

# Tema 2. Funcionamiento de un PLC

AUTOMATIZACIÓN. CURSO 2022-2023

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



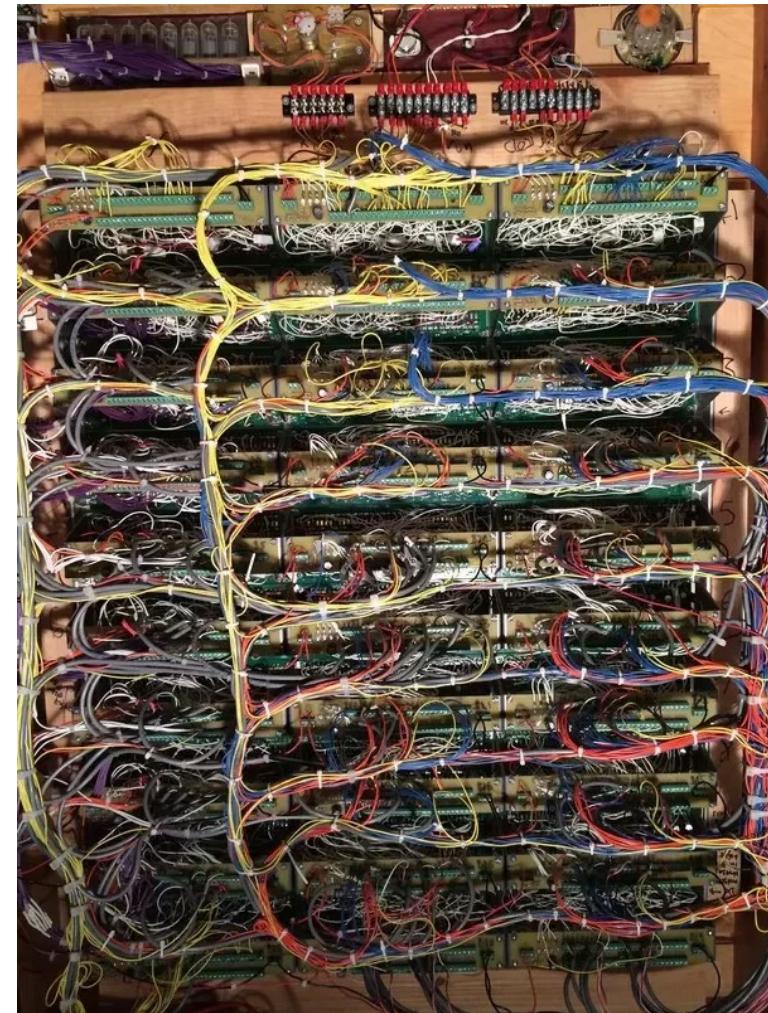


CNC

# Introducción. Historia de los sistemas de control

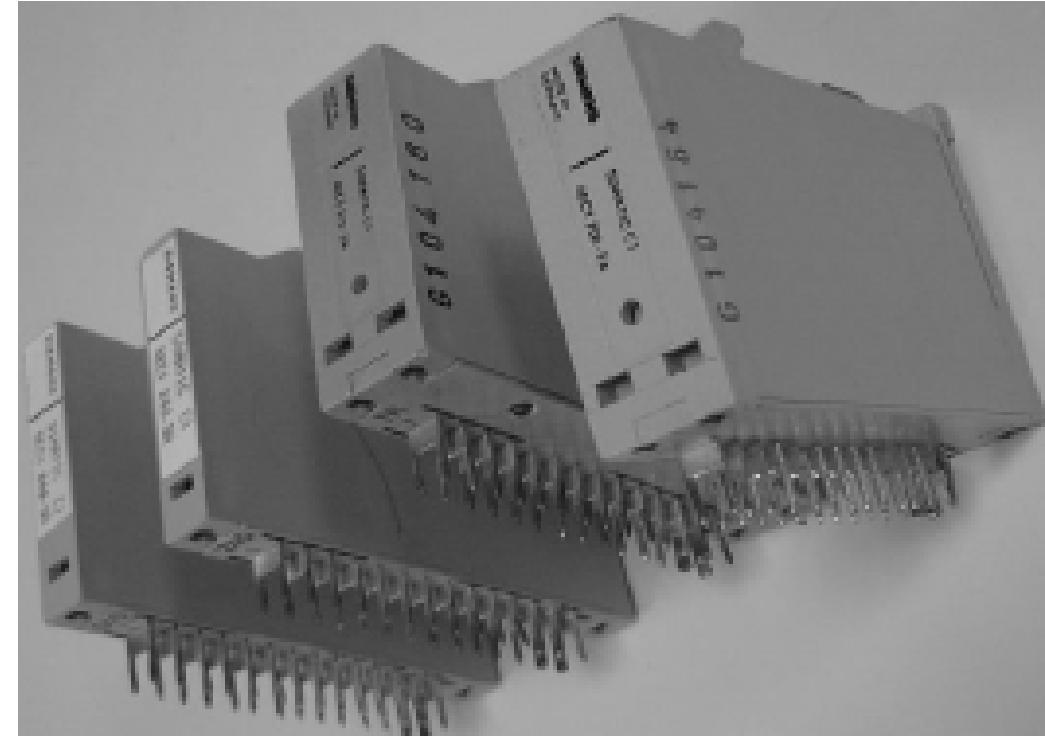
---

- Antes de la aparición de los transistores las fábricas operaban con lógica cableada.
- Había que programar el sistema de control mediante relés y bobinas



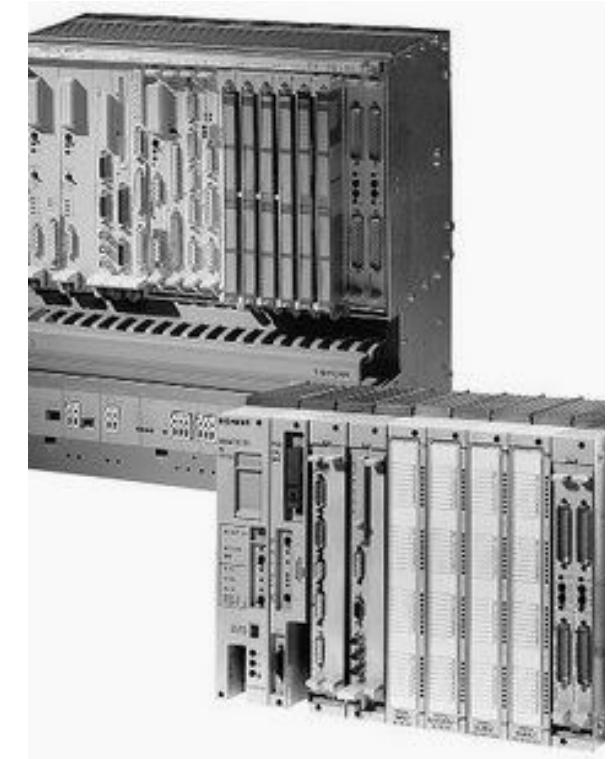
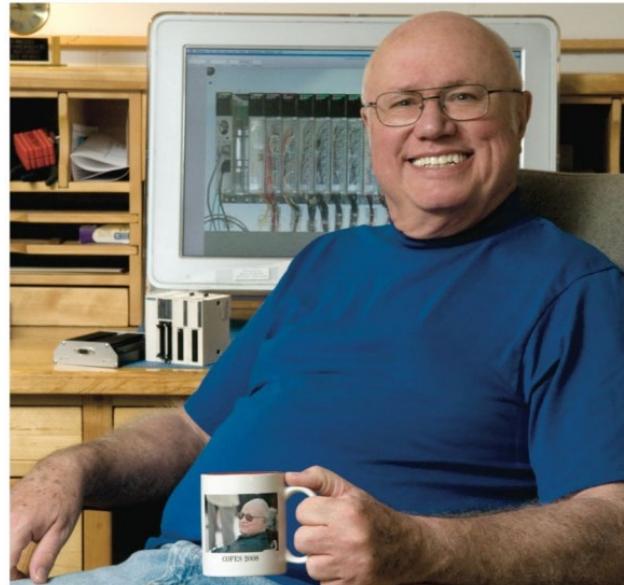
# Introducción. Historia de los sistemas de control

- Hasta los años 50 motores controlados por sistemas secuencias mediante relés interconectados.
- En 1960 aparecen los primeros controladores electrónicos de lógica cableada.



