



COMILLAS
UNIVERSIDAD PONTIFICIA

ICAI

ICADE

CIHS

Tecnologías para la automatización: Automatismos programados (PLC)

Prof. Dr. José Antonio Rodríguez Mondéjar
mondejar@comillas.edu

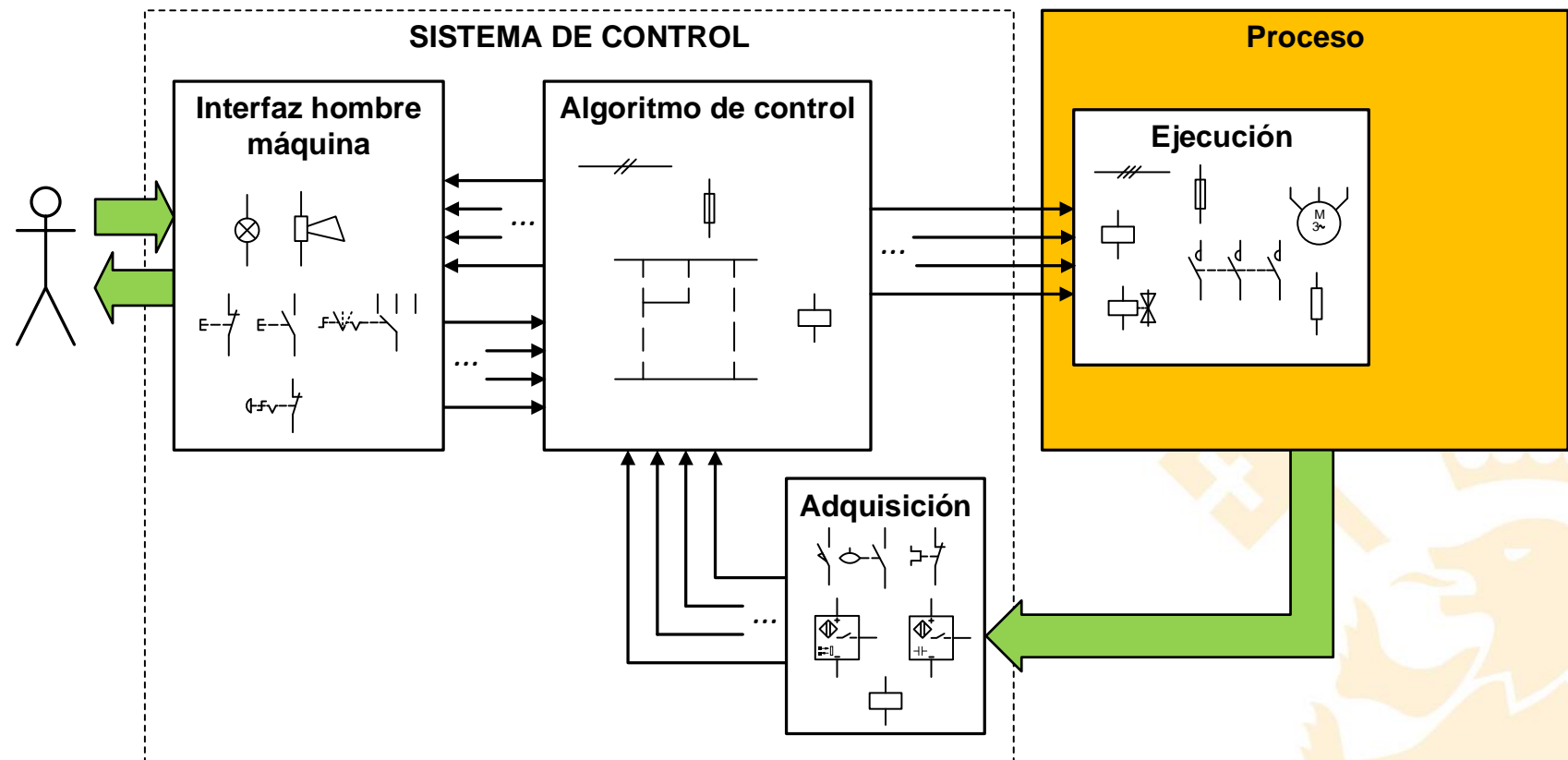
Escuela Técnica Superior de Ingeniería ICAI
Departamento de Electrónica, Automática y Comunicaciones

Enero 2023

comillas.edu

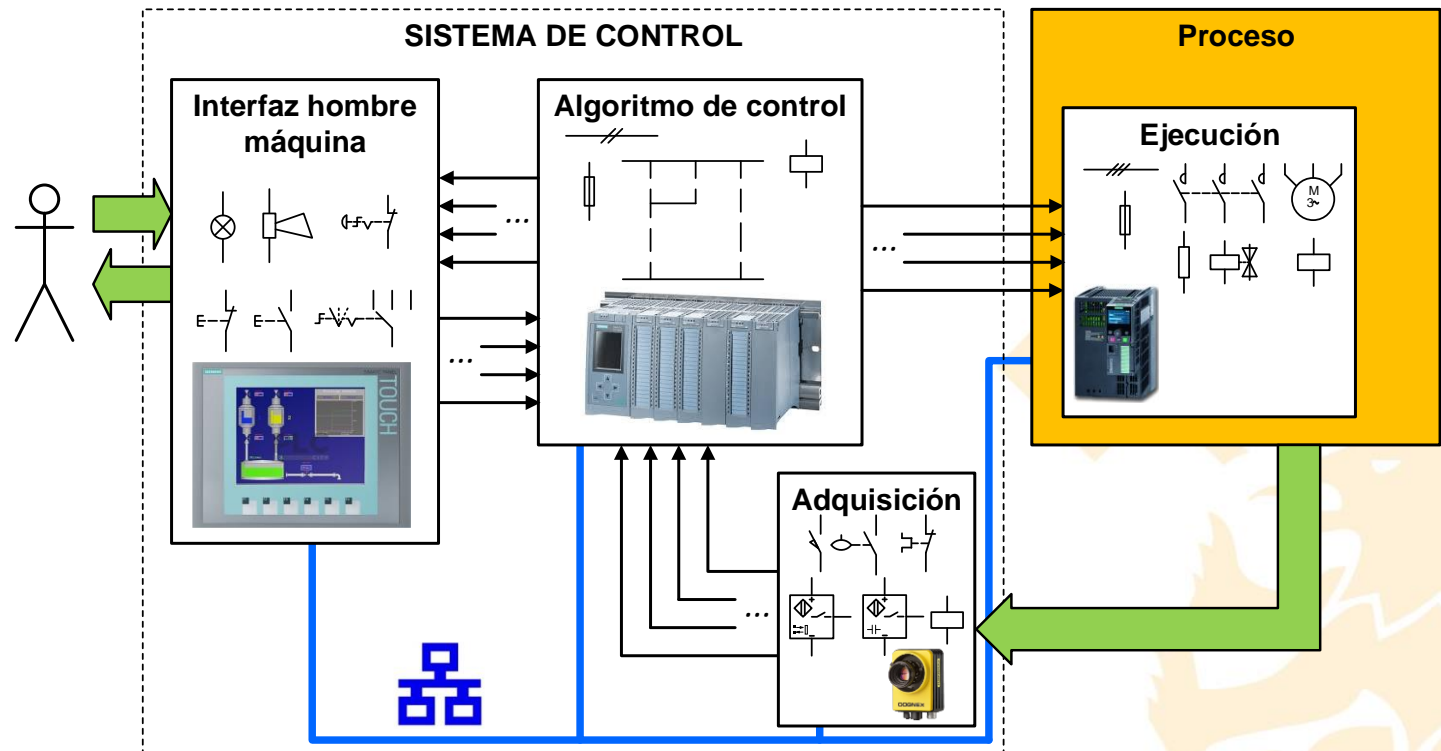
Visto: Automatismo cableado de tipo eléctrico

- Tres tipos de elementos: contactos, cables (conexiones) y consumidores de energía



Automatismo programado

- A los dispositivos de tipo cableado se añaden los de tipo programado
 - Parte de los cableados se sustituyen por programados
- Gran parte del algoritmo de control reside en un sistema programable: PLC o similar

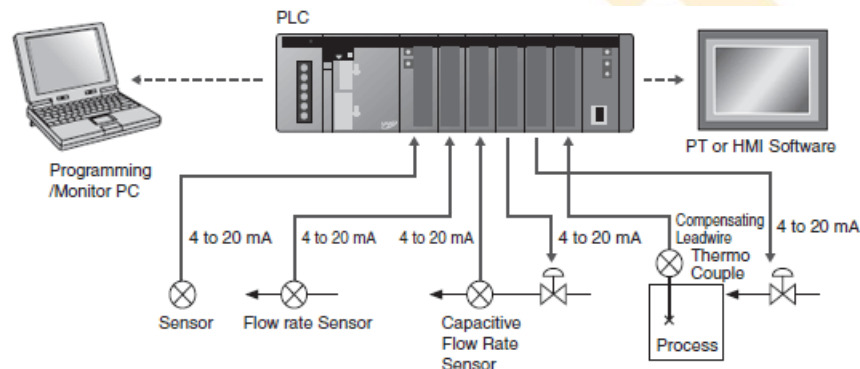


Definición de PLC según norma IEC 61131-1

Autómata programable (PLC): Sistema electrónico con funcionamiento digital, diseñado para su uso en un entorno industrial, que utiliza una memoria programable para el almacenamiento interno de instrucciones orientadas al usuario, para la implementación de funciones específicas de tipo lógico, secuencial, de temporización, de conteo, y aritmética, para controlar mediante entradas y salidas digitales o analógicas, diversos tipos de máquinas o procesos. Tanto el PLC como sus periféricos asociados están diseñados de manera que puedan ser fácilmente integrados en un sistema de control industrial y utilizados de acuerdo con sus funciones pretendidas.

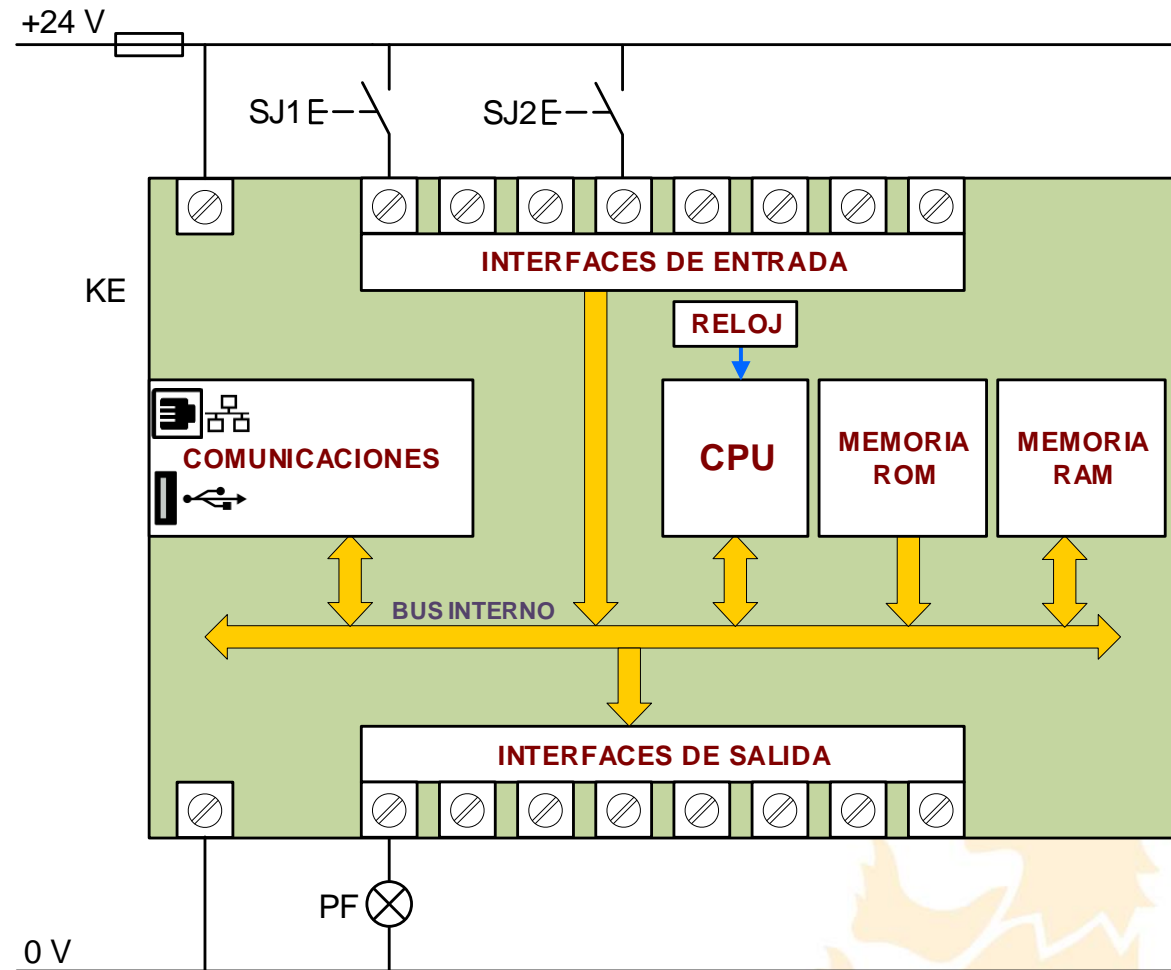
- *Nota: La abreviatura PLC (Programmable Logic Controller) se utiliza en esta norma para designar autómatas programables, como en la práctica común en la industria de la automatización. El uso de PC (Programmable Controller) como abreviatura de autómatas programables crea confusión con los ordenadores personales*

**Origen: sustituir la
lógica cableada por
programada**



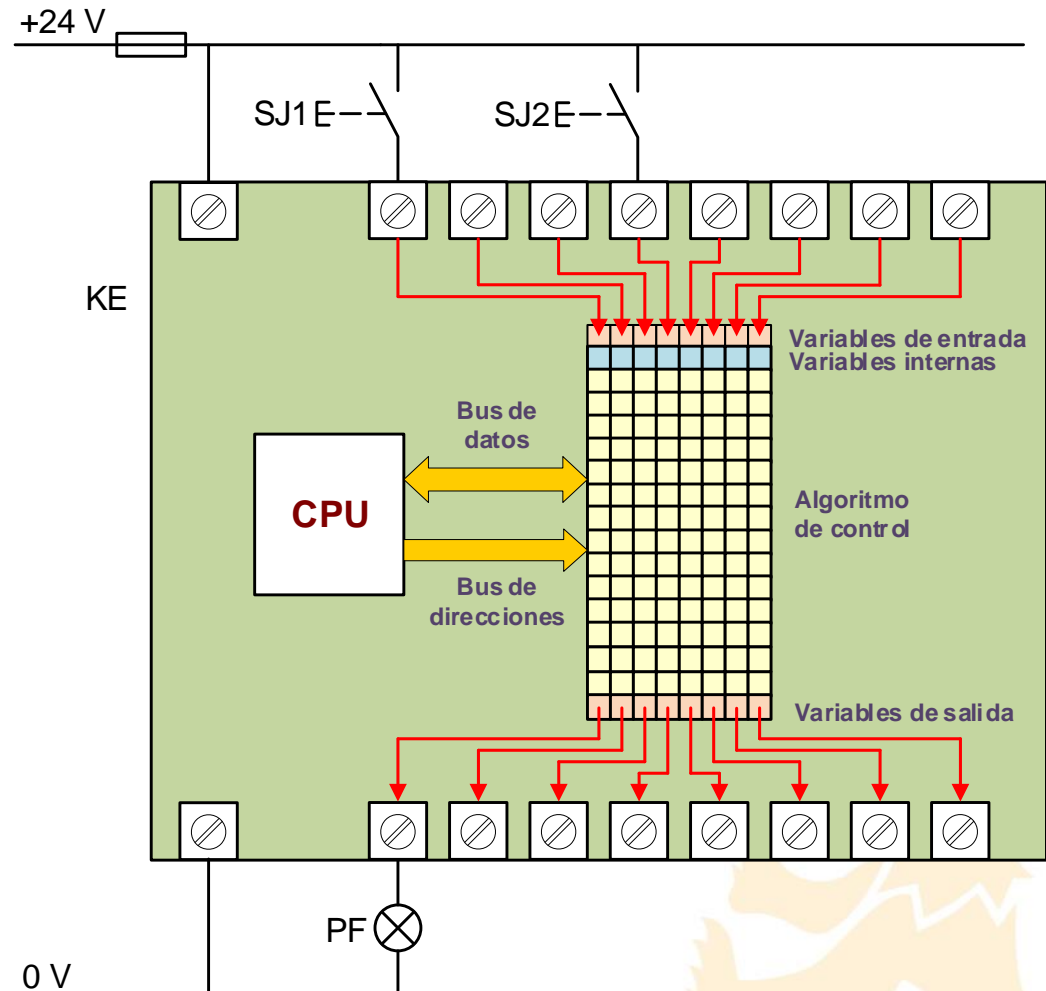
Arquitectura típica de un PLC

- Semejante a un ordenador
 - CPU
 - Reloj
 - Memoria ROM y RAM
 - Interfaces de entrada y salida
 - Comunicaciones
- En general, sin teclado ni pantalla
 - Entrada/salida por bornas (tornillos)
- Nombre IEC81346: KE



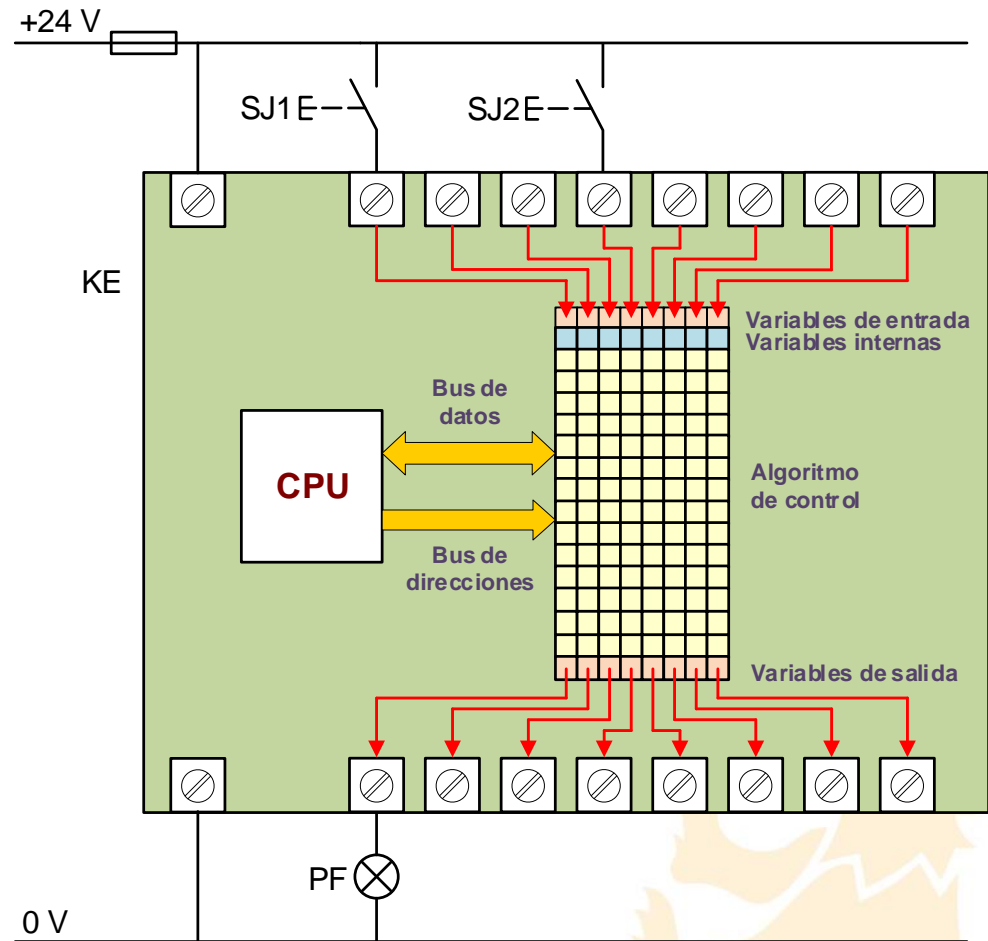
Función básica de la memoria: almacenar

- **Posiciones destinadas a recibir los valores de las entradas físicas convertidos a lógicos**
 - Rangos típicos
 - 0: [-30 V DC, 5 V DC]
 - 1: [11 V DC, 30 V DC]
 - Se denominan variables de entrada
- **Posiciones destinadas a recibir los valores lógicos que aparecerán convertidos en las salidas físicas**
 - 1: 24 V DC
 - 0: 0 V DC
 - Se denominan variables de salida
- **Posiciones destinadas a mantener valores internos del algoritmo de control**
 - Se denominan variables internas (también marcas)
- **Posiciones que almacenan las instrucciones del algoritmo de control y programas auxiliares**



