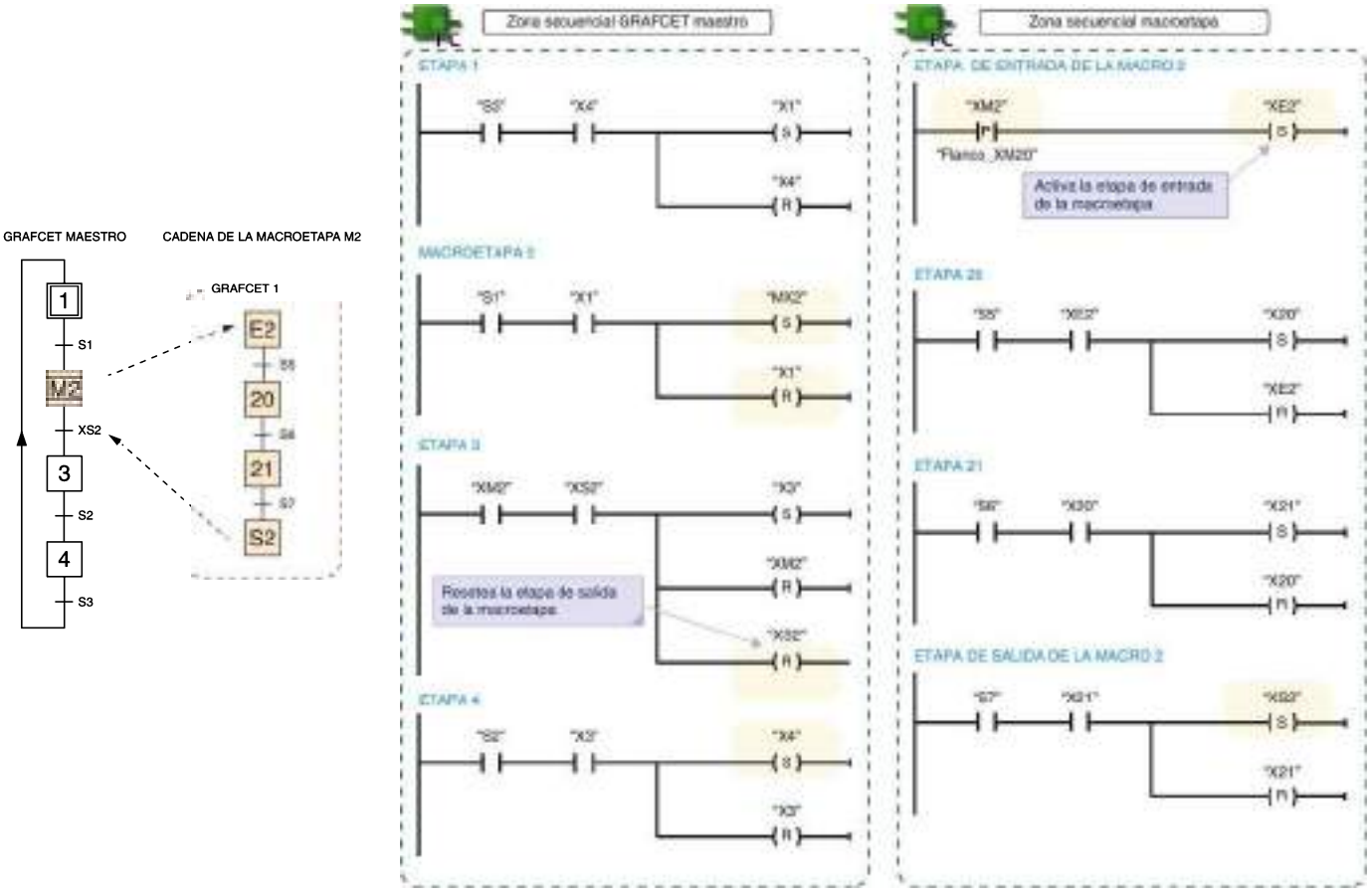


Figura 8.17. Ejemplo de programación de una macroetapa.

2.2. Etapas incluyentes

De igual forma que para las macroetapas, cuando se programa una etapa incluyente es aconsejable programar su zona secuencial en un FC diferente del bloque de la zona secuencial del GRAFCET maestro.

El GRAFCET parcial, llamado desde una etapa incluyente, puede tener bucle de retorno para un funcionamiento cíclico o ser una secuencia sin retorno basada en etapas fuente y pozo.

La zona secuencial del GRAFCET maestro se programa de forma similar a lo que se ha estudiado anteriormente en otros tipos de GRAFCET.

Es muy importante tener en cuenta que una etapa incluyente se puede abandonar aunque no se haya ejecutado por completo el GRAFCET parcial asociado, por lo que es necesario añadir un segmento de programación que permita resetear todas las etapas de dicho GRAFCET parcial cuando se sale de la etapa incluyente.

Recuerda

Las etapas incluyentes no se pueden repetir en un mismo GRAFCET.

Cuando la etapa incluyente está activa se debe poner a SET la etapa por la que comienza la secuencia del GRAFCET parcial, la cual ha de representarse con un asterisco (*). Esta operación debe hacerse con un flanco positivo del bit de la etapa incluyente. Por lo demás, esta cadena GRAFCET se programa de forma similar a lo estudiado anteriormente.

Los GRAFCET parciales pueden disponer de una etapa inicial. En este caso, es necesario activarla junto con la etapa inicial del GRAFCET maestro desde el bloque de inicialización (OB100).

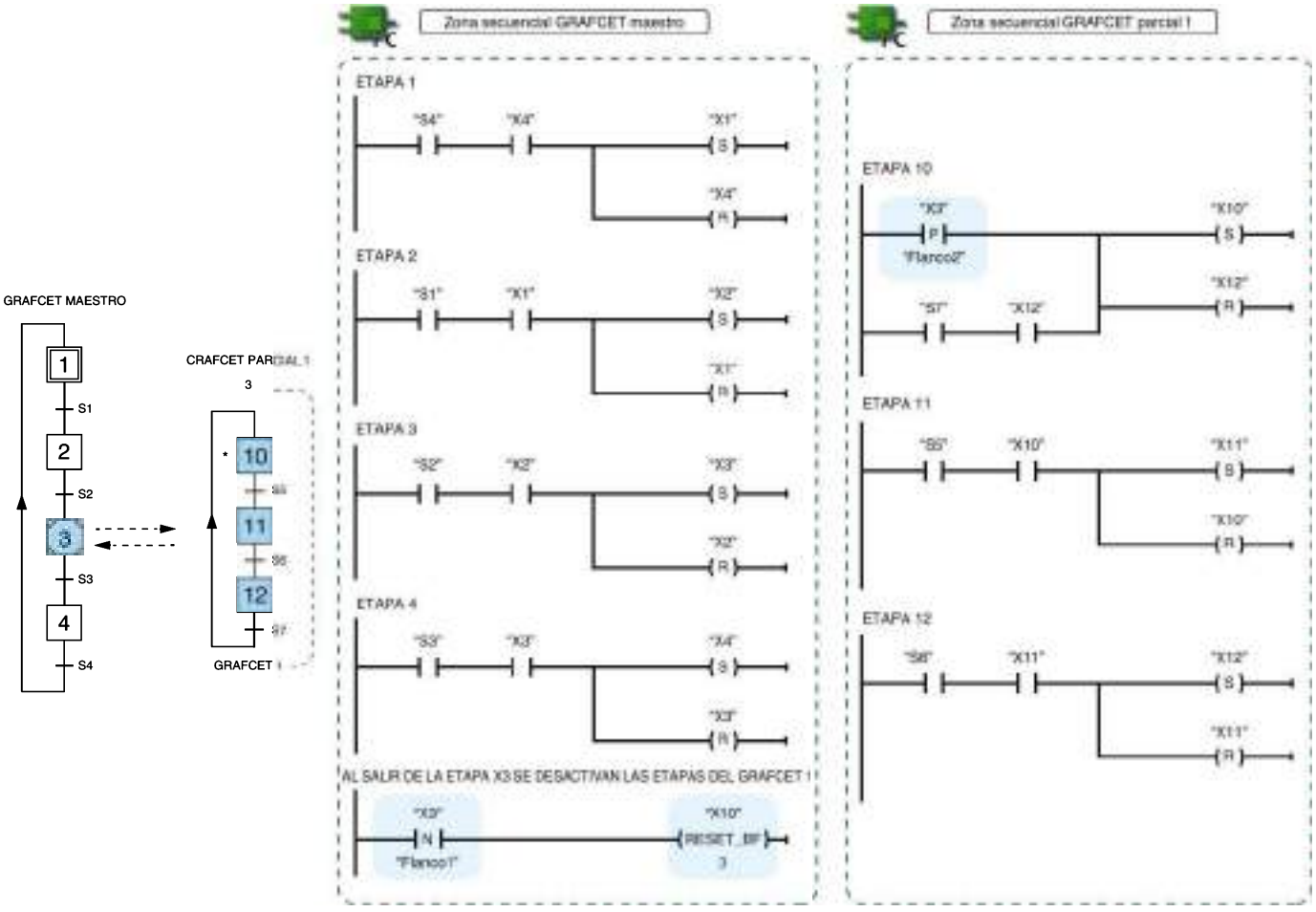


Figura 8.18. Programación de una etapa incluyente.

2.3. Acciones de llamada a GRAFCET parciales

La llamada a los GRAFCET parciales como si se tratase de acciones es una herramienta mucho más potente y flexible que el uso de macroetapas o etapas incluyentes.

Para realizar este tipo de programación se pueden usar diferentes soluciones. A continuación, vamos a desarrollar y explicar una solución que consiste en utilizar un bloque FC exclusivo para las acciones relacionadas con la llamada a GRAFCET parciales.

A continuación se muestran los diferentes casos que se pueden presentar para realizar este tipo de operaciones.

Ejecución de GRAFCET parciales

Hay que tener en cuenta que los GRAFCET parciales se pueden ejecutar de forma paralela al GRAFCET desde el que se hace la llamada.

2.3.1. Llamada a GRAFCET parcial activando una o más etapas

Se puede hacer una llamada a un GRAFCET parcial activando una o varias de sus etapas. La sintaxis para hacerlo es la siguiente: Gn{Xn_1, Xn_2,...}, donde Gn define el número del GRAFCET que llamar y Xn la etapa o etapas que activar cuando dicho GRAFCET es invocado.

En el ejemplo de la figura se muestra un GRAFCET parcial (G2) con dos cadenas secuenciales, una de funcionamiento cíclico (de X10 a X12) y la otra basada en una etapa fuente y una etapa pozo (de X20 a X22).

- Cuando el programa se ejecuta la primera vez (el PLC pasa de STOP a RUN) se activa la etapa X1 del GRAFCET principal.
- Cuando la secuencia se sitúa en la etapa X2 del GRAFCET principal se activan las etapas X10 y X20. En esa situación, y aunque el GRAFCET principal pase a la etapa X3, las cadenas secundarias evolucionan a medida que se van flanqueando sus respectivas transiciones.
- Cuando el flujo del programa activa la etapa X4 se realiza una llamada al GRAFCET secundario que activa la etapa X20 y desactiva cualquiera de las etapas de la cadena de X10 a X12, cuyo estado se mantiene, aunque el GRAFCET evolucione a la etapa X1 por la transición S4.

La programación de la zona de acciones global y las zonas secuenciales, tanto del GRAFCET principal como del parcial, son similares a lo visto en ejemplos anteriores, por lo que aquí no se representa.

Bit de marca

Recuerda que cada operación de flanco debe tener asociado un bit de marca (M), cuyo uso es exclusivo para esta operación. Si dicho bit se repite en el programa, el flanco no funcionará. En los ejemplos que se muestran a continuación estas marcas se han denominado *Flanco1*, *Flanco2*, etc.

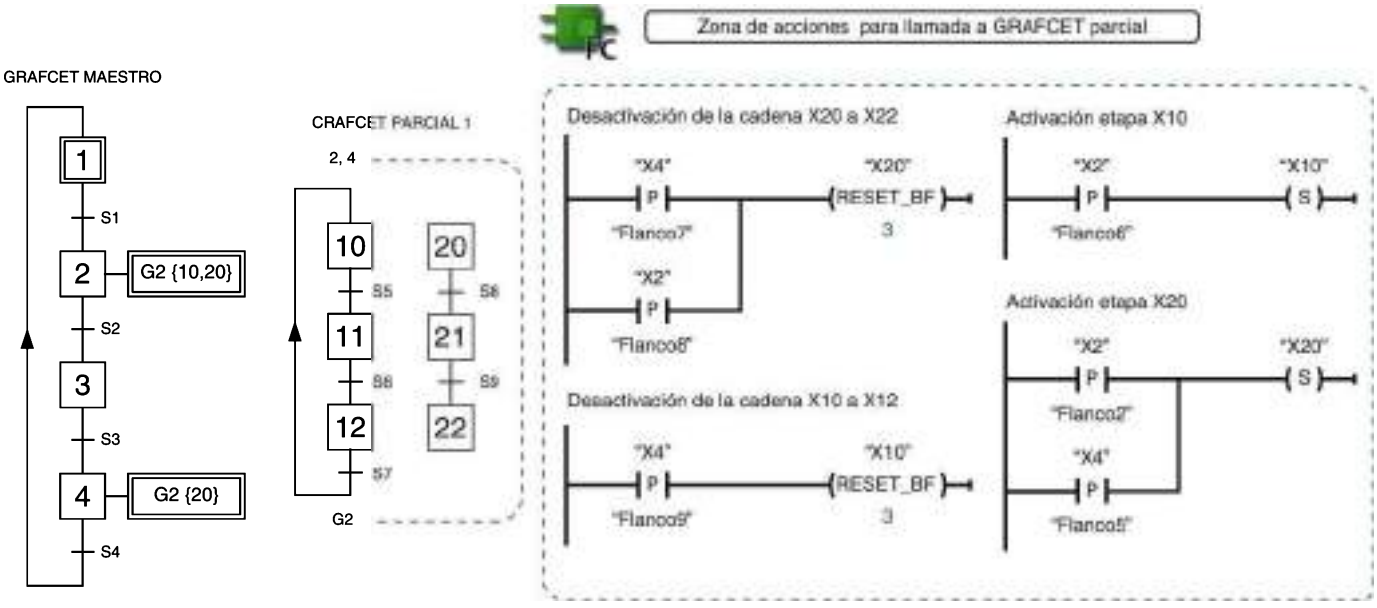


Figura 8.19. Llamada a GRAFCET parcial con activación de etapas.

2.3.2. Llamada a GRAFCET parcial con una o más etapas iniciales

La llamada a esta acción se hace usando la sintaxis: Gn{INIT}, donde Gn es el número del GRAFCET que se va a llamar e INIT es la etapa, o etapas, iniciales del GRAFCET parcial.

La programación de la zona de acciones global y las zonas secuenciales, tanto del GRAFCET principal como del parcial, son similares a lo visto en ejemplos anteriores, por lo que aquí no se representa.

Las tres etapas iniciales de la secuencia, una en el GRAFCET principal y dos en el parcial, se activan en el momento de inicializar el programa con el bloque OB100. Desde ese instante, las cadenas del GRAFCET parcial podrán evolucionar independientemente de lo que ocurra en la cadena del GRAFCET principal, salvo cuando se activa la etapa X4, donde se resetean los estados de las cadenas secundarias y se activan las etapas iniciales, que en este caso son X10 y X20.

Se debe crear un bloque FC propio para las acciones de llamada al GRAFCET parcial. En él se han de escribir las instrucciones para desactivar las etapas de las cadenas secundarias y activar las etapas representadas como iniciales.

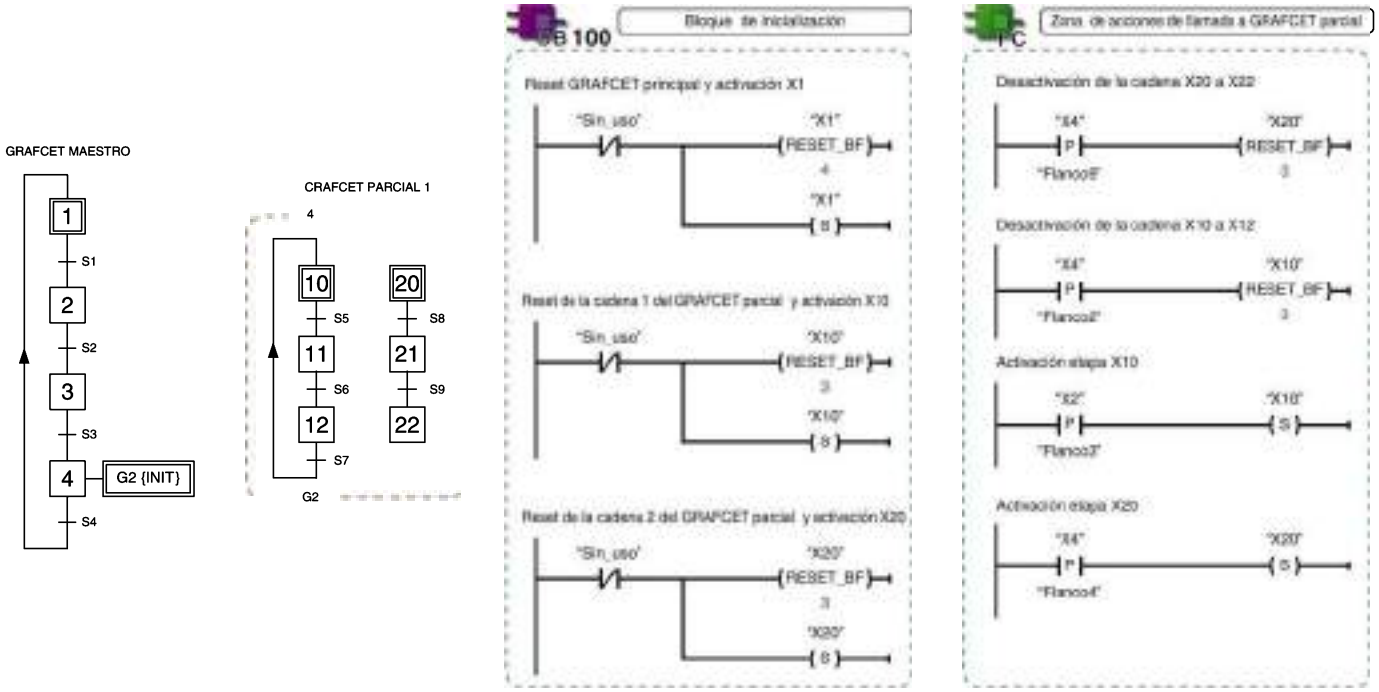


Figura 8.20. Llamada a GRAFCET parcial con etapas iniciales.

2.3.3. Inhabilitación momentánea de un GRAFCET parcial

Consiste en bloquear el funcionamiento del GRAFCET parcial en el momento que es invocado. La sintaxis que se va a emplear es Gn{*}, donde el asterisco representa que el GRAFECT secundario, nombrado en Gn, será deshabilitado.

Es importante tener en cuenta que esta acción no desactiva las acciones del GRAFCET secundario, simplemente «congela» su funcionamiento de forma temporal, lo que evita que la secuencia asociada evolucione, aunque se cumplan las condiciones lógicas de las transiciones que son receptivas en ese momento. Es decir, las señales de las acciones que están activas en el momento que se realiza la de llamada continuarán estándolo sin posibilidad de desactivarlas por la evolución lógica de la secuencia.

La acción de inhabilitación cesa en el momento que se accede a otra etapa en el GRAFCET principal, la cual puede tener o no otro tipo de llamada al GRAFCET secundario.

De igual forma que en los ejemplos anteriores, la zona secuencial y las zonas de acciones del GRAFCET maestro y del parcial son similares a los ya vistos en otros ejemplos. En este caso, la acción solamente consiste en condicionar el funcionamiento de la zona secuencial del GRAFCET parcial a la no activación del bit de la etapa en la que se produce la acción.

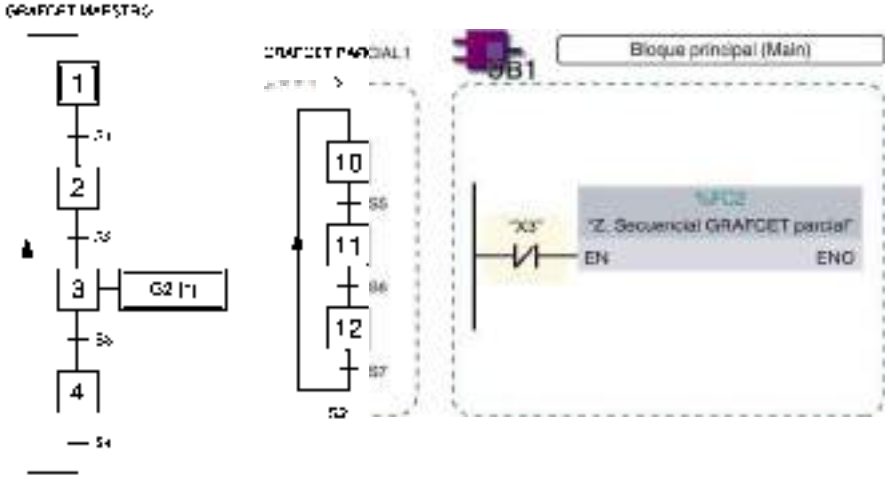


Figura 8.22. Llamada para la inhabilitación de un GRAFCET parcial.

GRAFCET parciales frente a etapas incluyentes

De igual forma que con las macroetapas, se puede condicionar la evolución del GRAFCET maestro a la finalización de la secuencia del GRAFCET parcial. La gran ventaja de usar la llamada a GRAFCET parciales como acciones se encuentra en que se pueden llamar cantas veces como sea necesario, algo que no se puede hacer con etapas incluyentes.

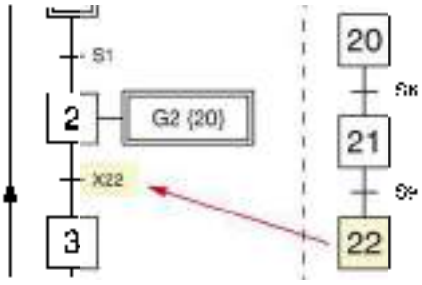


Figura 8.21. Condición de salida del GRAFCET parcial.

2.3.4. Desactivación de GRAFCET parcial

La desactivación de un GRAFCET parcial consiste en resetear todas sus etapas y, con ellas, también las señales que en sus acciones se ejecutan.

La sintaxis empleada es: Gn{ }.

En el siguiente ejemplo solo se muestra la programación relacionada con la desactivación del GRAFCET parcial. Las demás partes se realizan como se ha descrito anteriormente.

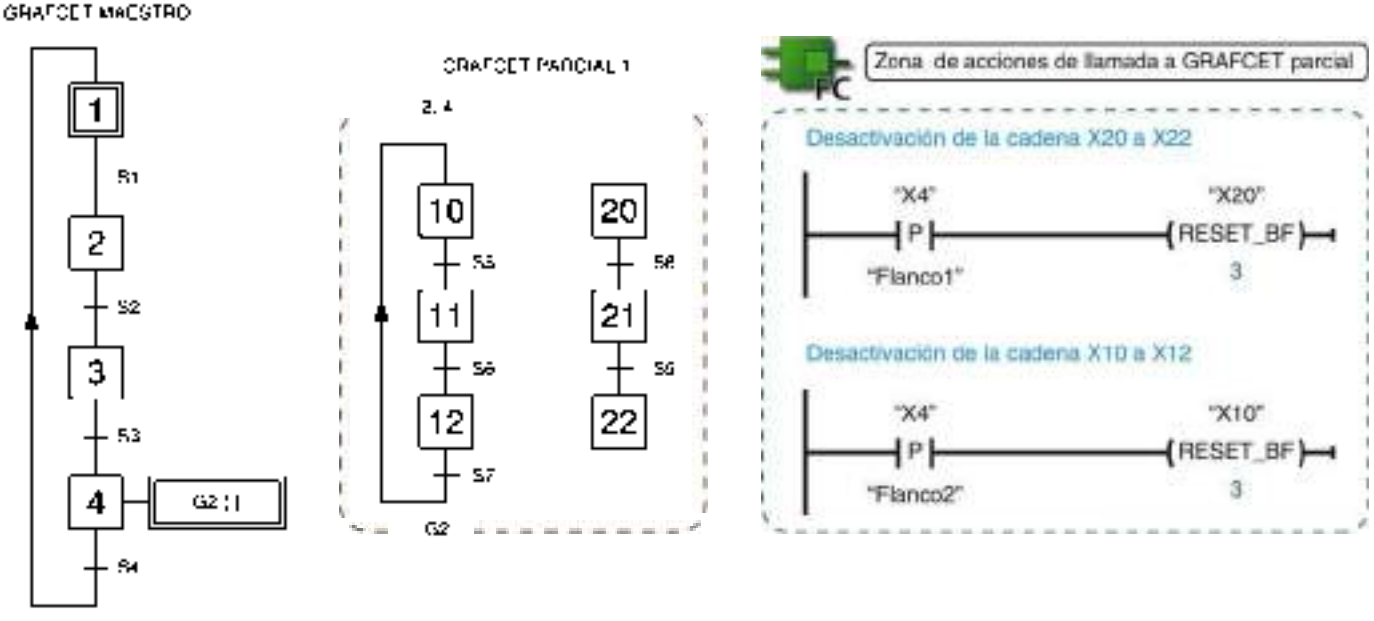


Figura 8.23. Llamada para la desactivación de un GRAFCET parcial.

Ejemplos

A continuación se muestra la programación completa de un GRAFCET estructurado. Este consta de un GRAFCET maestro, desde el cual se llama, de diferentes formas, a dos GRAFCET parciales (G1 y G1).

En la figura 8.23 se puede ver que en una misma etapa del GRAFCET maestro pueden convivir acciones de llamada a GRAFCET parciales con acciones estándar.