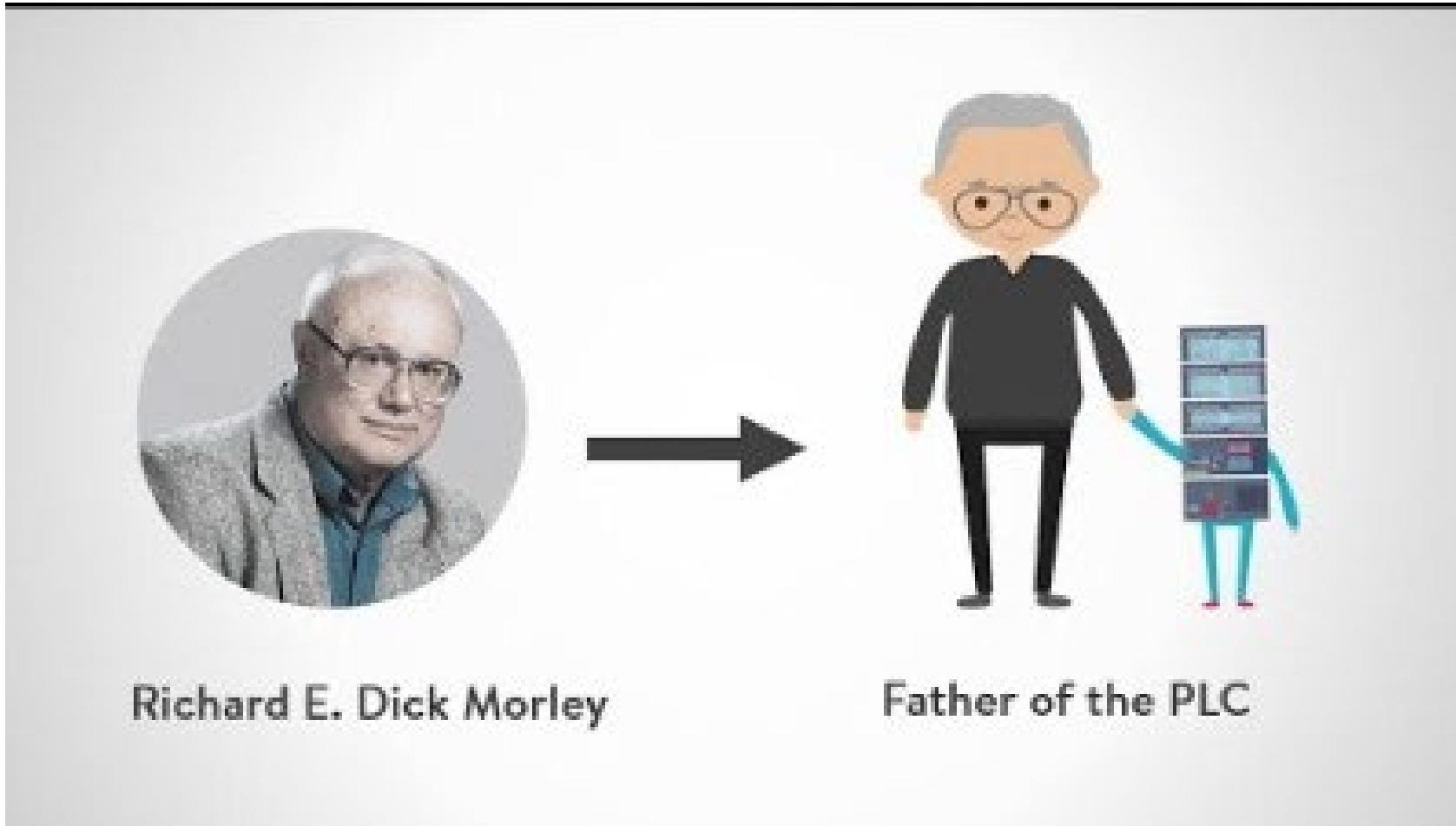
# Introducción. Historia de los sistemas de control

- En 1968 GM Hydramatic (la división de transmisión automática de General Motors) realiza un concurso para reemplazarlos sistemas cableados de relés por sistemas electrónicos
- Bedford Associates ganó el concurso con su idea de dispositivo de control de lógica Programable (Programmable Logic Controller).
- El primer PLC comercial fue el Modicon 084 (Modular Digital CONtroler). Dick Morley, es considerado como el creador del PLC.
- Modicon fue vendida en 1977 a Gould Electronics, luego fue comprada por AEG y finalmente por Schneider Electric.



<https://www.infopl.net/blogs-automatizacion/item/104790-obituario-dick-morley-padre-plc>

# Introducción. Historia de los sistemas de control



<https://www.youtube.com/watch?v=4BxOizho1UE>

# Introducción. Historia de los sistemas de control

- En 1974 ALLEN BRADLEY (hoy Rockwell Automation) presenta la solicitud de patente *Programmable Logic Controller*
- Odo Strugler (1931-1998) fue el impulsor del desarrollo del PLC en la compañía
- Participó en el desarrollo del estándar IEC1131-3



United States Patent [19]

Dummermuth

[11] 3,942,158

[45] Mar. 2, 1976

[54] PROGRAMMABLE LOGIC CONTROLLER

[75] Inventor: Ernst Dummermuth, Chesterland, Ohio

[73] Assignee: Allen-Bradley Company, Milwaukee, Wis.

[22] Filed: May 24, 1974

[21] Appl. No.: 473,149

[52] U.S. Cl. .... 340/172.5

[51] Int. Cl. .... G06F 7/00; G06F 9/06

[58] Field of Search ..... 340/172.5

[56] References Cited  
UNITED STATES PATENTS

3,624,611 11/1971 Wirsing ..... 340/172.5

3,651,484 3/1972 Smealie ..... 340/172.5

3,731,279 5/1973 Halsall et al. ..... 340/172.5

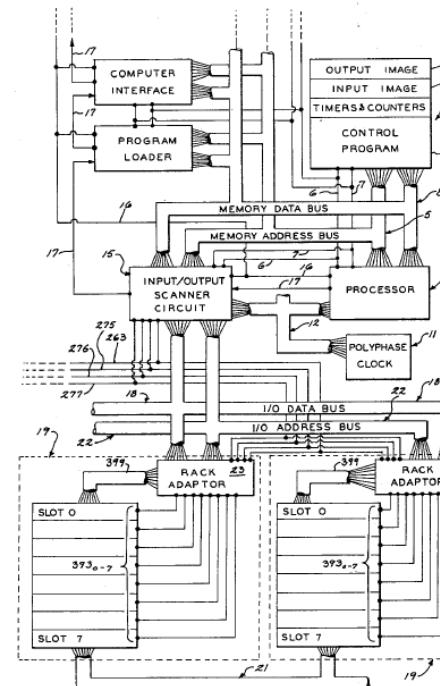
3,827,030 7/1974 Seipp ..... 340/172.5

Primary Examiner—David H. Malzahn  
Attorney, Agent, or Firm—Quarles & Brady

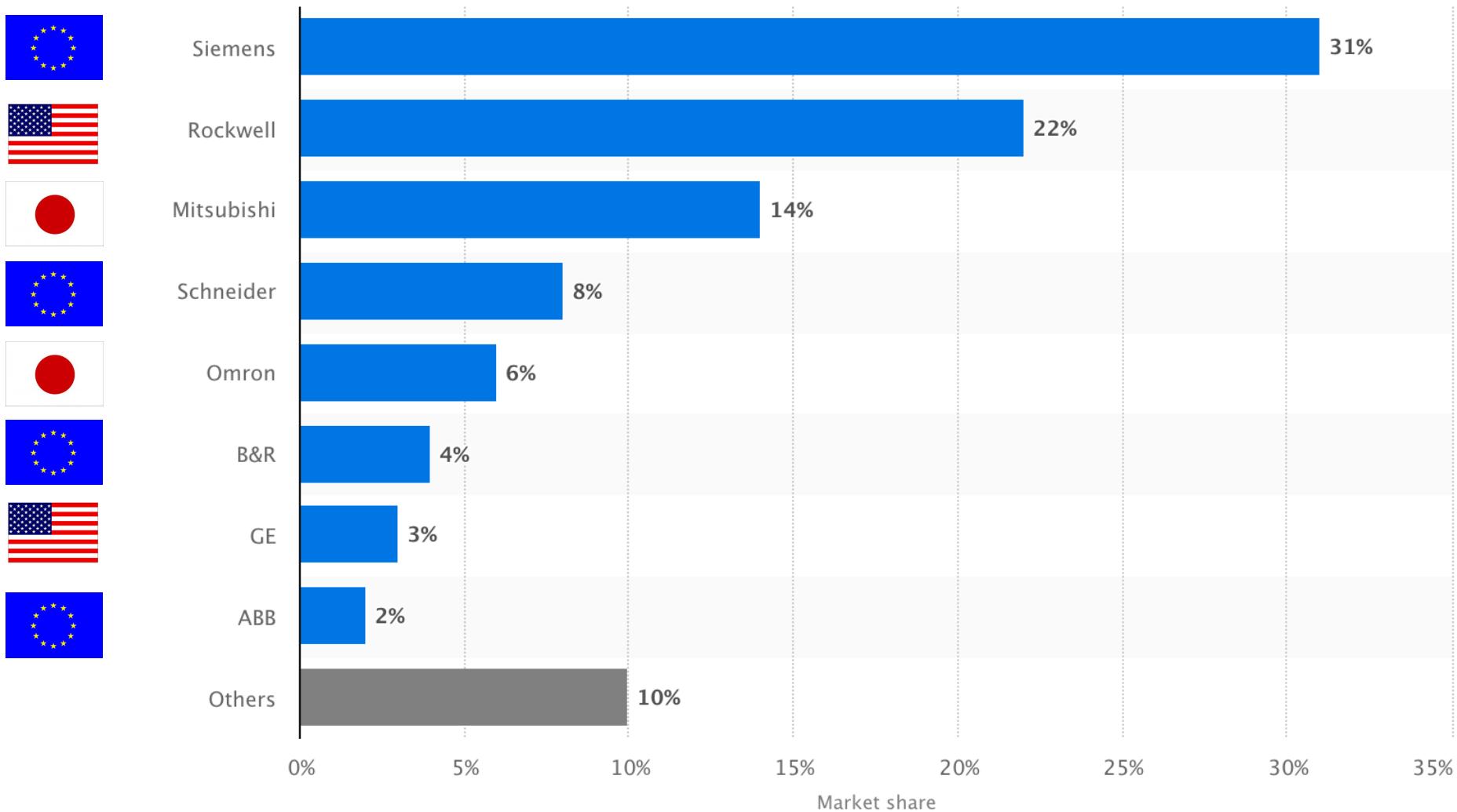
[57] ABSTRACT

A programmable controller includes a processor which executes a control program to alter the state of an output image table stored in a read/write memory in response to the state of an input image table stored in the memory. An input/output scanner circuit connects directly to the read/write memory and periodically steals a memory cycle from the processor to couple the status of input and output devices with corresponding bits in the input and output image tables. The rate at which the scanner circuit operates is independent of the processor speed and is selected to accommodate the input/output interface circuitry on the programmable controller.