Función básica de la CPU: ejecutar el programa en memoria

- CPU envía una dirección a la memoria para obtener una instrucción
- Memoria envía a la CPU la instrucción pedida
- CPU decodifica y ejecuta la instrucción pedida
 - Si la instrucción necesita un dato que está en memoria, la CPU pedirá el dato a memoria enviándole la dirección dónde está
 - Si la instrucción necesita escribir sobre una posición de memoria, la CPU envía a memoria la dirección de la posición y el valor a escribir
- La CPU continúa la ejecución con la siguiente instrucción
 - Continúa con la ejecutada
 - O a donde indique la instrucción ejecutada
- Igual que cualquier sistema basado en microprocesador

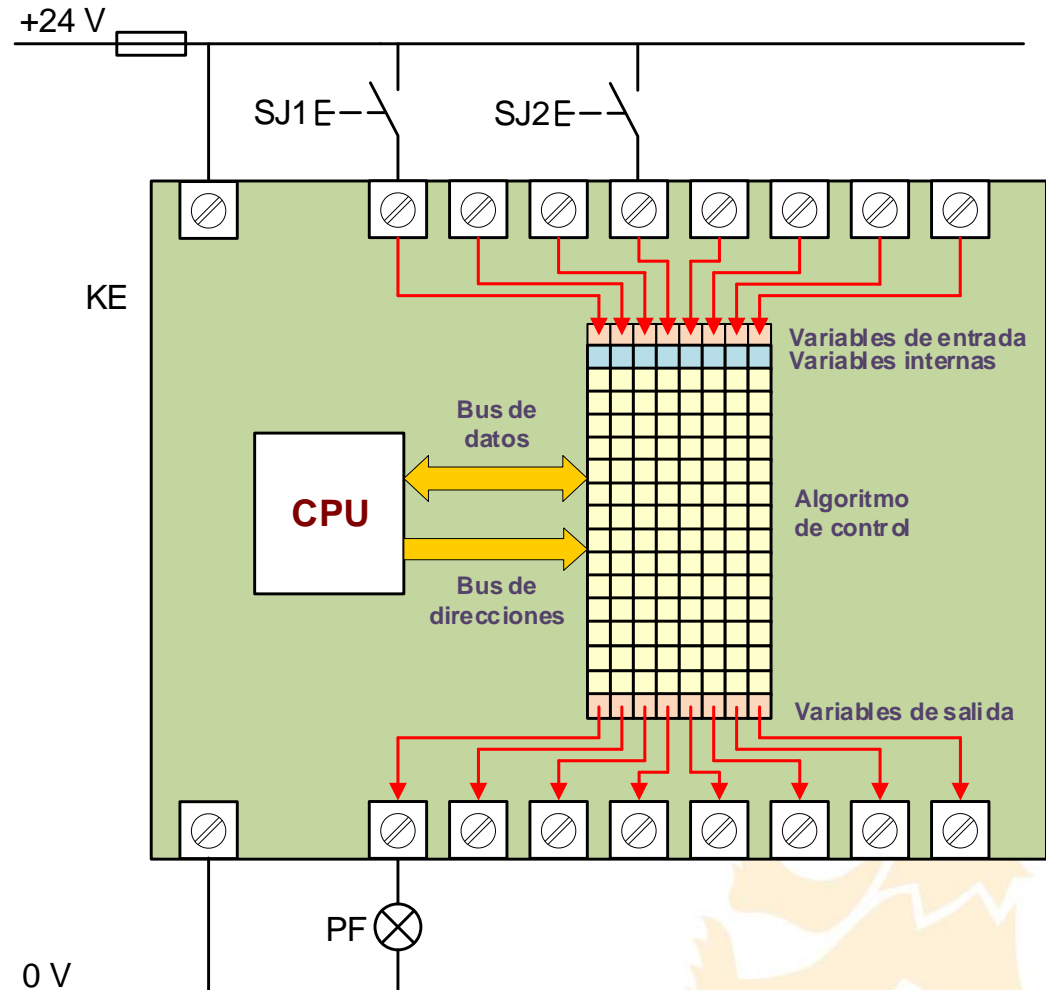


Programa típico que ejecuta la CPU del PLC

• CPU ejecuta de forma cíclica:

- Lee todas las entradas físicas y deposita su valor en las variables de entrada
 - El hardware puede hacer este trabajo
- Ejecuta el algoritmo de control: las funciones que calcularán nuevos valores para las variables de salida a partir de las variables de entrada y las variables internas
 - También puede actualizar las variables internas
- Escribe los valores de todas las variables de salida en las salidas físicas
 - El hardware puede hacer este trabajo

• Tiempo de ciclo: ciclo de scan



Ejemplo de ejecución: salida igual a producto de 2 entradas

• PLC equipado con

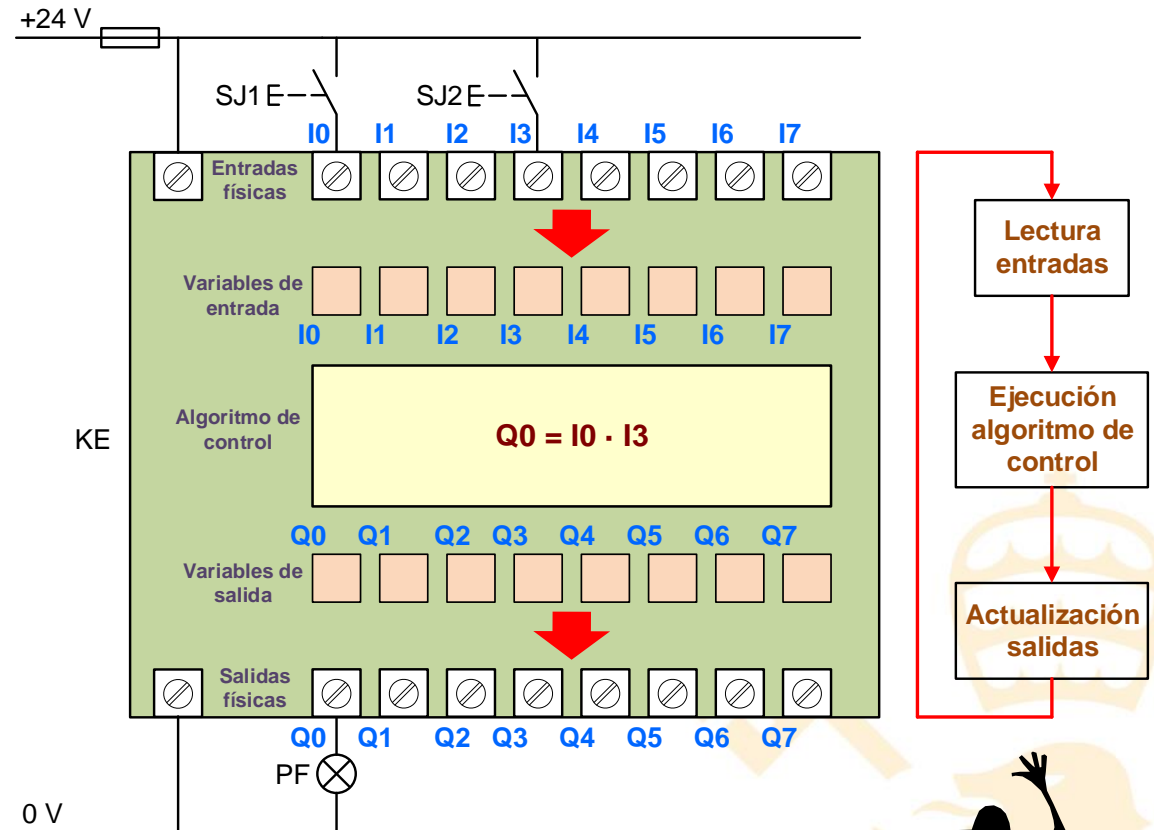
- 8 entradas físicas: I0...I7
 - Conexión lógica con 8 variables de entrada de igual nombre
- 8 salidas físicas: Q0...Q7
 - Conexión lógica con 8 variables de salida de igual nombre
- Nombres físicos y lógicos coinciden para simplificar
 - No confundir entrada o salida con variable

• Conexión física del PLC

- I0 con el pulsador SJ1
- I3 con el pulsador SJ2
- Q0 con el piloto PF

• Algoritmo de control

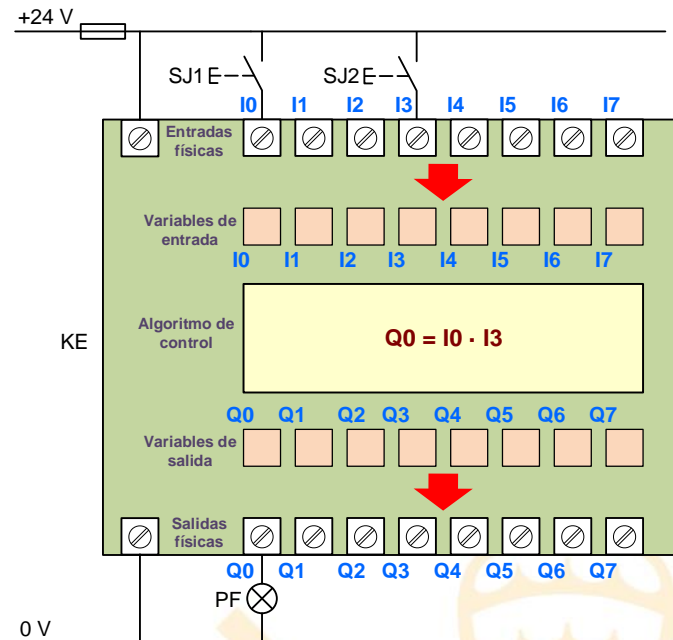
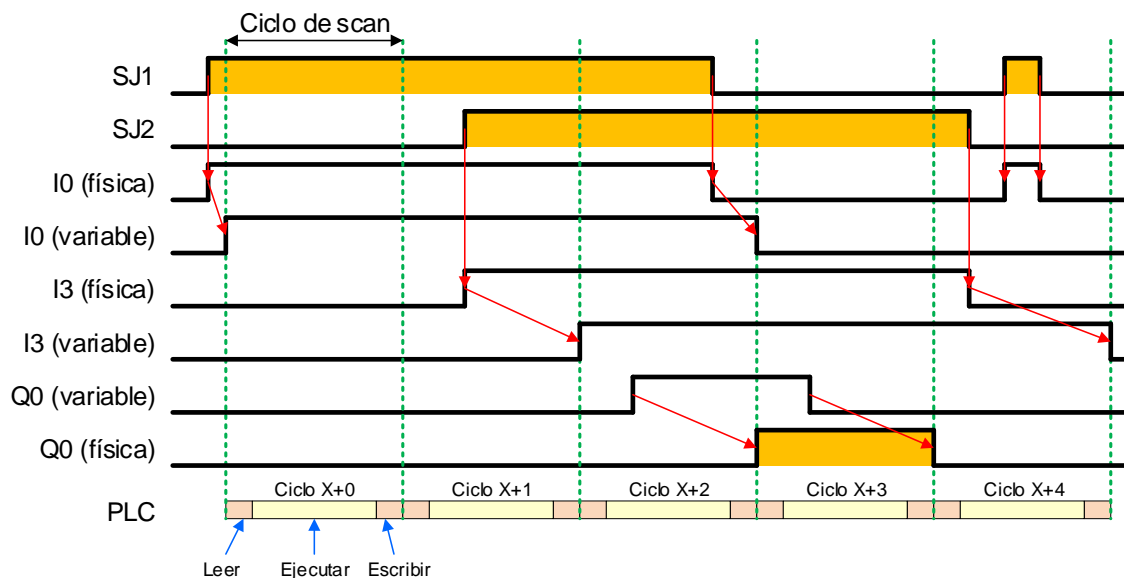
- $Q0 = I0 \cdot I3$



**Sólo necesito cambiar el programa
 para cambiar la función lógica.
 ¡No necesito recablear!**



Cronograma del ejemplo



• Limitaciones del PLC

- La respuesta tiene un retraso entre 1 y 2 ciclos de scan
 - Ciclo de scan típico: [1 ms, 5 ms]
- Se pueden perder cambios en las entradas menores de un ciclo de scan

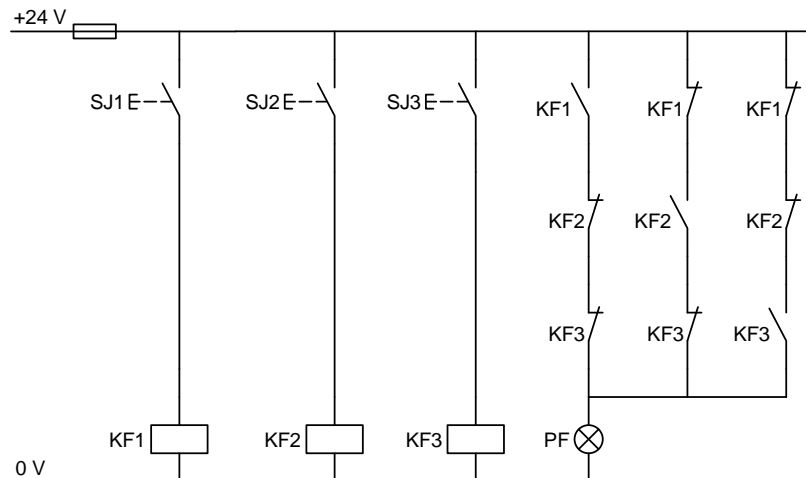


¿Es un problema el ciclo de scan? ¿Qué pasa si pulso muy rápido?

Automatismo cableado vs automatismo programado

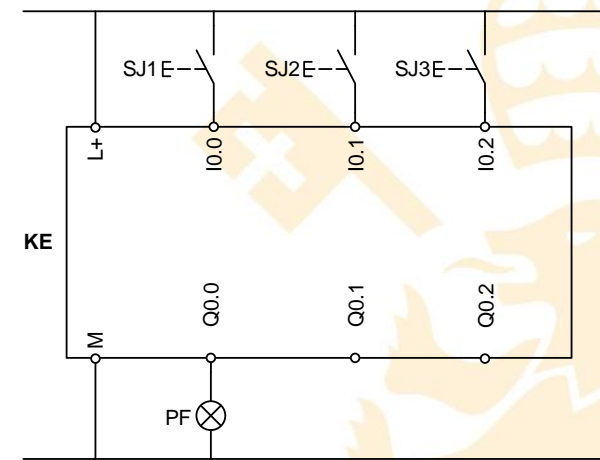
• Cableado

- Muy complicado los cambios de diseño
- Sólo en automatismos simples
- Sistemas de seguridad simples
 - Permite hacer un estudio de fiabilidad muy preciso








• Programado

- Muy fácil el cambio de programa
- Automatizaciones complejas
- La flexibilidad no siempre es una ventaja:
 - Programas más complejos: programas con más errores
 - Ciberseguridad
- Siempre hay una parte cableada



¿En qué lenguaje se programan los PLCs?