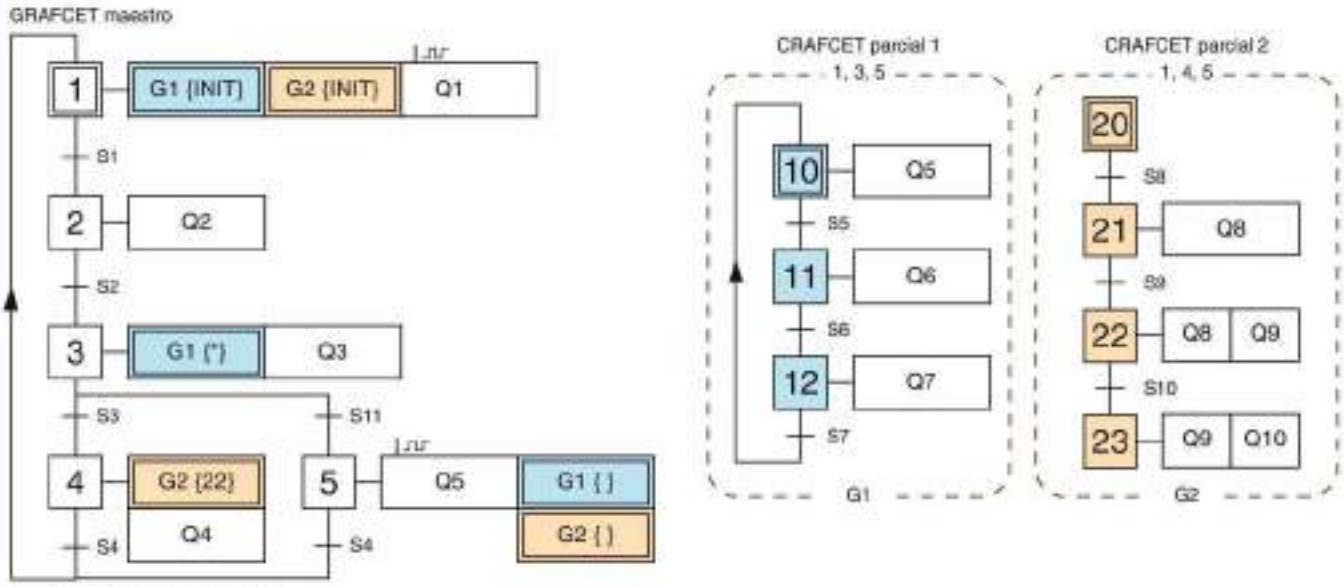


Figura 8.24. GRAFCET completo.

El programa completo en lenguaje de contactos es el siguiente:

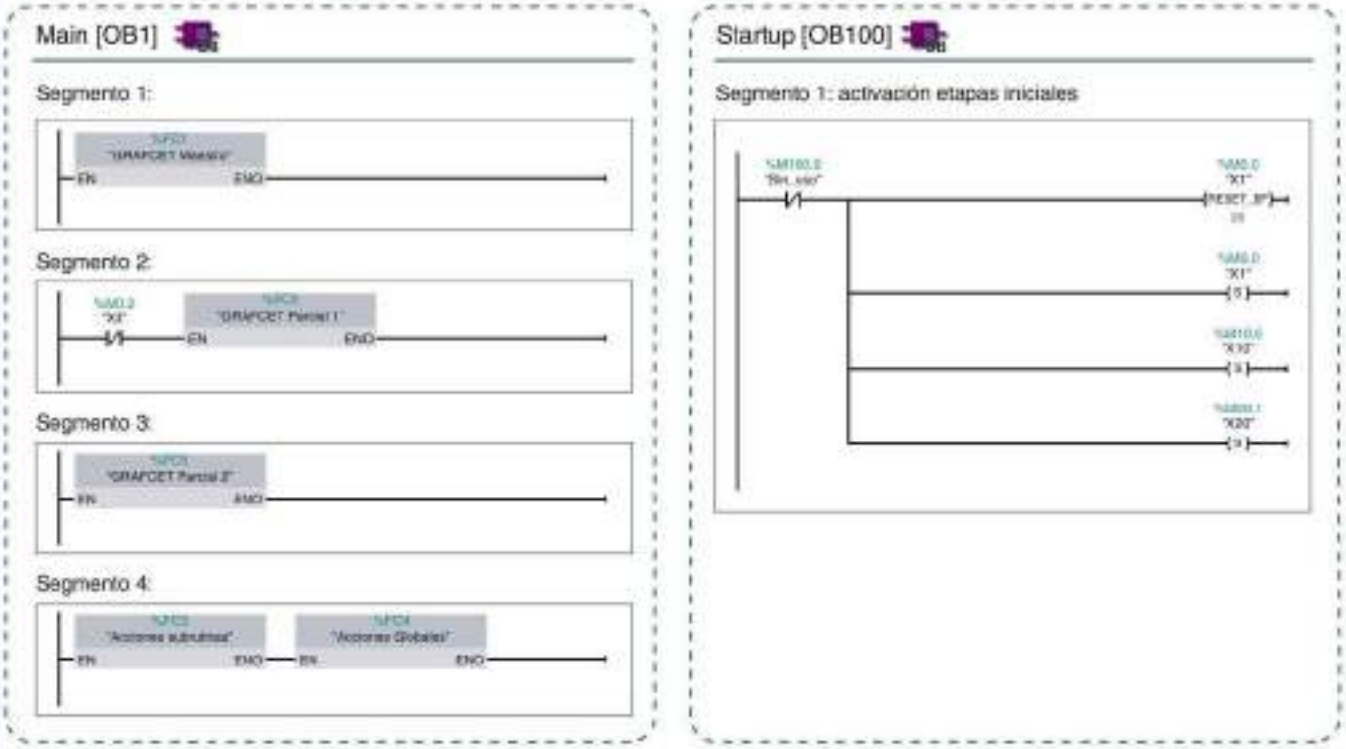


Figura 8.25. Bloques OB1 y OB100.

continúa >

» continuación

El GRAFCET maestro es:

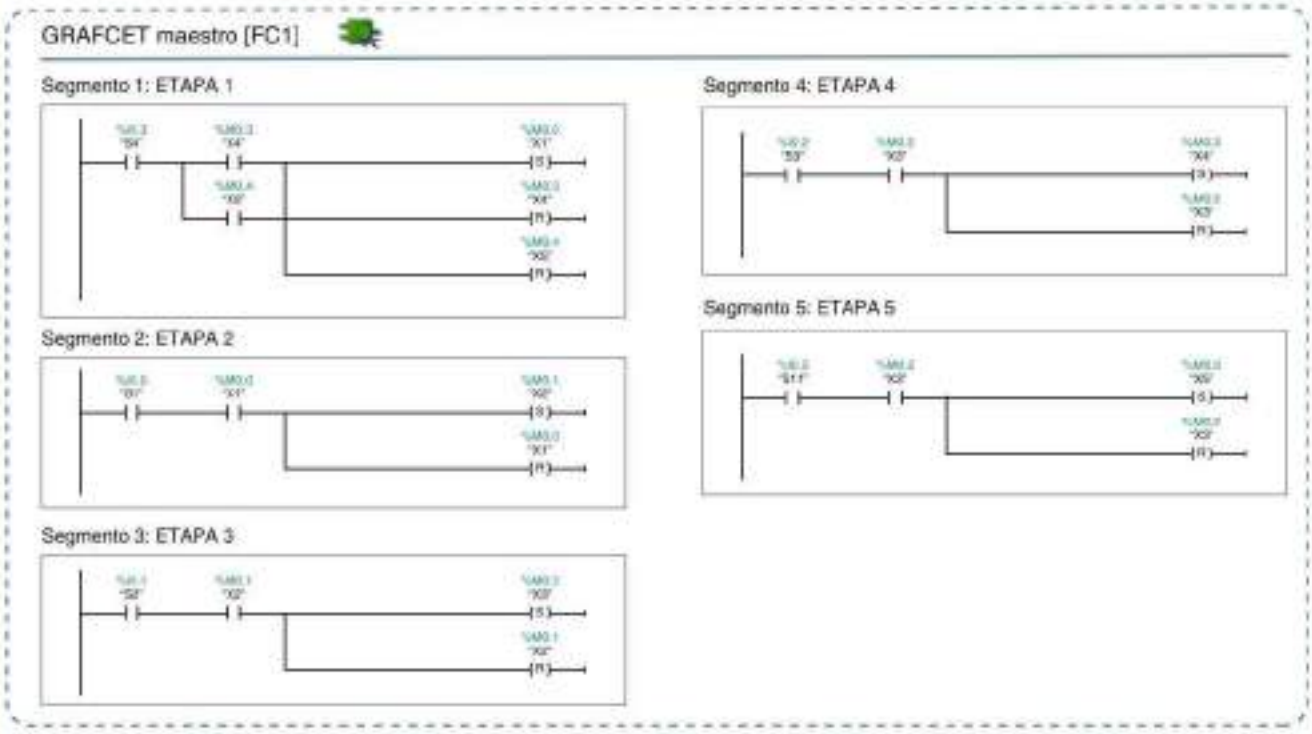


Figura 8.26. GRAFCET maestro.

Y los GRAFCET parciales son:

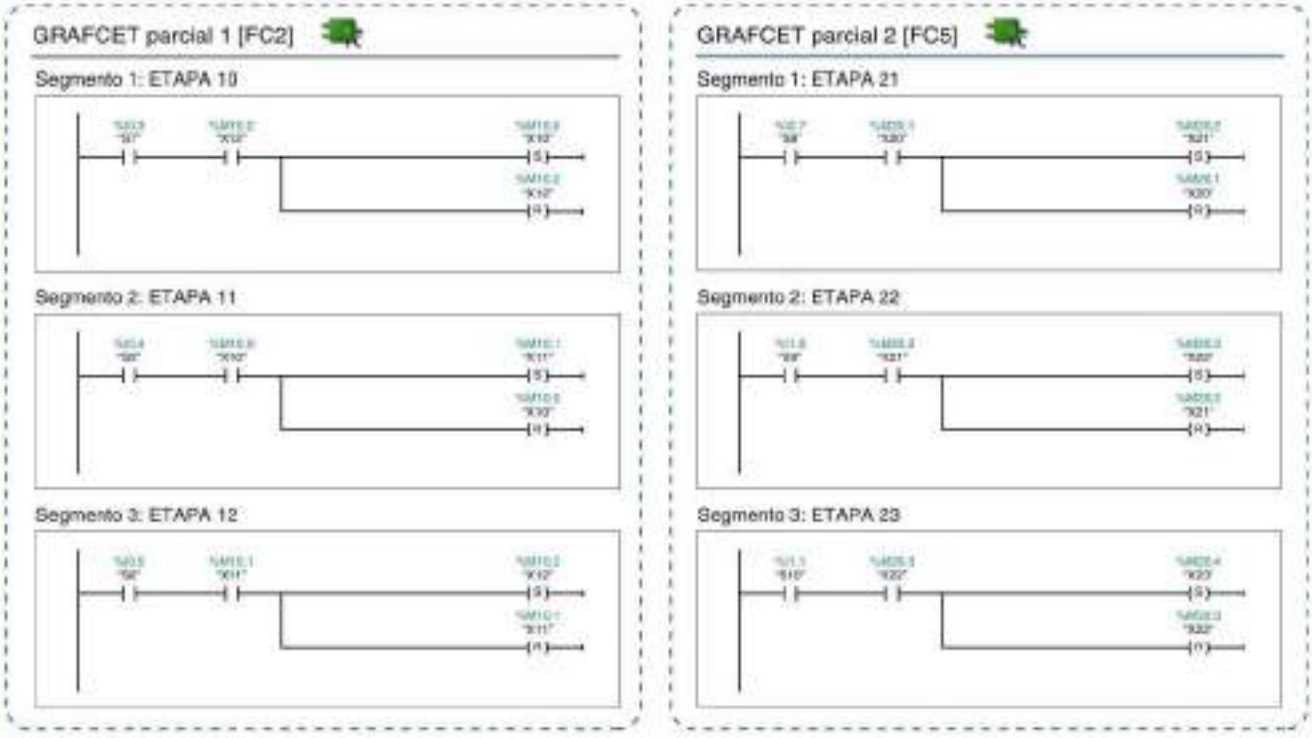


Figura 8.27. GRAFCET parciales.

continúa »

» continuación

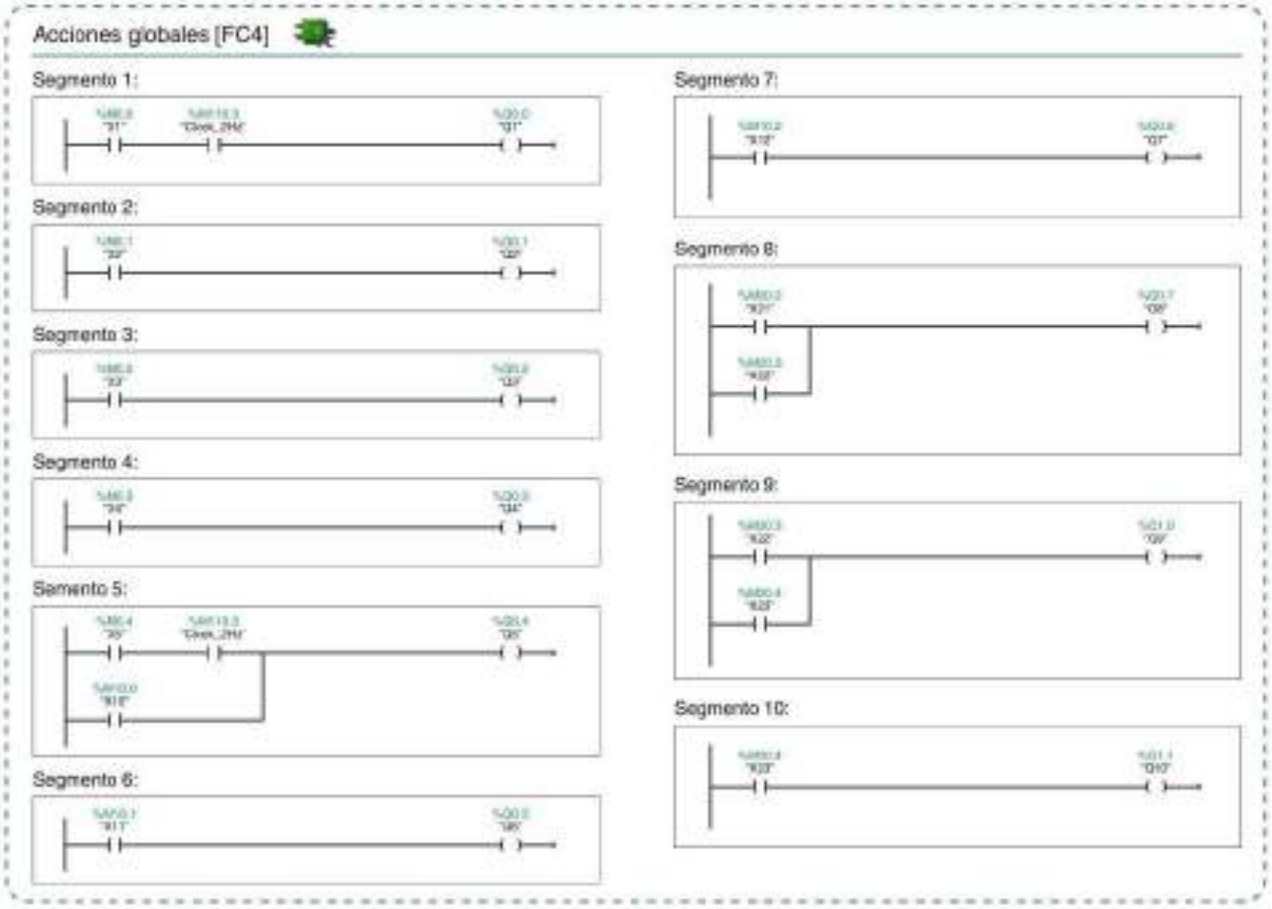


Figura 8.28. Zona de acciones globales.

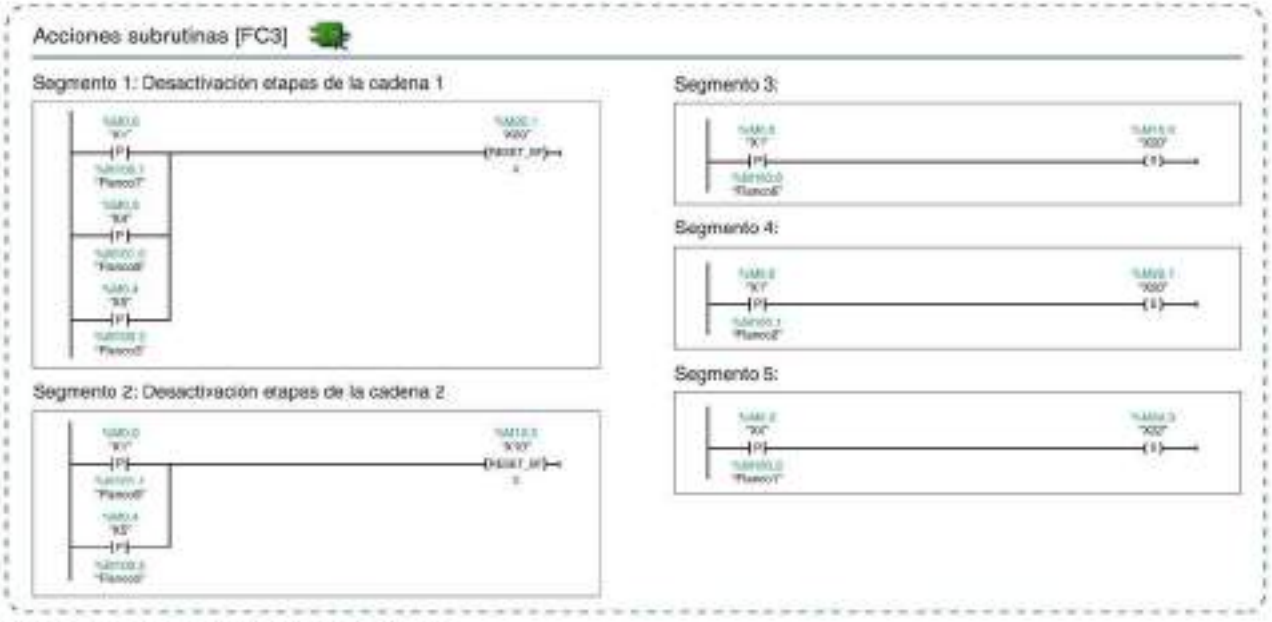


Figura 8.29. Zonas de acciones de llamadas a subrutinas.

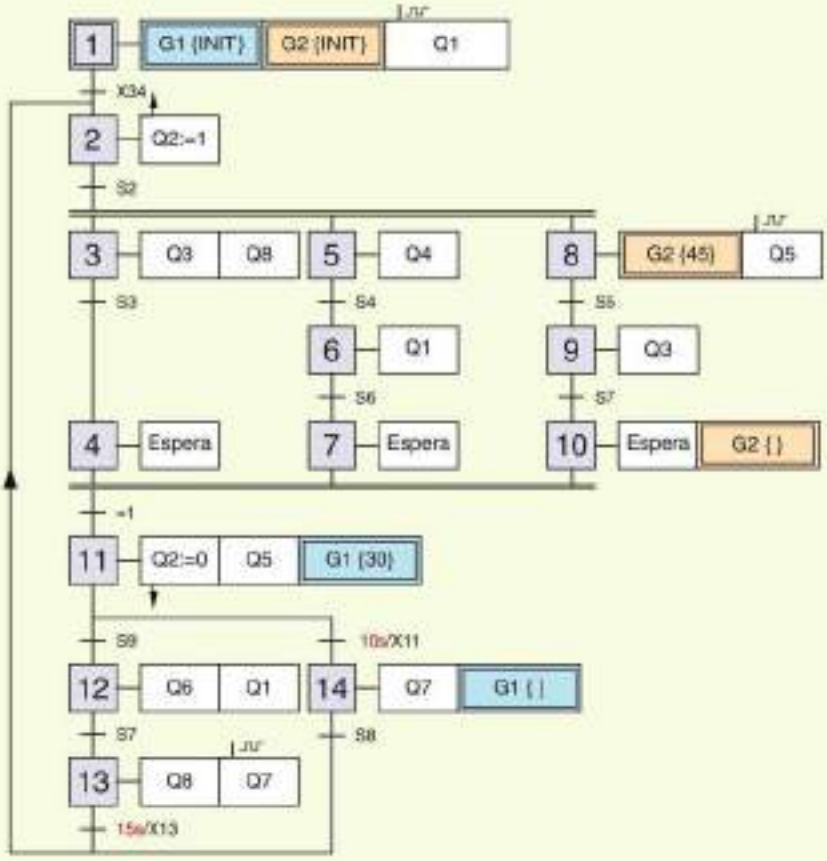
Con esto, quedaría completo el GRAFCET estructurado.

Actividades

1. Programa y comprueba el siguiente GRAFCET estructurado.

Añade la posibilidad de parar la secuencia con un pulsador de paro.

GRAFCET maestro



GRAFCET parciales

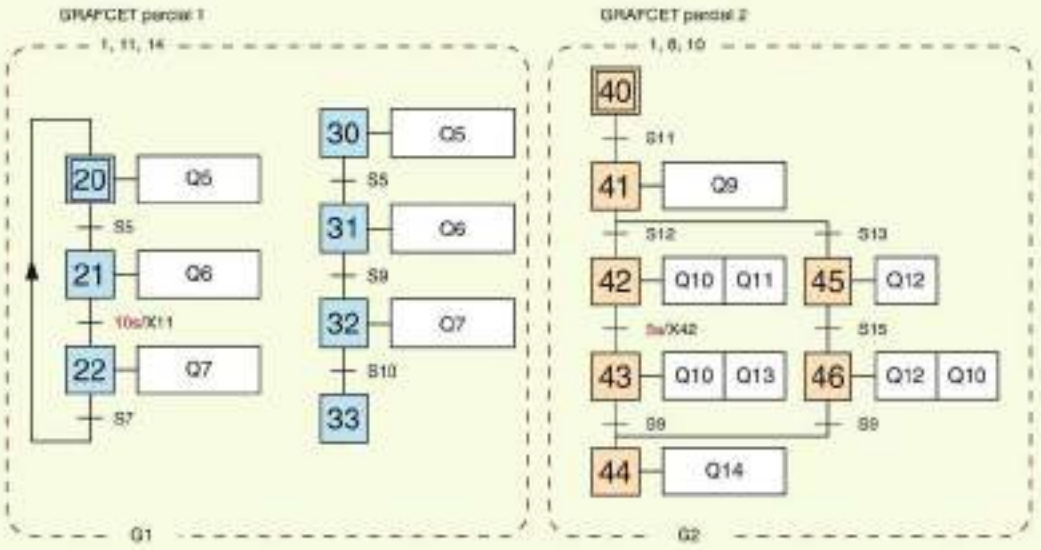


Figura 8.30. GRAFCET estructurado.

PRÁCTICA PROFESIONAL RESUELTA

Herramientas

- PC con software TIA PORTAL y PLCSIM

Material

- Un PLC compatible con TIA PORTAL
- Opcional: maqueta que permita la simulación del proceso

Programación del GRAFCET del taladro con desahogo

Objetivo

- Diseñar un GRAFCET estructurado para resolver un automatismo secuencial.
- Programar macroetapas y etapas incluyentes

Precauciones

- Hay que tener en cuenta que la subida y bajada del taladro se hace con un motor trifásico, por lo que no es posible invertir el sentido de giro de manera instantánea, ya que produciría un cortocircuito en el circuito de fuerza que controla el motor.
- Se deben prever los siguientes modos de funcionamiento: producción o automático, parada, por pasos y rearme.

Desarrollo

El proceso que se va a automatizar se basa en el taladro mostrado en ejemplos anteriores, al cual se le ha añadido un sistema para la carga automática de piezas y un final de carrera intermedio en la bajada del taladro que va a permitir realizar los orificios en dos tiempos. En el GRAFCET de producción o modo automático, después de accionar el pulsador de marcha, el cargador coloca la pieza debajo del taladro y, en esa posición, la máquina taladra en dos tiempos, según se muestra en la figura de movimientos, primero hasta la mitad de su recorrido (S7) y después hasta el final (S4). El cargador retira la pieza una vez que se ha finalizado la operación de taladro.

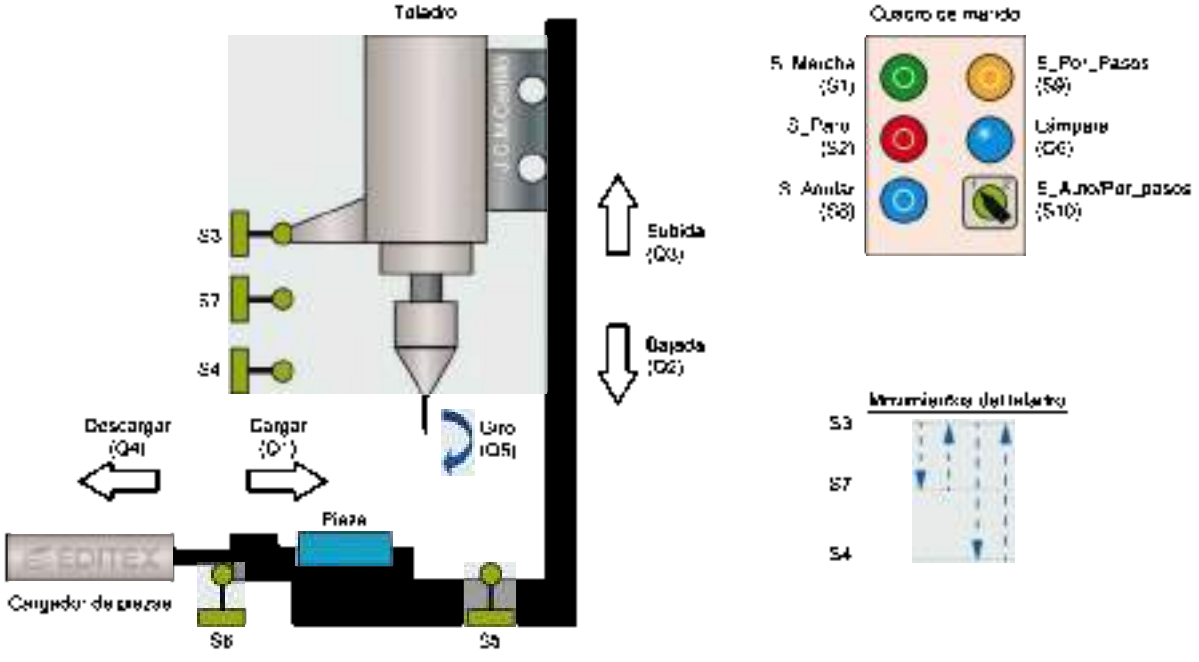


Figura 8.31. Taladro semiautomático con desahogo con cargador de piezas.