21 Claims, 18 Drawing Figures



# KEY PLAYERS



Global PLC market share as of 2017, by manufacturer

<https://www.statista.com/statistics/897201/global-plc-market-share-by-manufacturer/>

# KEY PLAYERS



Table of the Top 25 PLC Manufacturers Ranked in Order of Industrial Automation Net Annual Sales Revenue

Rank	PLC Manufacturers	Industrial Automation Revenue (millions of USD)	Consolidated Revenue (millions of USD)
1	Siemens (Simatic)	\$18,281	\$98,636
2	Mitsubishi Electric (Melsec)	\$13,346	\$41,120
3	Emerson (GE Fanuc)	\$12,202	\$18,372
4	Hitachi	\$8,654	\$86,250
5	Bosch (Rexroth)	\$8,523	\$88,319
6	Schneider Electric (Modicon)	\$7,172	\$30,861
7	Eaton (Cutler-Hammer)	\$7,148	\$21,390
8	Rockwell Automation (Allen Bradley)	\$6,694	\$6,694
9	ABB (B&R Automation)	\$6,273	\$27,978
10	Keyence	\$5,341	\$5,341
11	Honeywell	\$5,146	\$36,709
12	Omron (Sysmac)	\$3,564	\$7,819
13	Yokogawa	\$3,371	\$3,679
14	Fuji Electric (Micrex)	\$2,934	\$8,323
15	Phoenix Contact	\$2,818	\$2,818
16	Toshiba	\$2,302	\$33,602
17	Panasonic	\$1,859	\$72,805
18	Delta Electronics	\$1,216	\$7,675
19	Wago	\$1,074	\$1,074
20	Beckhoff	\$1,026	\$1,026

Source: <https://ladderlogicworld.com/plc-manufacturers/>



Siemens PLC S7-1200

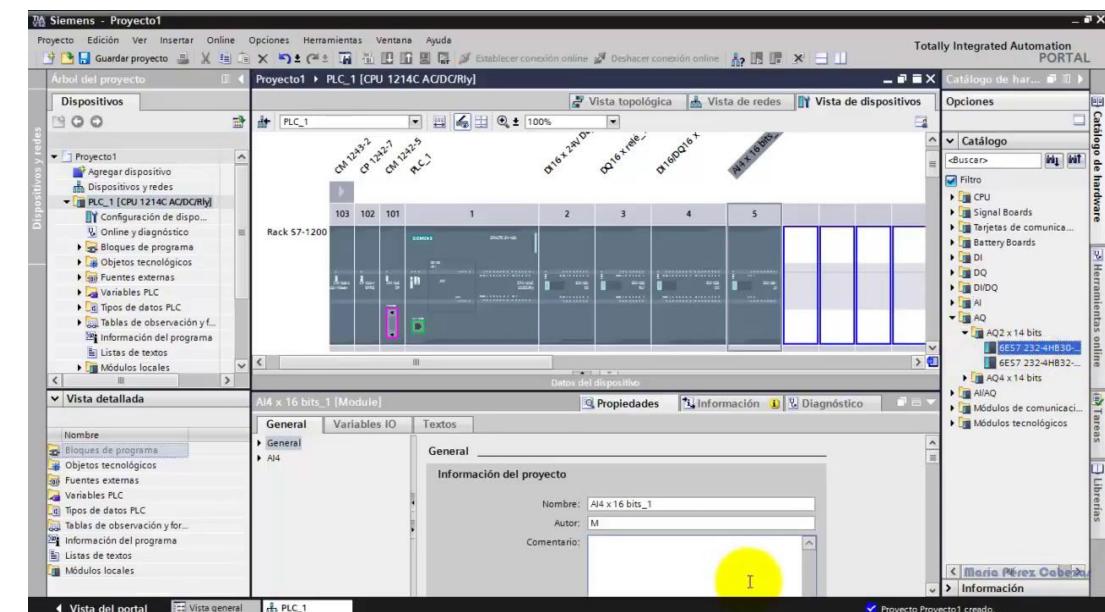
# SIEMENS



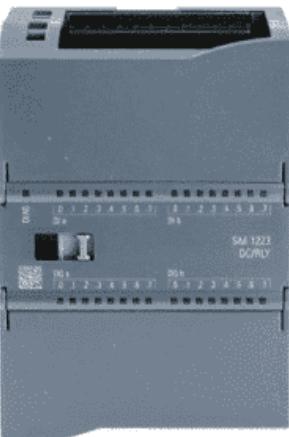
Siemens LOGO!



Siemens PLC S7-1500



TIA PORTAL  
(Totally Integrated  
Automation Portal)