- 5 lenguajes estandarizados por la IEC: norma IEC 61131-3

Nombre	Tipo	Comentario	Denominación IEC en inglés	Siglas IEC	Denominación en Siemens
Diagrama de contactos	Gráfico	Emula un automatismo cableado	Ladder diagram	LD	KOP 
Diagrama de funciones secuenciales (Grafcet)	Gráfico	Programación de sistemas secuenciales	Sequential Function Chart	SFC	S7-GRAPH 
Texto estructurado	Texto	Emula un lenguaje de alto nivel como JAVA	Structured text	ST	SCL 
Diagrama de bloques de funciones	Gráfico	Emula los circuitos digitales	Function block diagram	FBD	FUP 
Lista de instrucciones	Texto	Emula ensamblador	Instruction List	IL	AWL 

- IL sólo se utiliza en proyectos ya desplegados
- La asignatura utiliza preferentemente diagrama de contactos, grafcet y texto estructurado

Ejemplos de programa en cada lenguaje

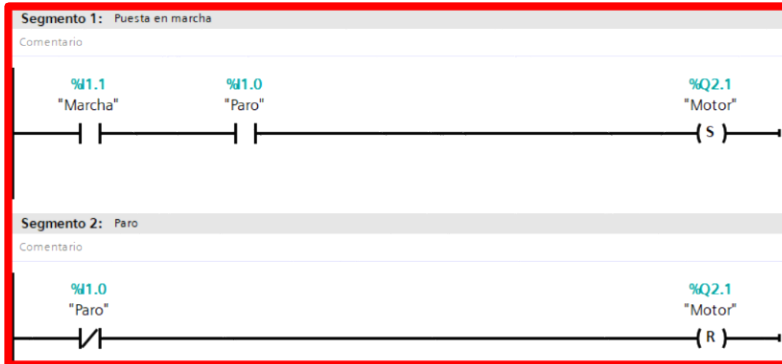
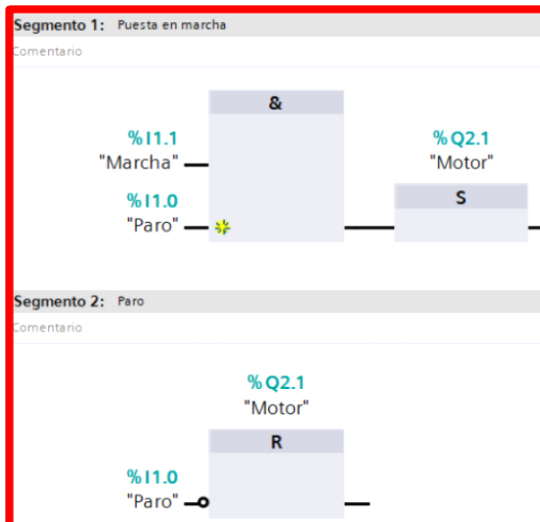
```
IF "Paro"=0 THEN
    "Motor" := 0;
ELSIF "Marcha"=1 THEN
    "Motor" := 1;
END_IF;
```

Segmento 1: Control Motor

Comentario

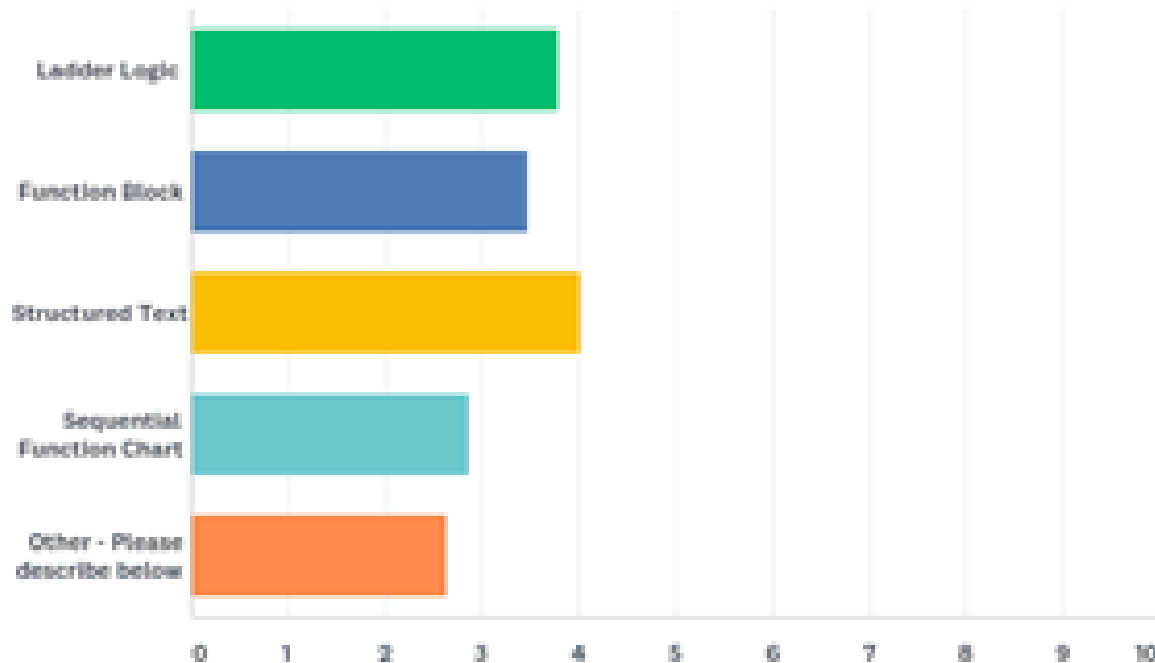
1	A	"Marcha"	%I1.1
2	A	"Paro"	%I1.0
3	S	"Motor"	%Q2.1
4	AN	"Paro"	%I1.0
5	R	"Motor"	%Q2.1

¿Qué ejemplo corresponde a cada tipo de lenguaje?



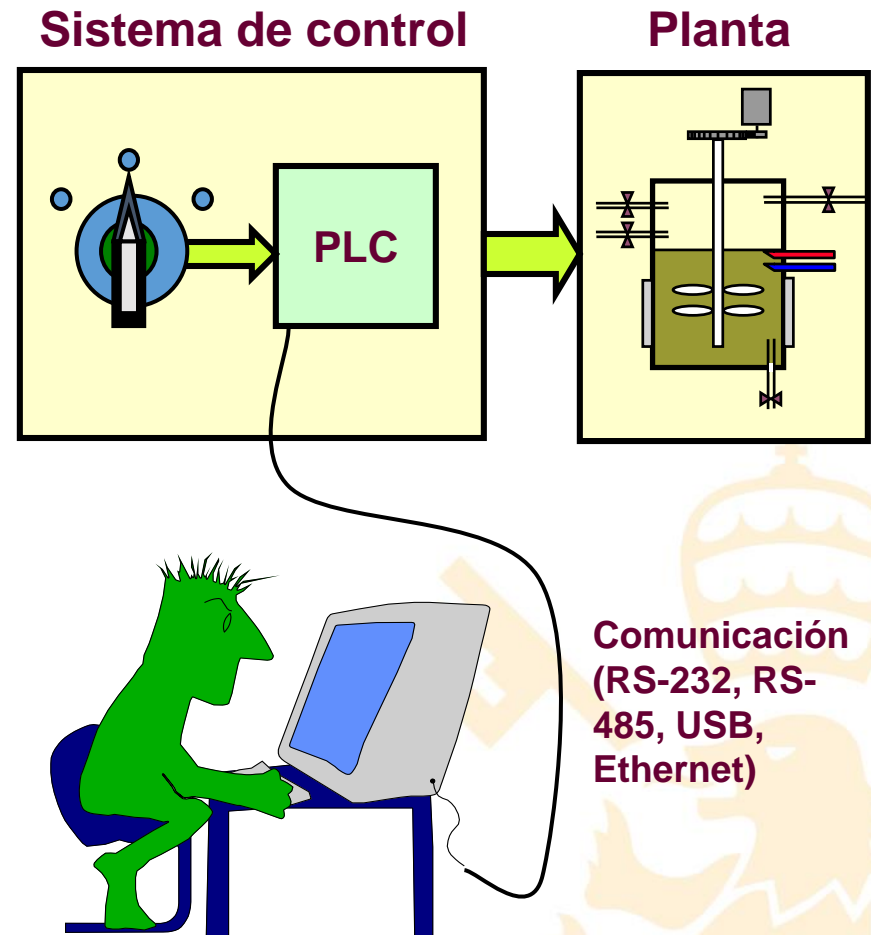
Métodos preferidos actualmente (fuente: PLCOpen)

Q4 What is your preferred method of programming a PLC?



¿Cómo se programan los PLCs?

- **Aplicación en estación de programación: ordenador o portátil**
 - Permite la edición y depuración del programa
 - Soluciones en el mercado
 - Programa del fabricante del PLC: TIA Portal (Siemens)
 - Programa multiPLC: CODESYS
- **Programación con el PLC físico**
 - Estación conectada al PLC
 - PLC conectado a la planta o a un simulador de planta
- **Programación sin el PLC físico**
 - Estación sin conexión al PLC
 - Estación con simulador de PLC
 - Estación con simulador de Planta



Pasos para realizar la programación del PLC

- Definir las funciones de control, observación y estado a implementar en el algoritmo de control
- Elegir el PLC adecuado para albergar el algoritmo de control y realizar el diseño de la parte cableada
- Definir la tabla de variables del PLC
 - Definir las variables de entrada y de salida para implementar el algoritmo de control
 - Indicar el nombre de la variable: típicamente el utilizado en el algoritmo de control o similar
 - Indicar la dirección: hace la correspondencia entre la variable y la entrada física o la salida física disponible
 - Ejemplo en Siemens para orden de marcha/paro a un motor:
 - Variables de entrada y salida en el PLC:
 - Nombre: "SJParo", "SJMarcha", "Motor"
 - Dirección: I0.0, I0.1, Q0.0
 - Entradas y salidas físicas del PLC: I0.0, I0.1, Q0.0
- Programar el algoritmo de control
 - Cada parte puede ir en un lenguaje diferente
- Realizar las pruebas

**Primero pensé
y después
programé**



¿Qué PLC se va a utilizar? Serie S7-1500 de Siemens

- **Características**

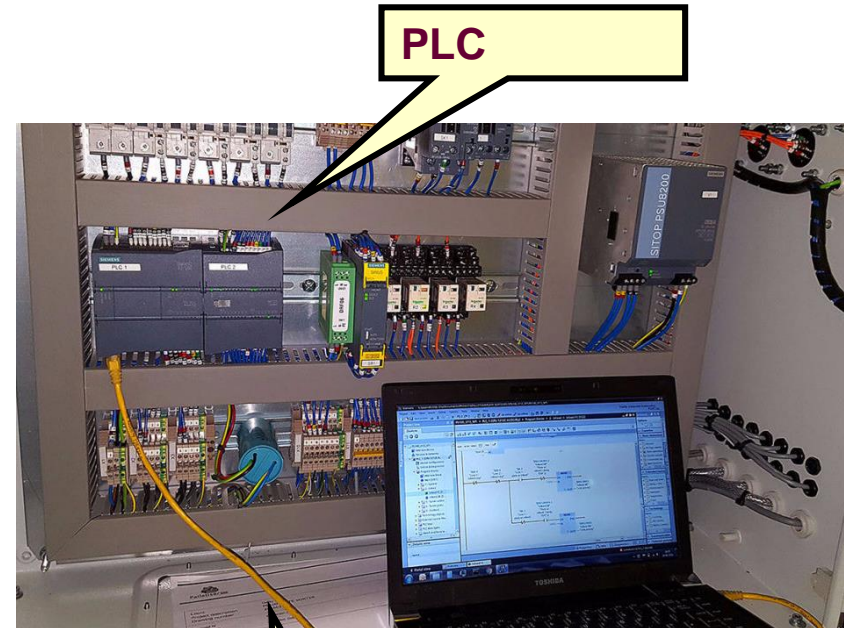
- Soportan todos los lenguajes
 - Algunas desviaciones respecto del estándar
- Herramienta de programación: TIA Portal

- **¿Qué lenguajes se van a utilizar?**

- Diagrama de contactos (mucho)
- Diagrama de bloques de funciones (poco)
- Listas de instrucciones (muy poco)
- Estructurado (medio)
- Grafcet (mucho)

- **Sólo se utilizan instrucciones IEC**

- Trasladar fácilmente lo aprendido a otros autómatas



**Conexión
con cable
Ethernet**

**Ordenador
con TIA
PORTAL**

Programación autómatas de Siemens: VARIABLES

3 conjuntos de variables lógicas directamente accesibles

• Variables de entrada:

- Direcciones: I0.0, I0.1, ..., I0.7, I1.0, ..., I1.7, I2.0, ..., I2.7, ...
- Conectadas de forma lógica a entradas físicas del PLC si las hay de igual dirección

• Variables de salida:

- Direcciones: Q0.0, Q0.1, ..., Q0.7, Q1.0, ...
- Conectadas de forma lógica a salidas físicas del PLC si las hay

• Variables internas o marcas

- Direcciones: M0.0, M0.1, ..., M0.7, M1.0, ...
- No están conectadas ni a entradas ni a salidas
- Guardan sus valores entre cada ciclo de scan

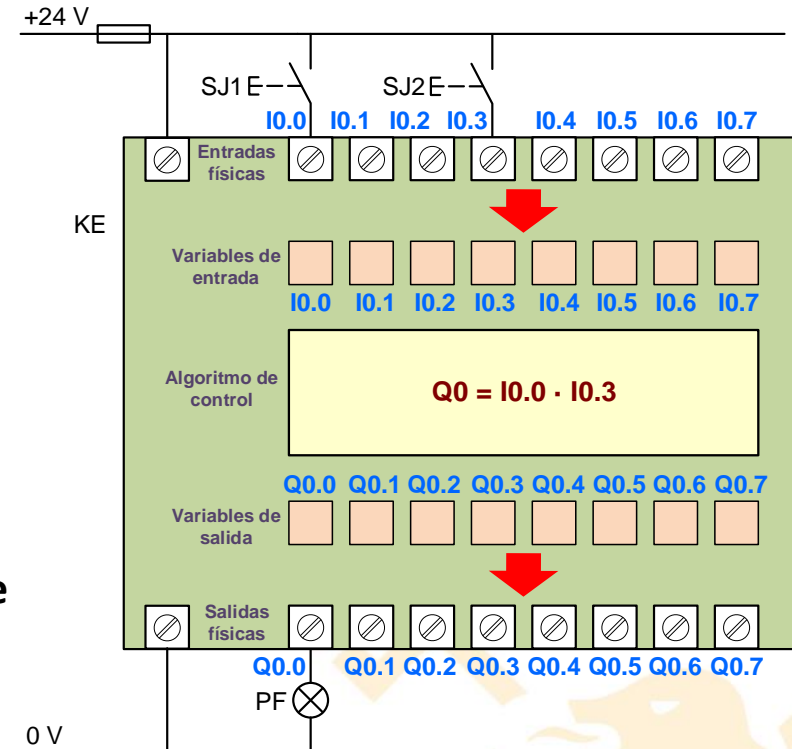
• I, Q y M son los identificadores de tipo de variable

- Según la IEC61131-3 deben llevar “%” delante: %I0.0

• Las variables se inicializan con valor 0 por defecto

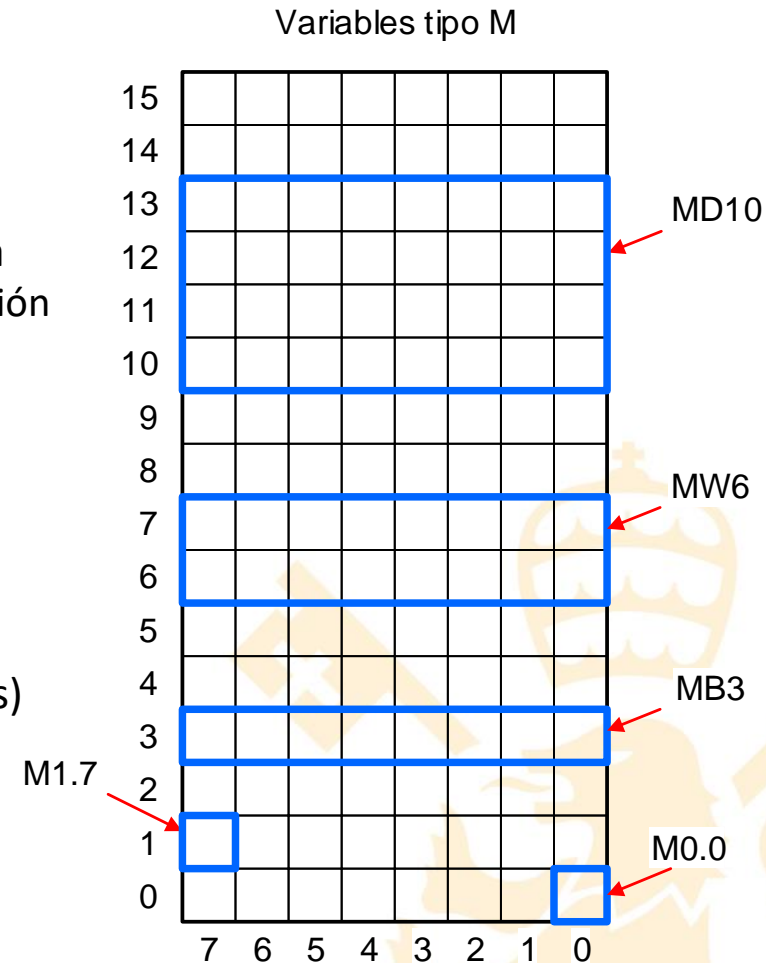
• No hay limitación para asignar nombre a las variables

- Típicamente los utilizados en el diseño del algoritmo de control



Direccionamiento de grupos de variables lógicas

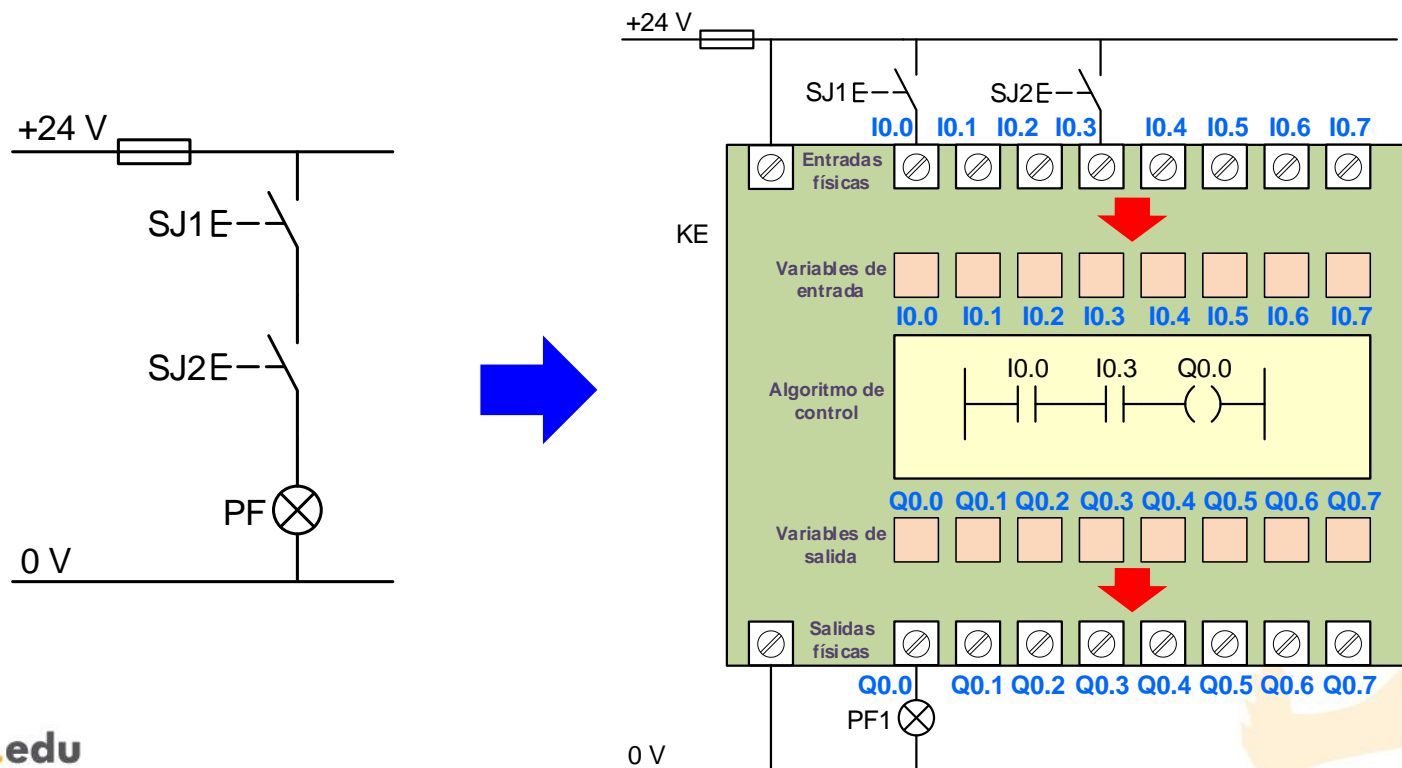
- El conjunto de las variables lógicas de un tipo tienen estructura de memoria
 - Cada variable lógica es un bit que pertenece a un byte de la memoria de ese tipo
 - Cada byte tiene una dirección
 - Explica el direccionamiento de una variable lógica
 - Tipo (I o Q o M) + dirección del byte + “.” + posición del bit
- Sobre la misma memoria se pueden definir variables que agrupan a variables lógicas
 - Variables de tipo Byte
 - Dirección: Tipo + “B” + dirección del byte
 - Variables de tipo palabra (2 bytes contiguos)
 - Tipo + “W” + dirección del byte más bajo
 - Variables de tipo doble palabra (4 bytes contiguos)
 - Tipo + “D” + dirección del byte más bajo
- Problemas con los solapamientos
 - MD10 y MD12 comparte 2 bytes
- Ojo con bytes de más de 8 bits: I0.8



Programación de PLCs con diagrama de contactos (aplicación a Siemens)

Diagrama de contactos

- Es un lenguaje gráfico para expresar las funciones lógicas del algoritmo de control basado en la normalización americana para dibujar esquemas eléctricos
- Rota 90° los esquemas IEC y utiliza otros símbolos gráficos



¿Cómo se ejecuta el ejemplo?