1. El primer paso es diseñar el GRAFCET del proceso, teniendo en cuenta que para la secuencia de inicialización o *Homing* se usa una macroetapa y para la de producción o modo automático, una etapa incluyente.
2. La parada se realiza por programación y, para simplificar el GRAFCET, no se ha representado en él.
3. La opción Anular funciona solamente cuando el GRAFCET está ejecutando la secuencia de producción. Si se acciona el pulsador S_Anular, el flujo de la secuencia salta hasta la macroetapa del *Homing* y lleva el taladro a las condiciones iniciales.
4. Se debe prever el modo de funcionamiento por pasos, el cual se selecciona mediante un conmutador. En este modo, cada vez que se avanza una etapa del GRAFCET de producción, los actuadores que suben y bajan el taladro, y cargan y descargan las piezas, deben ser desconectados con las señales de final de recorrido, como se ha indicado en la unidad

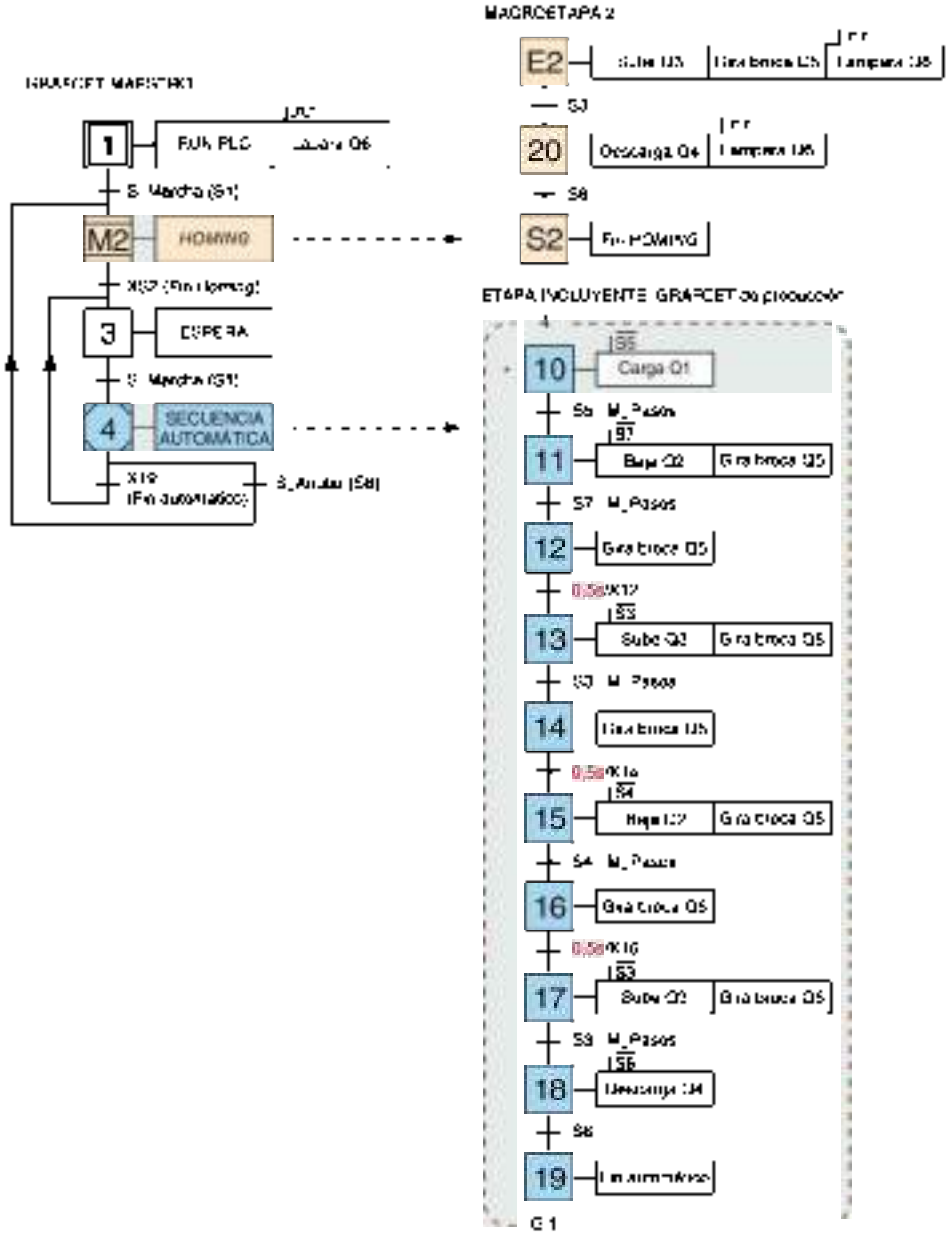


Figura 6.32. GRAFCET estructurado del taladro con destajoqu.

PRÁCTICA PROFESIONAL RESUELTA

continuación

5. Elaborar una tabla de variables con todas aquellas que aparecen en el GRAFCET. TIA PORTAL permite crear diferentes tablas, por lo que, en este caso, se podría separar por categorías: etapas, salidas, entradas, etc.

   Hay que tener la precaución de direccionar de forma consecutiva todos los bits de etapa.

   Dependiendo del modelo de PLC, el direccionamiento de las entradas y salidas puede ser diferente al que aquí se muestra.

Nombre	Dirección	Nombre	Dirección
Entradas		Salidas	
S_Marcha (S1)	%I10.0	Carga (Q1)	%Q0.0
S_Paro (S2)	%I10.1	Baja (Q2)	%Q0.1
S_Anular (S8)	%I10.2	Sube (Q3)	%Q0.2
S3	%I10.3	Descarga (Q4)	%Q0.3
S4	%I10.4	Giro (Q5)	%Q0.4
S5	%I10.5	Lámpara (Q6)	%Q0.5
S6	%I10.6		
S7	%I10.7		
S_Auto/Por_pasos (S10)	%I11.0		
S_por_pasos (S9)	%I11.1		
Etapas		Variables especiales	
X1	%M0.0	AlwaysTRUE	%M100.2
XM2	%M0.1	Clock_2Hz	%M101.3
X3	%M0.2	Clock_1Hz	%M101.5
X4	%M0.3	Clock_0.625Hz	%M101.6
X10	%M1.1	M_Paro	%M102.0
X11	%M1.2	M_Pasos	%M102.1
X12	%M1.3	Flanco_XM20	%M60.0
X13	%M1.4	Flanco_X10	%M60.1
X14	%M1.5	Flanco_S_Pulso	%M61.7
X15	%M1.6		
X16	%M1.7		
X17	%M2.0		
X18	%M2.1		
X19	%M2.2		
XE2	%M2.3		
X20	%M2.4		
XS2	%M2.5		

   Se pueden crear las variables en cualquier momento, incluso en la fase de programación.

6. Comenzar la programación de cada uno de los bloques. El orden no es importante, pero se aconseja comenzar a programar por los bloques secuenciales: GRAFCET maestro, GRAFCET del *Homing* y secuencia de producción.
7. Realizar comprobaciones parciales, con el PLC físico o con el simulador, que permitan depurar errores a medida que se elabora el programa.
8. En la programación del GRAFCET maestro o principal, cada vez que se cambia de etapa se ha optado por resetear todas las etapas del GRAFCET, incluidas las de los GRAFCETS parciales, e inmediatamente activar con SET la etapa a la que se llega. Esto evita tener que poner RESET individuales de los bits de cada una de las etapas correspondientes.

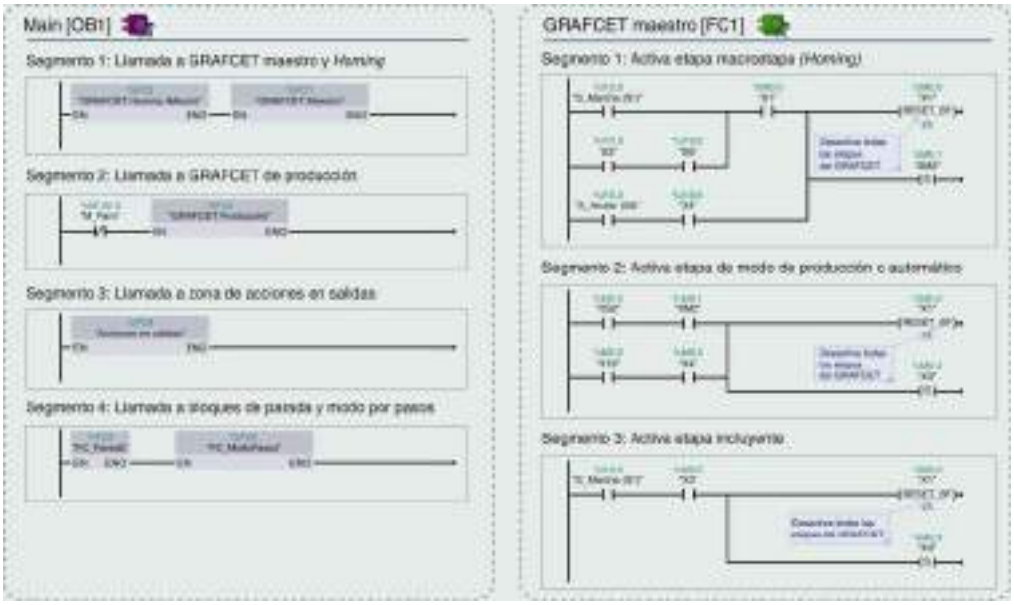


Figura 8.33. Bloque de llamadas (OB1) y FC del GRAFCET maestro.

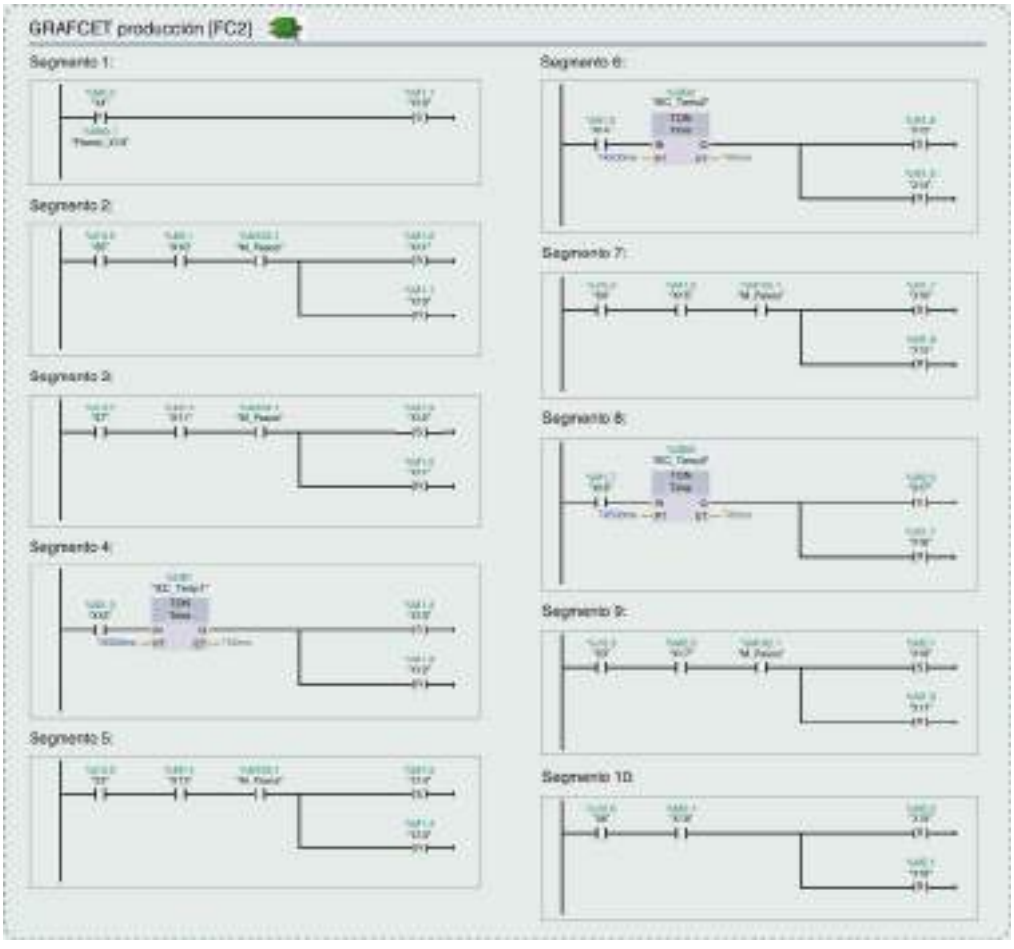


Figura 8.34. FC del GRAFCET de producción o automático.

PRÁCTICA PROFESIONAL RESUELTA

continuación

9. Una vez iniciada la secuencia de inicialización o referenciado *(Homing)* no se puede salir de ella hasta que no ha finalizado.
10. La parada debe implementarse en todas las zonas en las que la máquina se está moviendo, es decir, en el *Homing* y el GRAFCET de producción, tanto en su funcionamiento automático como por pasos.
11. El modo por pasos solamente se debe implementar en el GRAFCET de producción. En ningún caso debe afectar al GRAFCET de inicialización o *Homing*.
12. Se debe tener en cuenta que la subida y bajada del taladro se realiza con un motor eléctrico trifásico, por lo que es necesario insertar temporizaciones en aquellas etapas consecutivas en las que se realiza la inversión del sentido de giro de dicho motor. En este caso, hay que implementarlo entre las etapas X11-X13; X13-X15 y X15-X17.
13. En el OB100 se deben desactivar todas las etapas del GRAFCET y, posteriormente, activar la etapa de inicio X1. Para lo primero, utilizar una operación «Desactivar mapa de bits» (RESET_BF), indicando la cantidad de etapas que se quieren desactivar a la vez. En este caso hay un total de diecisiete etapas direccionadas en diferentes *bytes* de marcas. Aunque el *byte* MB0 no está completo, para evitar tener que utilizar varios bloques RESET_BF se puede programar sin problema el RESET de todos los bits consecutivos a partir del de la etapa X1, incluidos los que no se han usado en el programa.
14. Es importante que el bloque RESET_BF esté programado antes que el SET que activa la etapa inicial, ya que si está al revés no funcionará.

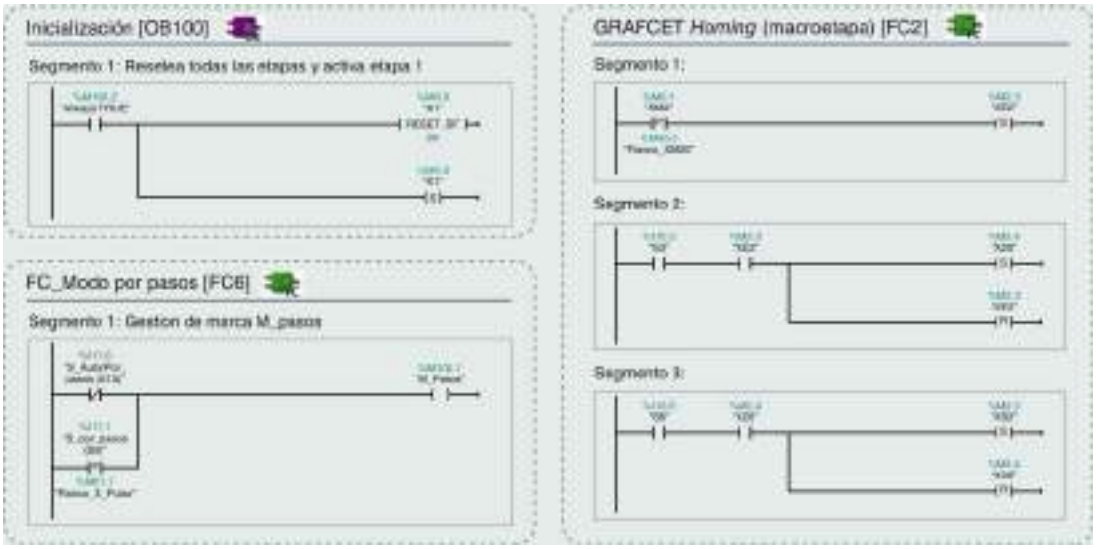


Figura 8.35. FC de bloque de inicialización, modo por pasos y macroetapa del *Homing*.

15. El pulsador físico de parada debe ser normalmente cerrado (NC), por lo que el contacto que activa con SET la marca de paro debe ser también cerrado.
16. La reanudación de la secuencia, después de una parada, se debe hacer con un pulsador normalmente abierto. En este caso se ha optado por usar el propio pulsador de marcha.

Figura 8.36. FC del bloque de parada.

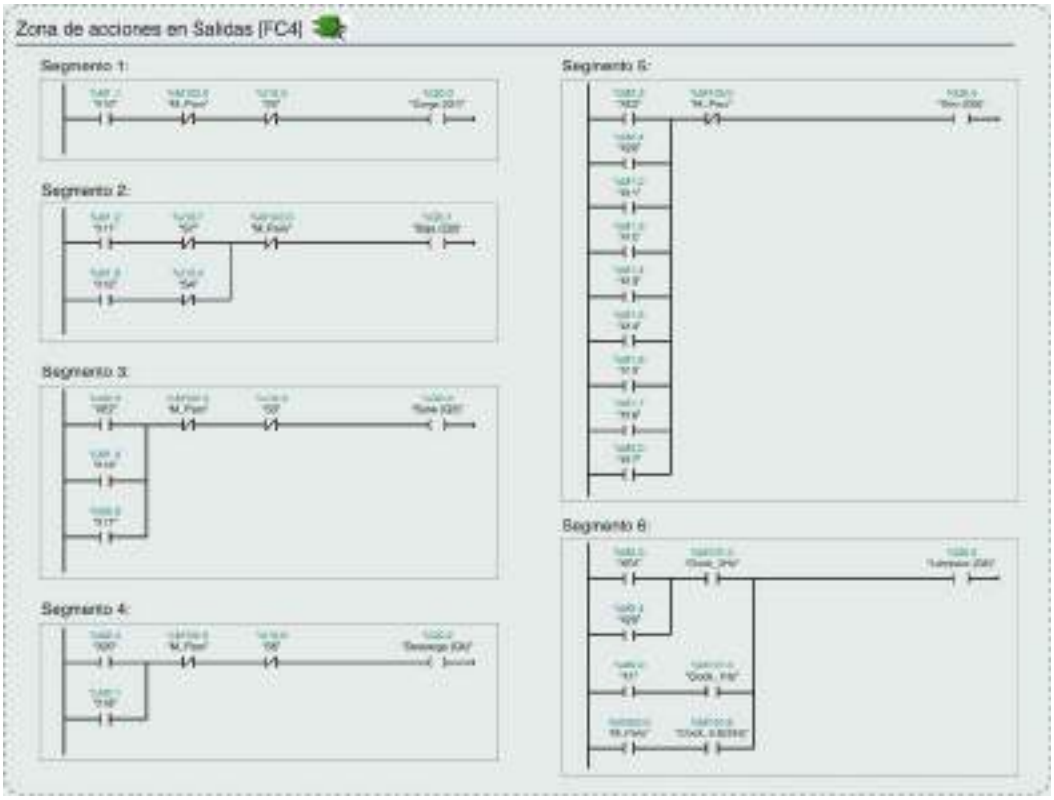


Figura 8.37. FC de la zona de acciones.

17. La organización de bloques en TIA PORTAL puede quedar como se muestra en la siguiente figura. Recuerda que haciendo clic con el botón derecho del ratón en «Bloques de programa» puedes crear grupos para organizar los bloques por similitudes funcionales.
18. Si el número de variables es elevado, es aconsejable utilizar varias tablas de variables organizadas según su funcionalidad en el programa.

Figura 8.38. Organización de bloques en el programa.

Figura 8.39. Organización de tablas de variabes.

TEST DE EVALUACIÓN

RESUELVE EN TU CUADERNO O BLOC DE NOTAS

1. **La parada en un GRAFCET se implementa:**
   a) Poniendo el pulsador de parada en serie en todas las transiciones.
   b) Realizando un salto a la etapa inicial.
   c) Condicionando las acciones a la marca de paro.
   d) Reiniciando el PLC.

2. **El modo por pasos:**
   a) Es similar al de parada.
   b) Se implementa solo en la zona de acciones.
   c) Se implementa solo en la zona secuencial.
   d) Se implementa tanto en la zona de acciones como en la secuencial.

3. **En el modo por pasos, en la acción sobre una electroválvula:**
   a) No es necesario hacer nada.
   b) Se debe desactivar el solenoide de la electroválvula.
   c) Se debe condicionar al pulsador por pasos.
   d) Se debe reiniciar el GRAFCET.

4. **En el modo manual/automático, los contactos del conmutador y el pulsador de marcha se deben conectar:**
   a) En serie y condicionar en las acciones.
   b) En paralelo y condicionar en las acciones.
   c) En paralelo y condicionar en todas las transiciones.
   d) En paralelo y condicionar en la primera transición que inicial la secuencia de producción.

5. **La programación de una macroetapa se debe hacer para que:**
   a) Se pueda salir de ella cuando sea necesario.
   b) No se pueda salir de ella hasta que no finalice.
   c) Se active al poner el PLC en RUN.
   d) Funcione de la misma forma que una etapa incluyente.

6. **La programación de una etapa incluyente se debe hacer para que:**
   a) Se pueda salir de ella cuando sea necesario.
   b) No se pueda salir de ella hasta que finalice.
   c) Se active al poner el PLC en RUN.
   d) Funcione de la misma forma que una macroetapa.

7. **Una llamada a un GRAFCET parcial con la sintaxis G2{*} significa que la secuencia del GRAFCET parcial:**
   a) Se detiene en la etapa en la que se encontraba.
   b) Se inicializa.
   c) Se desactiva.
   d) Se abandona.

8. **Una llamada a un GRAFCET parcial con la sintaxis G2{} significa que la secuencia del GRAFCET parcial:**
   a) Se detiene en la etapa en la que se encontraba.
   b) Se inicializa.
   c) Se desactiva.
   d) Se abandona.

9. **Para inicializar un GRAFCET se debe activar la etapa inicial y desactivar las demás en:**
   a) El bloque FC de la zona secuencial.
   b) El bloque FC de la zona de acciones.
   c) En el bloque OB100.
   d) En el bloque OB1.

10. **La marca de paro se controla con:**
    a) Tres pulsadores.
    b) Dos pulsadores, uno de marcha y otro de paro.
    c) Con un conmutador.
    d) Con una salida específica.

ACTIVIDADES FINALES

1. Diseña y comprueba el programa del GRAFCET que diseñaste en la unidad anterior para el sistema de llenado de contenedores en función de su tamaño. Debes implementar las funciones de parada, anular y Homing según lo descrito en dicha unidad. Utiliza GRAFCET estructurado basado en macroetapas y etapas incluyentes.

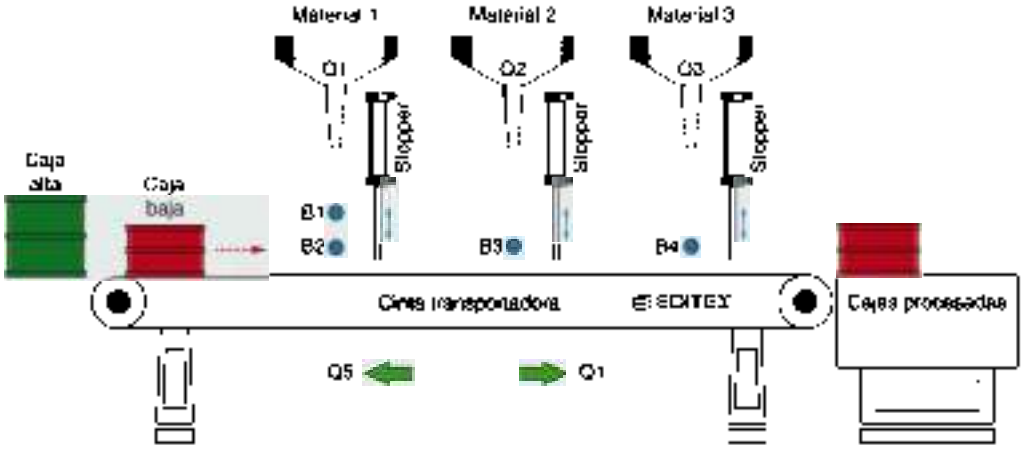


Figura 8.40. Sistema de llenado de cajas contenedoras.

2. Utilizando el GRAFCET que diseñaste en la Práctica Profesional Propuesta 2 de la unidad anterior, realiza y comprueba el programa en lenguaje de contacto del sistema de tratamiento de materiales que allí se mostraba. Además de las funciones de parada y anular, debes implementar el modo de funcionamiento por pasos. Debes tener en cuenta que la pieza no se debe soltar más que en las zonas de carga y descarga.

3. Haz lo mismo para el robot de soldadura cuyo funcionamiento se describe en la actividad de la unidad anterior. Debes ampliar el proceso con las siguientes funciones:

   a) Se debe contabilizar el número de veces que se ha completado la soldadura de piezas de forma que, si se supera un número determinado (por ejemplo, 10), se debe sustituir el electrodo del robot. Cuando esto ocurre, la secuencia debe pasar a un estado de espera en el que se active de forma intermitente la lámpara del cambio de electrodo. De dicho estado solamente se podrá salir si se actúa sobre el pulsador de acuse.

   b) Cuando se acciona alguno de los pulsadores que selecciona alguno de los tipos de soldadura, se debe señalizar, con la lámpara correspondiente, el punto o puntos por los que el robot va a pasar.

   c) Siempre que el robot esté soldando debe encenderse una lámpara que lo señalice.