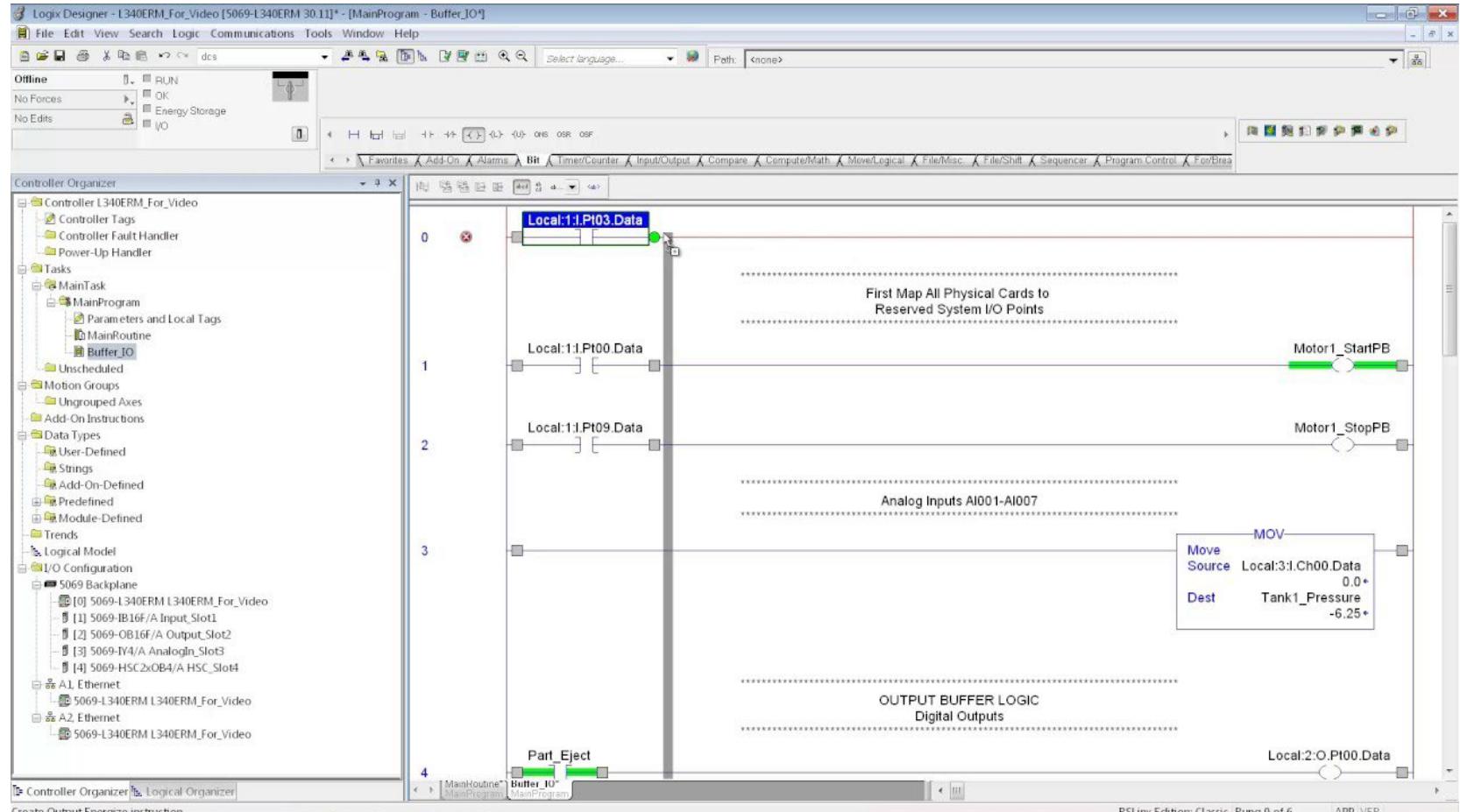
Módulo E/S



# Rockwell Automation



# Allen-Bradley



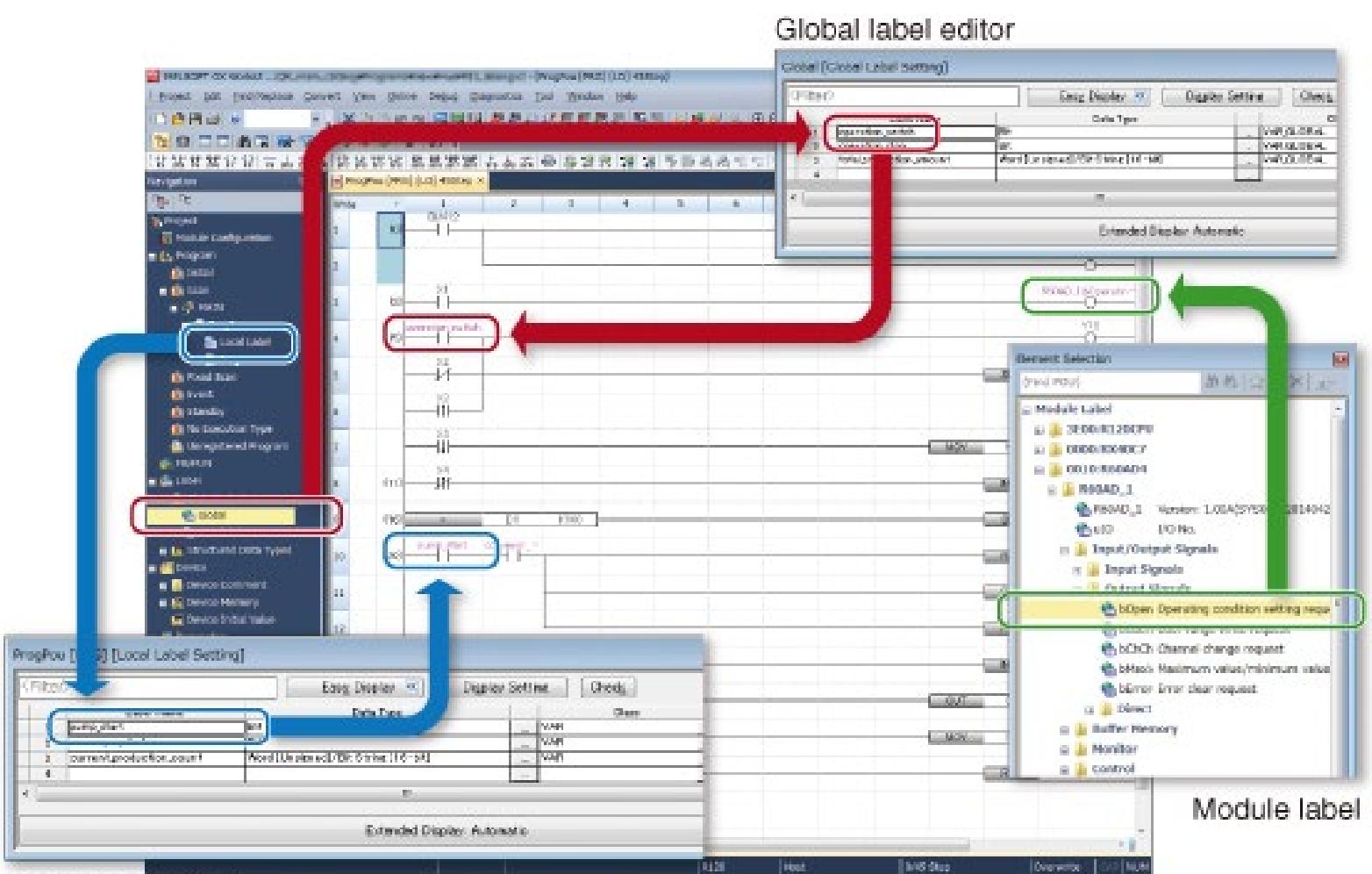
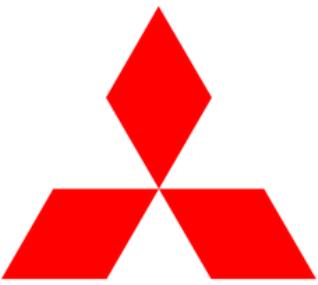
Studio 5000 Logix Designer



Módulo E/S



CompactLogix 5480



Módulo E/S



CPU para PLC Mitsubishi FX3S



Schneider Electric SoMachine Basic - V1.5 SP1

No selected device | Online | Run | Scan Time 0 µs | Unknown stop reason: 0

Properties Configuration Programming Display Commissioning

Tasks Tools

Messages

Animation tables

Animation table\_0

Memory objects

System objects

I/O objects

Network objects

Software Objects

PTO objects

Drive Objects

Communication Objects

Search and Replace

Symbol list

Memory consumption

Analog Inputs

Inputs Change Value

%IW0.0	30
%IW0.1	0

Add Insert

Address Symbol Value

%MW0	20
%MW1	0
%IW0.0	AI1
	30

Time Base 5 s Open Trace window

Modicon M221

Modicon M262

Módulo E/S

The screenshot shows the Schneider Electric SoMachine Basic software interface. The main area displays a ladder logic program for a POU named '1 - New POU'. The program consists of two rungs. Rung 1 has a normally open contact 'IN4 %I0.3' connected to the coil of a timer 'T' (Type TON, TB: 1 s, Preset: 20). Rung 2 has the output of the timer 'T' connected to a normally closed contact '20 < 30' which is then connected to the coil of a contact 'OUT3 %Q0.2'. On the left, there's a sidebar with 'Analog Inputs' showing values for %IW0.0 and %IW0.1. Below that is a table for memory objects (%MW0, %MW1) and an I/O table for Modicon M221/M262 modules. The bottom right shows three physical hardware components: a Modicon M221 module with 16 digital inputs and 8 digital outputs, a Modicon M262 module with 16 digital inputs and 16 digital outputs, and a separate Módulo E/S (Modular I/O) module.