- **Tres elementos del lenguaje en el ejemplo**
 - Contacto normalmente abierto (NO)
 - Bobina
 - Conexión de contactos y bobina entre línea de alimentación y línea de masa
- **Emula el funcionamiento de un automatismo cableado**
 - La línea vertical izquierda representa la barra de alimentación
 - Su valor es 1 (lógico)
 - Cada contacto NO tiene una variable asociada que controla su apertura y cierre
 - Si la variable es 1, el valor a la izquierda aparece a la derecha del contacto
 - Si la variable es 0, el contacto no impone ningún valor a la derecha del contacto
 - Los 1's se propagan a través del conexionado hasta llegar a la bobina si los contactos lo permiten
 - Si llega 1 a la bobina, se escribe 1 en la variable asociada
 - Si no llega 1 a la bobina, se asume que llega 0, y se escribe 0 en la variable asociada
 - Ejemplo: Q0.0 es 1 si I0.0 e I0.3 es 1, en otro caso es 0

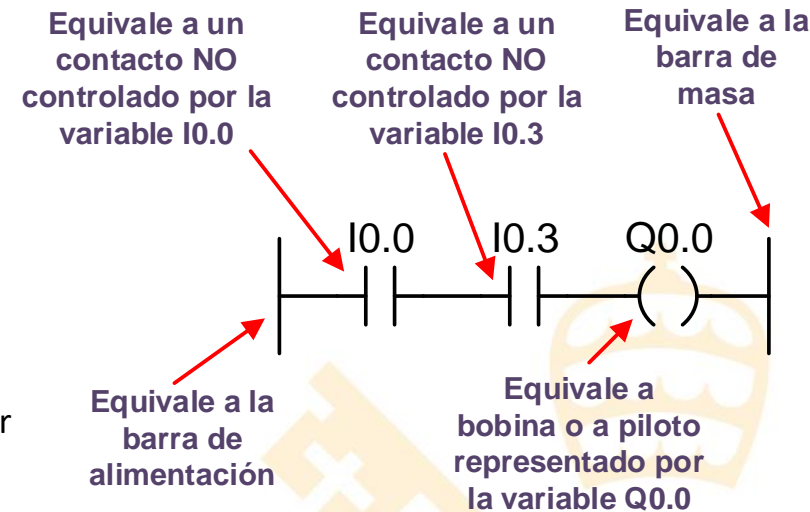
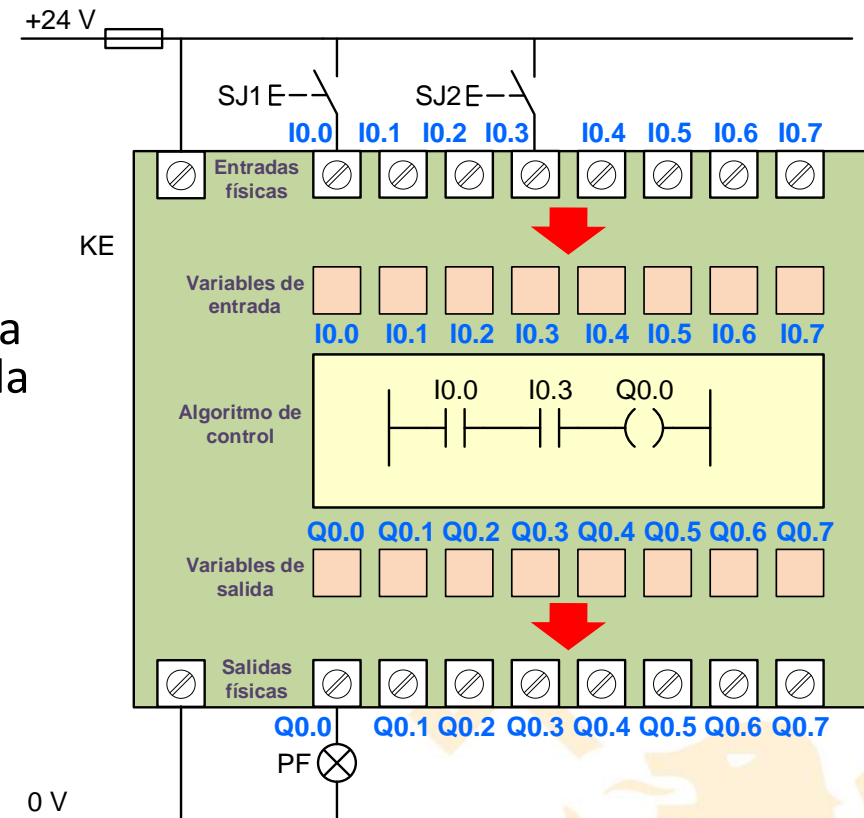


Diagrama de contactos y ciclo de scan

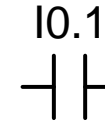
- **Ejecuta de forma cíclica**
 - Actualiza variables de entrada con entradas físicas
 - Comienza ejecución del programa por la izquierda
 - Si I0.0 tiene valor 1, el 1 lógico de la izquierda del contacto aparece en la derecha
 - I0.0 actúa como el dedo en el pulsador
 - Igual para I0.3
 - Si llega 1 lógico a la bobina, se escribe 1 en Q0.0
 - Si no llega 1 lógico a la bobina, se escribe 0 en Q0.0
 - Actualiza salidas físicas con los valores de las variables de salida



Elementos principales de los diagramas de contactos

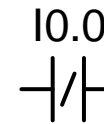
- **Contacto normalmente abierto**

- Si el valor de la variable asociada es
 - 1: el valor que hay a la izquierda del contacto se transmite a la derecha
 - 0: el valor que hay a la izquierda no se transmite a la derecha



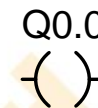
- **Contacto normalmente cerrado**

- Si el valor de la variable asociada es
 - 1: el valor que hay a la izquierda no se transmite a la derecha
 - 0: el valor que hay a la izquierda se transmite a la derecha



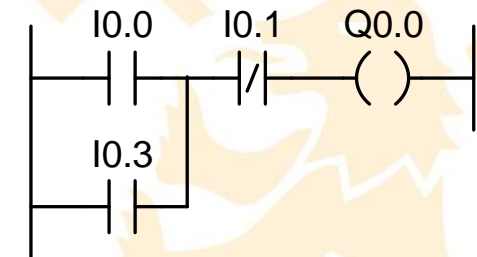
- **Bobina**

- Si a la bobina llega valor
 - 1: se escribe 1 en la variable asociada
 - 0: se escribe 0 en la variable asociada



- **Segmento (red de conexión)**

- Línea izquierda (valor 1), línea derecha, contactos, bobinas y su conexión
 - Si a una conexión no llega 1, tiene valor 0
 - La propagación siempre va de izquierda a derecha
- El programa está formado por segmentos
- Cada segmento representa una o varias funciones lógicas



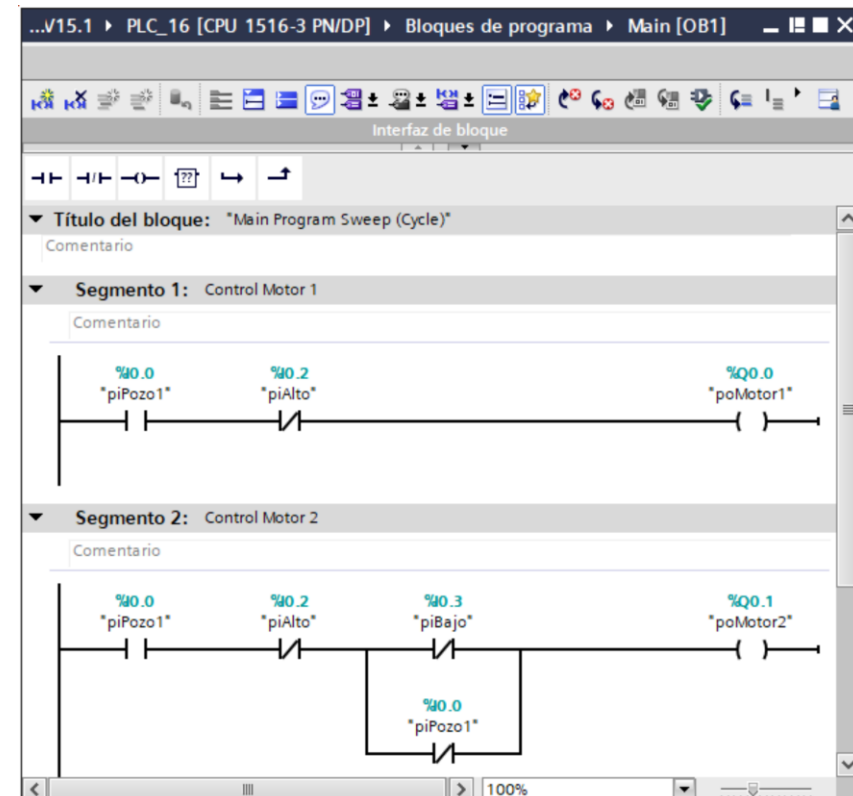
Funciona igual
 que un
 automatismo
 cableado, pero
 no tengo que
 cablear



Pasos para programar diagrama de contactos en TIA Portal

- Definir variables y funciones lógicas del algoritmo de control
- Definir tabla de variables en TIA Portal
 - Nombre de la variable
 - Dirección de la variable
 - TIA portal permite acceder a las variables por nombre o por dirección
- Programar las funciones lógicas utilizando un segmento por función
 - TIA Portal admite más de una función por segmento pero dificulta la legibilidad del programa
 - Desde la misma lógica se pueden actuar varias bobinas
 - Contactos a la izquierda, bobinas a la derecha
 - Flujo siempre de izquierda a derecha
- Ejecución: de izquierda a derecha, de arriba a abajo

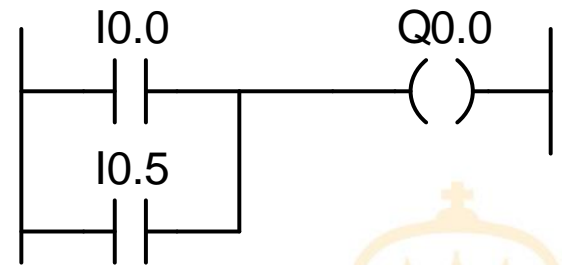
	Nombre	Tipo de datos	Dirección	Comentario
10	piPozo1	Bool	%I0.0	Sonda de nivel en el pozo 1
11	piPozo2	Bool	%I0.1	Sonda de nivel en el pozo 2
12	piAlto	Bool	%I0.2	Sonda de nivel alto en el depósito
13	piBajo	Bool	%I0.3	Sonda de nivel bajo en el depósito
14	poMotor1	Bool	%Q0.0	Contacto bomba 1
15	poMotor2	Bool	%Q0.1	Contacto bomba 2



¿He entendido el funcionamiento del PLC?

- **Requisitos:**
 - $Q0.0 = I0.0 + I0.5$
 - $Q0.7 = I0.0 \cdot I0.5$
- **Programo la función de control y cometo un error**
 - Q0.0 en vez de Q0.7 en el segundo segmento
- **¿Qué valor aparece en la salida Q0.0 si $I0.0=0$ y $I0.5=1$?**
 - 0
 - 1
 - Oscila entre 0 y 1 con frecuencia marcada por el tiempo de ejecución de las instrucciones
 - 0 y, además, el led de error del PLC se activa para indicar incoherencia en la salida Q0.0

Segmento 1: Control Q0.0

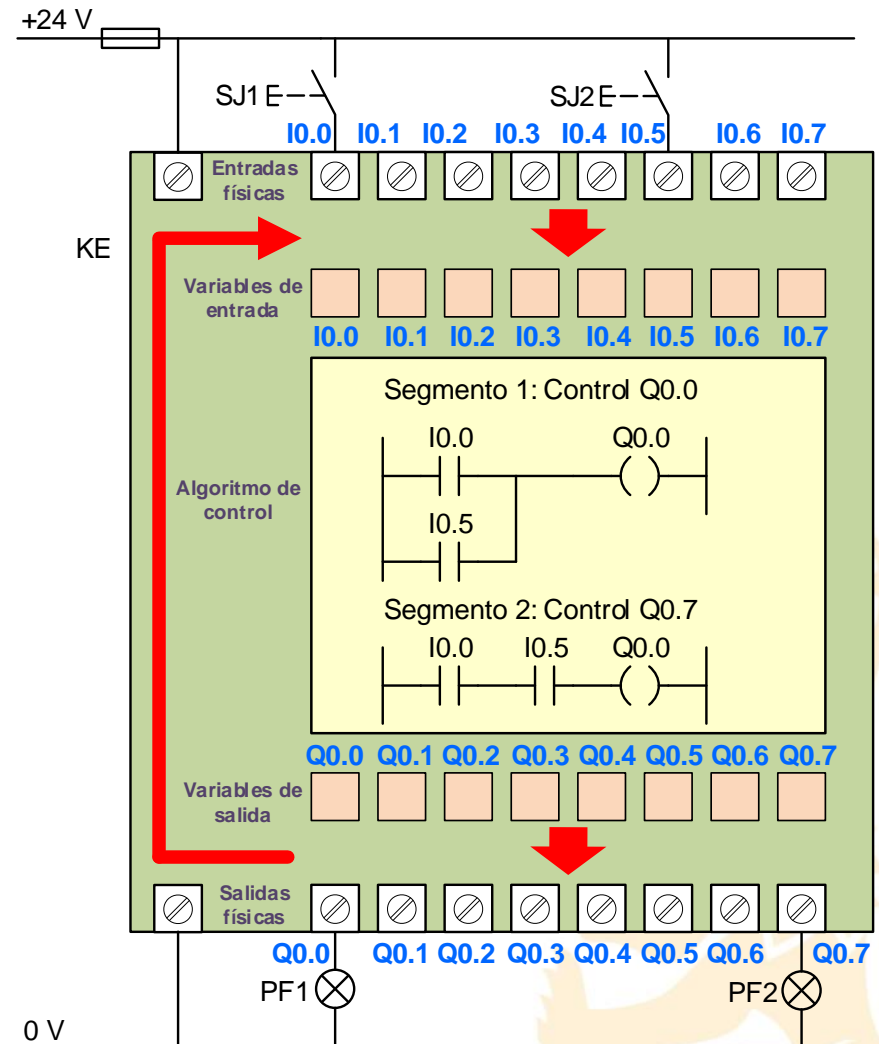


Segmento 2: Control Q0.7

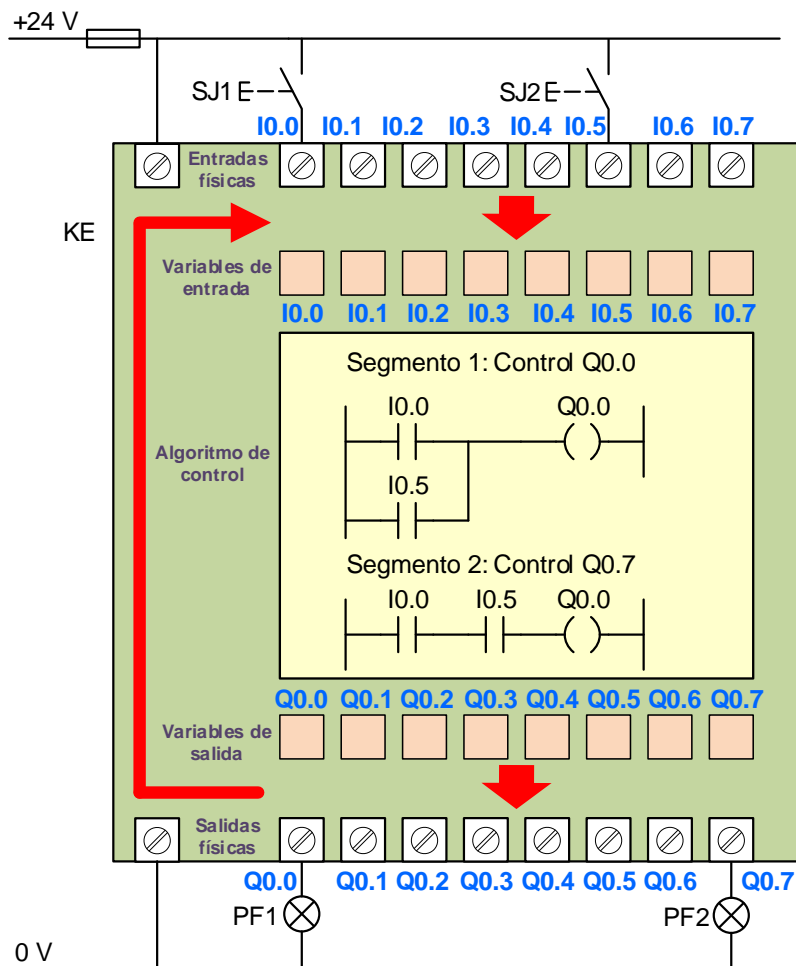


Ayuda para entender el funcionamiento

- ¿Qué valor aparece en la salida Q0.0 si I0.0=0 y I0.5=1?
- 0
- 1
- Oscila entre 0 y 1 con frecuencia marcada por el tiempo de ejecución de las instrucciones
- 0 y, además, el led de error del PLC se activa para indicar incoherencia en la salida Q0.0



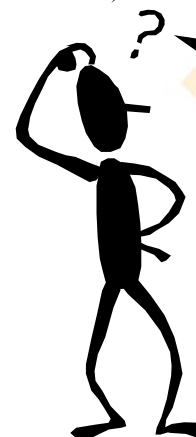
¿Ejecución en paralelo?



¿Los dos segmentos se ejecutan en paralelo?

¿Los dos contactos en paralelo se ejecutan en paralelo?

¿Cómo es el resultado de la ejecución visto desde las salidas físicas?