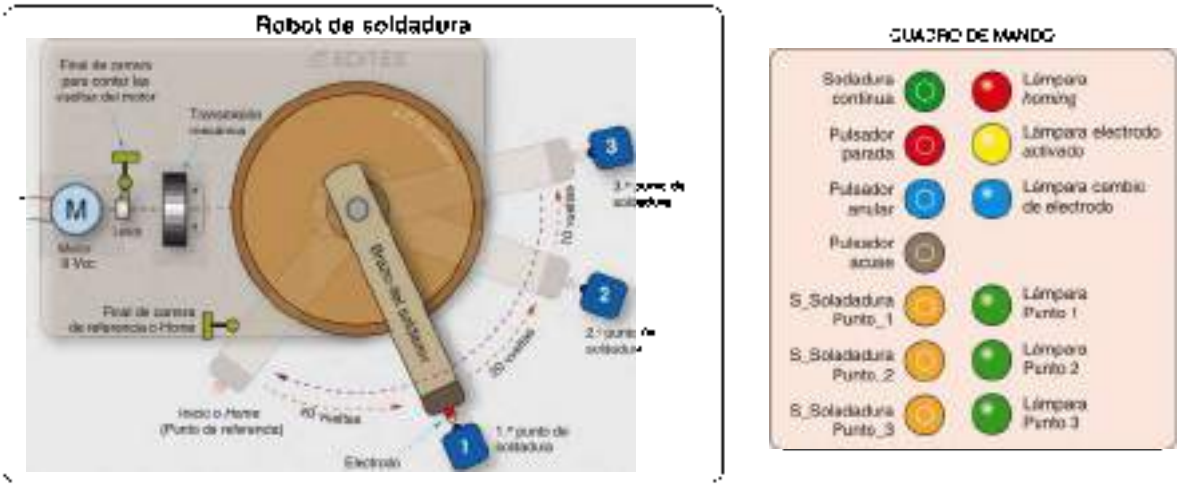


Figura 8.41. Robot de soldadura.

ACTIVIDADES FINALES

continuación

4. Diseña, programa y comprueba el GRAFCET estructurado, basado en la llamada a secuencias parciales como acciones, para el control del ascensor de tres plantas de la figura, cuyo funcionamiento se describe a continuación.
   - El sistema que se va a automatizar consiste en un ascensor de tres plantas, con puerta automática, que se desplaza junto con la cabina del ascensor.
   - Los botones del interior de la cabina tienen la misma función que los que están en cada una de las plantas. Es decir, están conectados eléctricamente en paralelo entre sí.
   - La puerta está cerrada con la cabina en reposo.
   - Cuando la cabina se encuentra en alguna de las plantas y se acciona el pulsador de llamada de cualquiera de las otras, la cabina se desplaza hasta donde fue llamada, con el siguiente funcionamiento:
     - La puerta se abre y, en dicha posición, se espera durante 3 segundos a que las personas salgan del ascensor.
     - Después de eso, la puerta se vuelve a cerrar.
     - Si la puerta se está cerrando y el detector fotoeléctrico de la puerta se acciona, se volverá a abrir y comenzará de nuevo la secuencia de control de la puerta.
   - Si el ascensor se encuentra en la misma planta en la que es llamado, la puerta se abre y posteriormente se cierra, según lo descrito anteriormente. Una vez en la cabina, se debe accionar el pulsador de la planta a la que se quiere llegar, con un funcionamiento similar al ya descrito.
   - Cuando el PLC pasa de STOP a RUN debe ocurrir lo siguiente:
     - Se deben encender las dos lámparas intermitentes en oposición para avisar de una situación de emergencia.
     - Se debe accionar el pulsador de llamada de la primera planta durante 5 segundos; para que la cabina se desplace hasta ella hasta se acciona el final de carrera correspondiente.
   - El proceso se podrá detener en cualquier momento mediante el pulsador de parada, que debe ser normalmente cerrado (NC), y la reanudación del funcionamiento, accionando cualquiera de los pulsadores de llamada de planta.

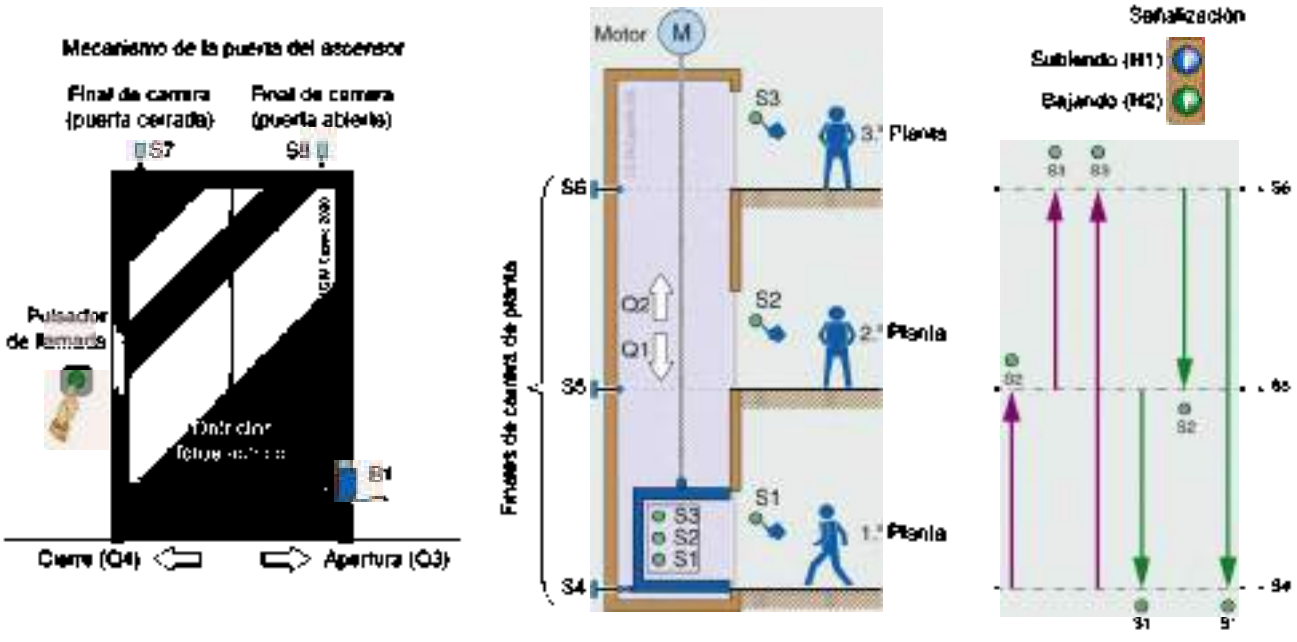


Figura 8.42. Ascensor de tres plantas con puerta automática.

PRÁCTICA PROFESIONAL PROPUESTA 1

Herramientas

- PC con TIA PORTAL y PLCSIM
- Herramientas básicas de electricista
- Herramientas para trabajar con dispositivos de neumática

Material

- Un cable para comunicar el PLC con el PC
- Un PLC compatible con TIA PORTAL
- 3 cilindros neumáticos de doble efecto
- 2 detectores magnéticos por cada cilindro
- 3 electroválvulas biestables de $24V_{DC}$
- Tubo para potencia neumática
- 1 piloto H1
- 3 contactores de $24V_{DC}$
- 1 motor trifásico
- 1 fuente de alimentación con salida a $24V_{DC}$
- Cable de 1 mm^2 para el circuito de control
- Cable de 2,5 mm^2 para el circuito de potencia
- 1 compresor

Programación de un GRAFCET de un circuito electroneumático

Objetivos

- Diseñar un GRAFCET estructurado para resolver una secuencia de un circuito electroneumático basado en un diagrama espacio-fase.
- Diseñar y montar los circuitos eléctricos y neumáticos necesarios para el control propuesto.

Precauciones

- La cadencia de parpadeo de la lámpara puede ser diferente, dependiendo de dónde se encuentre la secuencia.
- Para invertir el sentido de giro del motor trifásico son necesarios dos contactores, los cuales deben tener cableado su correspondiente circuito de fuerza a $230\,V_{AC}$. No obstante, si no se desea utilizar dicho circuito, con la observación visual de la activación de ambos contactores podría ser suficiente.
- Cada cilindro debe tener instalados dos detectores magnéticos, uno en cada extremo, que permitan saber cuándo están extendidos y cuándo recogidos.

Desarrollo

1. Crea un nuevo proyecto en TIA PORTAL para el PLC que vayas a cablear.
2. Configura la marca de ciclo en el dispositivo.
3. Observa el diagrama espacio fase de la figura y crea una lista de variables.
4. Diseña el GRAFCET con la siguiente estructura: una macroetapa para la zona inicial o *Homing* y tres GRAFCET parciales para las secuencias numeradas como 2, 3 y 4 en la figura.
5. Estructura la zona secuencial en cuatro FC, uno por cada secuencia del diagrama.
6. Crea un solo FC para la zona de acciones. En este bloque deben estar las acciones de todas las cadenas.
7. Crea el bloque OB100. En él debes desactivar todas las etapas y activar la etapa inicial de espera.
8. En el OB1, haz la llamada a todos los demás bloques.
9. Cablea el circuito al PLC.
10. Carga el programa en el autómata y comprueba su funcionamiento.

PRÁCTICA PROFESIONAL PROPUESTA 1

continuación

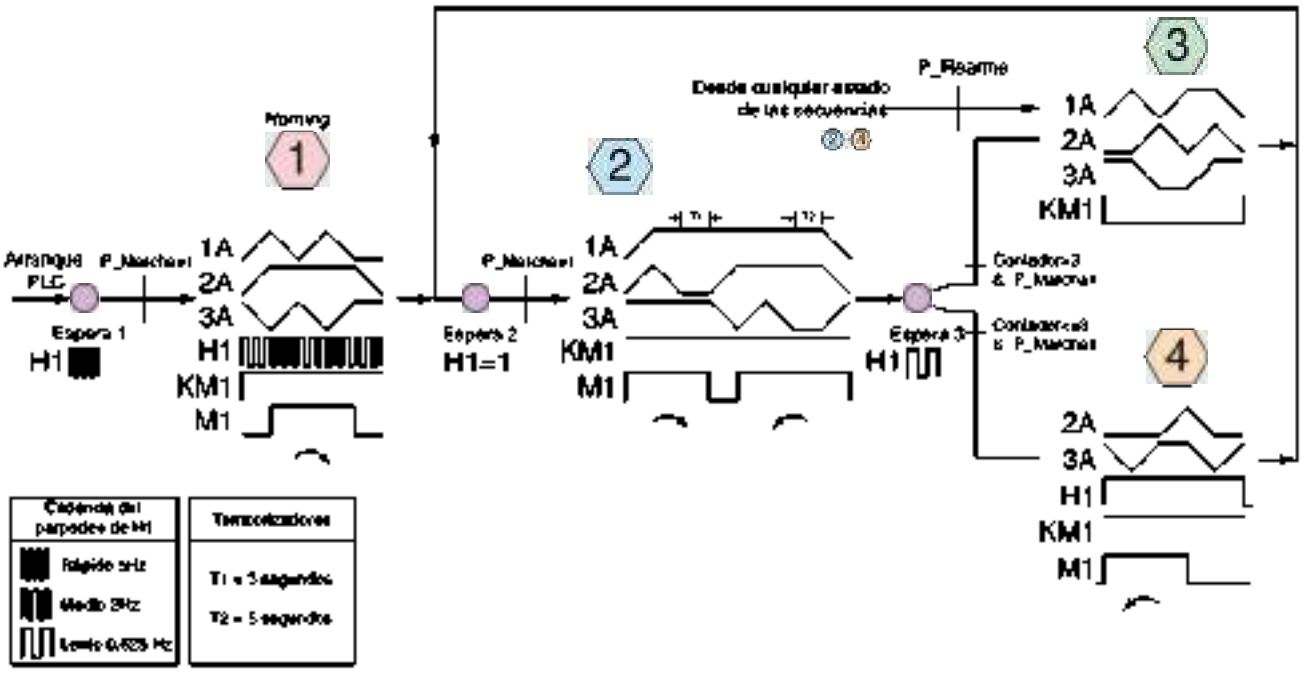


Figura 8.45. Diagrama espacio fase del circuito electroneumático.

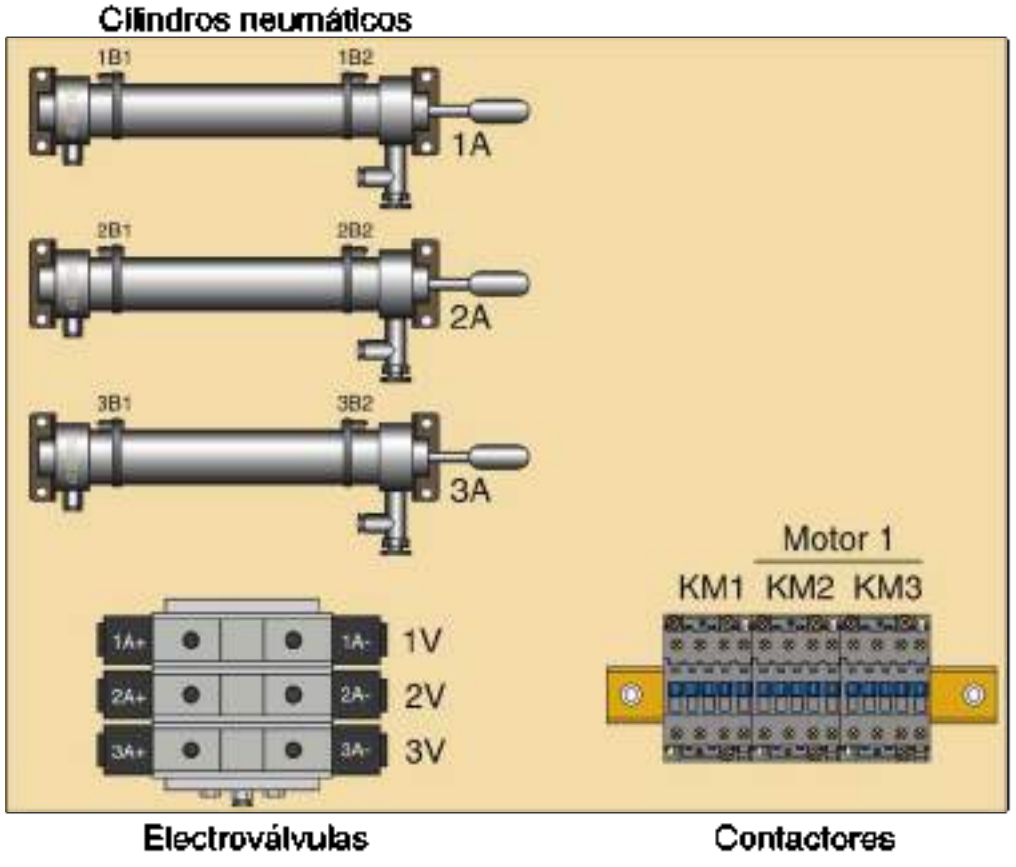


Figura 8.46. Elementos de potencia para comprobar el circuito.

PRÁCTICA PROFESIONAL PROPUESTA 2

Herramientas

- PC con TIA PORTAL y PLCSIM

Material

- Un cable para comunicar el PLC con el PC
- Un PLC compatible con TIA PORTAL
- Opcional: maqueta de simulación del proceso

Control de un sistema de selección de piezas por alturas

Objetivo

- Programar un sistema de manipulación de piezas por alturas.
- Uso de contadores en el GRAFCET.

Precauciones

- Los finales de carrera B_pos1 y B_pos3 no se encuentran en los extremos de la guía. Eso quiere decir que el carro puede sobrepasarlos tanto en un sentido como en el otro.
- La pinza está controlada por una electroválvula monoestable, de forma que si el solenoide no recibe alimentación eléctrica, esta debe permanecer cerrada para evitar que la pieza caiga.
- El punto de referencia del sistema se consigue cuando la pinza está abierta sobre el punto de carga.

Desarrollo

1. Diseña el GRAFCET del sistema automático de la figura.
2. Implementa en TIA PORTAL el programa y comprueba su funcionamiento.

Descripción del funcionamiento

- Cuando el PLC pasa de STOP a RUN, el carro debe buscar el punto de referencia.
- Se deben prever dos modos de funcionamiento: manual y automático. En el primero es requerida la acción sobre el pulsador de marcha. En el segundo, el sistema funciona de forma automática cuando se detecta una pieza en el punto de carga.
- Si el carro ha sobrepasado los límites marcados por los detectores B_pos1 y B_pos2, el sistema debe salir de estas zonas y localizar de igual forma el punto de referencia.
- Cuando hay una pieza en el punto de carga y se acciona el pulsador de marcha, la pinza la recoge y la deposita en el punto de descarga correspondiente en función de su altura.
- Cada punto de descarga permita almacenar cinco piezas. Cuando se ha sobrepasado este número se activa H1 de forma intermitente. En esta situación, si se manipula alguna pieza cuyo almacén está completo, debe ser rechazada dejándola caer por la rampa de la derecha.
- Los almacenes se vacían manualmente. Cuando esto ocurre, el operario debe accionar el pulsador RESET para poner a cero el contador asociado y que el proceso pueda depositar nuevamente piezas en ese punto de descarga.

PRÁCTICA PROFESIONAL PROPUESTA 2

continuación

- Cuando el sistema está manipulando alguno de los tipos de piezas y se accione el pulsador RESET, estas deben ser rechazadas dejándolas caer por la rampa de la derecha. Después de esto, el carro debe desplazarse hasta en el punto de referencia.

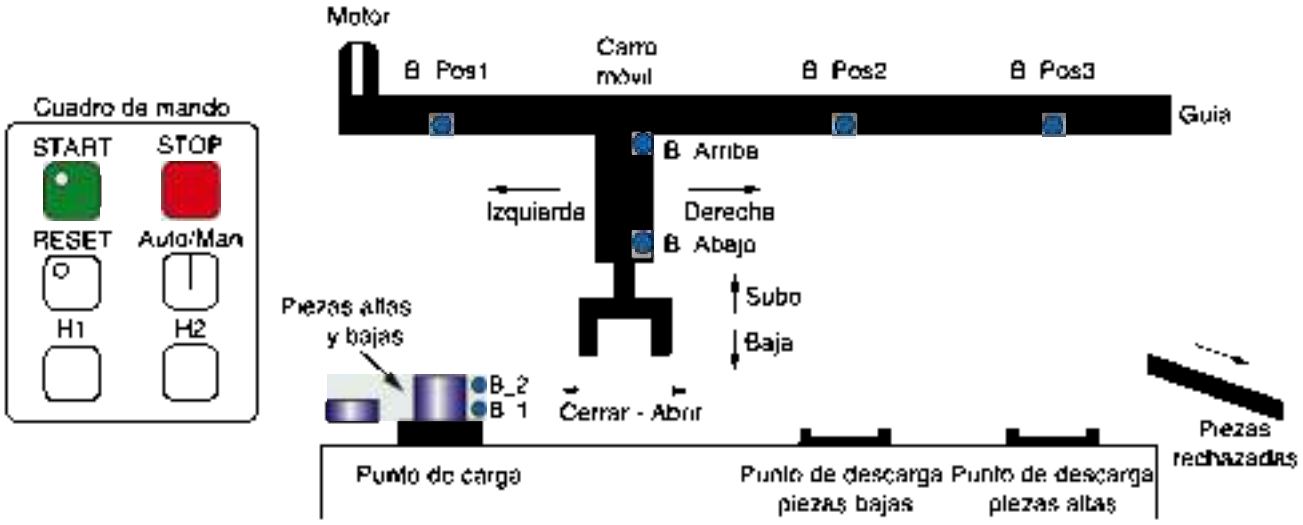
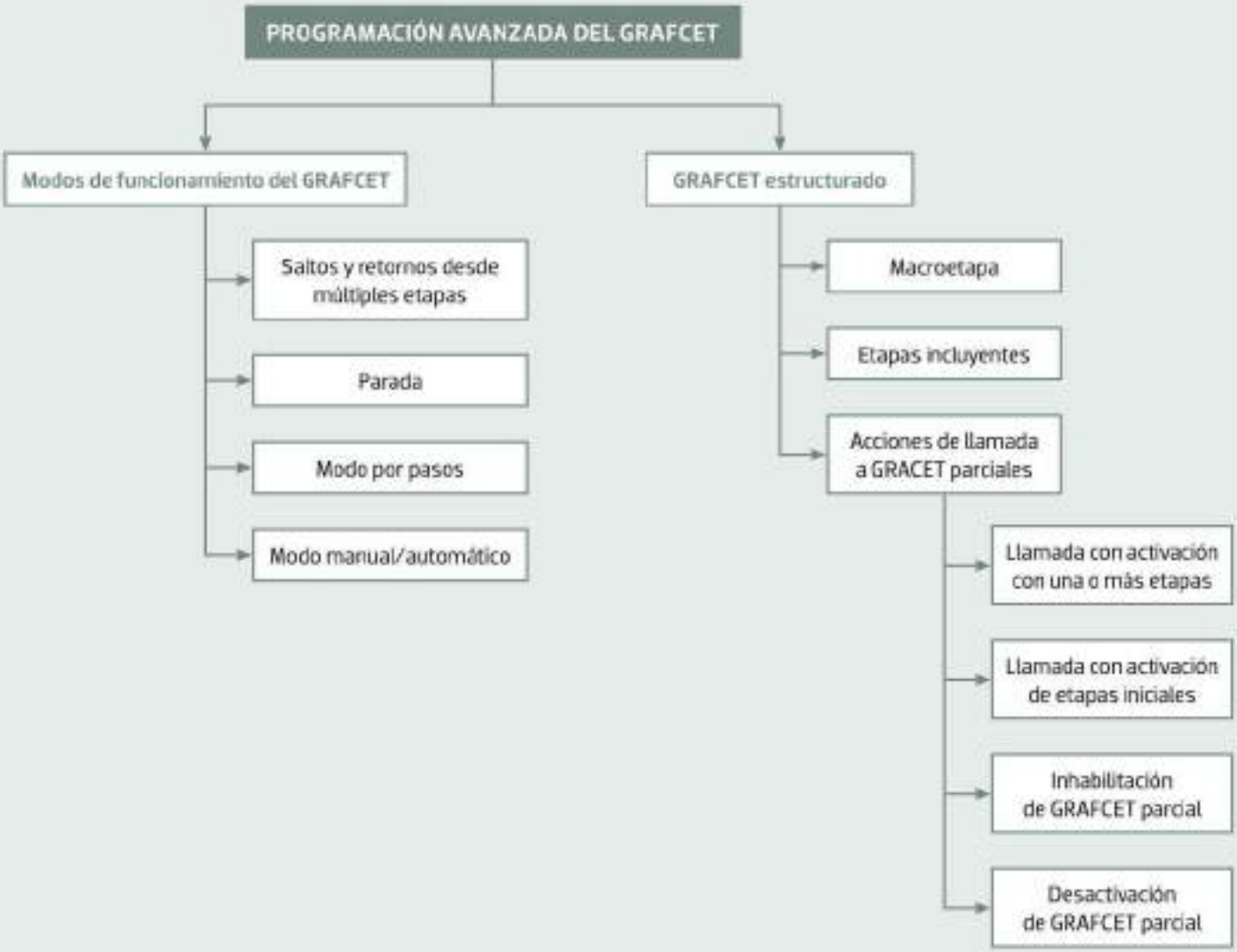


Figura 8.45. Sistema de selección de piezas por alturas.

Figura 8.46. Un selector de piezas por alturas se puede utilizar, por ejemplo, para clasificar paquetes según su altura.

EN RESUMEN


PROGRAMACIÓN AVANZADA DEL GRAFCET
Modos de funcionamiento del GRAFCET
Saltos y retornos desde múltiples etapas
Parada
Modo por pasos
Modo manual/automático
GRAFCET estructurado
Macroetapa
Etapas incluyentes
Acciones de llamada a GRACET parciales
Llamada con activación con una o más etapas
Llamada con activación de etapas iniciales
Inhabilitación de GRAFCET parcial
Desactivación de GRAFCET parcial

9 Tratamiento de datos y señales analógicas en STEP 7

Vamos a conocer...

1. Tipos de datos en el lenguaje STEP 7
2. Tipos de datos simples
3. Notación de números en STEP 7
4. Operación de transferencia (MOVE)
5. Operaciones matemáticas básicas
6. Tratamiento de señales analógicas

PRÁCTICA PROFESIONAL RESUELTA

Transferencia de datos escalados a canal analógico de salida

PRÁCTICAS PROFESIONALES PROPUESTAS

1. Incremento y decremento progresivo de una señal analógica de salida
2. Control de un semáforo mediante la transferencia de números en binario

Y al finalizar esta unidad...

- Identificarás los distintos tipos de datos y la notación numérica que tiene el lenguaje de programación STEP 7.
- Sabrás cómo se organizan los tipos de datos simples.
- Diseñarás programas para la transferencia de datos a variables con la instrucción MOVE.
- Usarás instrucciones de transferencia en las acciones de un GRAFCET.
- Programarás operaciones matemáticas para cambiar el valor numérico de variables.
- Conocerás los diferentes tipos de señales analógicas que pueden tratarse en un PLC.
- Ejecutarás programas para leer valores de entradas analógicas.
- Enviarás datos a canales analógicos de salida.

1. Tipos de datos en el lenguaje STEP 7

En anteriores unidades se ha trabajado, principalmente, con datos en formato booleano, ya que la mayoría de los ejemplos requerían conocer valores de señales entre 0 o 1 (verdadero o falso) y, después de operar, obtener resultados similares.

Sin embargo, cuando se han usado temporizadores y contadores, ha sido necesario conocer el dato de tiempo o de cómputo que en ellos se estaba procesando para, posteriormente, evaluarlo y utilizarlo en operaciones de comparación o de rango.