SoMachine (gratuito) / Unity Pro



Modicon M221

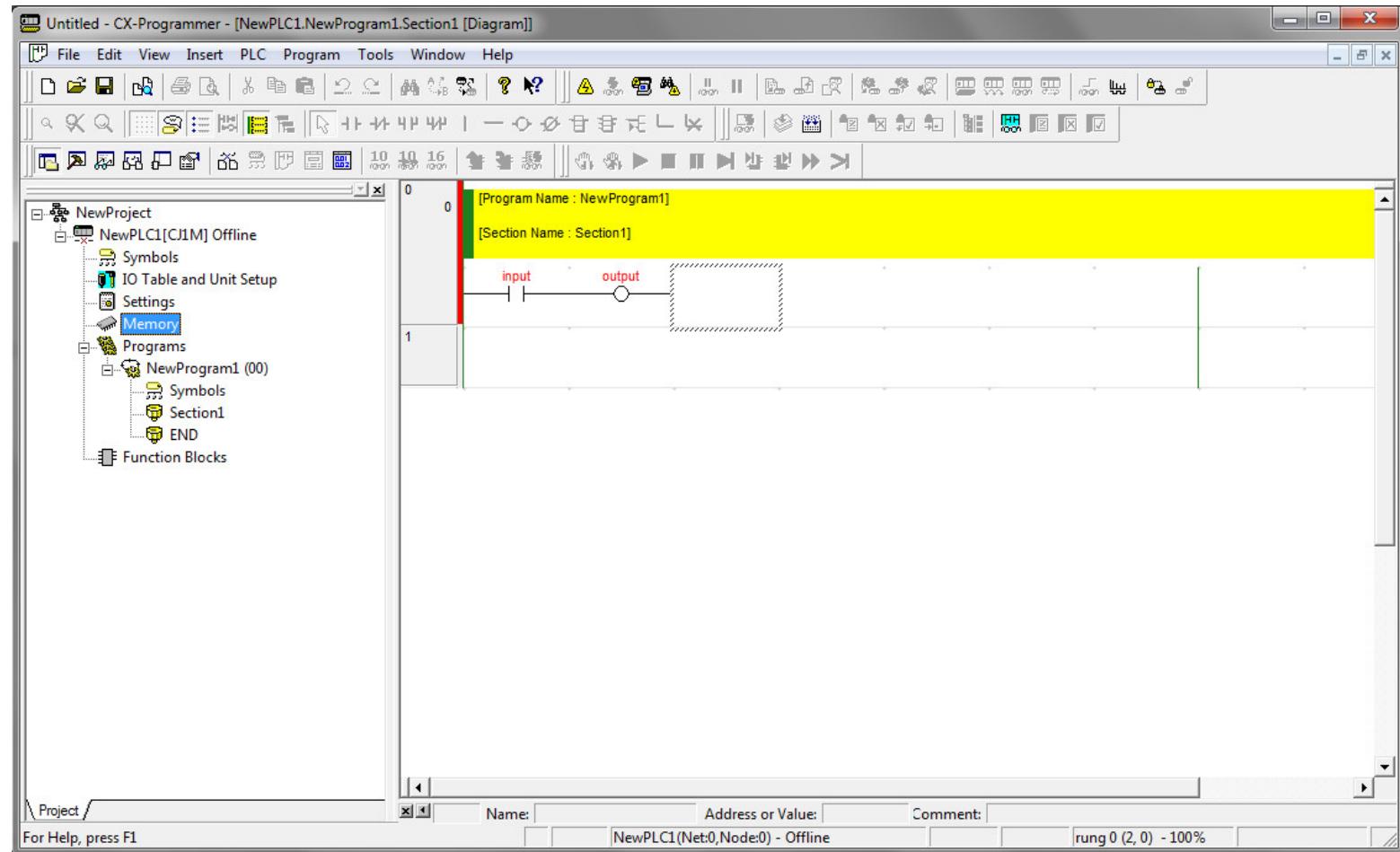


Modicon M262



Módulo E/S

# OMRON



CXONE - SYSMAC



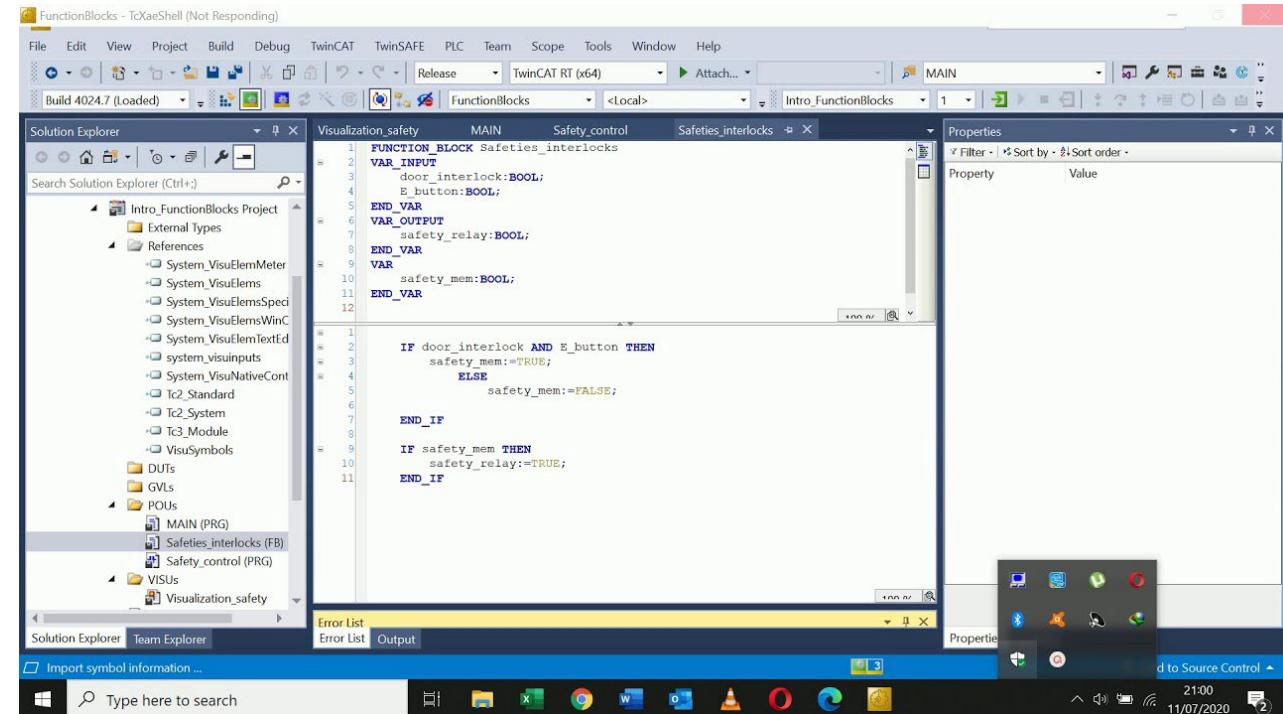
OMRON C2PE



Módulo E/S

# BECKHOFF

## New Automation Technology

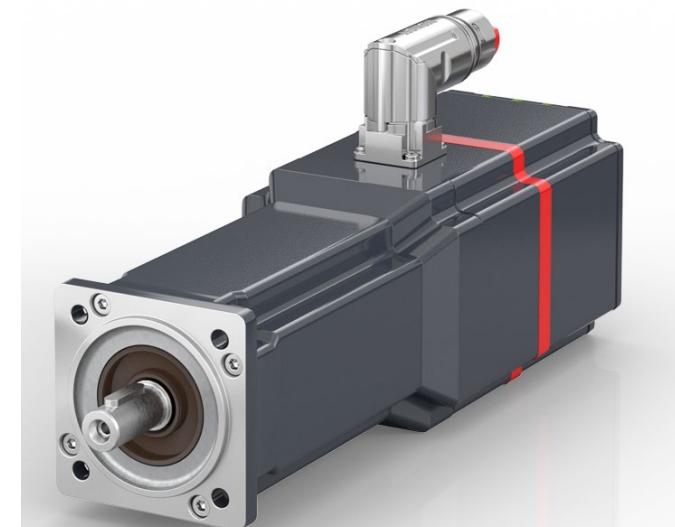


```
FUNCTION_BLOCK Safety_interlocks
VAR_INPUT
    door_interlock:BOOL;
    E_button:BOOL;
END_VAR
VAR_OUTPUT
    safety_relay:BOOL;
END_VAR
VAR
    safety_mem:BOOL;
END_VAR

IF door_interlock AND E_button THEN
    safety_mem:=TRUE;
    ELSE
        safety_mem:=FALSE;
END_IF

IF safety_mem THEN
    safety_relay:=TRUE;
END_IF
```

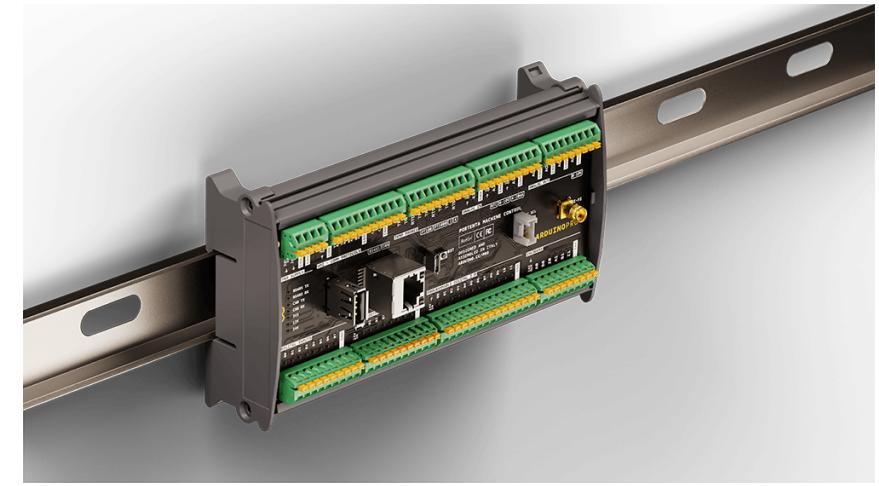
TWINCAT

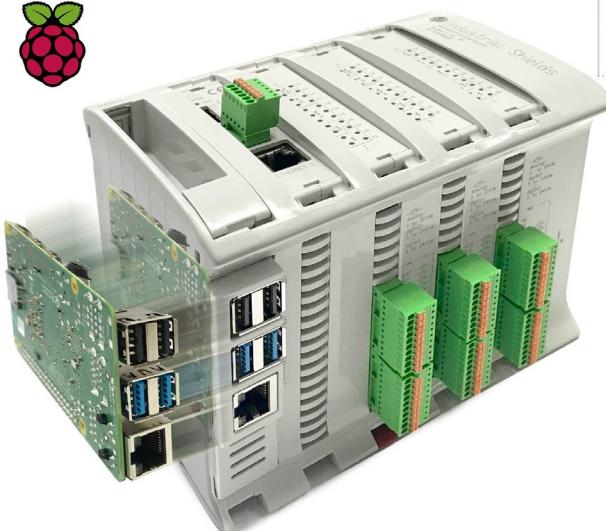




A screenshot of the Arduino PLC IDE. The interface includes a top menu bar with File, Edit, View, Project, On-line, Debug, Scheme, Variables, Window, Tools, and Help. Below the menu is a toolbar with various icons. The main workspace is divided into several panes: 'Project' (containing 'introduction Project' and 'myLDPProgram'), 'Local variables' (listing 'myCount' and 'increment' with their types and initial values), 'Watch' (a table for monitoring variable values), 'Library Tree' (a tree view of available functions like ABS, SIN, COS, etc.), and 'Catalog' (a list of device names). A central ladder logic editor shows a single coil labeled 'myCount' with two parallel rungs: one for 'myCount' and another for 'increment'. The bottom pane displays 'Output' logs, including memory usage and compatibility checks. The status bar at the bottom shows 'EDIT MODE', 'DIFF. CODE (SYM)', and 'CONNECTED'.

ARDUINO PLC IDE

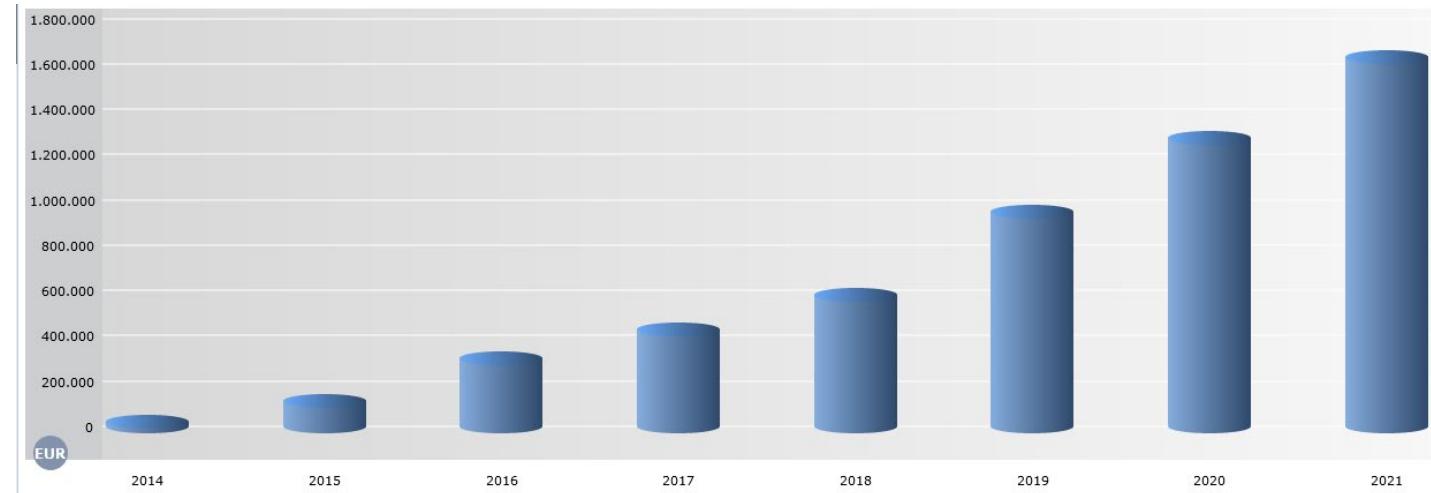




Industrial Shields®

<https://www.industrialshields.com>

# PLC CÓDIGO ABIERTO



Fuente: SABI ([Sistema de Análisis de Balances Ibéricos](#))

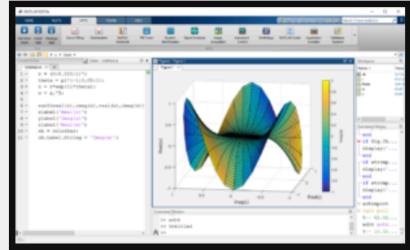
- Una gran variedad de PLC basados en Arduino. (PNP, NPN, Relé, Analógicos, etc.)
- Touch Screen con Sistema Linux Integrado (Panel PC + Kit IN/OUT 24Vcc)
- Sensores Industriales
- Automatismos y sistemas mecatrónicos
- Equipos y maquinaria

Entrevista fundador Industrial Shields:

<https://www.youtube.com/watch?v=6tQ6agFYHSE&t=113s>



# PLC CÓDIGO ABIERTO



## Pick the software of your choice.

You can program CONTROLLINO with **Arduino IDE**, as well as with many other software solutions like **Matlab**, **Atmel Studio** and more. Generally said: If the software is compatible with traditional Arduino boards, you can use it for CONTROLLINO too.