Bobinas con funciones adicionales

- **Bobina activar o Set**

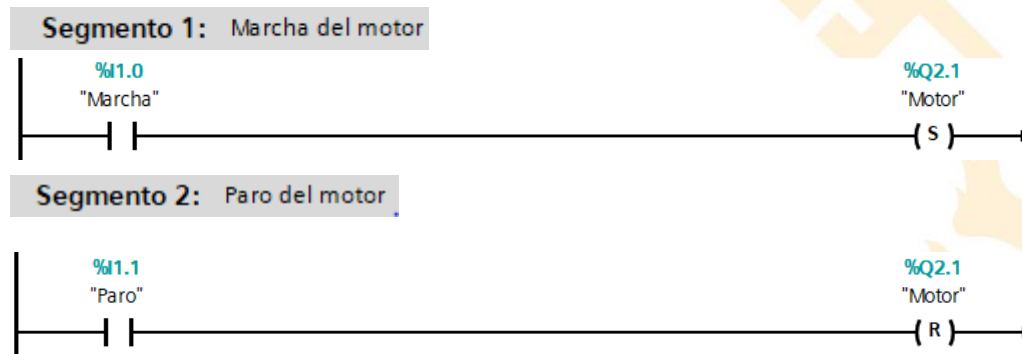
- Si a la bobina llega valor
 - 1: se escribe 1 en la variable asociada
 - 0: no se modifica el valor de la variable asociada

- **Bobina desactivar o Reset**

- Si a la bobina llega valor
 - 1: se escribe 0 en la variable asociada
 - 0: no se modifica el valor de la variable asociada

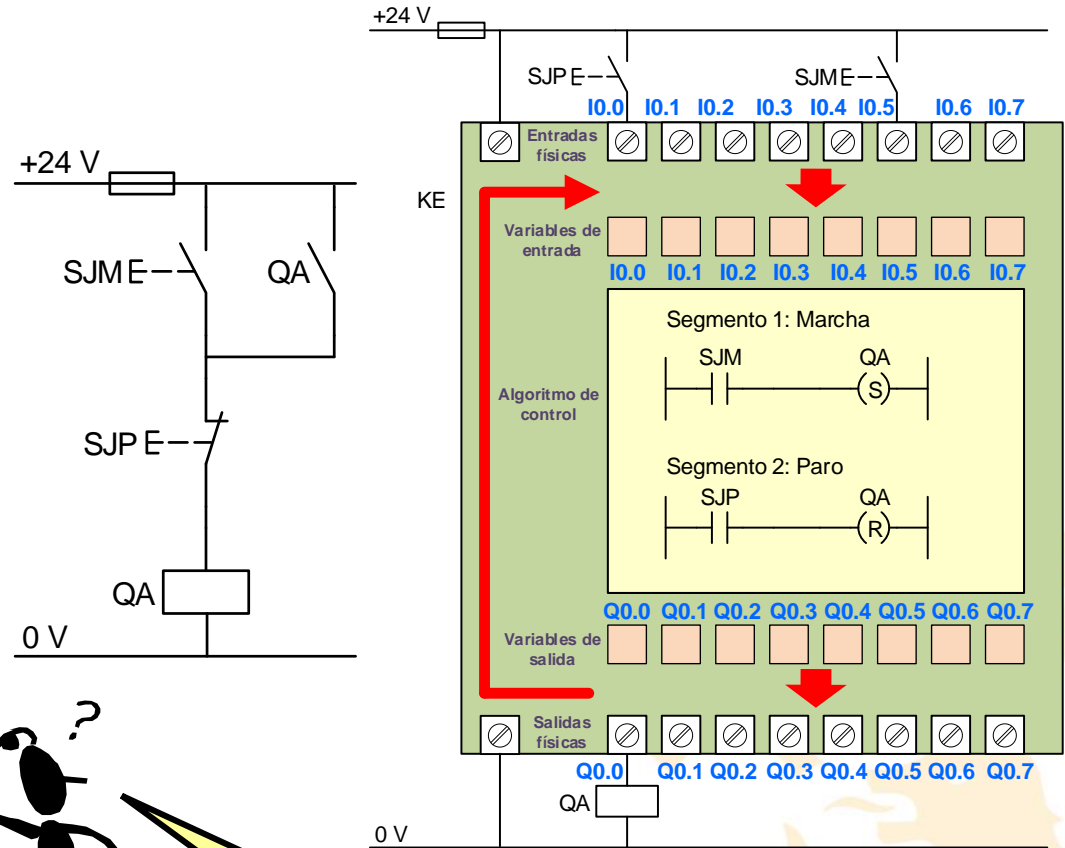
- **Uso**

- Emular los circuitos de marcha/paro
- Programación de graficets y máquinas de estado cuando no está disponible en el PLC la programación directamente mediante graficet (SFC)



¿Marcha/paro programada es igual a cableada?

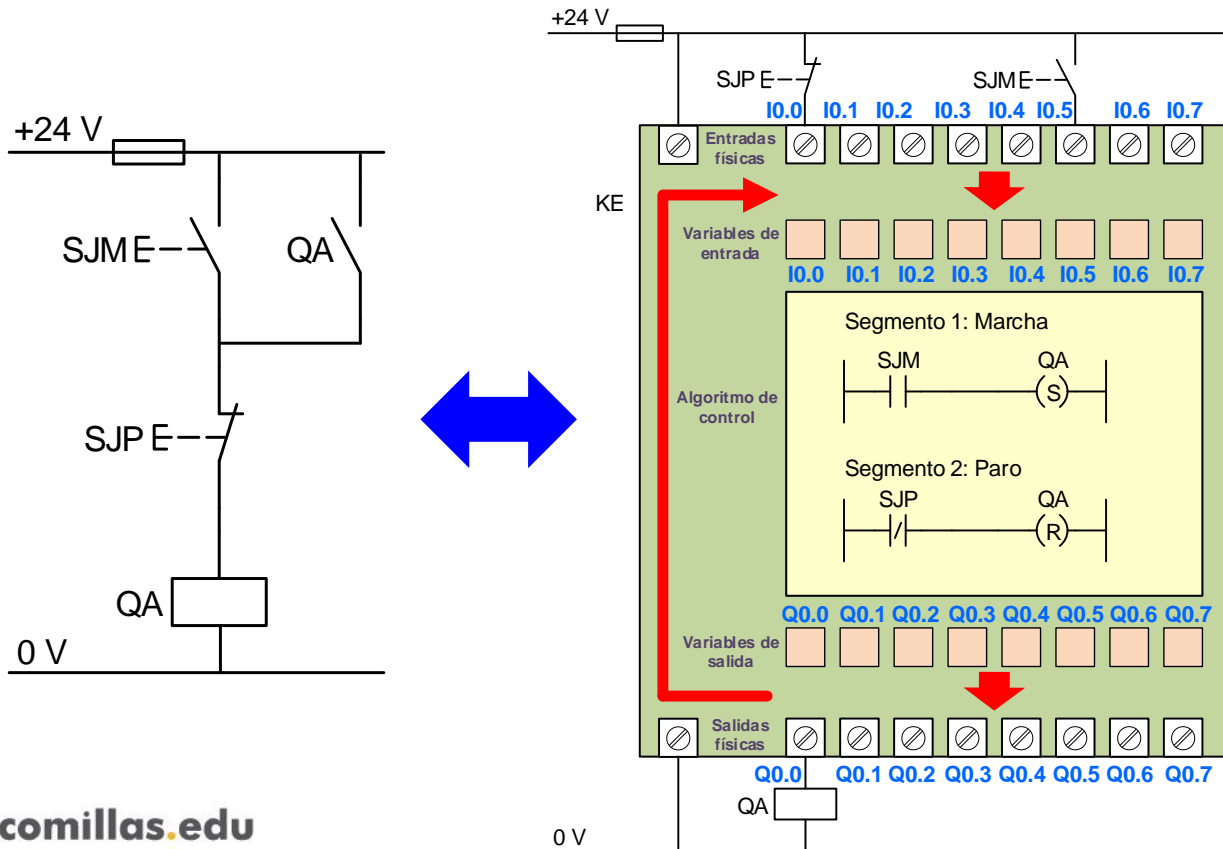
- En condiciones normales de funcionamiento son equivalentes
 - Salvo retardo del ciclo de scan
- ¿Qué ocurre si hay una rotura de cable (ejemplo SJP) con el motor en marcha?
 - Cableado: SE PARA
 - Programado: NO SE PUEDE PARAR
 - Siempre llega 0 a la variable SJP



¿El cableado es más seguro?

¿Cómo mejorar el programa?

- **Paro por nivel 0: pulsador de paro con contacto NC**
 - Pulsar paro y la falta de alimentación (ejemplo: cable roto) son equivalentes: si hay duda se para (si es la opción más segura)

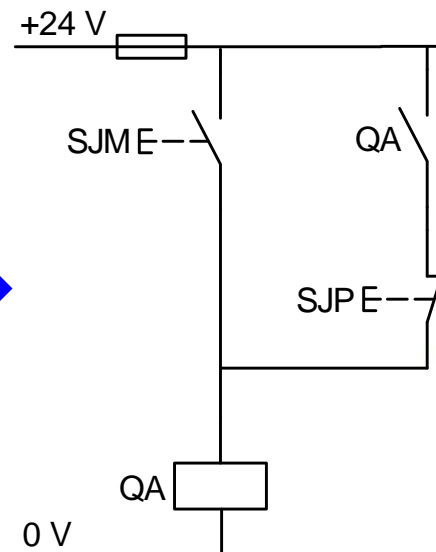
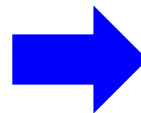
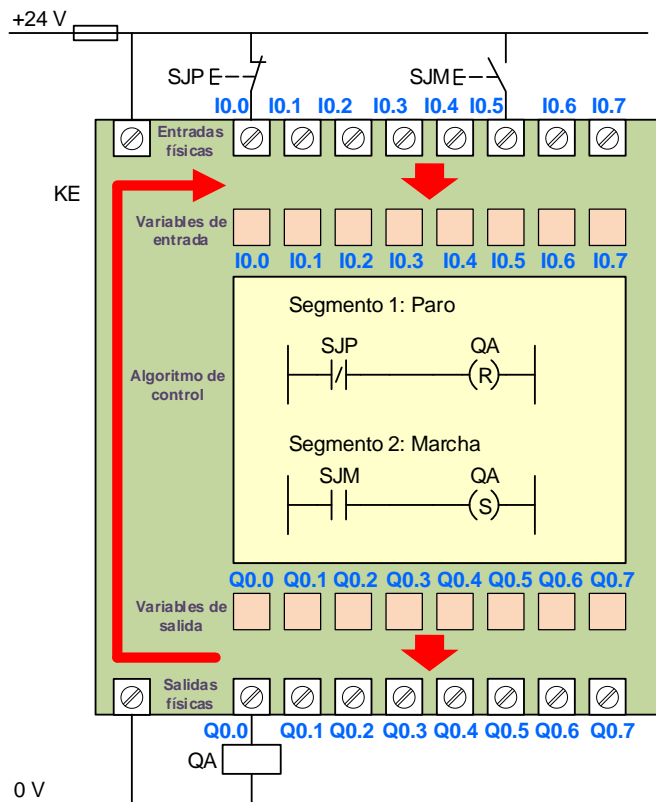


¿Quién dijo que había que simplificar siempre (a')'?



¿Qué ocurre si se cambia el orden?

- Es predominante a la marcha en vez de al paro en caso de pulsar simultáneamente marcha y paro



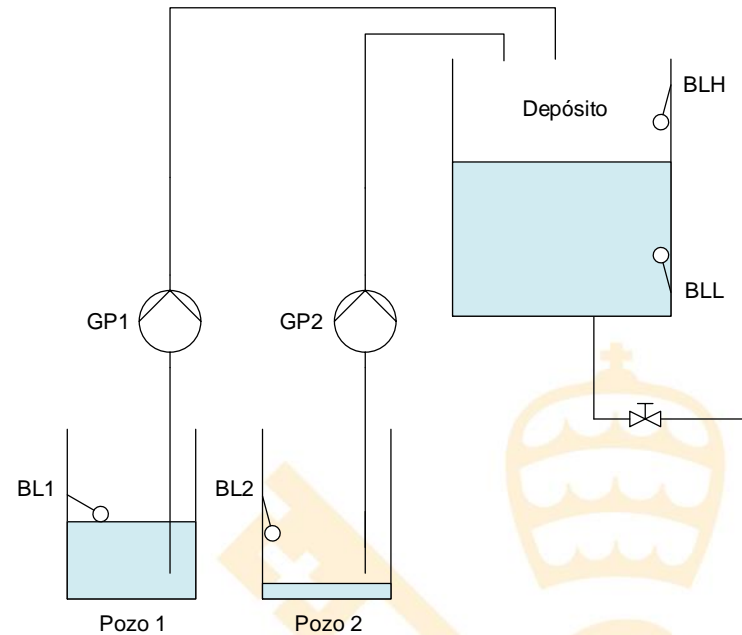
¿Cómo se soluciona si tiene que ser independiente del orden de ejecución?



Ejemplo: automatización con PLC del llenado de un depósito

• Requisitos

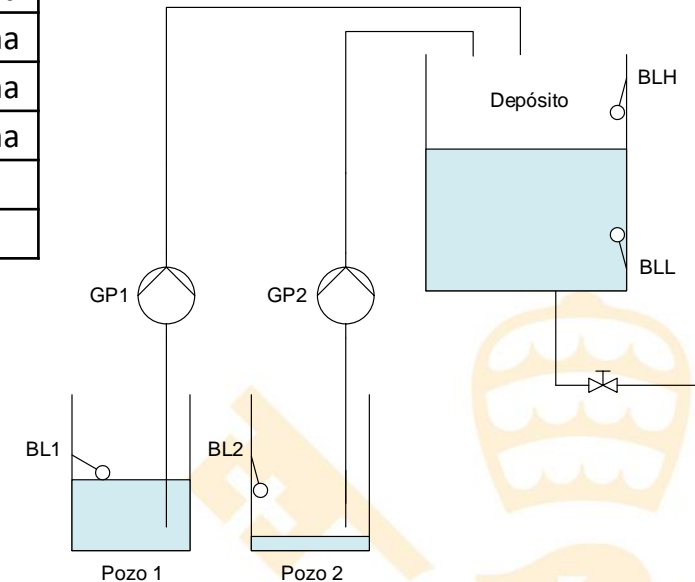
- Automatizar el sistema de llenado del depósito de la figura a partir de dos pozos. Mediante dos bombas, GP1 y GP2, se controla el nivel del depósito. Las bombas están movidas por los motores M1 y M2. El depósito tiene dos boyas, BLH y BLL, para indicar nivel alto y nivel bajo. Cada pozo tiene un sensor para detectar si hay agua (BL1 y BL2). Si no hay agua en un pozo, la correspondiente bomba no funciona.
- El sistema funciona de la siguiente manera:
 - Si el nivel del depósito supera la boya BLH, las bombas están paradas.
 - Si el nivel del depósito está entre la boya BLH y la BLL, funciona la bomba GP1, si hay agua suficiente en el pozo 1. Si no hay agua en el pozo 1 pero la hay en el 2, funciona la bomba GP2.
 - Si el nivel del depósito está por debajo de la boya BLL, se activa la bomba GP2, además de la GP1, si se puede.



Ejemplo: algoritmo de control

• Variables:

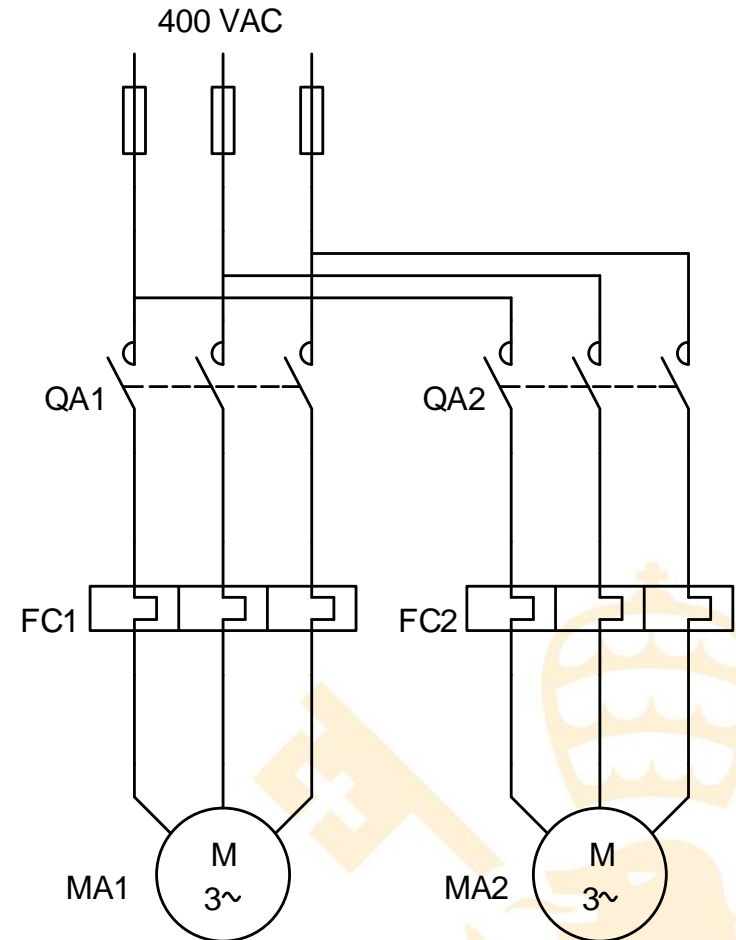
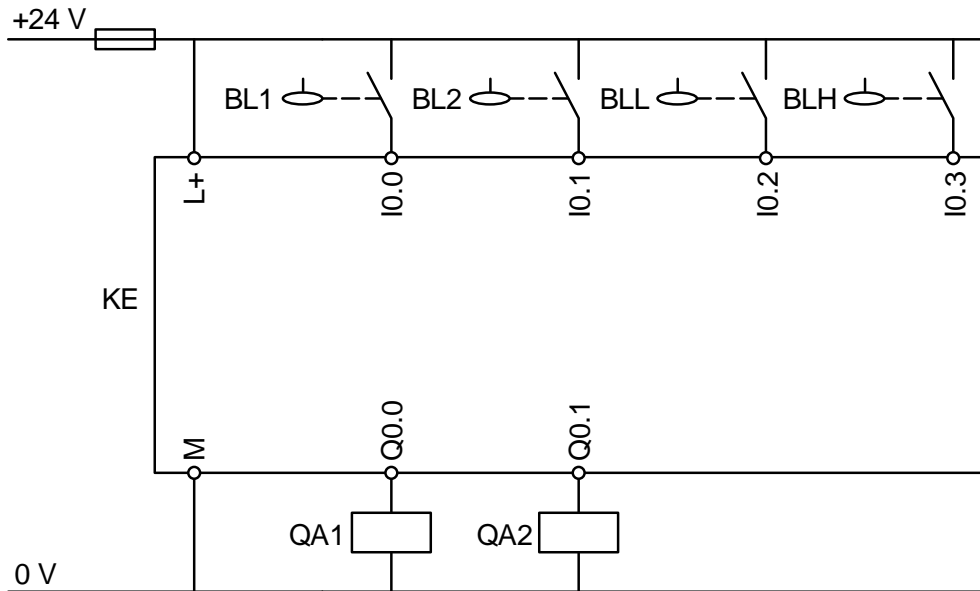
Fuente, receptor	Tipo	Variable	0	1
Sensor nivel pozo 1	Entrada	BL1	Agua por debajo	Agua por encima
Sensor nivel pozo 2	Entrada	BL2	Agua por debajo	Agua por encima
Sensor depósito nivel bajo	Entrada	BLL	Agua por debajo	Agua por encima
Sensor depósito nivel alto	Entrada	BLH	Agua por debajo	Agua por encima
Bomba 1	Salida	GP1	Parada	En marcha
Bomba 2	Salida	GP2	Parada	En marcha



• Algoritmo de control

- $GP1 = BLH'(BL1')'(BLL+BLL') = BLH' \cdot BL1$
- $GP2 = BLH'(BL2')'(BLL \cdot BL1' + BLL') = BLH' \cdot BL2(BL1' + BLL')$
- Funciones definidas según criterio de seguridad