Los procesos industriales, además de controlar valores booleanos, requieren tratar otros tipos de datos (numéricos, de texto, etc.) que faciliten realizar operaciones matemáticas, tratar señales analógicas o comparar valores numéricos, de forma que se puedan procesar magnitudes físicas en su entorno como velocidad, temperatura, presión, etc.

STEP 7, de igual forma que otros lenguajes de programación, emplea diferentes tipos de datos que se pueden clasificar en:

- Tipos de datos simples: utilizan variables con un tamaño predefinido para almacenar números, cadenas de caracteres y valores booleanos.
- Tipo de datos compuestos: almacenan datos formando estructuras de variables.

El estudio de los datos compuestos se sale de los objetivos de este libro, por lo que aquí solamente se trabajará, de forma básica, con los tipos de datos simples y su aplicación a los sistemas secuenciales programables.

Recuerda

STEP 7 es el lenguaje de programación utilizado por los PLC de Siemens. TIA PORTAL es el entorno de programación en el que, entre otras cosas, se pueden programar los PLC en lenguaje STEP 7.

2. Tipos de datos simples

Permiten almacenar datos numéricos (enteros y reales), caracteres y valores booleanos.

Los tipos de datos simples tienen un tamaño en bits que permite determinar el rango del valor que pueden almacenar. Las variables para los tipos de datos simples se deben definir con un nombre simbólico y la zona de memoria del PLC en la que direccionan.

El tamaño se determina mediante la potencia $2^{n\text{-bits}}$. En aquellos tipos de datos que lo admiten, la mitad del rango es positivo y la otra mitad negativo, teniendo en cuenta que el signo emplea un bit.

A continuación se describen algunos de los tipos de datos simples más característicos de STEP 7 y que son comunes en todas las series de autómatas programables de Siemens (S7-300, S7-400, S7-1200 y S7-1500).

Saber más

Los S7-1200 y S7-1500 tienen tipos de datos que no están presentes en los S7-300 y S7-400. Su descripción y uso se puede consultar en el sistema de ayuda de TIA PORTAL.

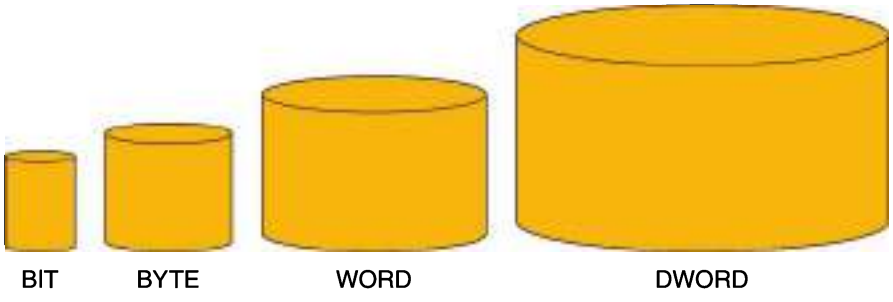


Figura 9.1. Representación gráfica no proporcional del tamaño de algunos de los principales tipos de datos.

En la siguiente tabla se listan todos los tipos de datos disponibles en STEP7 con sus características más importantes:

Tipos de datos				
Tipo de dato	Tamaño	Descripción	Rango	Ejemplos de direccionamiento
BOOL	1 bit	Valor booleano	1-0	I0.0 Q0.1 M10.4
BYTE	8 bits	Byte sin signo	0-255	IB0 QB1 MB12
WORD	16 bits	Entero sin signo	0 a 65 535	IW0 QW1 MW12
INT	16 bits	Entero con signo	-32 768 a +32 767	IW0 QW1 MW12
DWORD	32 bits	Doble entero sin signo	0 a 4 294 967 295	ID0 QD1 MD12
DINT	32 bits	Doble entero con signo	-2 147 483 648 hasta +2 147 483 647	ID0 QD1 MD12
REAL	32 bits	Valor de 32 bits en coma flotante	1,175 495e-38 hasta +3,402 823e+38	MD12
SSTIME	16 bits	Tiempo SIMATIC	SST=0H_0M_0S_10MS hasta SST=2H_46M_30S_0MS	MD10
TIME	32 bits	Tiempo IEC	-T=24D_20H_31M_23S_648MS hasta T=24D_20H_31M_23S_647MS	MD15
CHAR	8 bits	Carácter	A, B, C, etc.	MB20

Actividades

1. Determina cuál es el formato más adecuado para los siguientes datos numéricos. Ten en cuenta que el subíndice que aparece junto a ellos define el sistema de numeración en el que está escrito. Es decir: (2 es binario, (16 es hexadecimal, etc.

a) $-1110_{(2}$
b) $1100 / 10_{(10}$
c) $-1 / 11_{(2}$
d) 256
e) 4,5
f) $BA_{(16}$
g) $-A_{(16}$
h) $F_{(16}$
i) 64 200
j) -255

2.1. Organización de los tipos de datos simples

La estructuración en binario de los tipos de datos según su tamaño se realiza según se muestra en la figura:

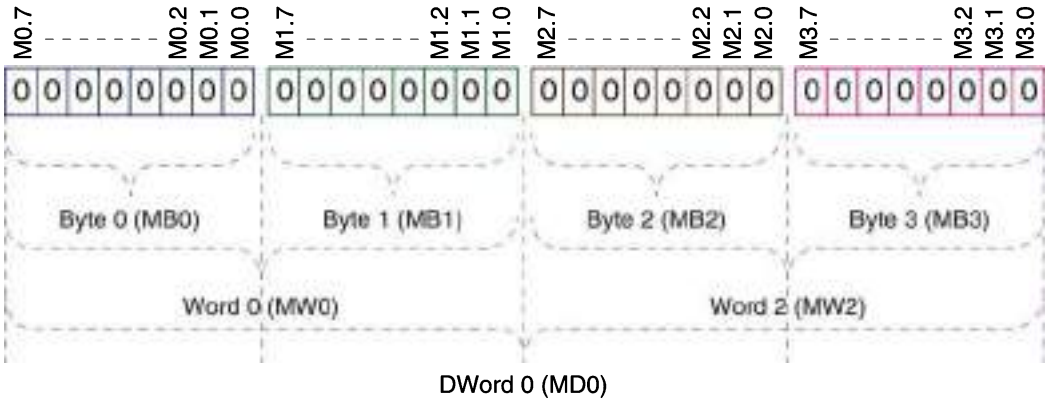


Figura 9.2. Formato de tipos de datos simples tomando como ejemplo la doble palabra MD0.

El direccionamiento absoluto a las zonas de memoria del PLC de las variables con formato superior a un bit se hace siempre a un *byte* de referencia. Así, el dato ocupa un número de *bytes* consecutivos dependiendo de su tamaño: 2 *bytes* para formato *word,* 4 *bytes* para el formato *double word,* etc.

A continuación se muestran algunos ejemplos de direccionamiento a zonas de marcas.

MB10	Direccionamiento que ocupa un solo byte.
MW12	Direccionamiento en formato de palabra (word) que ocupa dos bytes, el 12 y el 13.
MD26	Direccionamiento en formato de doble palabra (double word), que ocupa 4 bytes consecutivos, el 26, 27, 28 y 29.

Cuando se direccionan datos que se encuentran próximos entre sí hay que tener la precaución de que no se solapen entre ellos.

Ejemplo

¿Qué característica exige un programa que necesita almacenar dos datos en formato word? ¿Y si son en formato double word?

Para que un programa almacene dos datos en formato word podríamos, por ejemplo, direccionar los datos a MW2 y a MW4, respectivamente. En este caso, los datos no están solapados, ya que MW2 utiliza los bytes 2 y 3, y MW4 los bytes 4 y 5.

Sin embargo, si en el mismo programa se utilizasen los direccionamientos MW2 y MW3 para almacenar los mismos datos, se estaría produciendo un solapamiento entre ellos, ya que MW2 ocupa los bytes 2 y 3, y MW3 lo hace con los bytes 3 y 4. Es decir, el byte 3 es común en ambos y, por lo tanto, está solapado. Los datos de MW2 y MW3 estarían mal direccionados.

De igual forma, si el tamaño de los datos se hace en double word (doble palabra), hay que tener en cuenta que cada variable ocupa 4 bytes consecutivos. Así, si dos datos se direccionan a las dobles palabras MD30 y MD32, ambas estarían mal direccionadas, ya que los bytes 32 y 33 se estarían solapando.

Evitar solapamientos

Cuando se crean las variables, TIA PORTAL organiza el direccionamiento para evitar que los datos se solapen. No obstante, el usuario debe tener la precaución de que esto no ocurra, salvo que dicho solapamiento sea deseado.

3. Notación de números en STEP 7

En STEP 7 los números se pueden escribir de diferentes formas, según el sistema de numeración en el que se desee trabajar. Si bien la mayoría de las funciones de programación aceptan el sistema de numeración decimal, algunos bloques, especialmente en las series S7-300 y S7-400, requerían otro tipo de notación.

Los números con una notación diferente a la decimal deben estar precedidos de identificador del sistema de numeración, seguido del símbolo #, que actúa como elemento separador.

- Binario: 2#
- Octal: 8#
- Hexadecimal: 16#
- BCD: C#

A continuación se muestran las diferentes formas de escribir datos con distinta notación numérica:

Decimal	Binario	Hexadecimal	Octal	BCD
100	2#10011100	16#F0	8#24	C#23
32 000	2#00101010	16#1329	8#123	C#718
-356	2#101010	16#F0F0	8#77	C#45

Notación en desuso

Una forma en desuso para la notación hexadecimal, pero aceptada en TIA PORTAL, es la siguiente: W#16#F0, donde existen dos elementos de separación #, uno para indicar que el número está en formato de palabra (W) y el otro para indicar que el sistema de numeración es el hexadecimal (16).

4. Operación de transferencia (MOVE)

La operación de transferencia MOVE se usa para enviar datos a diferentes zonas de memoria del PLC.

En el lenguaje de contactos, esta instrucción se representa en forma de caja con una serie de entradas y salidas.

En el lado izquierdo dispone de las entradas EN e IN. La primera se usa para habilitar el bloque y la segunda para escribir el dato que se va a transferir.

En el lado derecho tiene la salida ENO, para habilitar otros bloques en cascada siempre que no se produzca un error en el procesamiento de la operación, y la salida OUT, donde se indica la zona de memoria o variable que va a recibir el dato.

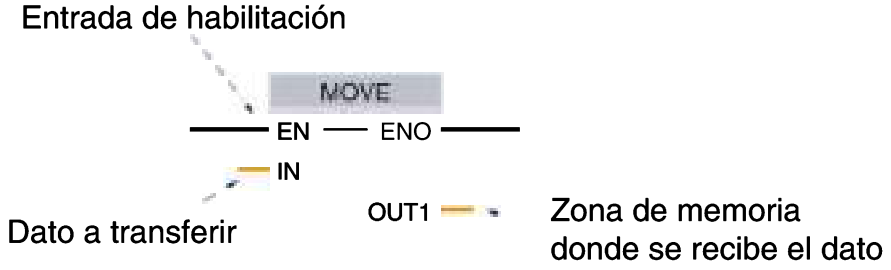


Figura 9.3. Operación de transferencia (MOVE).

La operación MOVE se puede utilizar para transferir datos entre variables que no tienen el mismo formato. No obstante, es necesario entender que si esto se realiza, se puede perder o modificar parte de la información a transferir.

Ejemplo

A continuación se muestran varios ejemplos del uso de la función de transferencia MOVE en STEP 7:

Transferencia incondicional de un número en decimal a una variable en formato de byte:

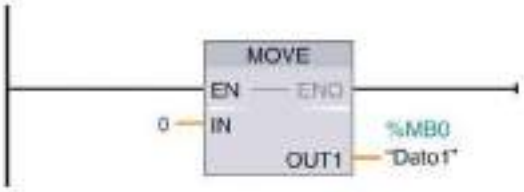


Figura 9.4. Operación de transferencia incondicional de decimal a byte.

Transferencia incondicional de una variable a otra:

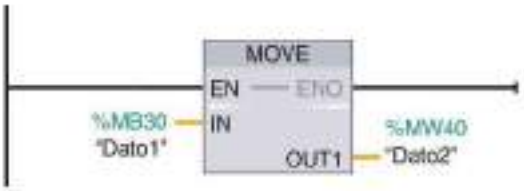


Figura 9.5. Operación de transferencia de una variable a otra.

Transferencia condicionada a la activación de una entrada, de un número en binario al *byte* 0 de salidas:

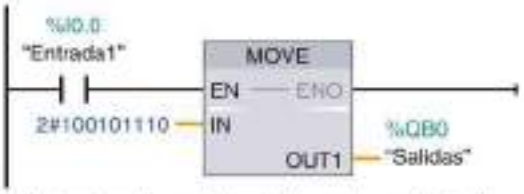


Figura 9.6. Operación de transferencia condicionada de un número al *byte* 0.

Transferencia condicionada a la activación de una entrada, de un número en decimal a una variable en formato word:

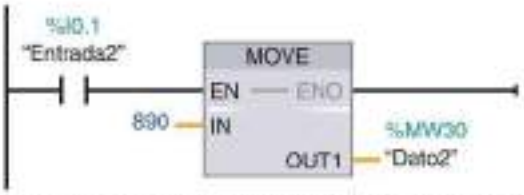


Figura 9.7. Operación de transferencia condicionada de un número a una palabra.

Transferencia condicionada a la activación de una entrada, de un número en hexadecimal al byte 2 de salidas:

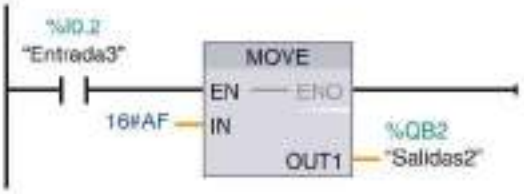


Figura 9.8. Operación de transferencia condicionada de hexadecimal al byte 2.

Operaciones de transferencia

La asignación de valores mediante las operaciones de transferencia no requiere mantener activa la entrada EN del bloque MOVE. La asignación se mantiene aunque desaparezca la señal que la ejecutó.

Actividades

2. Desarrolla el siguiente programa en TIA PORTAL y comprueba su funcionamiento en un PLC. El programa se ha realizado para un S7-1200, en el que el primer byte de salidas es el QB0. No obstante, este ejemplo se puede extrapolar a cualquier otro modelo de PLC, con otro direccionamiento.

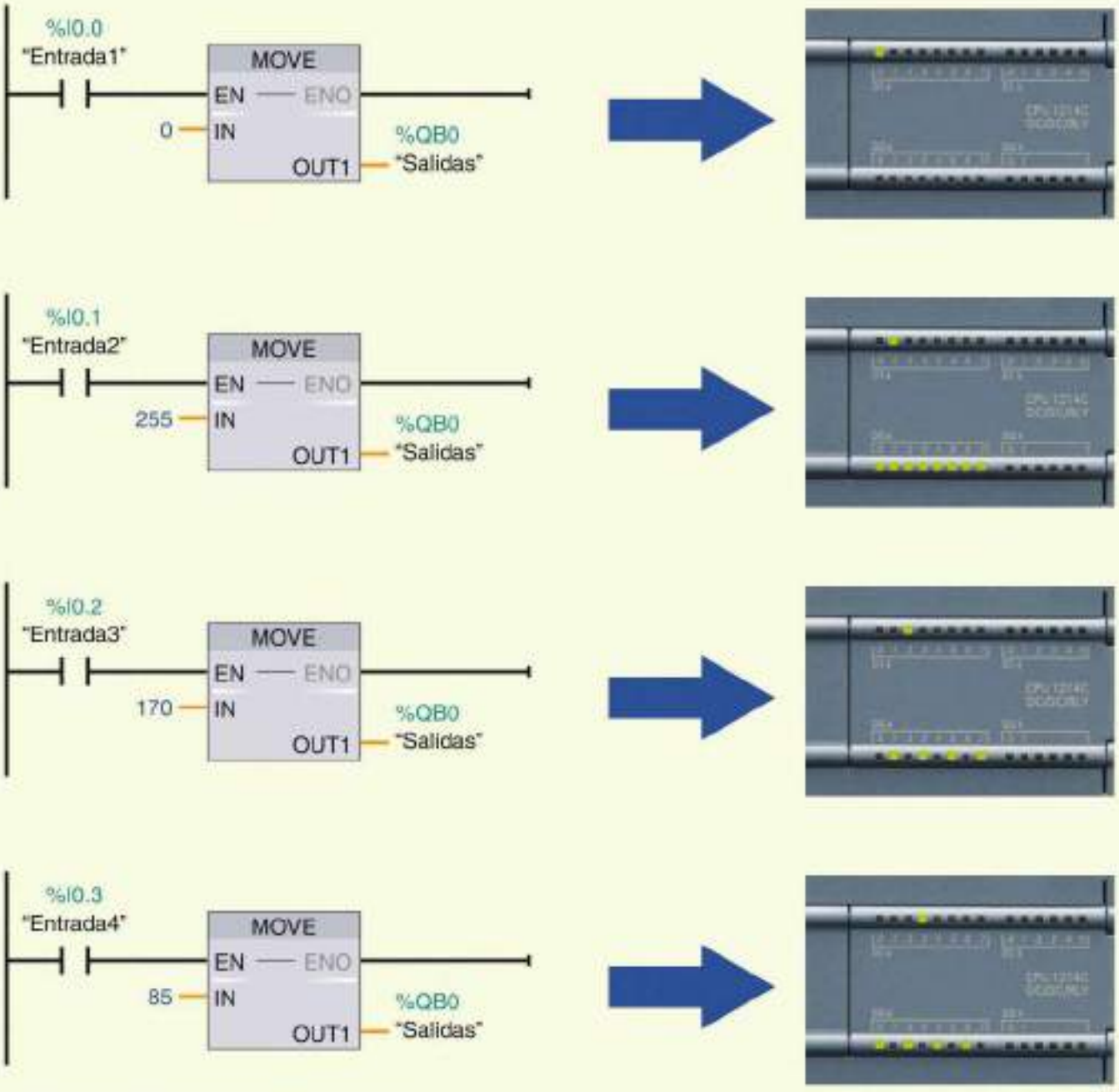


Figura 9.9. Programación de operaciones de transferencia

Importante

Los números en decimal enviados con la instrucción MOVE son convertidos a binario para activar los bits individuales de byte de salidas. Observa que, debido a la disposición de los led de salida en el PLC, la codificación en binario está representada al revés (en espejo) de como se escribiría el número. En este caso, el bit de menor peso es la salida Q0.0, la cual se encuentra físicamente ubicada a la izquierda del byte, cuando dicho bit, en su representación escrita, siempre debe colocarse a la derecha.

4.1. Transferencia de datos con un formato determinado

La instrucción MOVE permite transferir datos con un formato determinado, como puede ser un tiempo de un temporizador. En este caso, la variable que recibe el dato tiene que tener el tamaño y formato correcto para recibirlo sin causar error.

A continuación se muestra un sencillo ejemplo para configurar de forma dinámica el tiempo de un temporizador IEC a la conexión.

La variable para intercambiar el tiempo debe ser de tipo TIME y tener un tamaño de doble palabra.

Nombre	Tipo de datos	Dirección
Tiempo dinámico	Time	%MD20

En esta ocasión, en la entrada PT del temporizador no se escribirá un valor fijo de tiempo, sino que se asignará a través de la variable creada a tal efecto, lo que permitirá cambiar el tiempo de forma dinámica.

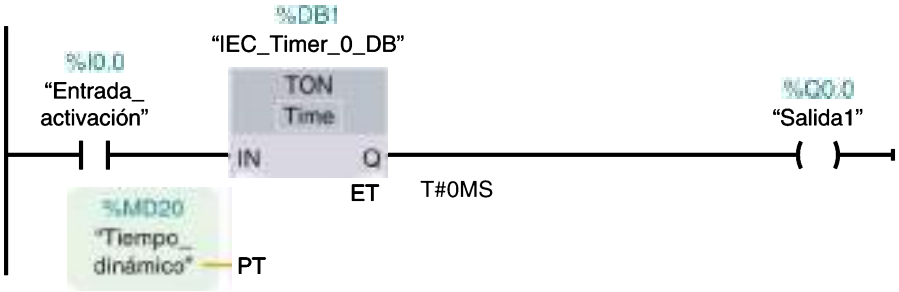


Figura 9.10. Tiempo dinámico en entrada PT de un temporizador.

Se crean tantos bloques MOVE como tiempos se deseen configurar en el temporizador. En este caso solamente se han creado dos, con 5 y 10 segundos respectivamente, que son asignados mediante contactos, uno cerrado y otro abierto, de una misma entrada, a través de instrucciones MOVE.

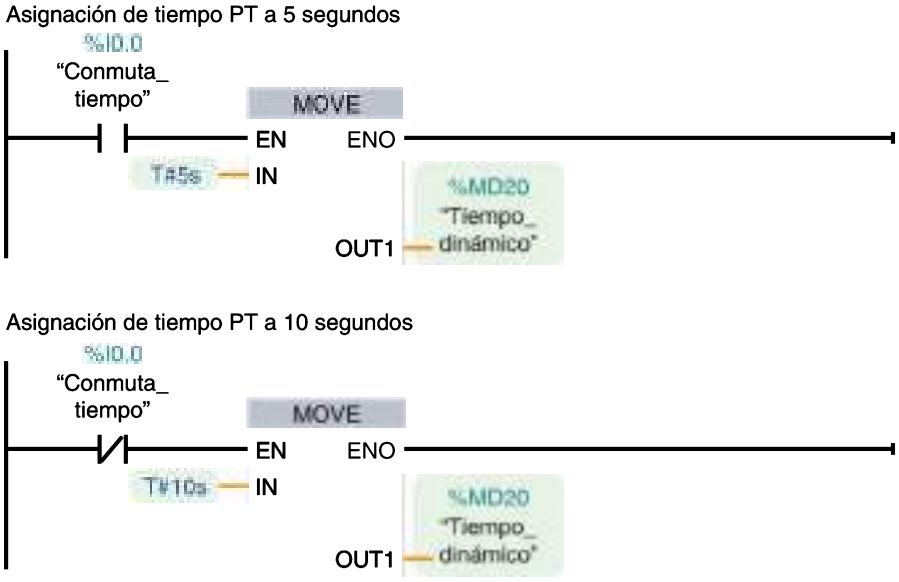


Figura 9.11. Asignar valores de tiempo a una variable.

Actividades

3. Programa y comprueba el ejemplo con el temporizador dinámico mostrado en esta página. Realiza los cambios que consideres oportunos para que se puedan asignar cuatro valores de tiempo diferentes.

4.2. Operaciones de transferencia en el GRAFCET

En un GRAFCET las operaciones de transferencia se representan en la zona de acciones y no es necesario realizar nada más en otras zonas de programación.

En este caso, cada vez que se realiza una asignación (:=) de un valor numérico a una variable, en la zona de acciones se debe representar la operación de transferencia MOVE condicionada con un contacto del bit de etapa en el que se realiza dicha acción.

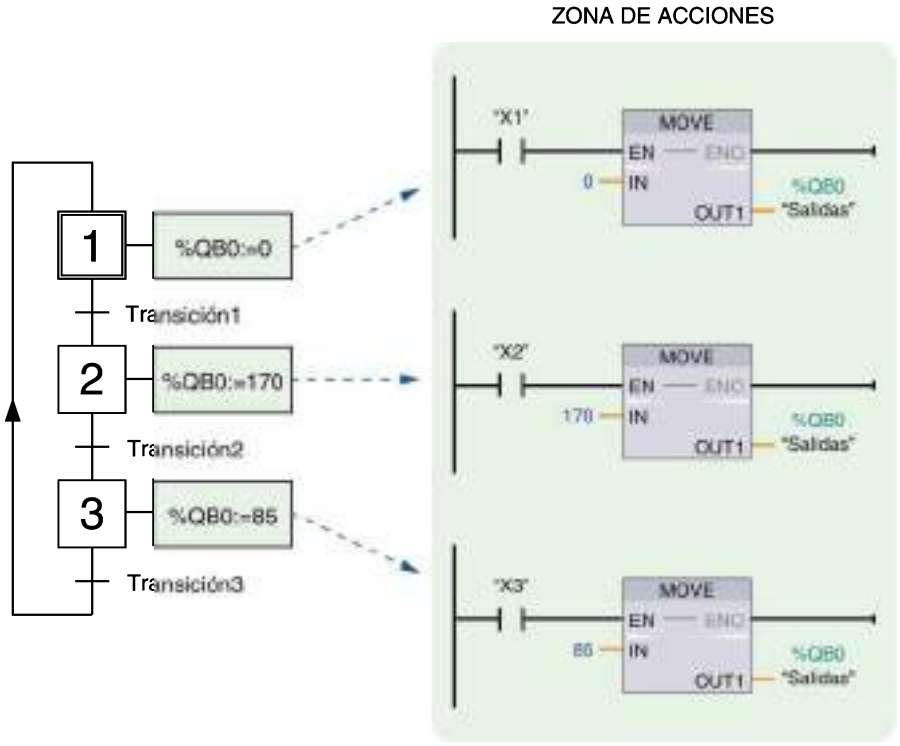


Figura 9.12. Acciones con operaciones de transferencia.

Actividades

4. Programa y comprueba el funcionamiento del GRAFCET de la figura con acciones con operaciones de transferencia.

Una vez realizada la acción con la instrucción MOVE el valor queda asignado a la variable, aunque la entrada EN del bloque se desactive. Eso significa que no se necesario mantener dicha entrada a 1 para la asignación del valor.

En ocasiones es aconsejable que la condición de activación del bloque se realice mediante operaciones de flanco, para evitar mantener de forma permanente la acción de transferencia.

En el caso del GRAFCET, la acción mediante flanco positivo permite realizar la transferencia cuando se entra en una etapa y con flanco negativo cuando se sale de ella.