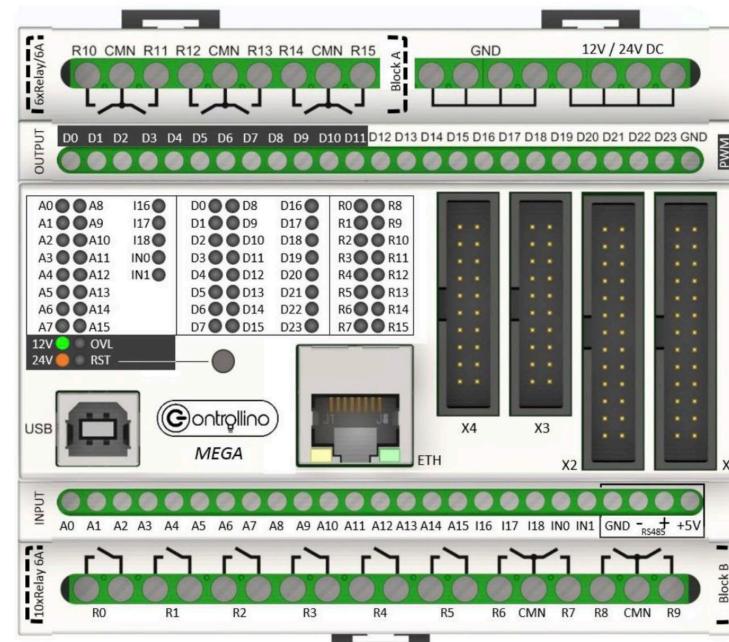


Figure 3:  
CONTROLLINO MEGA

## PLC MARKET REPORT SCOPE

REPORT ATTRIBUTE	DETAILS
MARKET SIZE (REVENUE)	USD 38 Billion (2020)
CAGR	5% (2020-2025)
BASE YEAR	2019
HISTORIC YEAR	2018
FORECAST YEAR	2020-2025
MARKET SEGMENTS	Type (Modular, Rackmount, and Unitary), End-users (Automotive, Chemical, Food & Beverage, Mining & Metallurgy, Water, Oil & Gas, Paper & Packaging, and Others)
GEOGRAPHIC ANALYSIS	North America, Europe, APAC, Latin America, and Middle East & Africa
COUNTRIES COVERED	US, Canada, UK, Germany, France, Italy, Spain, China, Japan, South Korea, Australia, India, Brazil, Mexico, GCC, and Turkey

*“The global PLC market size is projected to reach revenues of around \$38 billion by 2020, growing at a CAGR over 5% by 2025.”*

<https://www.arizton.com/market-reports/plc-market-analysis>



The global programmable logic controller (PLC) market size reached US\$ 14.6 Billion in 2022. Looking forward, IMARC Group expects the market to reach US\$ 20.2 Billion by 2028, exhibiting a growth rate (CAGR) of 5.38% during 2023-2028

<https://www.imarcgroup.com/programmable-logic-controller-market>

# Electrónica digital. Funciones básicas

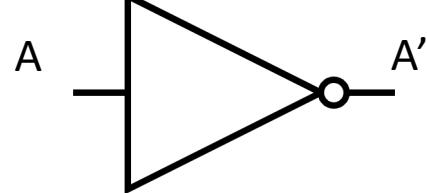
- La unidad de información digital binaria es el bit que puede tomar valor 1 ó 0 (1 Byte = 8 bits).
- La función implementada por un sistema digital viene determinada por su Tabla de Verdad, que proporciona el valor de la salida en función de las entrada.
- Álgebra de BOOLE. Funciones Booleanas: AND, OR, NOT, NAND, NOR, EXOR, EXNOR.
- Literal: Variable o su complemento.

# Electrónica digital. Funciones básicas

NOT

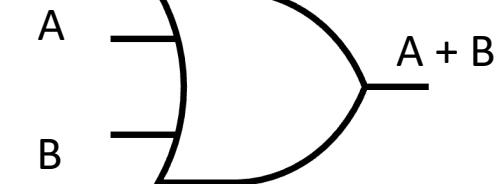
A	A'
0	1
1	0

$$A' = \bar{A} \quad \text{Dos notaciones}$$



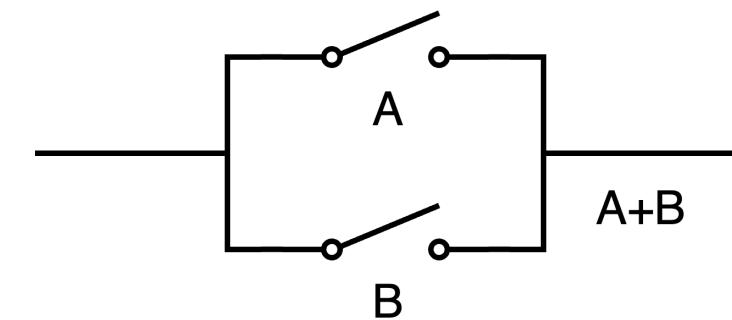
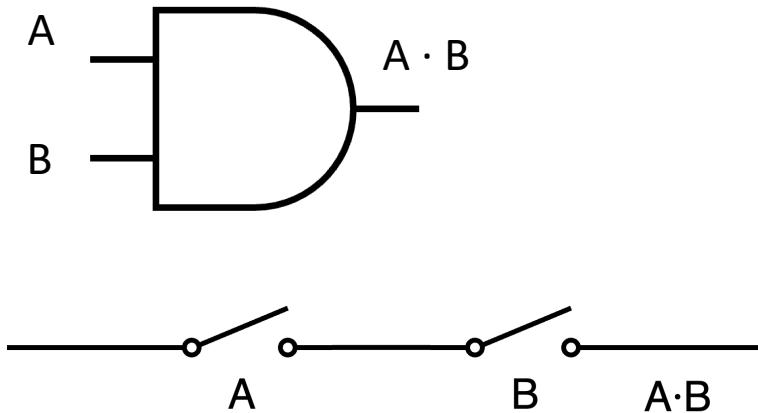
OR

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1



AND

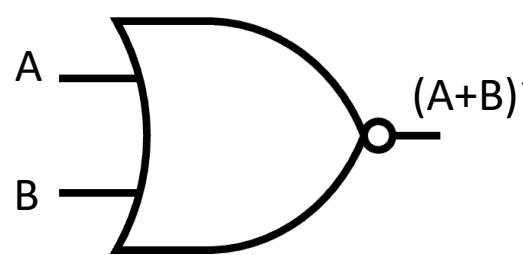
A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1