Ejemplo: cableado



Confundir cablear el PLC con resolver mediante lógica cableada



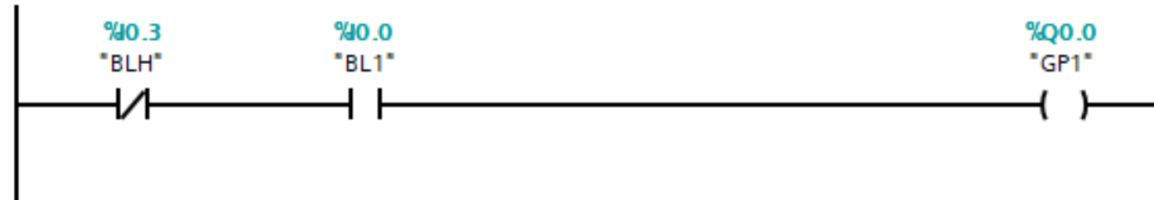
Ejemplo: tabla de variables y programación

Nombre	Tipo de datos	Dirección	Comentario
BL1	Bool	%I0.0	Nivel pozo 1: 1 con agua
BL2	Bool	%I0.1	Nivel pozo 2: 1 con agua
BLL	Bool	%I0.2	Nivel bajo depósito: 1 por encima
BLH	Bool	%I0.3	Nivel alto depósito: 1 por encima
GP1	Bool	%Q0.0	Bomba pozo 1: 1 bombear
GP2	Bool	%Q0.1	Bomba pozo 2: 1 bombear

Muy importante indicar en la tabla el significado de 1 y 0 en cada variable

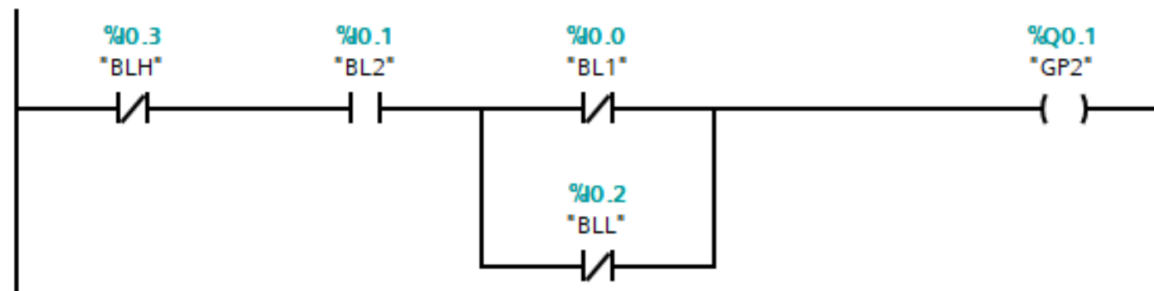
Segmento 1: Control bomba 1

Comentario



Segmento 2: Control bomba 2

Comentario

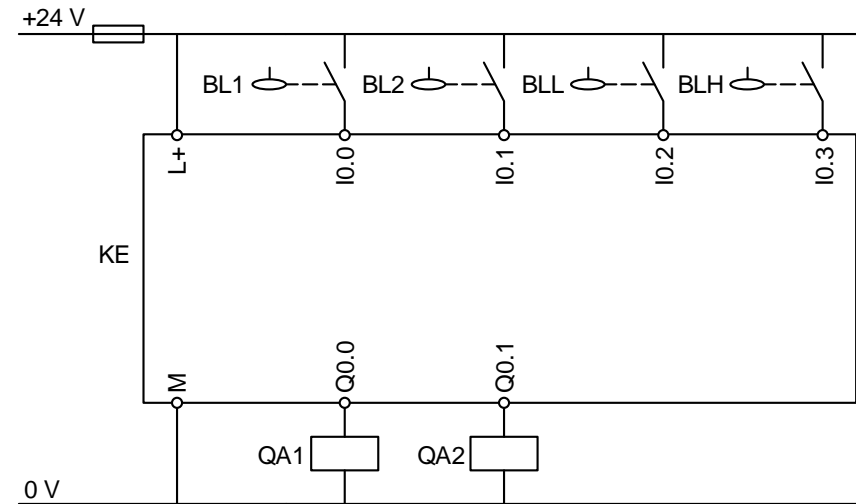


Pruebas

- Situaciones normales y situaciones anormales pero posibles

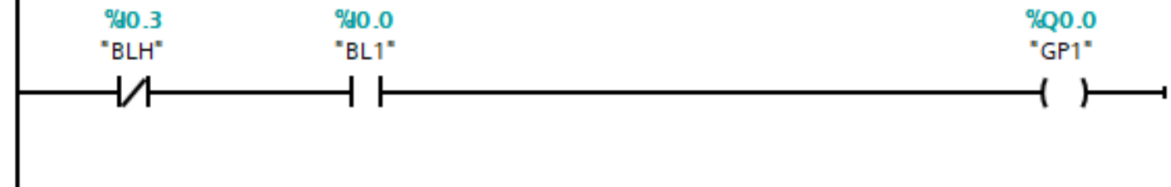
- Rotura de cables

Todo funciona bien hasta que se rompe el cable de BLH. ¡¡EL DEPÓSITO SE DESBORDA!!



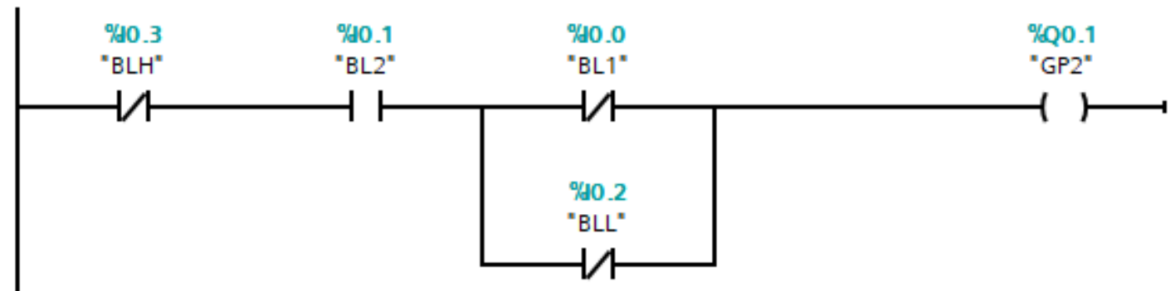
Segmento 1: Control bomba 1

Comentario



Segmento 2: Control bomba 2

Comentario



Ejemplo: lógica cableada es más tolerante a fallos

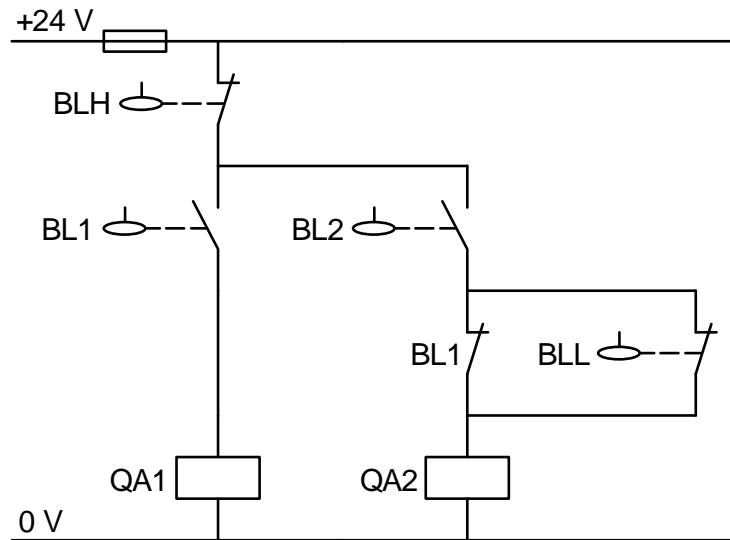
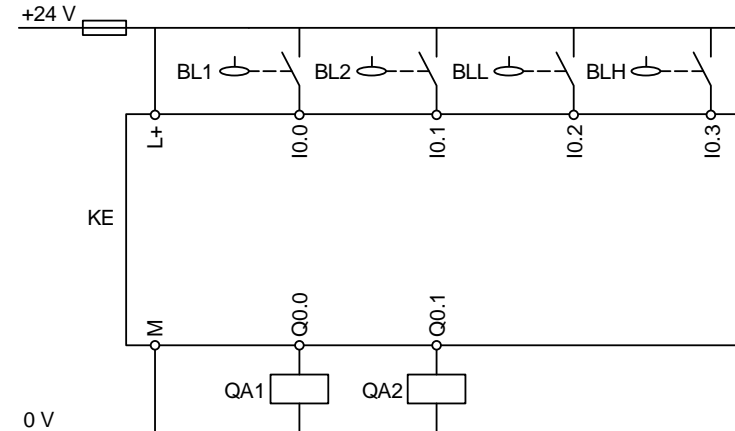
Rotura del cable del sensor BLH



Se desborda el depósito

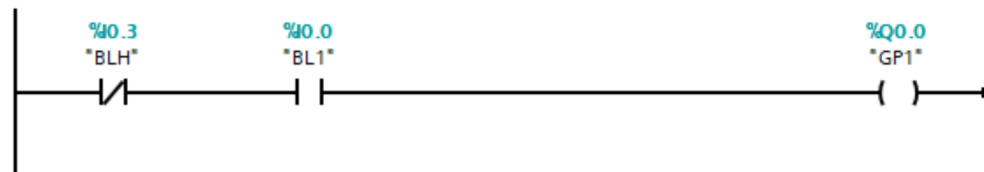


Se paran las bombas



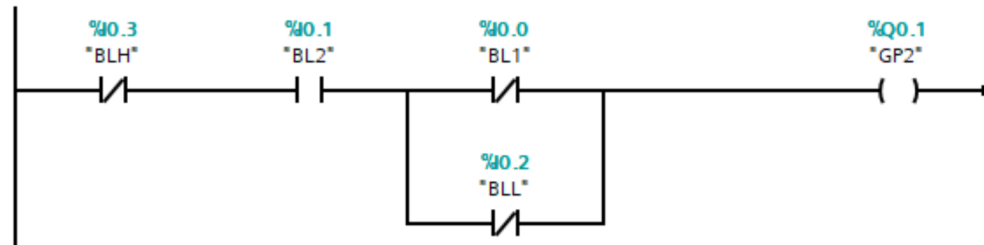
Segmento 1: Control bomba 1

Comentario



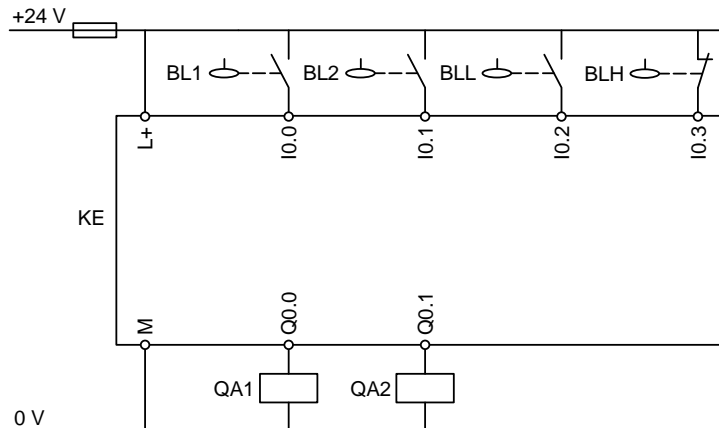
Segmento 2: Control bomba 2

Comentario



Ejemplo: mejorar la tolerancia a fallos del programado

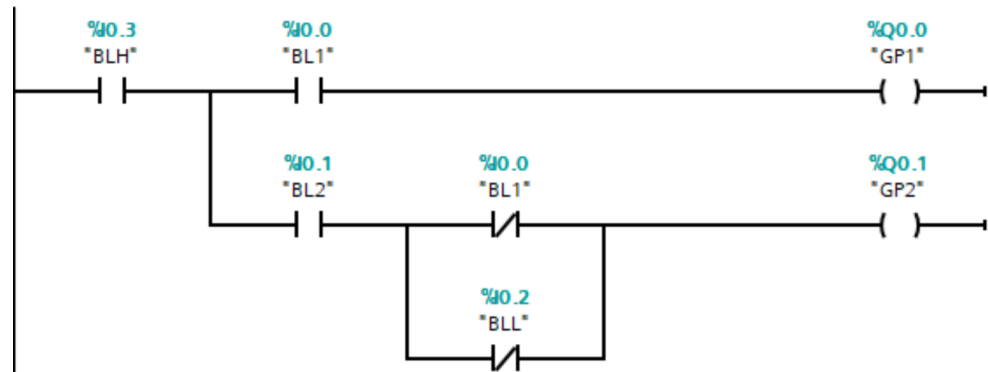
- Utilizar la misma solución que en el paro del marcha/paro programado
 - Paro y falta de alimentación (ejemplo: cable roto) son equivalentes
- Común en entradas de paro y de seguridad
 - Alimentación (1): normal, no hay orden de paro
 - Sin alimentación (0): anormal, hay orden de paro



Nombre	Tipo de datos	Dirección	Comentario
BL1	Bool	%I0.0	Nivel pozo 1: 1 con agua
BL2	Bool	%I0.1	Nivel pozo 2: 1 con agua
BLL	Bool	%I0.2	Nivel bajo depósito: 1 por encima
BLH	Bool	%I0.3	Nivel alto depósito: 0 por encima
GP1	Bool	%Q0.0	Bomba pozo 1: 1 bombear
GP2	Bool	%Q0.1	Bomba pozo 2: 1 bombear

Segmento 1: Control bombas

Comentario



Otro ejemplo de diseño inherentemente seguro

Ejercicio: control de puerta de nave industrial con PLC

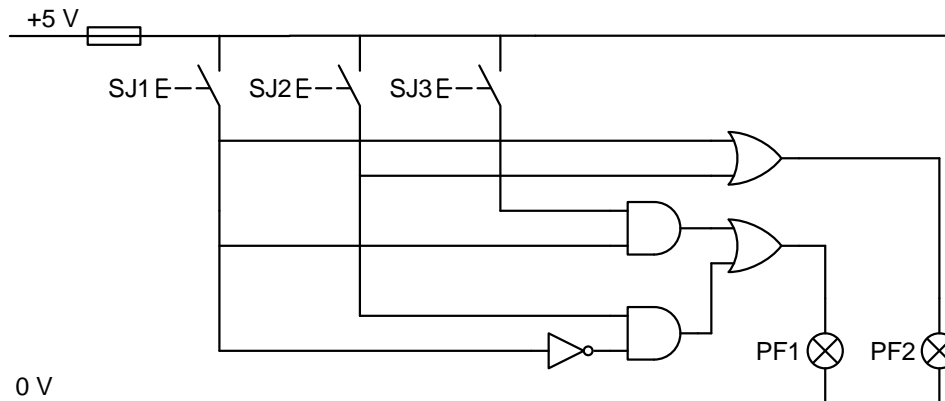
Una nave industrial tiene una puerta movida con un motor trifásico 400VAC. El operador tiene un pequeño pupitre para dar las órdenes de abrir, cerrar y parar la maniobra (pulsadores SJA, SJC y SJP). Además hay una seta de emergencia (SGE) y tres pilotos que indican puerta abierta (PFA), puerta cerrada (PFC) y puerta en movimiento (PFM). La puerta tiene dos finales de carrera para indicar puerta cerrada (BGC) y puerta abierta (BGA). El control se va a realizar mediante un PLC similar al del laboratorio. Al pulsar la seta de emergencia se corta la alimentación del motor con independencia de lo indicado por el PLC. No obstante, el PLC necesita saber que la seta de emergencia se ha pulsado y si se ha disparado la protección térmica del motor.

- Dibuje los esquemas de control y potencia.
- Diseñe el programa a implementar en el PLC utilizando diagrama de contactos

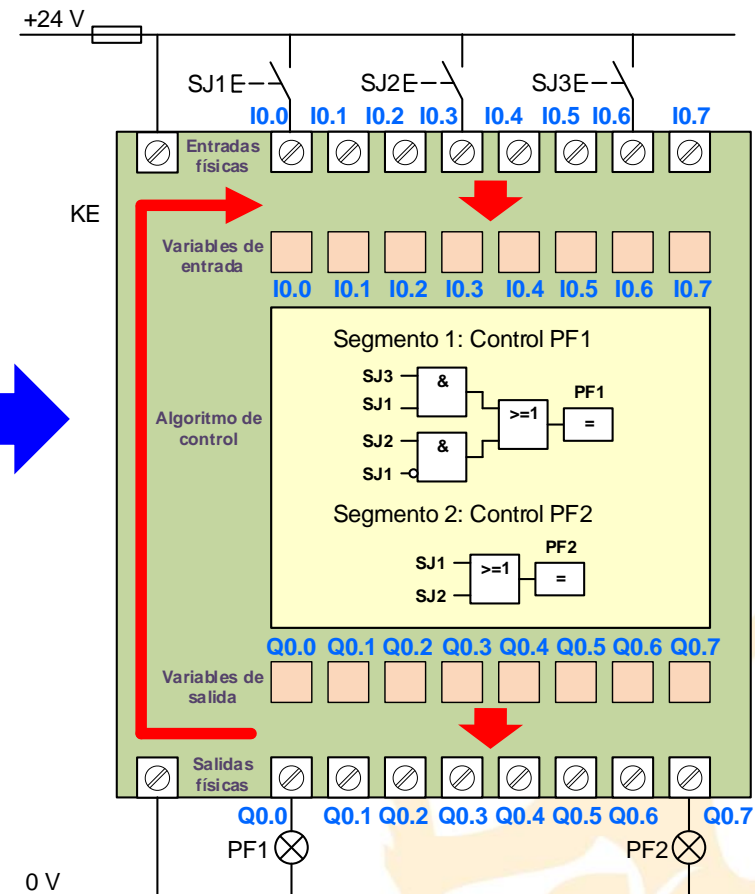
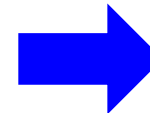
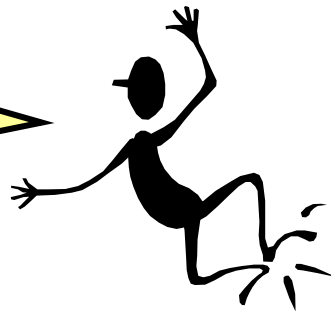
Programación de PLCs con bloques de funciones (aplicación a Siemens)