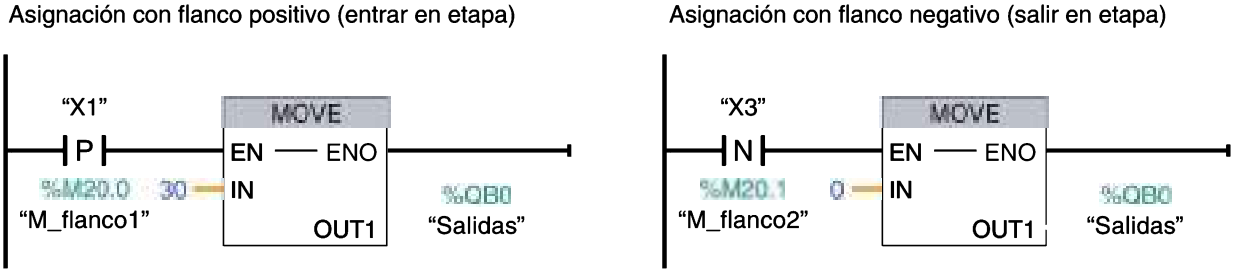


Figura 9.13. Asignaciones condicionadas a flancos.

5. Operaciones matemáticas básicas

Los autómatas programables tienen una alta potencia de cálculo. Con ellos se pueden realizar operaciones matemáticas empleando funciones de trigonometría, cálculo logarítmico, potenciación, etc.

Aquí solamente se describirán las operaciones aritméticas básicas: adición, sustracción, multiplicación y división, cuyo empleo permite dar una visión general de este tipo de instrucciones y así facilitar el uso de cualquiera de las otras.

En el lenguaje de contactos las operaciones aritméticas se representan en forma de caja.

- En la parte superior se indica el tipo de operación.
- En el lado izquierdo se encuentran las entradas (IN), en las que se escriben los números con los que hay que operar.
- En el lado derecho se encuentra la salida (OUT), donde se recibe el resultado de la operación.

Tipo de datos

Se pueden usar para operar con números enteros o números reales, por lo que hay que elegir adecuadamente el tipo de datos de las variables para realizar estas operaciones. Hay que ser especialmente cuidadoso con la operación división, ya que, al operar con dos números, que pueden ser enteros, el resultado puede ser un número real.

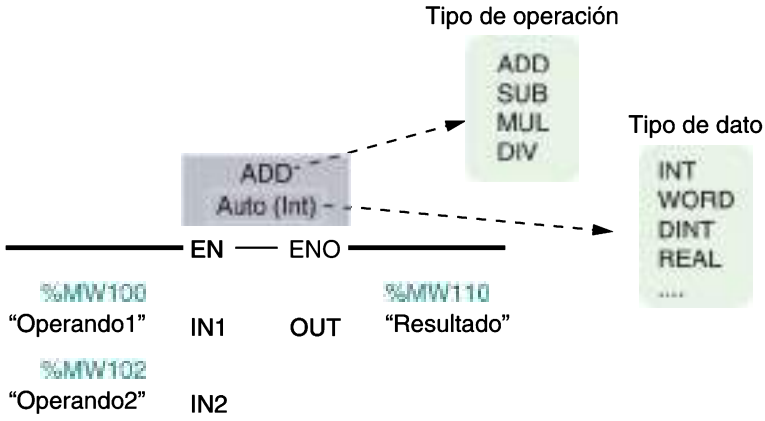


Figura 9.14. Bloque de operación matemática.

Si se mantiene a 1 la entrada EN de las operaciones matemáticas, estas se ejecutan en cada ciclo de SCAN del PLC. Esto significa que, en aquellos casos en los que la variable de entrada es la misma que la de salida, la operación se ejecuta de forma repetitiva y se puede perder el control sobre ella, si no es lo deseado.

El siguiente ejemplo muestra cómo el valor de una variable se incrementa con la operación suma (ADD). En el segmento de la izquierda, la operación se ejecuta cada vez que el contacto conectado a la entrada EN tiene valor 1. En este caso, la suma se realiza de forma repetitiva cada ciclo de SCAN, lo que hace que, con una simple activación de la entrada, el valor de la variable que se va a operar cambie de forma abrupta. Para evitar esto, la entrada del bloque debe dispararse por flanco.

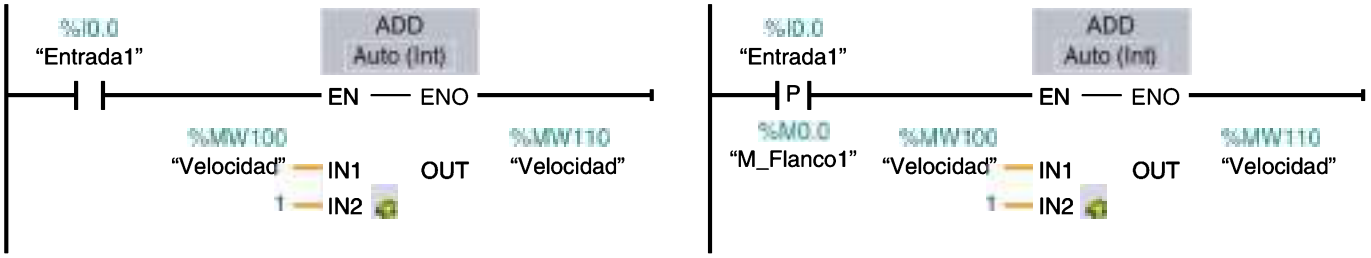


Figura 9.15. Ejecución de una operación de suma con activación con y sin flanco.

6. Tratamiento de señales analógicas

En la unidad 2 se estudiaron los diferentes tipos de señales analógicas que un PLC puede incorporar. A continuación se muestra cómo tratar los valores de este tipo de señales en el lenguaje de contactos, tanto de entrada como de salida.

6.1. ¿Qué se debe conocer para trabajar con las señales analógicas?

Antes de comenzar a realizar programas para el tratamiento de señales analógicas debemos conocer varios datos sobre ellas.

6.1.1. Tipo de señal

Para una correcta conexión del *hardware* es necesario conocer el tipo de señal admitido por el canal analógico con el que se va a trabajar.

Se debe saber si trabaja en tensión o en corriente, y si la señal es de tipo unipolar (solo valores positivos) o bipolar (con valores positivos y negativos).

Así, por ejemplo, algunas de las señales analógicas más empleadas son:

1. En tensión:
   - De 0 a 10 V (unipolar).
   - De –10 a 10 V (bipolar).
2. En corriente:
   - De 0 a 20 mA (unipolar).
   - De 4 a 20 mA (unipolar).

En algunos dispositivos, las señales analógicas tienen valores fijos que no se pueden cambiar. Sin embargo, en los equipos más avanzados se puede elegir el tipo de señal que se va a utilizar, pudiéndose seleccionar vía *hardware* o vía *software*.

6.1.2. El direccionamiento físico al canal analógico

Es la zona de memoria, o de periferia, a donde hay que «apuntar» para trabajar con la señal analógica.

El dato ocupa dos *bytes* (16 bits) y en el operando debe aparecer el número del primero de ellos.

Se puede hacer de dos formas:

- **Direccionamiento a la periferia,** donde el operando debe estar precedido por la letra P.

  Así, para leer entradas analógicas la sintaxis es PIWxxx y para escribir en las salidas analógicas, PQWxxx, donde xxx es el número del canal analógico direccionado en el *hardware*. Esta forma de direccionamiento es habitual en las series S7-300 y S7-400.

  Si en TIA PORTAL se escribe el direccionamiento por periferia, automáticamente se convierte en %IWxxx:P.

- **Direccionamiento a zonas de memoria,** que consiste en apuntar directamente al *byte* de referencia de la señal, indicando el tamaño de la misma, que en este caso es de una palabra (W).

  Así, para direccionar a señales analógicas de entrada la sintaxis es %IWxxx, y a las de salida, %QWxxx. Esta es la forma de direccionar en los modelos S7-1200.

Ejemplo

El S7-1200 de la figura dispone de dos entradas y tres salidas analógicas. De estas últimas, dos están integradas en el mismo módulo que la CPU y la tercera se encuentra en una tarjeta Signal Board que el PLC tiene instalada.

El direccionamiento de estas señales es el siguiente:

Entradas analógicas	Salidas analógicas	
IW64	QW64	Integradas
IW66	QW66	
——	QW80	Módulo Signal Board

En TIA PORTAL el direccionamiento se muestra en la configuración del dispositivo:

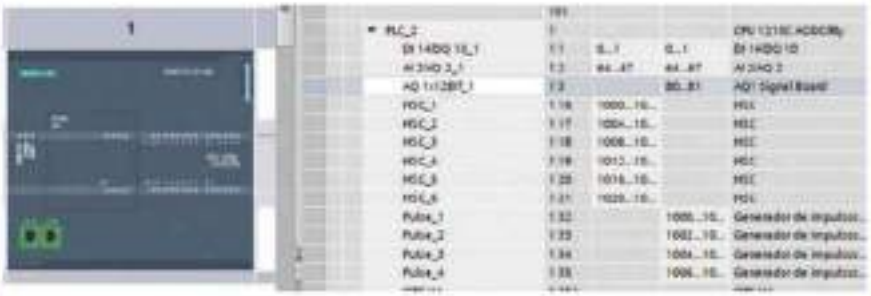


Figura 9.16. Entradas y salidas analógicas en un S7-1200.

De igual forma, si tenemos un dispositivo S7-300 como el de la figura, se puede comprobar que tiene cinco entradas y dos salidas analógicas, todas ellas integradas, ya que es una CPU compacta y no dispone de módulos de ampliación.

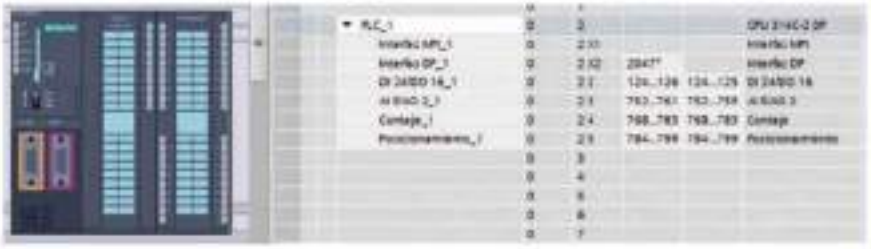


Figura 9.17. Entradas y salidas analógicas en un S7-300.

El direccionamiento en este dispositivo es el siguiente:

Entradas analógicas	Salidas analógicas	
PIW752	PQW52	Integradas
PIW754	PQW54	
PIW756	——	
PIW758	——	
PIW760	——	

Recuerda que el direccionamiento también puede establecerse según la nemotecnia SIMATIC. En ese caso, PEW será para las entradas y PAW para las salidas.

6.1.3. Valor del dato analógico

Los valores analógicos se almacenan en zonas de memoria de 16 bits *(word)*, por lo que su valor de procesamiento puede ser de 216 = 65 536.

Si la señal analógica es de tipo bipolar, el valor se divide en números positivos y números negativos. Es decir, de –32 768 a 32 768.

Si la señal es de tipo unipolar solamente se trabaja con números positivos, de 0 a 32 768.

De este valor, el rango nominal de medición de la señal analógica se hace entre 0 y 27 648 en las señales unipolares, y de –27 648 a 27 648 en las señales bipolares.

Los valores que superen estos rangos, tanto en positivo como en negativo, hasta 32 768, se utilizan para detectar el rebase y desbordamiento de la señal.

Vocabulary

- Desbordamiento: *overflow*.
- Rebase: *overshoot*.
- Dato analógico: *analog data*.
- Rango: *rank*.
- Direccionamiento: *addressing*.
- Corriente: *current*.
- Tensión: *voltage*.
- Suma: *addition*.
- Resta: *subtraction*.
- Multiplicación: *multiplication*.
- División: *division*.
- Partido: *split*.
- Periferia: *periphery*.
- Notación: *notation*.
- Datos compuestos: *composite data*.
- Visión general: *overview*.
- Escalado: *scaled*.

Ejemplo

Supóngase una señal analógica en tensión de tipo bipolar entre –10 y +10 V. Determina el rango numérico y nominal para la señal.

El rango numérico correspondiente a dicha señal es:

Señal eléctrica	Valor de dato	
11,85 V	**+32 768**	**Rebase de la señal**
+10 V	+27 648	Valor nominal
0 V	0	
–10 V	–27 648	
–11,85 V	**–32 768**	**Rebase de la señal**

Es decir, el rango nominal para la señal de –10 a +10 V está comprendido entre –27 648 y +27 648. Si dichos valores se superan, tanto por arriba como por abajo, se puede considerar que la señal se ha rebasado y podría ser debido a algún tipo de defecto.

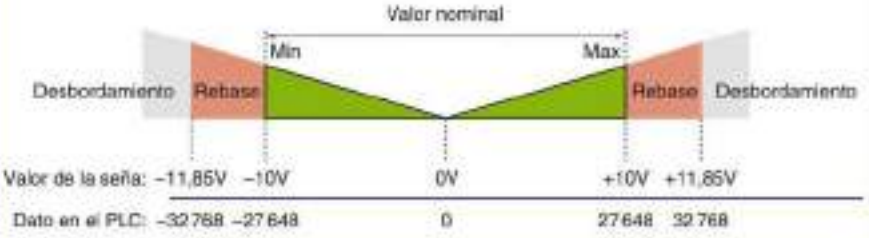


Figura 9.18. Representación gráfica de la señal.

Este planteamiento puede ser usado para cualquier tipo de señal analógica, tanto en tensión como en corriente. En el caso de las señales unipolares, solamente se usarán los valores positivos.

Actividades

5. En una señal analógica de 4 a 20 mA, ¿a qué valores de corriente corresponden a los valores del dato en el PLC de 5 000, 10 000, 20 530, 27 000 y 32 000?

6.2. Lectura de señales analógicas de entrada

La lectura de señales analógicas de entrada se hace direccionando al canal correspondiente, con la sintaxis %IWxxx, donde xxx es el primer *byte* del canal.

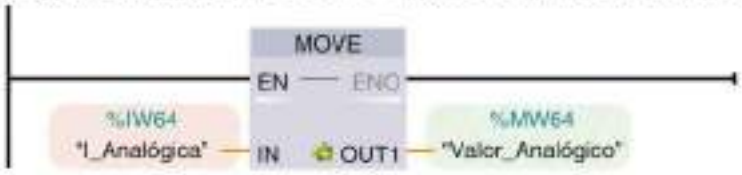
Es recomendable transferir el valor de una entrada analógica a una variable de tipo INT del PLC y utilizar esta para realizar las operaciones correspondientes.

Figura 9.19. Transferencia del valor de una entrada analógica a una variable.

Ejemplo

El siguiente ejemplo muestra cómo el valor de la señal analógica de entrada %IW64 se transfiere con la instrucción MOVE a la marca %MW64 y esta se utiliza en operaciones de comparación para activar salidas digitales en función del valor analógico alcanzado.

Segmento 1: transferencia de señal analógica de entrada a una variable



Segmento 2: activación de la salida digital 1 si el valor analógico supera 10 000

Segmento 3: activación de la salida digital 2 si el valor analógico supera 20 000

Segmento 4: activación de la salida digital 3 si el valor analógico supera 32 000

Figura 9.20. Ejemplo de uso de una señal analógica de entrada.

Actividades

6. Programa y comprueba el funcionamiento del programa del ejemplo para activar salidas digitales en función del valor de una entrada analógica.

6.3. Escritura de señales analógicas de salida

Para escribir un dato en una salida analógica basta con transferir un valor numérico comprendido entre 0 y 27 648.

De igual forma que con las entradas, es aconsejable asociar una variable en formato INT con el canal de la salida analógica. Esta variable se usa para recibir los datos de las operaciones previas a su transferencia al canal de salida.

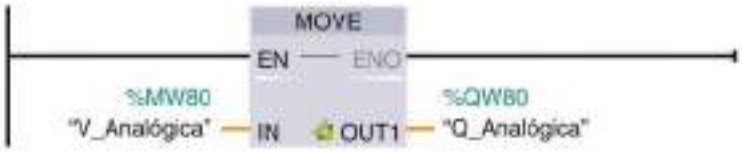
El direccionamiento a una salida analógica se hace mediante la sintaxis: %QWxxx, donde xxx es el *byte* del canal.

Figura 9.21. Transferencia a una salida analógica desde una variable.

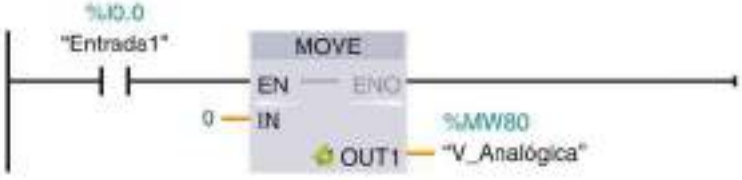
Ejemplo

En el siguiente ejemplo se muestra cómo mediante tres entradas digitales se transfieren tres valores fijos a la salida analógica %QW80.

Segmento 1: transferencia del valor de una variable a la salida analógica



Segmento 2: transferencia del valor 0 a la variable



Segmento 3: transferencia del valor 16 000 a la variable

Segmento 4: transferencia del valor 32 000 a la variable

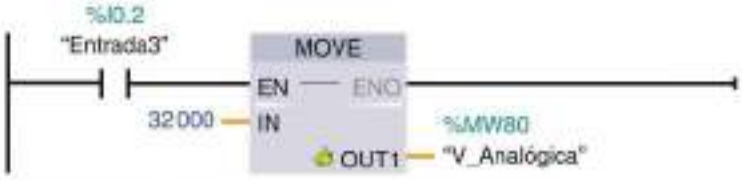


Figura 9.22. Ejemplo de transferencia de valores a una salida analógica.

Actividades

7. Programa y comprueba el funcionamiento del programa del ejemplo para enviar datos a una salida analógica. Si estás utilizando la salida analógica del S7-1200 en tensión puedes conectar un polímetro en dicha salida y comprobar cómo varía el valor de la señal según el número transferido.

6.4. Normalización y escalado de señales analógicas

Trabajar con números dentro del rango entre 0 y 27648 puede resultar confuso si lo que se desea es relacionarlos proporcionalmente con valores de señales físicas conocidas, como velocidad, presión, temperatura, longitud, etc. Por ejemplo, no es lo mismo hacer referencia a una temperatura en un rango numérico abstracto entre 0 y 27648 que asignar dicho valor en grados centígrados entre 0 y 100 °C.

STEP 7 dispone de instrucciones de programación que facilitan la tarea de relacionar valores de magnitudes conocidas con el rango numérico de la señal.

NORM_X: normaliza el valor recibido por una variable en la entrada VALUE entre un máximo y un mínimo. El valor procesado se transfiere a una variable de salida en formato de número en coma flotante (REAL).

SCALE_X: escala el valor de una variable con formato de número en coma flotante, como puede ser el depositado en la función NORM_X, para ajustarlo entre un rango de valores comprendido entre un mínimo y un máximo, cuyo valor procesado se transfiera a una variable en formato INT.

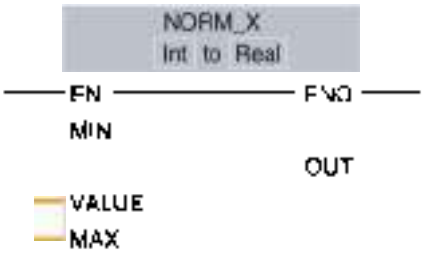


Figura 9.23. Función de normalización (NORM_X).

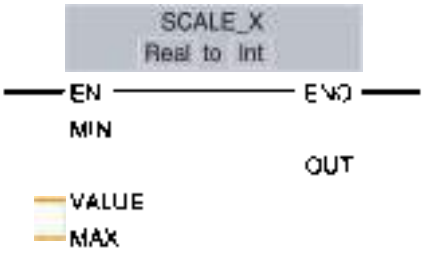


Figura 9.24. Función de escalado (SCALE_X).

6.4.1. Normalizado y escalado de una entrada analógica

El siguiente ejemplo muestra cómo el valor de una entrada analógica (IW64) se normaliza (0...27648) y se escala para que la variable de salida MW64 reciba valores entre 0 y 10, que podrían ser los correspondientes a los valores en tensión (0...10 V) de la señal analógica.

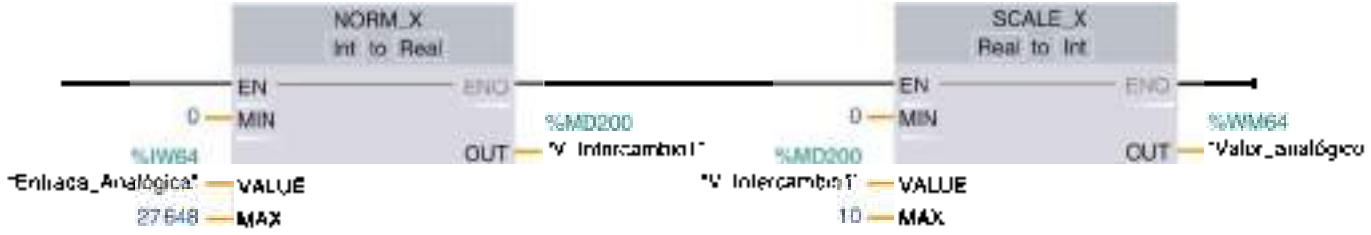


Figura 9.25. Normalizado y escalado de una entrada analógica.

Así, si la entrada analógica procesa la mitad de la señal (13824), significa que en la variable de salida se recibe el valor 5, que corresponde a los voltios que en ese momento tiene la entrada analógica.

6.4.2. Normalizado y escalado de una salida analógica

En este caso, una variable (MW80) recibe valores dentro del rango 0-10, que podrían ser los valores de tensión de la señal analógica de salida, y los escala entre 0 y 27648 para transferírselos a dicha salida (QW80).

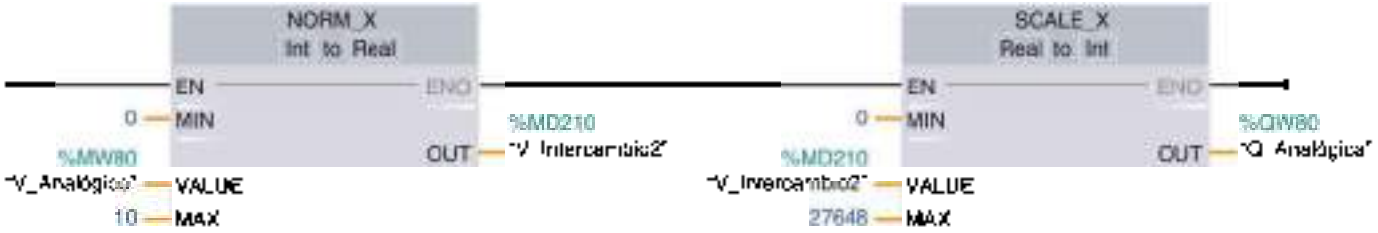


Figura 9.26. Normalizado y escalado de una salida analógica.

Así, si la variable MW80 recibe el número 5, significa que la salida analógica saca un valor en tensión de 5 V.

Números en coma flotante

La variable para intercambiar valores entre ambos bloques debe estar definida para almacenar números en coma flotante (REAL).

PRÁCTICA PROFESIONAL RESUELTA

Herramientas

- PC con TIA PORTAL y PLCSIM
- Un polímetro

Material

- Un PLC compatible con TIA PORTAL

Transferencia de datos escalados a canal analógico de salida

Objetivo

- Enviar datos a la salida analógica de un PLC.
- Utilizar la operación de transferencia.
- Normalizar y escalar señales.

Precauciones

- Comprueba que el PLC tiene configurada la salida analógica en tensión.
- Localiza dónde se encuentran las salidas analógicas en el autómata programable. Pueden estar integradas en el mismo módulo que la CPU o en módulos de expansión.
- Recuerda que los datos analógicos deben ser tratados con formato de palabra (*word*).

Desarrollo

1. Observar en el *hardware* del dispositivo donde se encuentra direccionada la salida analógica. En el caso del equipo de la figura, solamente dispone de una salida analógica, en un módulo Signal Board que se ha colocado en la CPU. El direccionamiento se hace en %QW80.

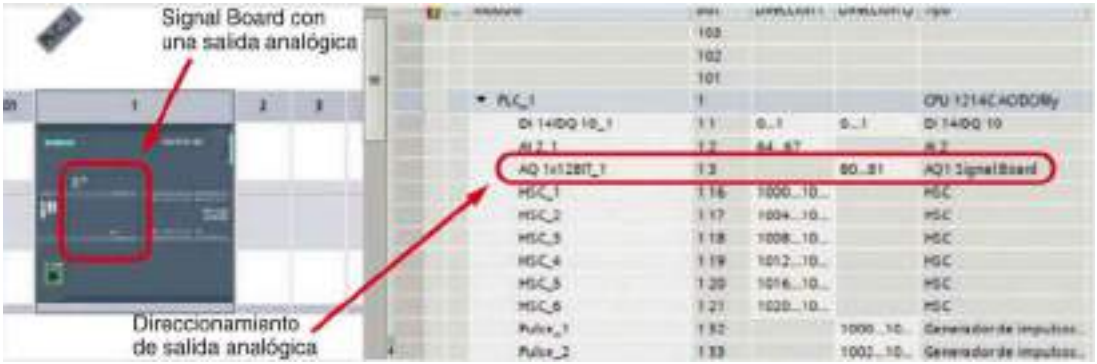


Figura 9.27. Direccionamiento de salida analógica.

2. Entrar en la configuración del canal analógico y comprobar que el Tipo de salida está en Tensión.

   El módulo del equipo mostrado en la figura permite dos configuraciones para la salida analógica:

   - En tensión de –/+10 V.
   - En corriente de 0 a 4 mA.

3. Crear un bloque FC en el que se escribirá todo el programa. No hay que olvidar que este bloque debe ser llamado desde el OB1.

Figura 9.28. Configuración del canal de salida analógica.

4. En el primer segmento, añadir las siguientes instrucciones para realizar la normalización y escalado de la señal. Se desea normalizar los valores de tensión –10 a 10 V, por lo que el escalado hay que hacerlo entre –27 648 y +27 648.