# Electrónica digital. Funciones básicas

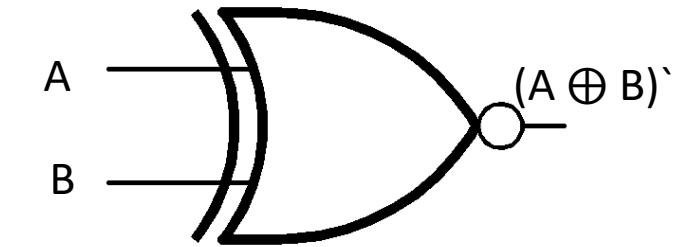
NOR

A	B	$(A+B)'$
0	0	1
0	1	0
1	0	0
1	1	0



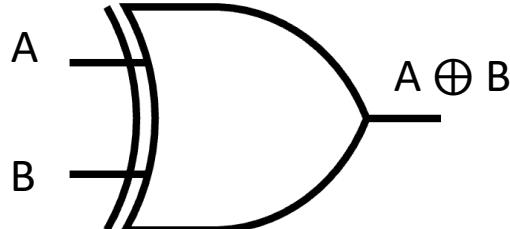
EXNOR

A	B	$(A \oplus B)'$
0	0	1
0	1	0
1	0	0
1	1	1

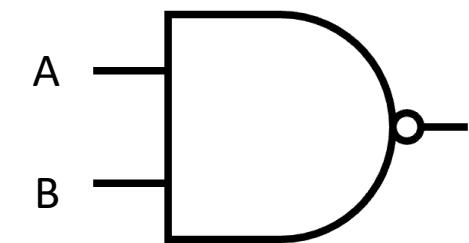


EXOR

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



A	B	$(A \cdot B)'$
0	0	1
0	1	1
1	0	1
1	1	0



# Electrónica digital. Identidades básicas del álgebra booleana

$$1) \quad x + 0 = x$$

$$2) \quad x + 1 = 1$$

$$3) \quad x \cdot 0 = 0$$

$$4) \quad x \cdot 1 = x$$

$$5) \quad x \cdot x = x$$

$$6) \quad x + x = x$$

$$7) \quad x + x' = 1$$

$$8) \quad x \cdot x' = 0$$

$$9) \quad (x')' = x$$

$$10) \quad x + y = y + x$$

$$11) \quad x \cdot y = y \cdot x$$

$$12) \quad x + (y + z) = (x + y) + z$$

$$13) \quad x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

$$14) \quad x \cdot (y + z) = x \cdot y + x \cdot z$$

$$15) \quad x + y \cdot z = (x + y) \cdot (x + z)$$

$$16) \quad (x + y)' = x' \cdot y'$$

$$17) \quad (x \cdot y)' = x' + y'$$

Comutativa

Asociativa

Distributiva

De Morgan

# Electrónica digital. Leyes de De Morgan

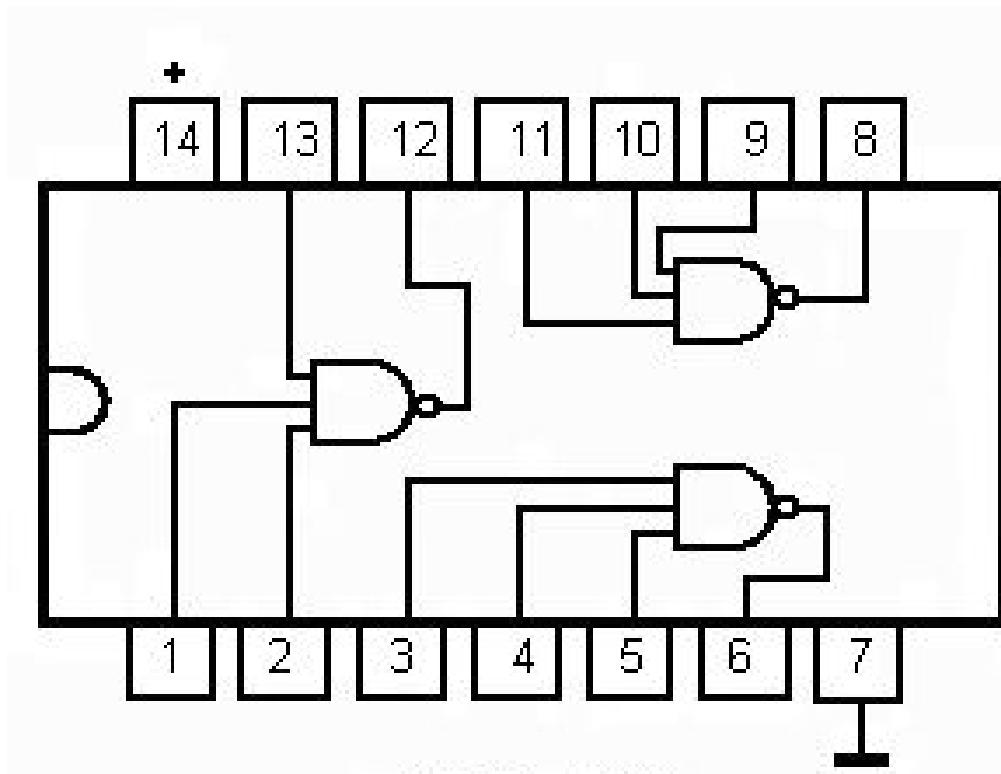
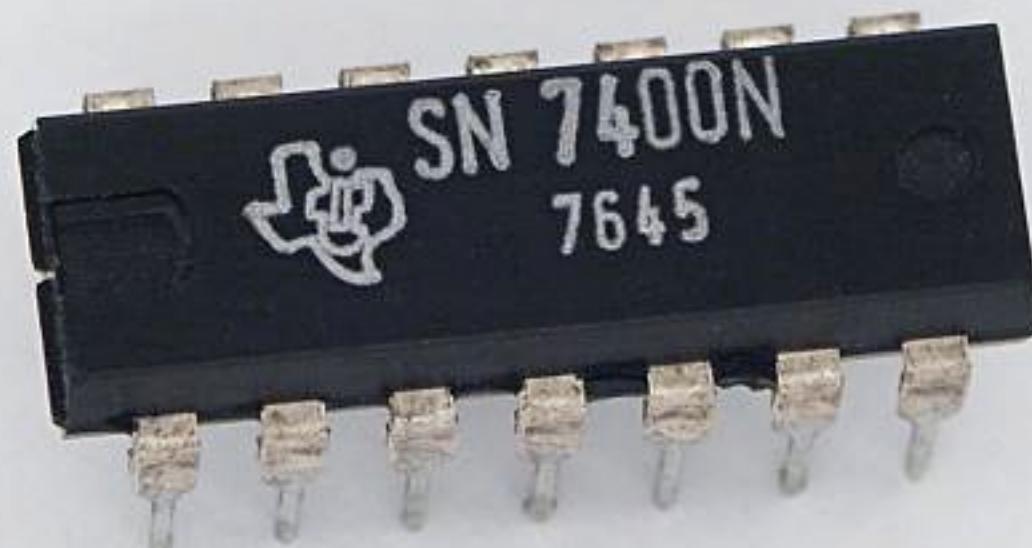
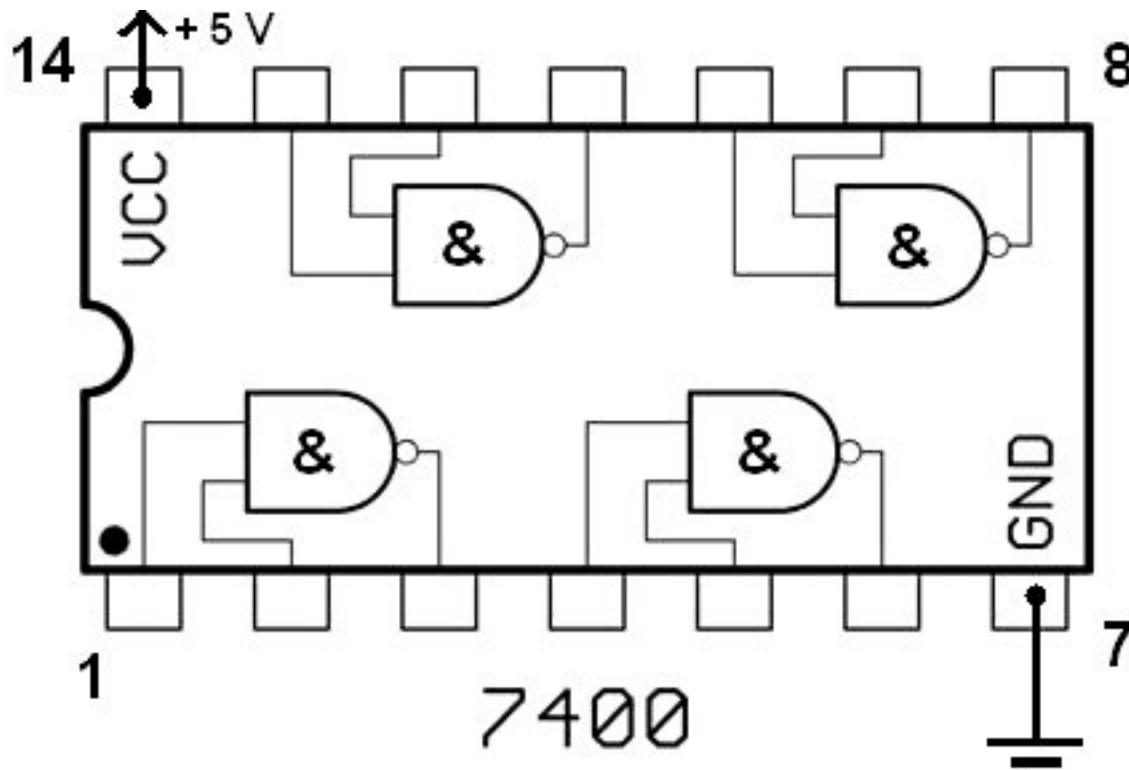
x	y	x'	y'	x·y	(x·y)'	x'+y'
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

$$(x \cdot y)' = x' + y'$$

x	y	x'	y'	x+y	(x+y)'	x'·y'
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

$$(x + y)' = x' \cdot y'$$

$$F'(A, A', \cdot, +, 1, 0) = F(A', A, +, \cdot, 0, 1)$$



7410 - 7412

# Electrónica digital. Karnaugh Maps

EJERCICIO - Comparador de números :

Tenemos dos números de dos bits ( $A_1, A_0$ ) y ( $B_1, B_0$ ). Diseña un sistema digital que implemente la siguiente funcionalidad:

Si  $(A_1, A_0) > (B_1, B_0)$  se pone a 1 la salida G (greater than)

Si  $(A_1, A_0) < (B_1, B_0)$  se pone a 1 la salida L (less than)

Si  $(A_1, A_0) = (B_1, B_0)$  se pone a 1 la salida E (equal to)