Diagrama de bloques de funciones

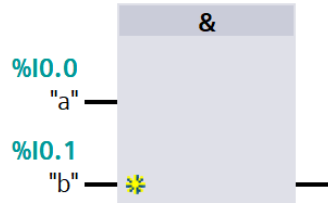
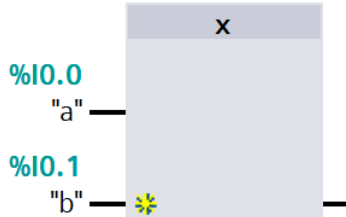
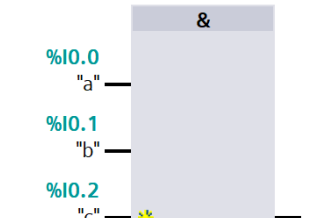
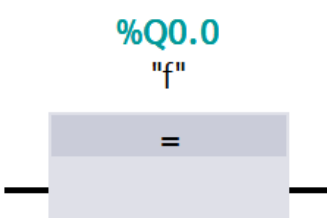
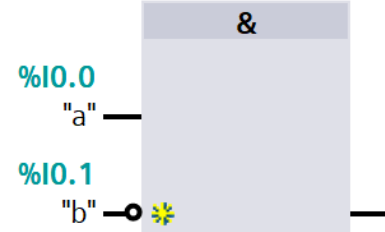
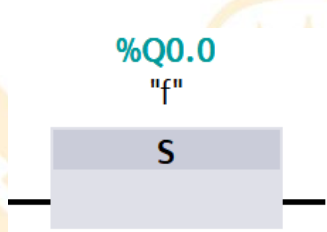
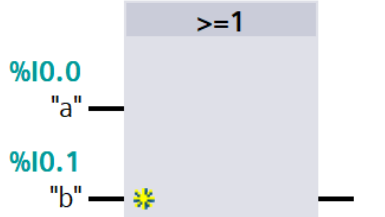
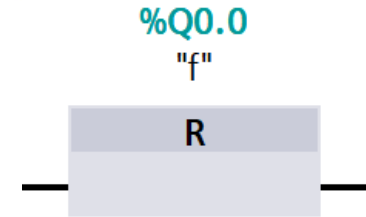
- Es un lenguaje gráfico para expresar las ecuaciones lógicas del algoritmo de control basado en la norma para dibujar circuitos lógicos (IEC 60617-12)



Se programa el PLC como si fuese diseño con puertas lógicas

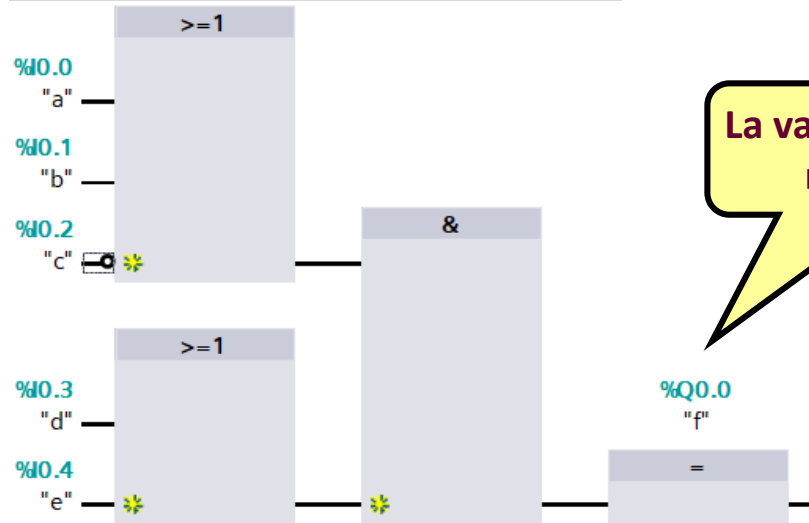


Elementos principales de la programación en FBD (FUP)

Y lógica o AND (2 entradas)		O-exclusiva o XOR (igual para más entradas)	
Y lógica de 3 entradas (igual para más entradas)		Asignar valor a variable: siempre encima	
Entrada negada		Asignar 1 a variable si entrada está a 1 (SET)	
O lógica (OR) (igual para más entradas)		Asignar 0 a variable si entrada está a 1 (RESET)	

Ejemplo de programa en FBD (FUP)

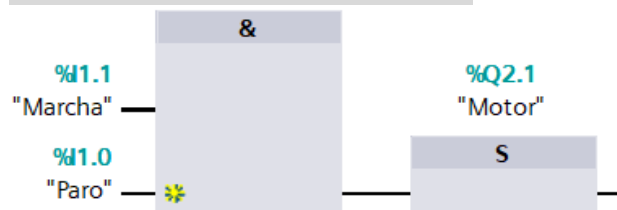
Segmento 1: Función lógica $f=(a+b+c')(d+e)$



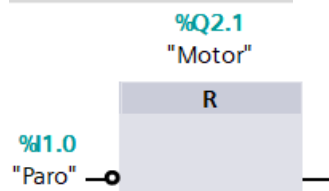
La variable va encima:
no a la salida

¿Los segmentos se
ejecutan en paralelo?

Segmento 2: Puesta en marcha



Segmento 3: Paro



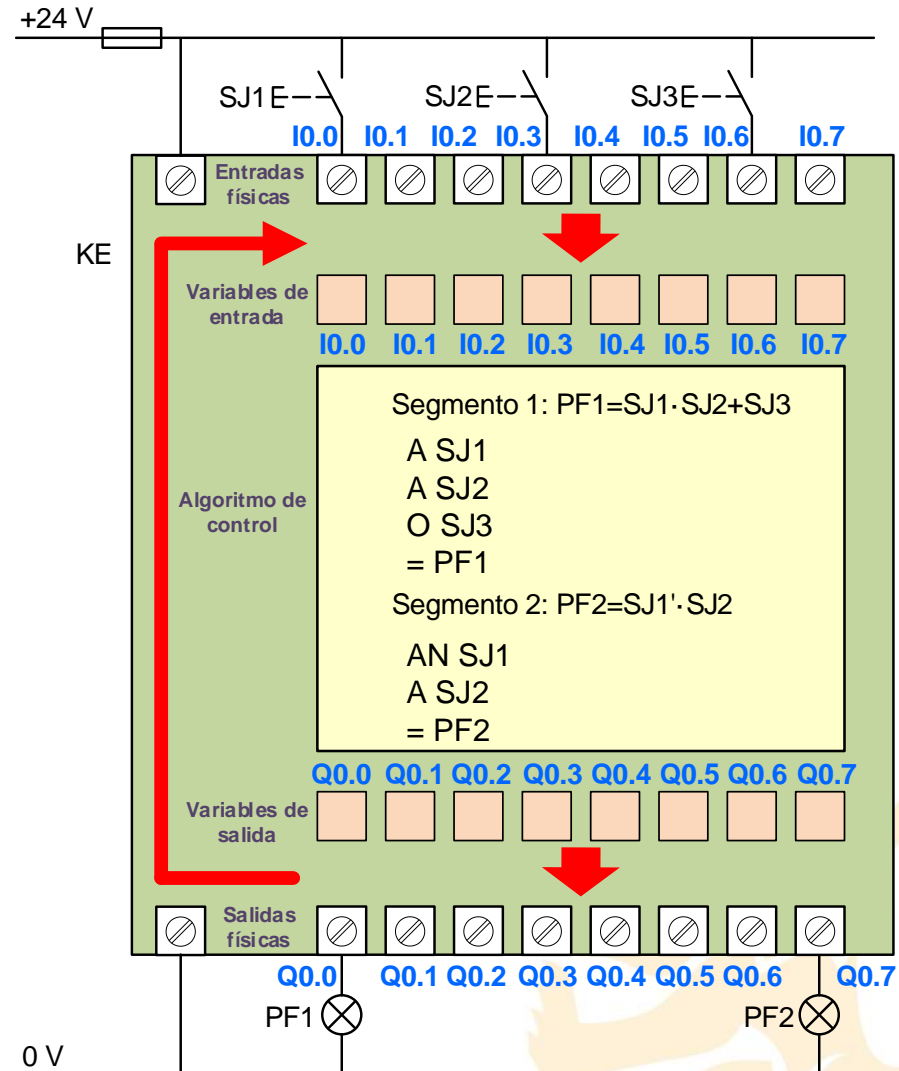
Ejercicio: programa con bloques de función

Programe según diagrama de bloques de función la siguiente función lógica: $f = ((a + b)c' + (b' + d)c)e'$

Programación de PLCs con lista de instrucciones (aplicación a Siemens)

Ejemplo de programación en lista de instrucciones

- **Similar a programación en ensamblador**
 - Una instrucción por línea
- **Formato de la instrucción**
 - Código de la operación
 - Lógicas: A, O, X
 - Asignación: =, S, R
 - Modificador: N
 - Puede no aparecer
 - Primer operando (implícito)
 - Se denomina RLO
 - Segundo operando (explícito): nombre de la variable o una constante
 - Puede no aparecer
 - Si aparece el modificador, el valor del operando que utiliza la instrucción es el negado



Ejecución de un programa en lista de instrucciones

- Antes de ejecutar la primera instrucción del programa, el RLO toma el valor del operando explícito o su negado si aparece el modificador
 - Igual ocurre antes de ejecutar una instrucción lógica después de una instrucción de asignación
 - Esta situación se denomina inicio de secuencia lógica
- Mecánica para ejecutar una instrucción que no es la primera de una secuencia lógica
 - Sin modificador
 - $RLO = Operación(RLO, operando)$
 - Con modificador
 - $RLO = Operación(RLO, operando')$

Programa

Ejecución

O	I0.0	RLO <= I0.0
ON	I0.1	RLO <= RLO + I0.1'
=	Q0.0	Q0.0 <= RLO
=	Q0.1	Q0.1 <= RLO
AN	I0.0	RLO <= I0.0'
O	I0.2	RLO <= RLO + I0.2
A	I0.1	RLO <= RLO · I0.1
=	Q0.2	Q0.2 <= RLO

Ecuaciones lógicas:

$$Q0.0 = I0.0 + I0.1'$$

$$Q0.1 = I0.0 + I0.1'$$

$$Q0.2 = (I0.0' + I0.2) \cdot I0.1$$

Ejemplo de programación

$f = ac + b \cdot c' + b'c$

- Programa incorrecto

A	a
A	c
O	b
AN	c
ON	b
A	c
=	f

- Expresión lógica programada

- $f = ((ac + b)c' + b')c$

- Programación con variables temporales

A	a
A	c
=	M0.0
A	b
AN	c
O	M0.0
=	M0.0
AN	b
A	c
O	M0.0
=	f

- Programación utilizando la salida como variable temporal

A	a
A	c
=	f
A	b
AN	c
O	f
=	f
AN	b
A	c
O	f
=	f

Programación con paréntesis

- Ejemplo: $f = a + b(c' + de) + d'$

- Ejemplo: $f = a(b+c)$ y $f = a(b+c)'$

```

A a
A (
    O b
    O c
)
= f
  
```

```

A a
AN (
    O b
    O c
)
= f
  
```

Funcionamiento

- Al encontrar una instrucción con paréntesis se posterga su ejecución hasta que aparezca el paréntesis derecho
 - Al aparecer la instrucción con paréntesis, el RLO actual y el código de la instrucción (incluido modificador) se introduce en una pila. Se inicia nueva secuencia lógica.
 - Al aparecer el paréntesis derecho se saca de la pila la última pareja instrucción (incluido modificador) + RLO guardada, y se ejecuta la instrucción con el RLO actual y el RLO sacado de la pila
- Se recomienda su uso frente a variables intermedias

```

A a
O (
    A b
    A (
        ON c
        O (
            A d
            A e
        )
    )
)
ON d
= f
  
```

El TIA Portal no admite el indentado

Segmentos y RLO

- Nuevo segmento no significa inicio de secuencia lógica
- Si la última instrucción de un segmento es una operación lógica es muy probable un error
 - Es un error muy difícil de detectar

Segmento 1: Lógica piloto H1			
Comentario			
1	O	"SM1 "	
2	ON	"SM2 "	

Segmento 2: Lógica piloto H2			
Comentario			
1	A	"SP1 "	
2	A	"SP2 "	
3	=	"H2 "	

$$H2 = (SM1 + SM2') SP1 \cdot SP2$$

Otras instrucciones

• S (Set)

- Si el RLO actual es 1, el operando toma valor 1.
- Si el RLO actual es 0, el operando no cambia de valor.

• R (Reset)

- Si el RLO actual es 1, el operando toma valor 0.
- Si el RLO actual es 0, el operando no cambia de valor.

• Ejemplo de marcha/paro

• SET

- Pone a 1 el RLO

Segmento 1: Marcha			
Comentario			
1	O	"SM1"	
2	O	"SM2"	
3	S	"H1"	

Segmento 2: Parada			
Comentario			
1	O	"SP1"	
2	O	"SP2"	
3	R	"H1"	

Ejercicio: programa en lista de instrucciones

Programar en lista de instrucciones la función $f = a' \cdot (c+b)' (d'c+dc')$

Programación del algoritmo de control

- **Combinacional**

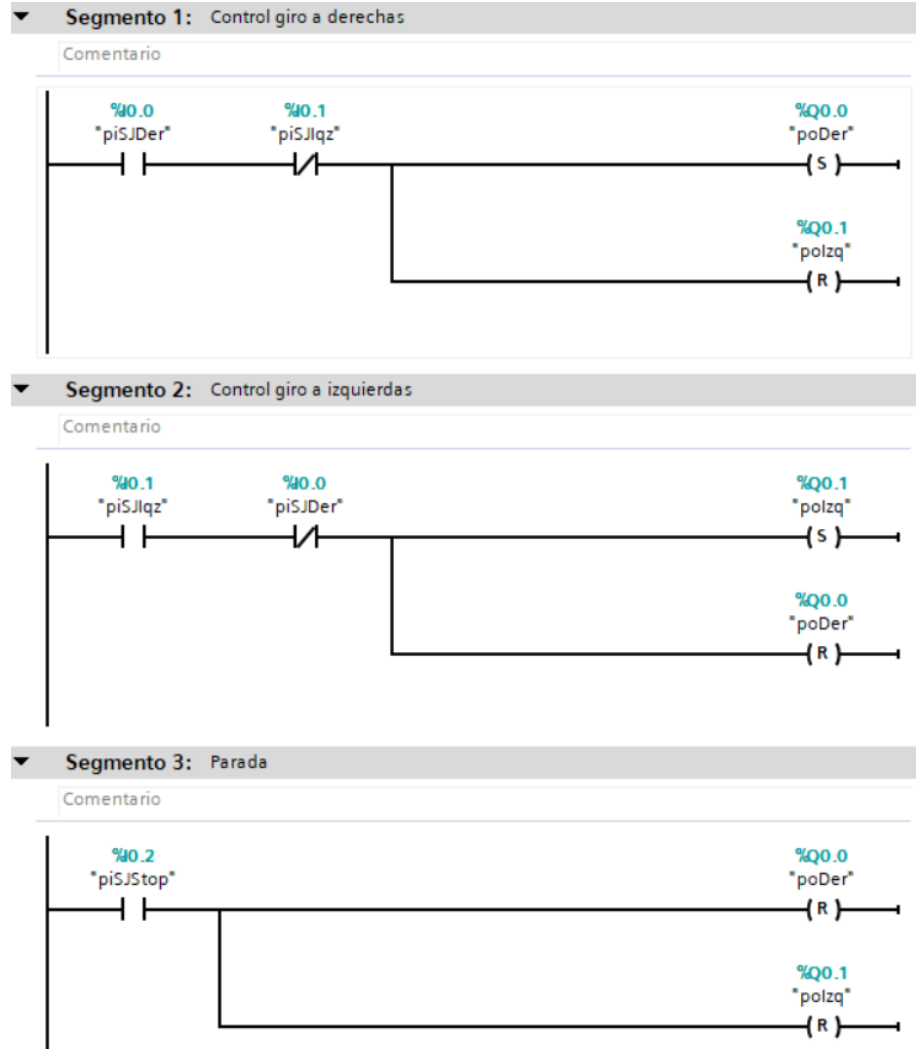
- Una función lógica o asimilable por segmento

- **Secuencial**

- Es una implementación síncrona
 - Tres opciones principales:
 - Reproducir los esquemas simples de marcha/paro cableado
 - Lenguajes LD, FBD, IL, ST
 - Implementar máquina de estados genéricas
 - Lenguajes LD, FBD, IL
 - Un bit por estado posible (Facilita la depuración de programas)
 - Cálculo del siguiente estado mediante el método de la copia y utilizando instrucciones de set y reset
 - Aunque en algunos casos no sea necesaria la copia, conviene hacerla para evitar errores en modificaciones futuras
 - Lenguaje ST
 - 1 bit por estado posible
 - Cálculo del siguiente estado mediante la instrucción de CASE
 - Grafcet: Lenguaje SFC (más adelante)
 - Fácil diseño y depuración
 - Otras opciones para casos sencillos: contador, contador con temporizador

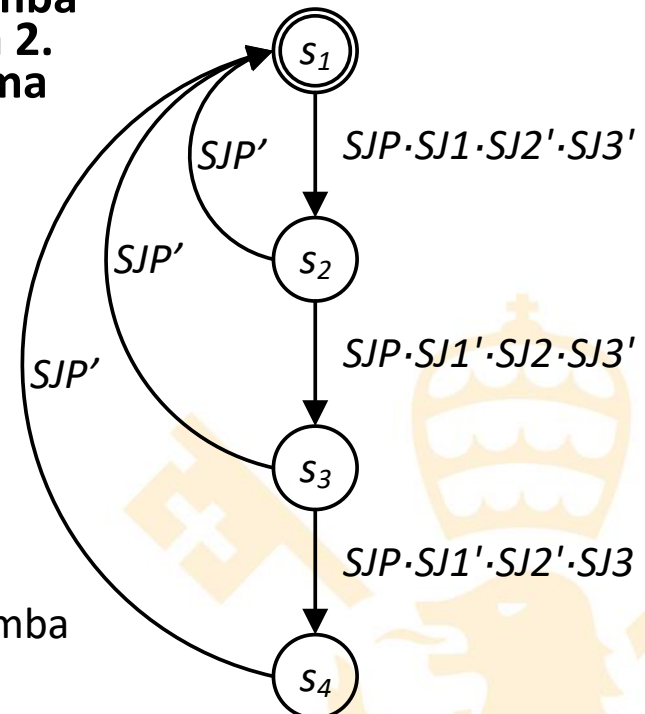
Ejemplo esquema simple de marcha/paro

- Pulsador SJDer para dar orden de giro a derechas
- Pulsador SJIzq para dar orden de giro a izquierdas
- Pulsador SJStop para dar orden de parada



Arranque de tres bombas en cascada mediante PLC

- El arranque de tres bombas (GP1, GP2 y GP3) se realiza mediante 3 pulsadores: SJ1, SJ2 y SJ3. El paro se realiza mediante un pulsador común (SJP). La bomba 2 sólo se puede arrancar si está arrancada la bomba 1. La bomba 3 sólo se puede arrancar si está arrancada la bomba 2. Si se pulsa más de un pulsador de arranque el sistema no modifica el estado de las bombas
- **Primer paso: identificar variables del sistema de control**
 - Entradas: SJ1, SJ2, SJ3, SJP
 - Salidas: GP1, GP2, GP3
- **Segundo paso: identificar tipo de control**
 - Secuencial: Al pulsar SJ2 sin pulsar el resto, unas veces arranca la bomba 2 y otras veces no.
- **Tercer paso: diseño de la máquina de estados**
 - Estados posibles: $S = \{s_1, s_2, s_3, s_4\}$ s_1 : Parado; s_2 : bomba 1; s_3 : bombas 1 y 2; s_4 : todas las bombas
- **Cuarto paso: funciones de control (y observación)**
 - $GP1 = s_2 + s_3 + s_4$; $GP2 = s_3 + s_4$; $GP3 = s_4$;



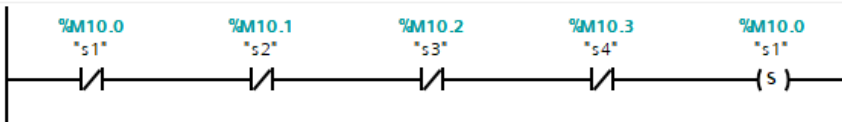
Programación arranque tres bombas: variables