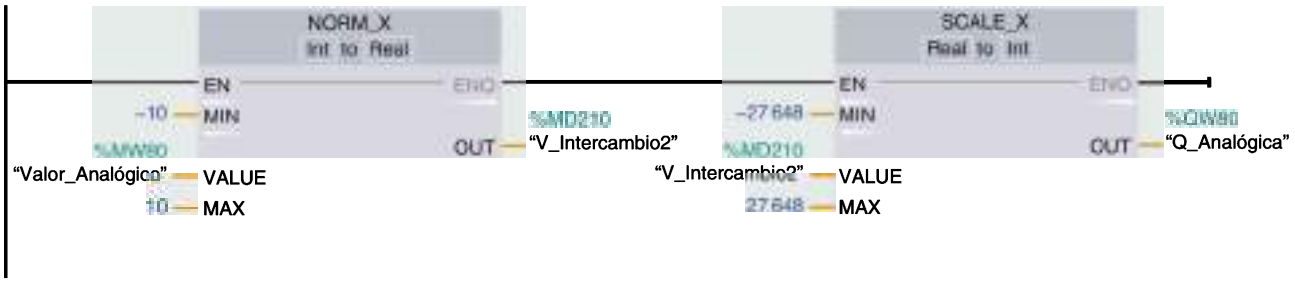


Figura 9.29. Normalización y escalado.

5. Añadir seis segmentos para enviar valores de tensión a la salida analógica. Cada operación MOVE debe estar precedida de un contacto asociado a una entrada digital, de forma que cada vez que se accione una de ellas se envíe un valor en tensión a la variable usada para realizar la operación de normalización.

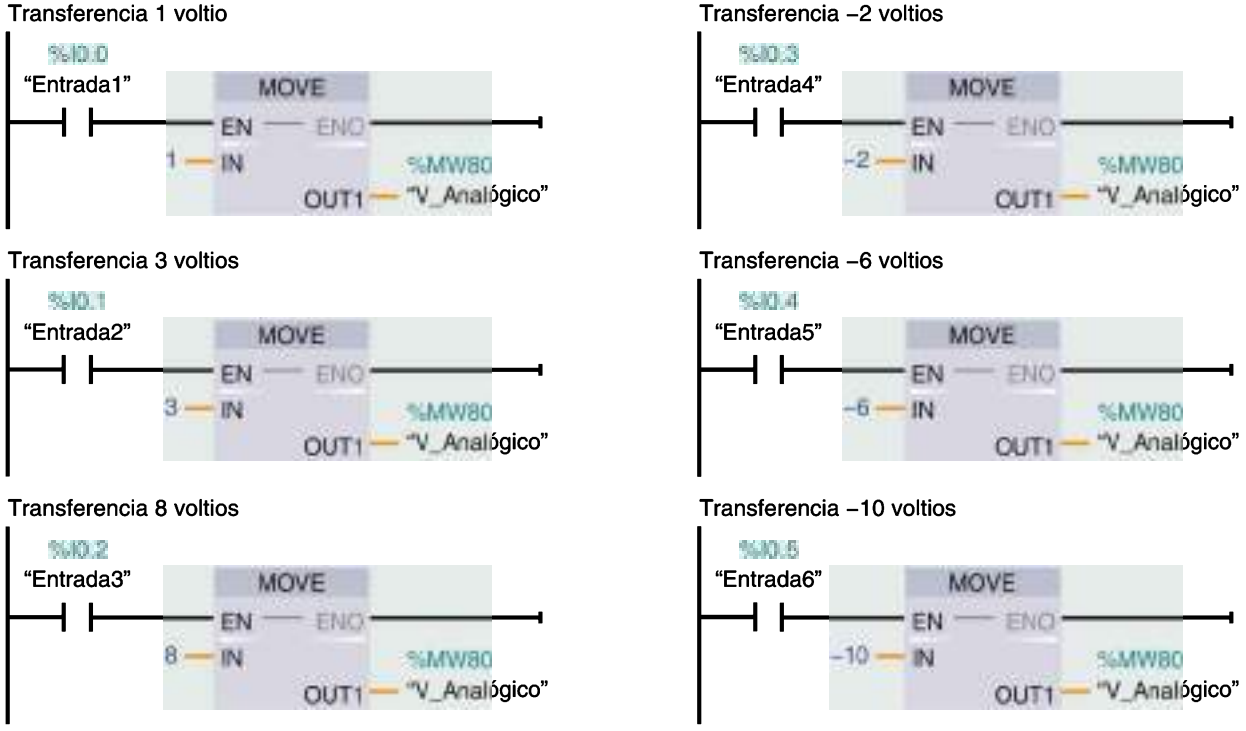


Figura 9.30. Transferencia de datos de tensión a variable de intercambio analógico.

6. Conectar un polímetro en los dos bornes de la salida analógica.
7. Conmutar el polímetro para medir tensión en corriente continua en un rango que admita –10 y 10 V.
8. Cargar el programa en el PLC.
9. Accionar las entradas digitales y observar cómo el valor de tensión del polímetro coincide con los valores asignados mediante la instrucción de transferencia.
10. Hay que tener en cuenta que si se acciona más de una entrada, solamente tendrá efecto la que esté escrita más abajo en el bloque del programa.

TEST DE EVALUACIÓN

RESUELVE EN TU CUADERNO O BLOC DE NOTAS

1. **Un número que ocupa dos *bytes* es:**
   a) Una palabra (*word*).
   b) Dos palabras (*double word*).
   c) TIME.
   d) Un carácter.

2. **Una variable direccionada a MD24 ocupa los *bytes* de marcas:**
   a) 24 y 25.
   b) 21, 22, 23 y 24.
   c) 24, 25, 26 y 27.
   d) 24, 25 y 26.

3. **El número 16#FE es el mismo que:**
   a) 8#367.
   b) 2552.
   c) #1111 1101.
   d) #1111 1110.

4. **¿Cómo se denomina la operación de transferencia en SETP 7?**
   a) ADD.
   b) SUB.
   c) MUL.
   d) MOVE.

5. **Si se transfiere el número 177 a *bytes* de salidas, ¿cuáles se activarán?**
   a) Todas.
   b) Ninguna.
   c) Las impares.
   d) Las pares.

6. **¿Cuáles de estos son valores de señales analógicos estandarizados?**
   a) –5 a 10 V.
   b) 0-10 V.
   c) 2-4 mV.
   d) 0-4 mV.

7. **Una señal analógica de entrada se lee como una variable en formato:**
   a) *Byte*.
   b) *Word*.
   c) *Double word*.
   d) Real.

8. **¿Cuál es el rango para el dato de una señal analógica de salida de tipo unipolar?**
   a) –32 768 a 32 768.
   b) 0 a 32 768.
   c) –27 648 a 27 648.
   d) 0 a 27 648.

9. **La variable de salida de la instrucción NORM_X es de tipo:**
   a) *Byte*.
   b) *Word*.
   c) *Double word*.
   d) Real.

10. **Si se envía el valor 30 000 a un canal de salida analógica en tensión:**
    a) No ocurrirá nada.
    b) La señal estará rebasada.
    c) Sacará el valor máximo de la salida.
    d) Se obtendrá el valor mínimo de la salida.

ACTIVIDADES FINALES

1. Asocia los números de la columna de la izquierda con el tipo de datos que resulte más adecuado de la columna de la derecha.

Dato	Tipo de dato de la variable
a) $-1110_{(2}$	
b) $1100 / 10_{(10}$	1. Byte
c) $-1 / 11_{(2}$	
d) 256	2. Word
e) E4.5	
f) BA(16	
g) –A(16	3. DWord
h) F(16	
i) 64 200	4. Int
j) –12	
k) 80 100	5. Dint
l) 255	
m) 3,5	6. Real
n) –3,5	
ñ) Falso	7. Bool
o) MB20	
p) E0.4	

Nota: algunos de los números podrían asociarse a diferentes tipos de datos, pero debes elegir el más adecuado para almacenar dicho valor.

2. Programa, en el mismo proyecto, los ejemplos de las siguientes transferencias de números codificados en diferentes sistemas de numeración:

Al accionar las siguientes entradas se envían los siguientes datos al primer *byte* de salidas del autómata:

- Entrada_1 → 0
- Entrada_2 → 170
- Entrada_3 → 255
- Entrada_4 → $11110000_{(2}$
- Entrada_5 → $00001111_{(2}$
- Entrada_6 → $11110000_{(2}$
- Entrada_7 → $55_{(16}$
- Entrada_8 → $AA_{(16}$

3. Haz lo mismo que en la actividad anterior; en este caso el dato se debe enviar a la primera palabra (*word*) de salidas del PLC:

- Entrada_1 → $3F01_{(16}$
- Entrada_2 → 34952
- Entrada_3 → $F0F0_{(16}$
- Entrada_4 → $0F0F_{(16}$

ACTIVIDADES FINALES

continuación

4. Utilizando instrucciones de transferencia MOVE, diseña un programa que permita activar de forma intermitente, con un bit de la marca de ciclo, un *byte* completo de salidas del autómata. En el funcionamiento intermitente se deben alternar las salidas pares con las impares cada 0,5 segundos.

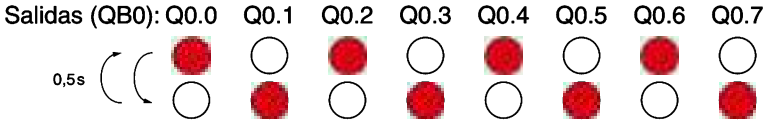


Figura 9.31. Transferencia con funcionamiento intermitente.

5. Diseña, programa y comprueba el funcionamiento de un GRAFCET con secuencias opcionales, de al menos diez etapas, que cumpla las siguientes condiciones de funcionamiento:

- Todas las salidas se deben activar mediante operaciones de transferencia, de forma que en cada etapa del GRAFCET el número enviado al *byte* de salidas sea diferente.
- Debe tener una zona de inicialización de al menos dos etapas.
- En la etapa de espera, que debe ser a la que se acceda después de la zona de inicialización, todas las salidas deben estar desactivadas.
- Cuando el PLC entra en la etapa inicial, las salidas impares del *byte* de salidas deben funcionar de forma intermitente hasta que se salga de ella.
- El paso entre etapas debe hacerse mediante entradas del PLC y, al menos en dos de los casos, con temporizadores.

6. Utilizando operaciones de suma y resta, realiza el programa de un selector digital controlado con dos entradas binarias cuyo funcionamiento es el siguiente:

- Cada vez que se acciona el primer pulsador una variable se incrementa. Cuando la acción es sobre el segundo pulsador, la variable disminuye. En ambos casos, el número de incremento y decremento es de 1.
- Cuando el valor de la variable es 0 todas las salidas digitales deben estar desactivadas. Sin embargo, cuando el valor de dicha variable está entre 1 y 4 las salidas se activan según lo mostrado en la figura.
- Se debe evitar que el valor de la variable baje a valores negativos.
- Si el valor es superior a 4, la salida Q0.4 debe activarse de forma intermitente para indicar que se ha sobrepasado el valor permitido.
- Si cualquiera de las dos entradas se acciona durante más de 5 segundos, la variable se pone a valor 0.

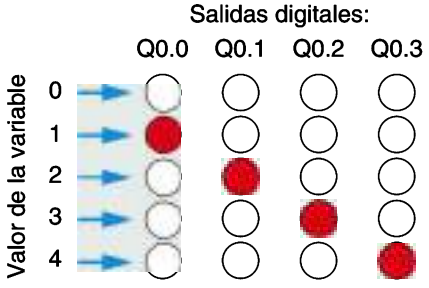


Figura 9.32. Secuencia de funcionamiento del selector digital.

7. Realiza un programa que actúe sobre el *byte* de salidas digitales de la siguiente forma:

- Al pasar el PLC de STOP a RUN se debe transferir el número 1 al *byte* de salidas.
- Si se mantiene accionado un interruptor conectado a una de las entradas digitales, el valor del *byte* de salidas debe aumentar cada segundo al doble del valor anterior. Es decir, cuando está a 1 el valor debe aumentar a 2, cuando está a 2 el valor debe aumentar a 4 y así sucesivamente.
- Cuando el valor transferido al *byte* de salidas es superior a 255 la secuencia debe comenzar de nuevo según lo descrito, siempre que el interruptor de la entrada se mantenga activado.

PRÁCTICA PROFESIONAL PROPUESTA 1

Herramientas

- PC con TIA PORTAL y PLCSIM
- Un polímetro

Material

- Un PLC compatible con TIA PORTAL

Precauciones

- Cada operación de flanco debe utilizar un bit de marcas que no puede usarse para ninguna otra operación en el programa.
- Si en el programa se crea un bloque FC, este debe ser llamado desde el OB1.

Incremento y decremento progresivo de una señal analógica de salida

Objetivos

Usar las operaciones matemáticas de suma y sustracción para incrementar progresivamente el valor de una señal analógica de salida.

Desarrollo

1. Utilizando el mismo dispositivo de la Práctica Profesional Resuelta de esta unidad, crea un bloque FC y escribe en el siguiente programa:

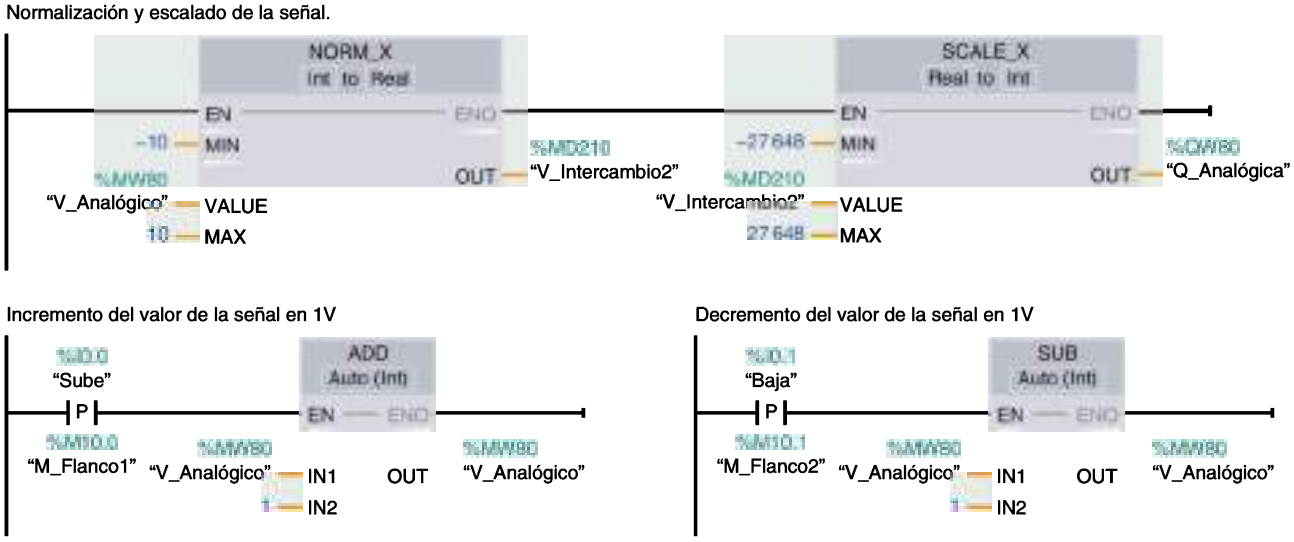


Figura 9.33. Programa básico para el incremento/decremento progresivo del valor de una salida analógica.

2. Conecta un polímetro para medir la tensión de la salida analógica.
3. Llama al FC desde el OB1.
4. Carga el programa en el PLC.
5. Comprueba su funcionamiento y observa qué ocurre con la tensión de salida cada vez que se acciona uno de los pulsadores de entrada.
6. ¿Qué ocurre cuando se superan los valores –10 y 10? ¿Qué debes hacer para evitar que puedan sobrepasar, por arriba y por abajo, ese rango?
7. ¿Qué debes hacer para que el valor de la señal analógica se ajuste a 0 manualmente mediante un tercer pulsador?
8. Haz los cambios necesarios para poder realizar lo indicado en los dos últimos puntos y comprueba su funcionamiento.

PRÁCTICA PROFESIONAL PROPUESTA 2

Herramientas

- PC con TIA PORTAL y PLCSIM

Material

- Un cable para comunicar el PLC con el PC
- Un PLC compatible con TIA PORTAL

Precauciones

- Ten en cuenta que el bit de menor peso de un número en binario es el que está más a la derecha, que en este caso se corresponde con la salida %Q0.0.
- En el modo de ámbar intermitente puedes optar por controlar las salidas con bobinas individuales para cada una de ellas o mediante instrucciones MOVE para hacer parpadear los bits correspondientes a dichas salidas. La primera opción es la que ya has utilizado en unidades anteriores, así que es aconsejable que intentes dar la solución utilizando operaciones de transferencia.

Control de un semáforo mediante la transferencia de números en binario

Objetivo

Programar un GRAFCET para el control de un semáforo y utilizar acciones para transferir números en binario al *byte* de salidas digitales.

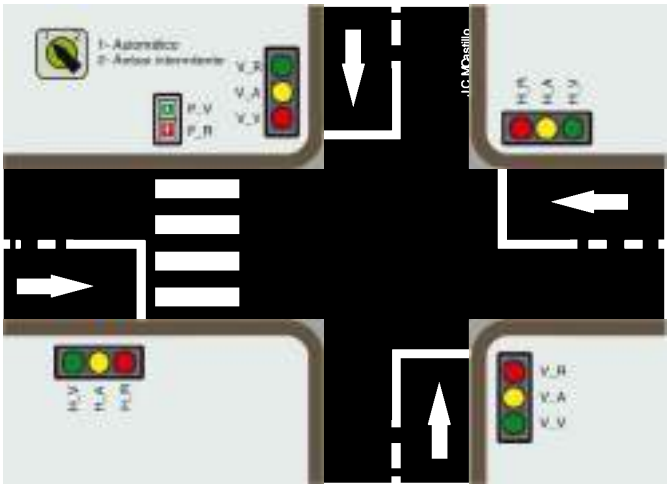
Desarrollo

1. Diseña el GRAFCET para el funcionamiento del semáforo de la figura 9.34. Debes tener en cuenta que la activación de salidas en la zona de acciones se hace mediante operaciones de transferencia (MOVE) de los números codificados en binario, correspondientes al funcionamiento de la tabla de la misma figura.
2. Debes contemplar dos modos de funcionamiento elegibles con el conmutador rotativo:

   a) Automático: en el cual la secuencia funciona según la tabla de la figura.

   b) Ámbar intermitente: en el cual la lámpara de color ámbar se activa de forma intermitente.

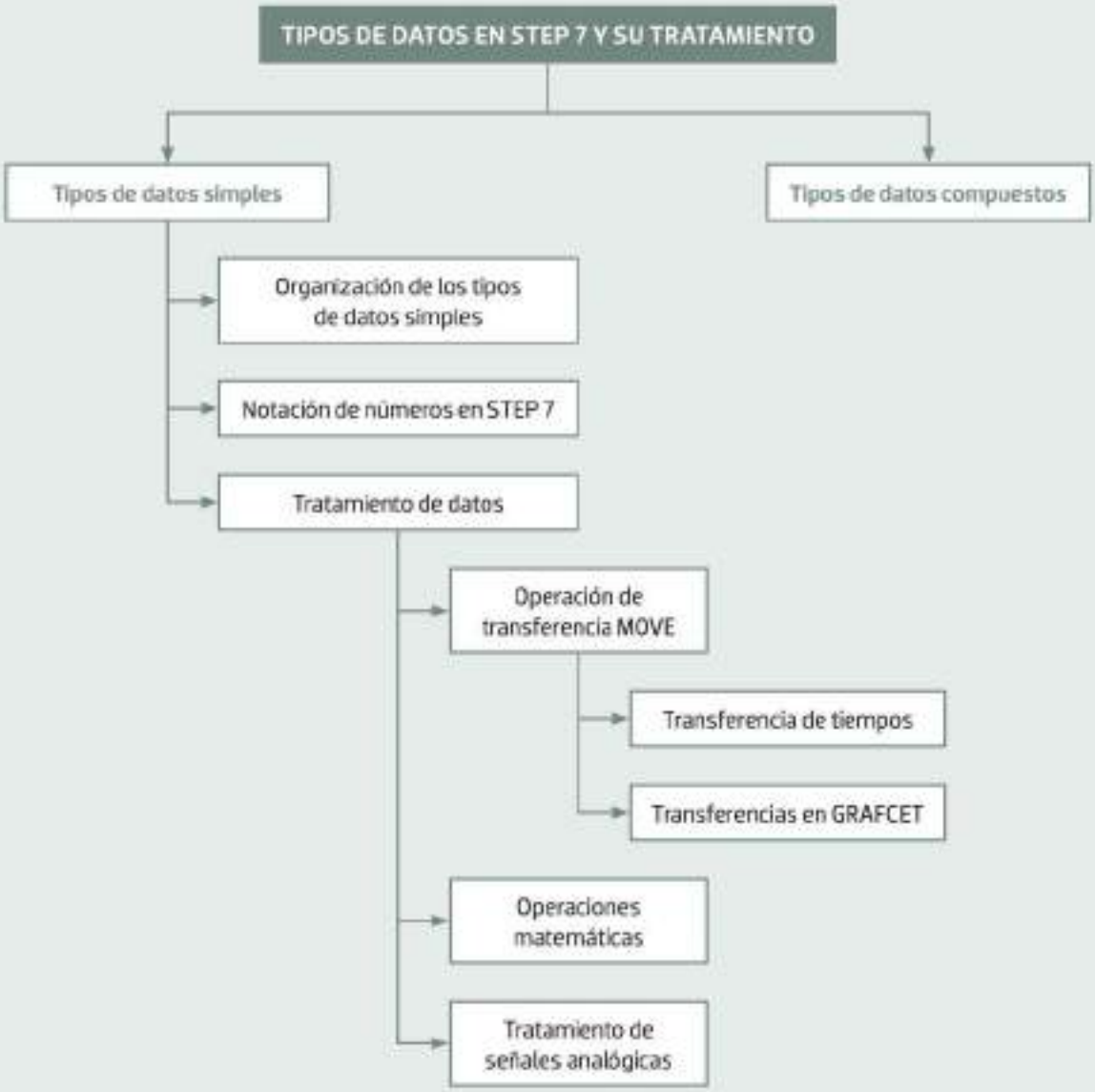
   Cuando se sale del segundo modo, la secuencia principal del modo automático comienza desde la primera etapa.



Tiempos	HORIZONTAL R	HORIZONTAL A	HORIZONTAL V	VERTICAL R	VERTICAL A	VERTICAL V	PEATONES R	PEATONES V	%QB0 Dato que transferir
2s	1	0	0	1	0	0	1	0	2#10010010
10s	0	0	1	1	0	0	1	0	2#00110010
2s	0	1	0	1	0	0	1	0	2#01010010
2s	1	0	0	1	0	0	1	0	2#10010010
10s	1	0	0	0	0	1	0	1	2#10000101
2s	1	0	0	0	1	0	1	0	2#10001010

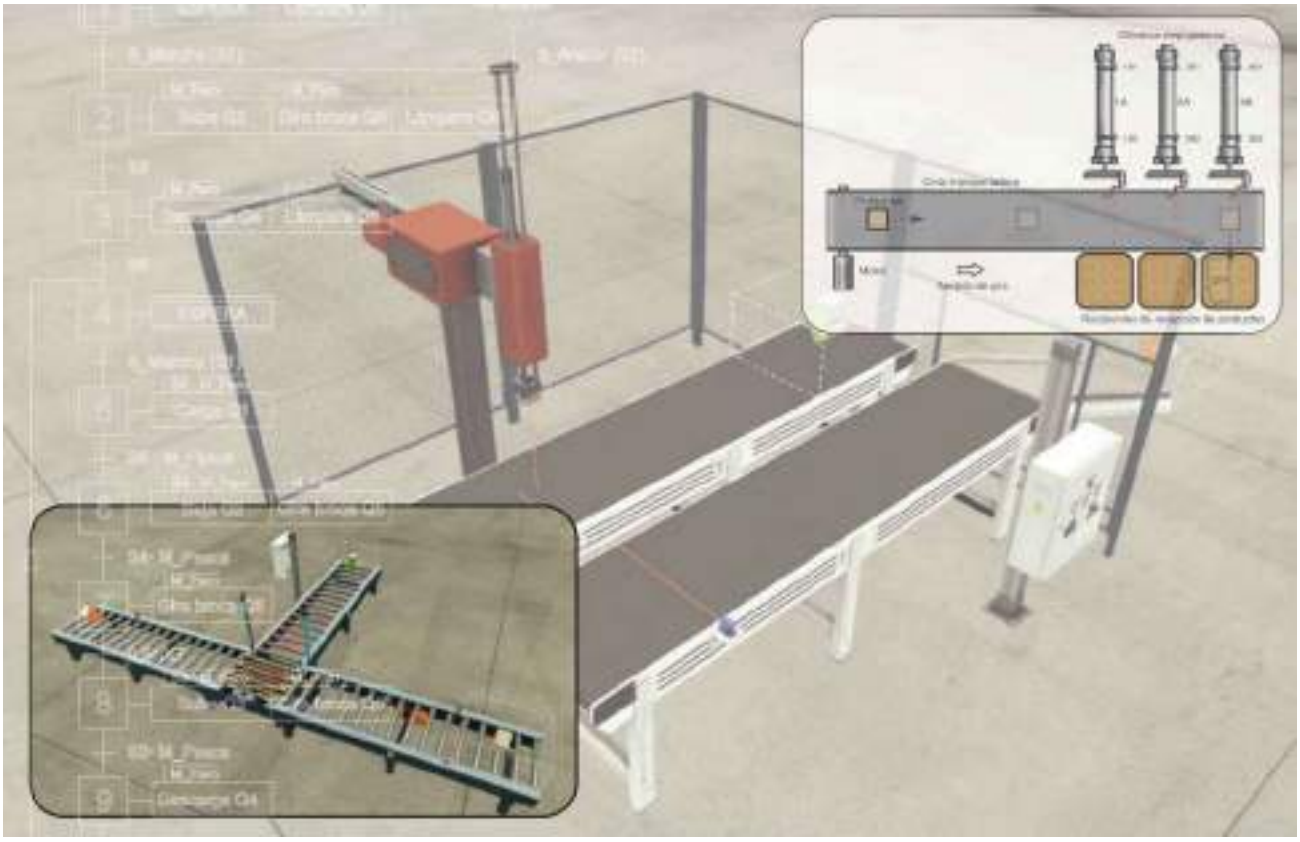
Figura 9.34. Control de un sistema de semáforos con transferencia de números en binario.

EN RESUMEN


TIPOS DE DATOS EN STEP 7 Y SU TRATAMIENTO
Tipos de datos simples
Tipos de datos compuestos
Organización de los tipos de datos simples
Notación de números en STEP 7
Tratamiento de datos
Operación de transferencia MOVE
Transferencia de tiempos
Transferencias en GRAFCET
Operaciones matemáticas
Tratamiento de señales analógicas

PROYECTOS

- Control de un proceso industrial de amasado
- Mezclado de productos líquidos
- Llenado de cajas por número de objetos
- Almacén de cajas por alturas (Factory I/O)
- Separación de objetos por colores (Factory I/O)
- *Pick & Place* (Factory I/O)



Descripción

Los proyectos que se describen a continuación tienen como objetivo afianzar las diferentes técnicas de diseño y programación estudiadas a lo largo del libro. Aunque algunos de ellos se pueden desarrollar a partir de la unidad 5, es aconsejable realizarlos cuando se haya concluido el estudio de todas las unidades.

Tareas a realizar en los proyectos

Para el desarrollo de todos los proyectos, se deben realizar las siguientes tareas:

1. Elaborar de una tabla de variables de E/S.
2. Diseñar esquemas de fuerza en los que se represente cómo se controlan los actuadores de potencia propuestos en las descripciones: motores, cilindros neumáticos, etc.
3. Diseñar de esquemas de control en los que se representen con detalle las conexiones de todos los dispositivos a las entradas y salidas del PLC.
4. Representar gráficamente de la secuencia mediante GRAFCET en la que se incluyan los diferentes modos de funcionamiento propuestos.
5. Programar en lenguaje KOP de los GRAFCET diseñados para dar solución a las secuencias de funcionamiento.
6. Comprobar el funcionamiento de los programas. Siempre que sea posible, se utilizará un *software* de simulación de procesos, como puede ser Factory IO, SPS Visu, PLC-Lab, SIMIT, o cualquier otro de similares características.

Los proyectos quedan abiertos a que el lector aporte cualquier mejora que se le pueda ocurrir y que se adapte a los contenidos descritos en el libro.

Control de un proceso industrial de amasado

Descripción del funcionamiento

Producción

- El pulsador S1 pone en marcha la cinta trasportadora hacia la derecha
- Cuando el material pasa por el final de carrera de varilla, se activa el motor del agitador.
- A los 10 minutos se activa la bomba extractora, mientras el agitador continúa en funcionamiento.
- La bomba y el agitador se desactivan cuando no haya producto en la tolva.

Parada

- El circuito se debe poder detener en cualquier momento mediante el pulsador de parada. En ese caso, todos los motores dejan de funcionar.
- El proceso debe continuar por donde se encontraba cuando se vuelva a accionar el pulsador de marcha.

Anular

El pulsador de anular permite cancelar el proceso con el siguiente funcionamiento:

- La cinta transportadora gira a izquierdas hasta tirar el producto al exterior. Como no existe ningún detector en el extremo derecho de la cinta, esta se detiene después de un tiempo de aproximadamente 10 segundos.
- La bomba y el agitador funcionan hasta que la tolva se queda vacía.

Arranque en caliente

- La lámpara H1 debe funcionar de forma intermitente después de un arranque en caliente y avisar al operario de que se debe accionar el pulsador de marcha.
- Después de esto, si hay producto en la cinta transportadora, este se descarga por el lado izquierdo. La cinta deja de funcionar después de unos segundos (por ejemplo, 10 s).
- Si la tolva de agitación tiene producto, esta se vacía mediante la bomba extractora.

Señalizaciones

Las lámparas deben señalizar las siguientes situaciones:

- H1: proceso en marcha.
- H2: disparo de alguna de las protecciones de los motores.
- H3: anulación de mezcla.

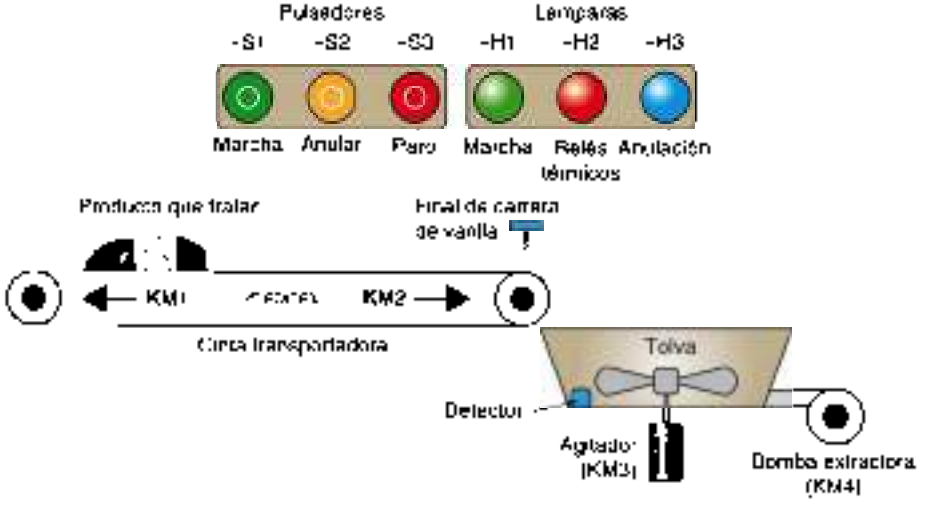


Figura 10.1. Proceso industrial de amasado de productos.

Mezclado de productos líquidos

Descripción del funcionamiento

Producción

- El proceso permite hacer tres tipos de mezclas en función de la posición en la que se encuentre el selector y atendiendo a la siguiente tabla:

Pos. selector	Producto 1	Producto 2	Producto 3	Producto 4	Agitador
1	2 s	2 s	3 s	2 s	10 s
2	4 s	5 s	–	5 s	15 s
3	–	6 s	7 s	–	5 s

- Una vez seleccionado el tipo de mezcla, el proceso se inicia con el pulsador de marcha, siempre que no haya ningún depósito sin producto.
- Las válvulas de cada depósito se abren para echar los productos en la tolva de agitación según los tiempos prelijados en la tabla. Hay que tener en cuenta que las válvulas son de tipo monoestable, y permanecen cerradas cuando su solenoide no está alimentado
- Cuando todas las válvulas se han cerrado el agitador comienza a funcionar según el tiempo seleccionado. Después de eso, se pone en marcha la bomba extractora hasta que la tolva se vacía por completo

Anular

Todas las válvulas se deben cerrar, el agitador debe dejar de funcionar y se tiene que poner en marcha la bomba extractora hasta que la tolva se vacíe por completo.

Parada

Se cierran todas las válvulas y se paran todos los motores. Al accionar de nuevo el pulsador de marcha el proceso continúa por donde estaba antes de la parada. La falta de producto en alguno de los depósitos también debe detener el proceso

Arranque en caliente

Una de las lámparas se debe encender de forma intermitente. Cuando se acciona el pulsador de marcha, si aún queda producto en la tolva, esta se vacía mediante la bomba extractora 1

Señalización

Se debe activar una lámpara de señalización por cada uno de los estados de funcionamiento del proceso (mezclando, agitando, anulando y extrayendo producto).

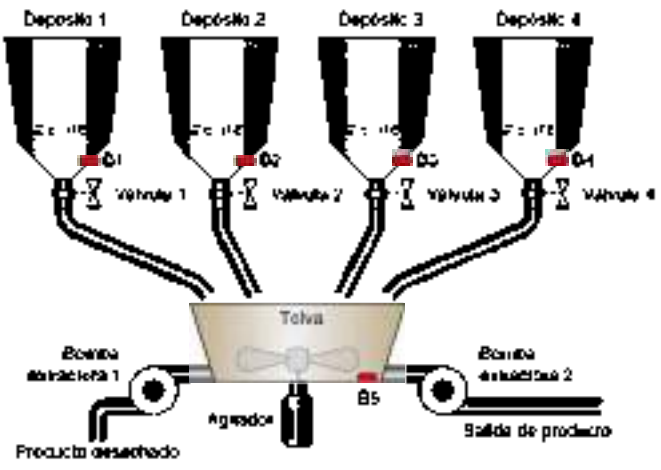


Figura 10.2. Mezcladora de productos.

Llenado de cajas por número de objetos

Descripción del funcionamiento

Producción

- Los productos se trasladan por la cinta transportadora hacia la zona de los cilindros.
- Los detectores fotoeléctricos B3, B4 y B5 detectan los productos cuando están en la zona de empaquetado.
- Cuando un producto es detectado por uno de los finales de carrera la cinta se detiene y el cilindro correspondiente la empuja a la caja contenedora que está enfrente de él.
- Los cilindros están controlados por electroválvulas biestables.
- Cada caja puede almacenar cuatro objetos.
- Las cajas se llenan de derecha a izquierda.
- Cuando todas las cajas están llenas se activa una lámpara de forma intermitente que indica al operario que debe retirarlas. En este estado, todos los contadores se resetean para comenzar un nuevo ciclo de llenado.

Anular

- Hay que recoger los cilindros y los productos se deben desechar por el lado derecho de la cinta transportadora. Los contadores no se deben resetear, ya que si hay algún producto en las cajas, el proceso debe continuar por donde estaba cuando se reanude el funcionamiento mediante el pulsador de marcha.
- La cinta se parará de forma automática a los 5 segundos de que el objeto desechado pase por el detector B5.

Parada

El proceso se detiene al accionar el pulsador de parada y se reanuda al accionar el pulsador de marcha.

Arranque en caliente

Una de las lámparas se debe encender de forma intermitente. Cuando se acciona el pulsador de marcha, la cinta transportadora se pone en marcha y desecha los objetos que en ella se encuentran de forma similar a lo descrito en la operación anular. Si no hay ningún producto en la cinta, esta se parará después de 20 segundos.

Señalización

- Se deben utilizar tres pilotos de señalización para indicar cuándo está completa cada una de las cajas.
- Además, se debe señalizar cuándo el proceso está en funcionamiento de producción, en arranque en caliente y anulado.

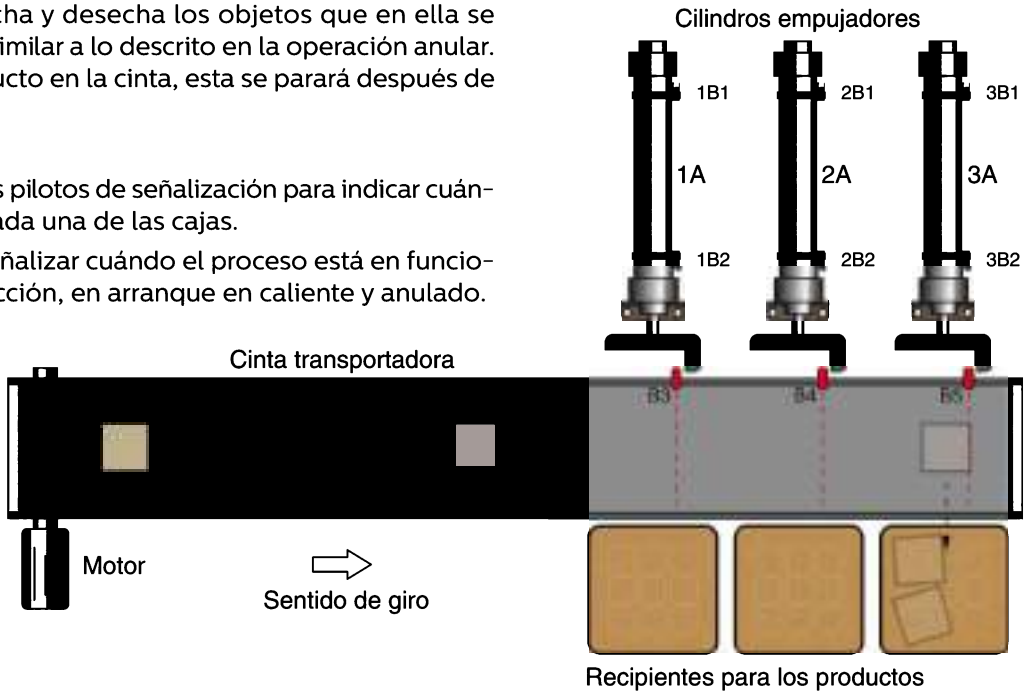


Figura 10.3. Llenado de recipientes desde una cinta transportadora.

Almacén de cajas por alturas (Factory I/O)

Descripción del funcionamiento

Producción

- Las cajas pueden tener dos alturas y aparecen de forma aleatoria por el transportador de rodillos principal.
- Los tres transportadores están unidos en uno de sus extremos por un cuarto transportador de rodillos de tipo bidireccional. Este permite, una vez que se ha situado sobre él un palé con una caja, desviarlo a la izquierda o a la derecha, según la atura detectada por la barrera fotoeléctrica ubicada en su entrada. Este transportador dispone de un actuador que realiza las siguientes funciones:
  - Elevar el palé con la caja para facilitar su desplazamiento.
  - Desviarlo en un sentido u otro para descargarlo sobre los transportadores secundarios.
- Cuando un palé con una caja es depositado en cualquiera de los transportadores de rodillos secundarios estos los desplazan hasta sus extremos para realizar la descarga por alturas.

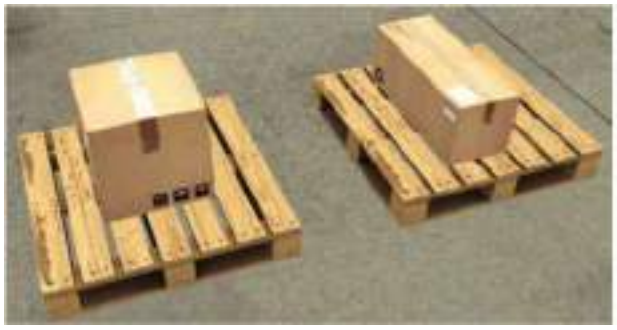


Figura 10.4. Tipos de objetos que procesar.

Figura 10.5. Puerta del cuadro de control.

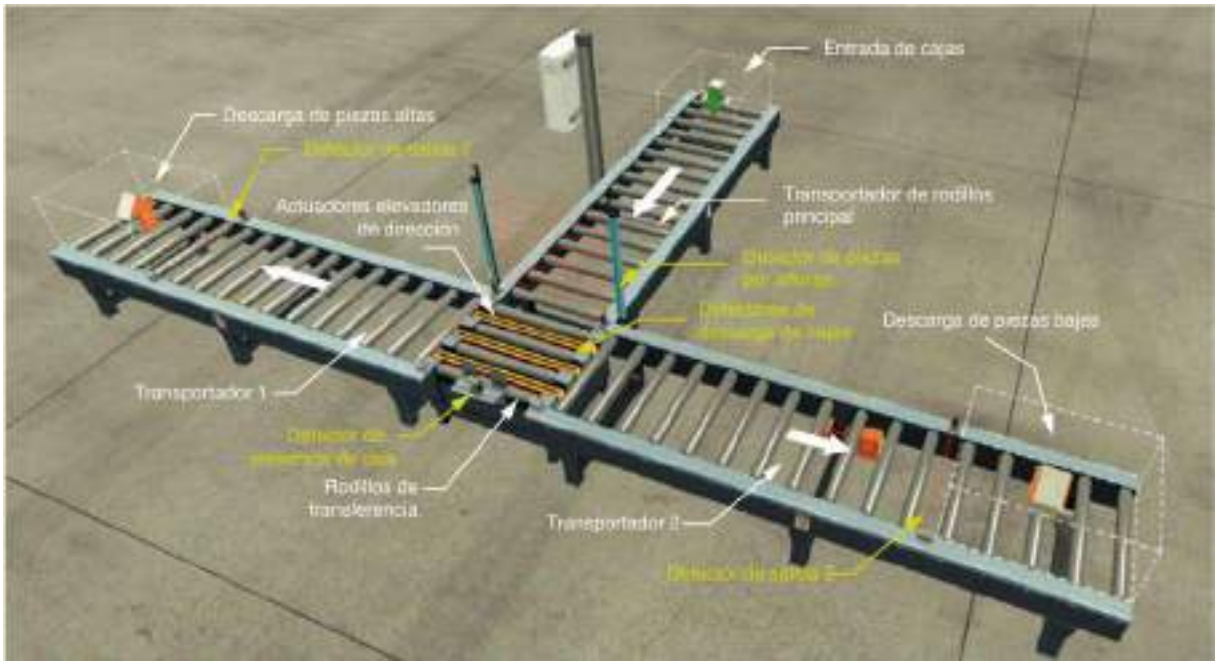


Figura 10.6. Sistema de selección de cajas por alturas. (Fuente: Real Games – Factory IO).

Parada

El proceso se detiene al accionar el pulsador de parada y se reanuda al accionar el pulsador de marcha.

Modo por pasos

El conmutador rotativo ubicado en la puerta del cuadro de control permite elegir entre el funcionamiento automático (producción) o por pasos.

El avance de cada paso debe hacerse accionando el pulsador de color amarillo.

En el modo de funcionamiento por pasos, si se conmuta el selector a modo automático, el proceso continuará por donde se encontraba.

Arranque en caliente

La lámpara de color rojo se debe encender de forma intermitente y el proceso quedará a la espera de accionar el pulsador de marcha.

Después de la acción sobre este pulsador, se deben vaciar los transportadores secundarios. Si en ellos hay algún palé, estos se llevan hasta los extremos de salida y se descargan. Si, por el contrario, no hay ninguno, como los detectores de salida no se activarán nunca, los transportadores se detendrán de forma automática después de un tiempo.

Si la parada del sistema se realizó cuando un palé se encontraba en los rodillos centrales, se debe prever que, después de un arranque en caliente, se conozca la atura de la caja para procesarla según lo descrito anteriormente.

Señalización

El sistema dispone de tres pilotos que señalizarán las siguientes acciones:

- Funcionamiento automático (verde).
- Funcionamiento por pasos (amarillo intermitente).
- Proceso parado (rojo).
- Arranque en caliente (rojo intermitente).

Figura 10.7. Detalle de la detección de la altura de las cajas con la barrera fotoeléctrica en la entrada de los rodillos centrales. (Fuente: Real Games).

Separación de objetos por colores (Factory I/O)

Descripción del funcionamiento

Producción

- La estación consta de cuatro cintas transportadoras, dos para la alimentación y detección del tipo de pieza, y otras dos para su desplazamiento hasta la zona de descarga.
- La alimentación de piezas se hace de forma aleatoria, pudiendo salir piezas tanto de color azul como de color verde de cualquiera de las cintas de alimentación.
- La detección del tipo de pieza se realiza mediante dos sensores cromáticos situados al final de las cintas de alimentación.
- Si las piezas llegan correctamente (verdes en la cinta superior y azules en la cinta inferior), estas se desplazan hasta las rampas de descarga por sus respectivas cintas de transporte.
- Si, por el contrario, las piezas llegan cambiadas, es decir, azules por la parte superior y verdes por la inferior, la cinta de descarga sobre la que se encuentran se detiene, para que el cilindro neumático correspondiente desplace la pieza a la cinta correcta. Una vez sobre ella, la cinta vuelve a ponerse en marcha hasta que se dejan caer sobre las rampas de descarga.
- El control de los cilindros neumáticos debe realizarse con electroválvulas de tipo biestable.

Parada

El proceso se detiene al accionar el pulsador de parada y se reanuda al accionar el pulsador de marcha.

Modo por pasos

Un conmutador rotativo ubicado en la puerta del cuadro de control permite elegir entre el funcionamiento automático (producción) o por pasos.

El avance de cada paso debe hacerse accionando un pulsador específico para tal fin.

Señalización

Mediante pilotos montados en el cuadro, se deben señalizar las siguientes operaciones:

- Funcionamiento automático.
- Funcionamiento por pasos.
- Detección de piezas invertidas.

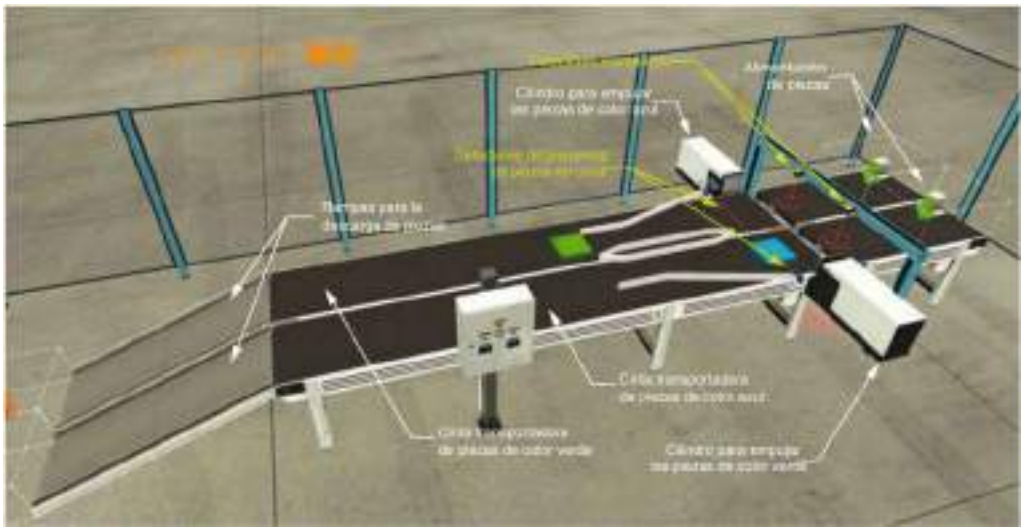


Figura 10.8. Estación para la separación de piezas por colores. (Fuente: Real Games – Factory IO.

Pick & Place (Factory I/O)

Descripción del funcionamiento

Producción

- La estación consta de dos cintas transportadoras, una para la alimentar las piezas hasta el manipulador y otra para la descarga de las mismas.
- El brazo de manipulación consta de dos ejes, uno que avanza y retrocede entre las dos cintas transportadoras y otro que sube y baja para recoger las piezas con la ventosa instalada en su extremo inferior.
- Las piezas salen desde el punto de alimentación. Cuando llegan al punto de recogida la cinta se detiene y el manipulador baja hasta que las detecta. En este momento, la ventosa se activa y succiona el objeto. El brazo se eleva y el eje horizontal se desplaza hasta la cinta de descarga. El brazo baja hasta la cinta de descarga y suelta las piezas sobre ella. Cuando el brazo sube la cinta se pone en marcha y lleva la pieza hasta la rampa de descarga.
- Después de esta acción, el manipulador vuelve a la posición de reposo, que es la que se muestra en la figura.

Parada

El proceso se detiene al accionar el pulsador de parada y se reanuda al accionar el pulsador de marcha. Si la parada se realiza cuando la ventosa está succionando una pieza, esta debe mantenerse en este estado para que no se caiga.

Modo por pasos

Un conmutador rotativo ubicado en la puerta del cuadro de control permite elegir entre el funcionamiento automático (producción) o por pasos.

El avance de cada paso debe realizarse accionando un pulsador específico para tal fin.

Señalización

Mediante pilotos montados en el cuadro, se deben señalizar las siguientes operaciones:

- Funcionamiento automático.
- Funcionamiento por pasos.
- Momento de manipulación de piezas cuando son trasladadas de una cinta a otra.

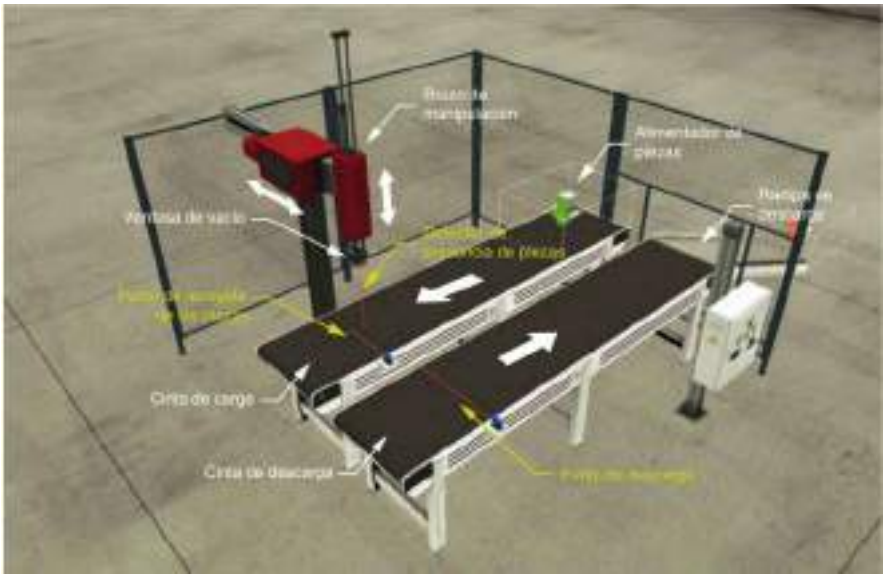


Figura 10.9. Estación de *Pick and Place*. (Fuente: Real Games – Factory IO).

Redacción y selección de contenidos: Juan Carlos Martín Castillo

Coordinación editorial: Javier Ablanque

Edición: Sergio Nombela

Diseño de colección: Juan Pablo Rada / Paso de Zebra

Fotocomposición, maquetación y realización de gráficos: Alfredo Casado

Fotografías: 123RF Limited, Getty Images, RealGames, Siemens AG y archivo Editex

Dibujos: 123RF Limited, autor, Getty Images y archivo Editex

Preimpresión: José Cira

Dirección de producción: Santiago Agudo

El presente material didáctico ha sido creado por iniciativa y bajo la coordinación de Editorial Editex, S. A., conforme a su propio proyecto editorial.

© Editorial Editex, S. A.

Vía Don Castillos, 33. C.E. Ática 7, edificio 3, planta 3ª, oficina 8

28224 Pozuelo de Alarcón (Madrid)

ISBN papel: 978-84-1321-228-9

ISBN eBook: 978-84-1321-281-8

ISBN LED: 978-84-1321-294-4

Depósito Legal: M-2115-2021

Impreso en España - *Printed in Spain*