- **Tabla de variables de entrada, salida e internas**
 - Las variables “copia” permiten que la evolución del diagrama de estados siempre sea la misma

SJP	Bool	%I0.0	Parada (0: orden)
SJ1	Bool	%I0.1	Arranque bomba 1 (1: orden)
SJ2	Bool	%I0.2	Arranque bomba 2 (1: orden)
SJ3	Bool	%I0.3	Arranque bomba 3 (1: orden)
GP1	Bool	%Q0.0	Bomba 1
GP2	Bool	%Q0.1	Bomba 2
GP3	Bool	%Q0.2	Bomba 3
s1	Bool	%M10.0	Estado 1: todas paradas
s2	Bool	%M10.1	Estado 2: bomba 1
s3	Bool	%M10.2	Estado 3: bomba 2
s4	Bool	%M10.3	Estado 4: bomba 3
cs1	Bool	%M20.0	Copia estado 1
cs2	Bool	%M20.1	Copia estado 2
cs3	Bool	%M20.2	Copia estado 3
cs4	Bool	%M20.3	Copia estado 4

Programación arranque tres bombas: máquina de estados

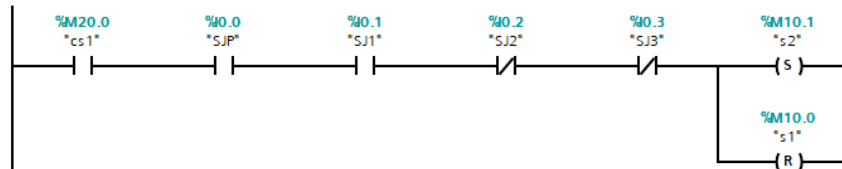
Segmento 1: Activar el estado 1 si no lo está



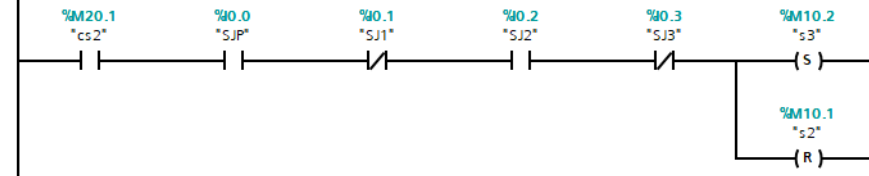
Segmento 2: Copia del estado



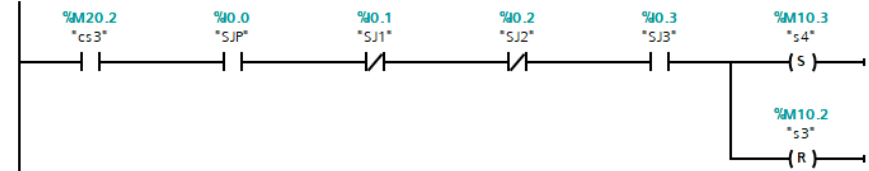
Segmento 3: Ir de estado 1 a estado 2



Segmento 4: Ir de estado 2 a estado 3



Segmento 5: Ir de estado 3 a estado 4

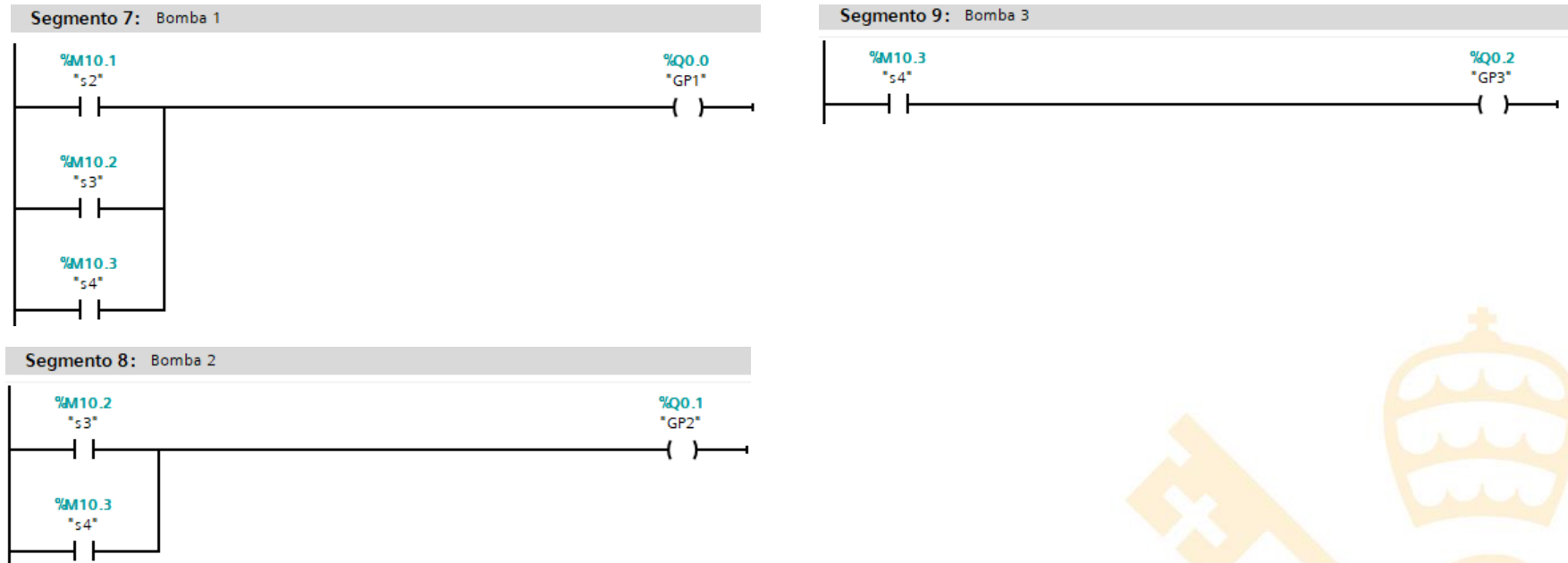


Segmento 6: Ir a estado 1 desde cualquier otro estado



Programación arranque tres bombas: variables de salida (I)

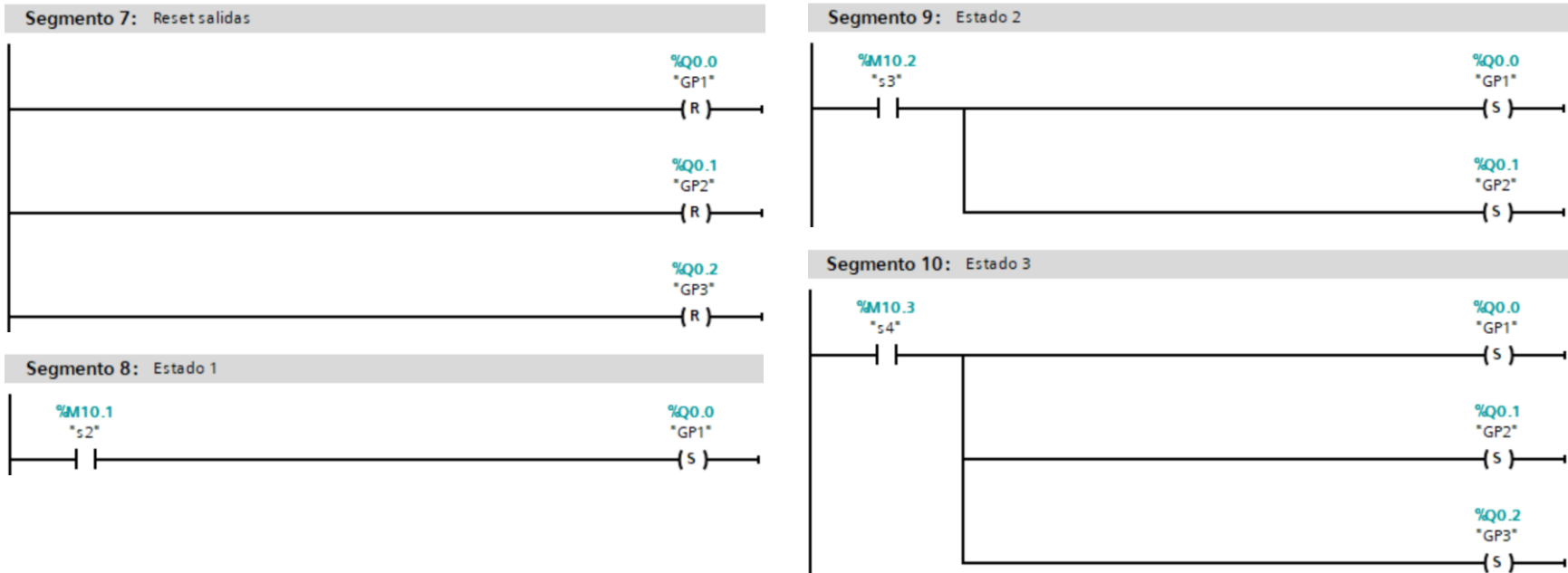
- **Versión: una única función lógica por salida**



- **Si añade un nuevo estado puede ser necesario cambiar todas las funciones lógicas de las variables de salida**

Programación arranque tres bombas: variables de salida (II)

- Versión: salidas que se activa en cada estado



- Si se añade un nuevo estado no hay que cambiar las funciones lógicas para activar las variables asociadas al resto de estados

Recomendaciones para programar máquinas de estado

- **Seguir el método propuesto de programación**
 - Llevar la máquina a un estado conocido (parada) si no lo está
 - El paro realiza la misma función
 - Si hay una etapa previa de inicialización se recomienda añadir una entrada física de reset para inicializar la máquina de estados cuando sea necesario
 - Calcular el nuevo estado mediante la copia y utilizando set/reset
 - Calcular el valor de las variables de salidas
 - Utilizar la asignación de salidas por etapa
- **Se puede simplificar el método propuesto**
 - Disminuye la legibilidad
 - Una programación sistemática frente a una programación llena de trucos
 - Aumenta el tiempo de depuración en caso de errores

¿Qué lenguaje utilizar?

- **Un programa está compuesto de bloques con diferentes necesidades**
 - Bloques expuestos a operarios sólo familiarizados con automatismos cableados
 - Diagrama de contactos
 - Diagrama de bloques de funciones
 - Bloques que implementan una o varias secuencias:
 - SFC: a ver en capítulo Grafcet
 - Diagrama de contactos con el método indicado
 - ST utilizando CASE
 - Bloques que procesan datos (matemáticas, comunicaciones, bases de datos): ST

Diferencia entre programar un PLC y un sistema digital

- **PLC admite:**

- Programación cíclica
 - El programa no espera a que un evento ocurra
 - El programa consulta de forma periódica si el evento ha ocurrido
 - Todas las fuentes de eventos (entradas, temporizadores, ...) están atendidas
 - Ventaja: programación fácil para atender a múltiple eventos
 - Desventaja: tiempo entre la ocurrencia del evento y su atención
- Programación por interrupciones (no vista)
 - Para aquellos eventos que necesitan ser atendidos de forma inmediata

- **Sistema digital admite:**

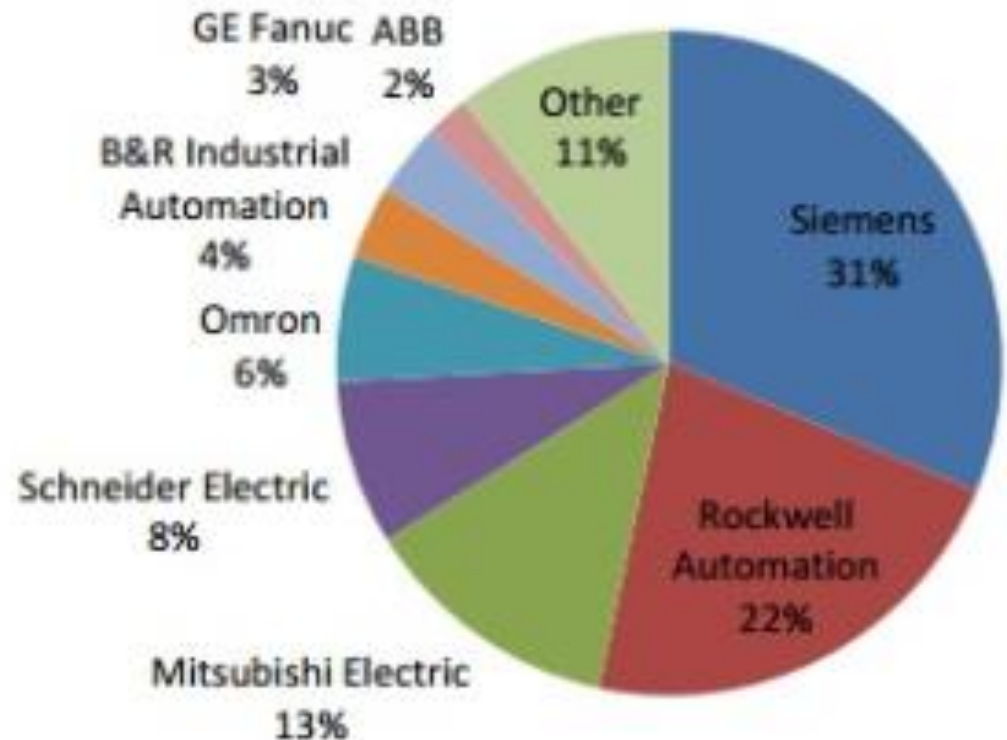
- Programación esperando un evento (no recomendado salvo casos sencillos)
 - El programa queda esperando a que un evento ocurra
 - El resto de entradas o fuentes de eventos no son atendidos
 - Ventaja: tiempo rápido de atención al evento cuando ocurre
 - Desventaja: el resto de eventos no son atendidos
- Programación por interrupciones
- Programación cíclica

Criterios para elegir un PLC

- **Entradas y salidas físicas**
 - Número
 - Capacidad de ampliación
 - Tipo de entradas
 - Digitales
 - Analógicas
- **Funciones especiales**
 - PID
 - Conexión con encoders de alta velocidad
 - Generación de pulsos de alta velocidad
 - Conexión con equipos de pesado
 - Algoritmos de inteligencia artificial
- **Comunicaciones**
 - Medio físico
 - Protocolos soportados
- **Tamaño físico**
- **Tamaño de memoria**
- **Velocidad de procesamiento**
 - Ciclo de scan
- **Lenguajes de programación**
- **Tipo de ejecución**
 - Cíclica
 - Interrupción
- **Fiabilidad**
- **Requerimientos de seguridad**
 - Sistemas críticos (SIL3)
 - Bucle de seguridad
- **Sistema de desarrollo**
- **Soporte del fabricante**
 - Servicio técnico/Repuestos
 - Entrenamiento
- **Coste**
- **Preferencias del cliente**

Fabricantes de PLCs

- Siemens
- Rockwell (Allen-Bradley)
- Mitsubishi
- Schneider (Modicon)
- Omron
- Bosch Rexroth
- GE Fanuc
- ABB
- Toshiba



Soluciones en Siemens

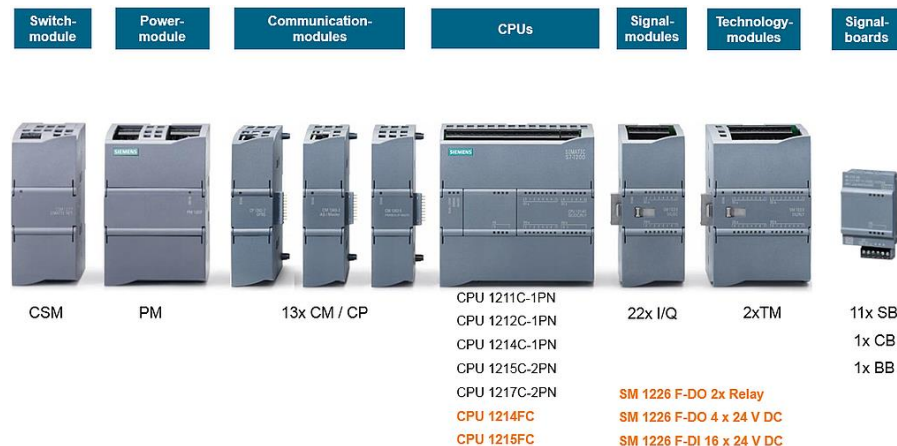
• Pequeña escala: LOGO!8

- Bloques funcionales
 - En pantalla
 - Mediante aplicación
- Diagrama de contactos
 - Mediante aplicación



• Complejidad baja o media: S7-1200

- Bloques funcionales
- Diagrama de contactos
- Texto estructurado



• Complejidad media o alta: S7-1500

- Todos los lenguajes
- Hay otras soluciones para muy alta complejidad



Soluciones en otros fabricantes

- **Allen-Bradley (Rockwell)**

- Complejidad alta: ControlLogix
- Complejidad media: CompactLogix
- Baja: MicroLogix



- **Schneider**

- Complejidad alta: Modicom Premium
- Complejidad media: Modicom TSX Micro
- Complejidad baja: Zelio



¿Siempre PLCs?

- **Automatización con PLCs**

- Idóneo en serie cortas
 - Disponibilidad inmediata
 - Ahorro de precio: hardware certificado, herramientas de desarrollo comprobadas, repuestos garantizados.
 - Flexibilidad para crecer: arquitectura modular
- Métodos de programación consolidados, fiables y con alta productividad
- Restricciones en el diseño:
 - Al ser un hardware genérico, no siempre está disponible la velocidad o la funcionalidad necesaria o el tamaño necesario
- Típico en instalaciones industriales

- **Automatización con sistemas digitales a la medida**

- Idóneo en series largas
 - Coste aquilatado a lo que se necesita
 - Coste de desarrollo, pruebas, certificación perfectamente asumible
 - No dependencia de terceros (no siempre: componentes)
- Métodos de programación con ciertos problemas
 - C no es el mejor lenguaje para equipos con cero errores
- Libertad para diseñar
 - Sólo límite tecnológico para conseguir prestaciones de velocidad o funcionalidades muy específicas
- Típico en equipos compactos: domótica, salud, automóvil

Resumen

- **Definición de PLC**
 - Arquitectura típica
 - Ejecución cíclica de las funciones lógicas
- **Diferencias entre automatismo cableado y programado**
- **Lenguajes estandarizados para programar los PLCs**
- **Organización de la memoria en los PLC de Siemens**
- **Programación con diagrama de contactos**
- **Aspectos de seguridad en el cableado del PLC**
- **Programación con bloques de funciones**
- **Programación en lista de instrucciones**
- **Programación de una máquina de estados (sólo si se ha visto)**
- **Criterios para elegir un PLC**
- **Típicas soluciones de los fabricantes de PLCs**

Ejemplos de preguntas

- Explica la organización interna de un PLC
- Explica mediante un cronograma la interacción entre realidad, entradas físicas, variables de entrada, variables internas, variables de salida y salidas físicas
- Dibujar un esquema con el cableado del PLC para un problema concreto teniendo en cuenta criterios de seguridad
- Diseño de un automatismo combinacional programado con diagrama de contactos
- Implementar una función lógica mediante bloques de funciones
- Implementar una función lógica mediante lista de instrucciones
- Diseño de un automatismo secuencial programado con diagrama de contactos (**sólo si se ha visto**)
- ¿Cuáles son los criterios para elegir un PLC?
- ¿Cuándo es recomendable utilizar un PLC frente a utilizar un sistema digital desarrollado a la medida?