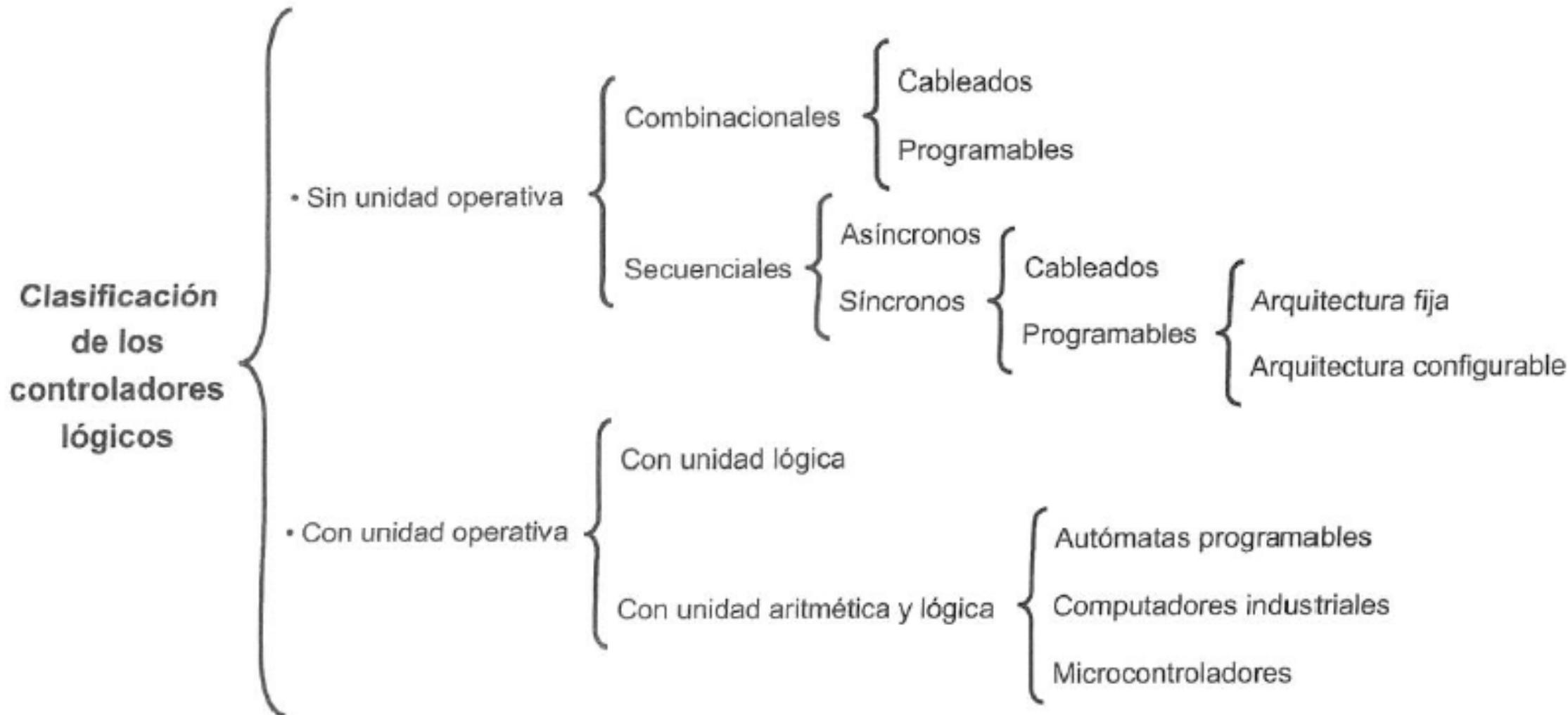
Ejemplo:  $A_1 = 0, A_0 = 1, B_1 = 1, B_0 = 0 \Rightarrow L = 1$

# Electrónica digital. Karnaugh Maps

- En el tribunal hay 2 juezas (A y B) y 2 jueces (C y D). Te han encargado diseñar un sistema de voto electrónico para determinar el sentido del veredicto. En caso de empate la presidenta del tribunal (jueza A) tiene el voto de calidad que decide el sentido de la votación.



# Introducción. Controladores lógicos



# Controladores lógicos sin unidad operativa

- Controladores sin unidad operativa:
  - Sistemas combinacionales
  - Sistemas secuenciales
- Sistema combinacional: la salida depende únicamente de las entradas del sistema (de su combinación)
- Controladores lógicos combinacionales: Operan de acuerdo al comportamiento de un sistema combinacional.
- Sistemas sin memoria: No sirven para tomar una decisión de control en función del estado del sistema.
- Pueden ser **cableados o programables**.

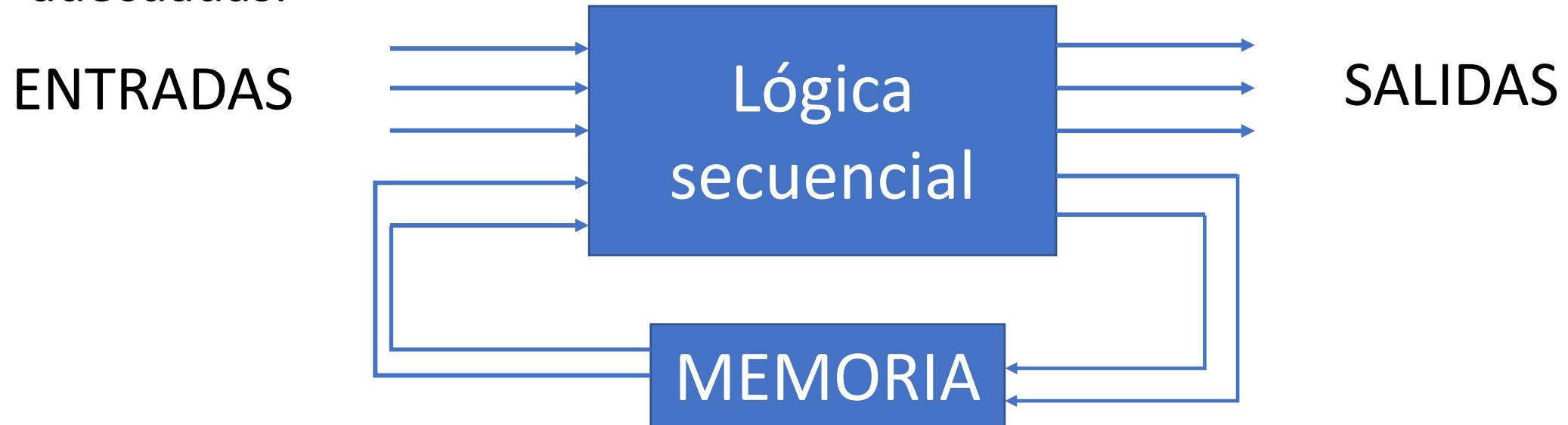
# Controladores lógicos sin unidad operativa. Combinacional

## Ejercicio:

Un proceso químico tiene tres sensores de temperatura cuyas salidas  $T_1$ ,  $T_2$  y  $T_3$  proporcionan dos niveles de tensión diferenciados según la temperatura sea mayor o menor que  $t_1$ ,  $t_2$  y  $t_3$  ( $t_1 < t_2 < t_3$ ). Se usa un valor 1 para indicar que la temperatura  $T$  del sensor es mayor que el valor fijado  $t$  para el sensor y un 0 si la temperatura es menor. Diseña un sistema que genere una señal de control activa cuando la temperatura esté comprendida entre  $t_1$  y  $t_2$  o cuando es superior a  $t_3$ .

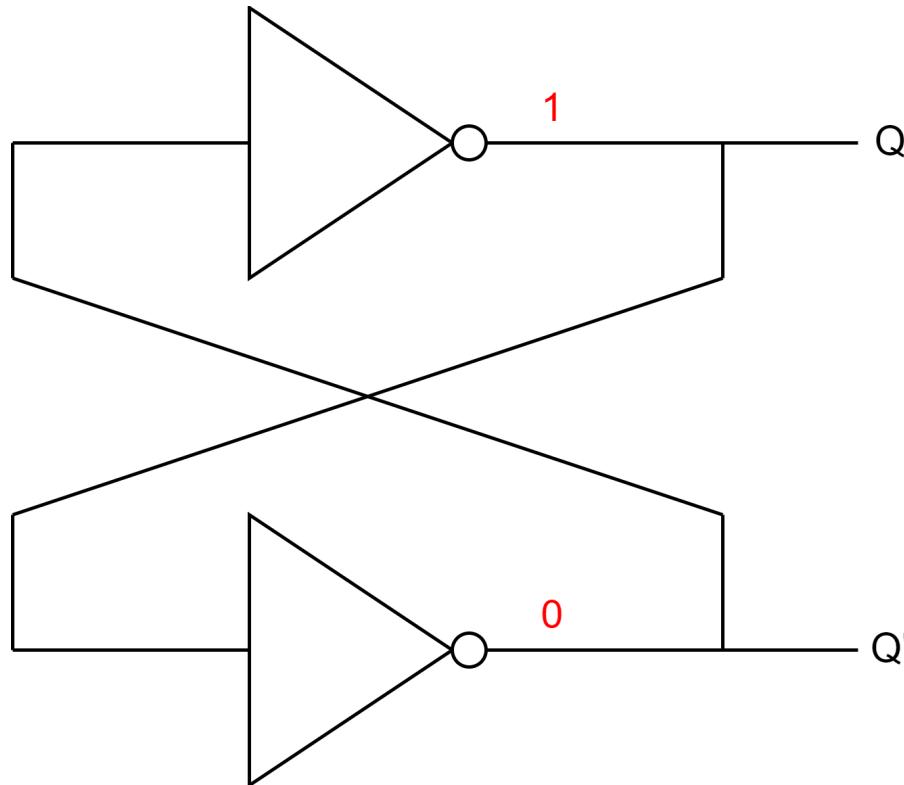
# Controladores lógicos sin unidad operativa. Secuencial

- Un sistema combinacional genera salidas en función de las entradas. No es posible generar salidas en función de la historia del sistema (estado previos).
- Los sistemas **secuenciales** permiten memorizar el estado del sistema para ser utilizado como una entrada y tomar las decisiones adecuadas.



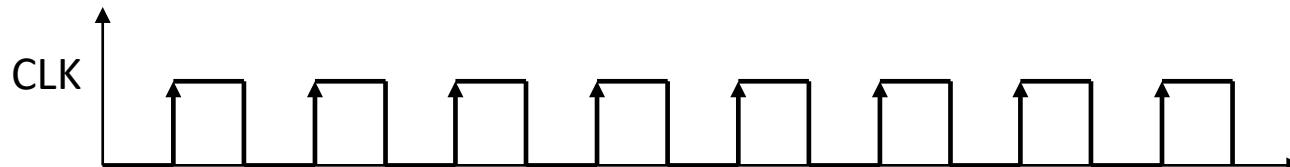
# Controladores lógicos sin unidad operativa. Secuencial

- Se utilizan elementos de memoria (biestables) para almacenar información.



# Controladores lógicos sin unidad operativa. Secuencial

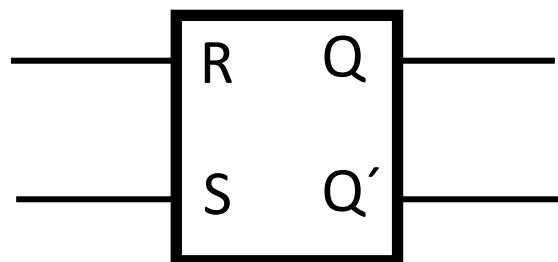
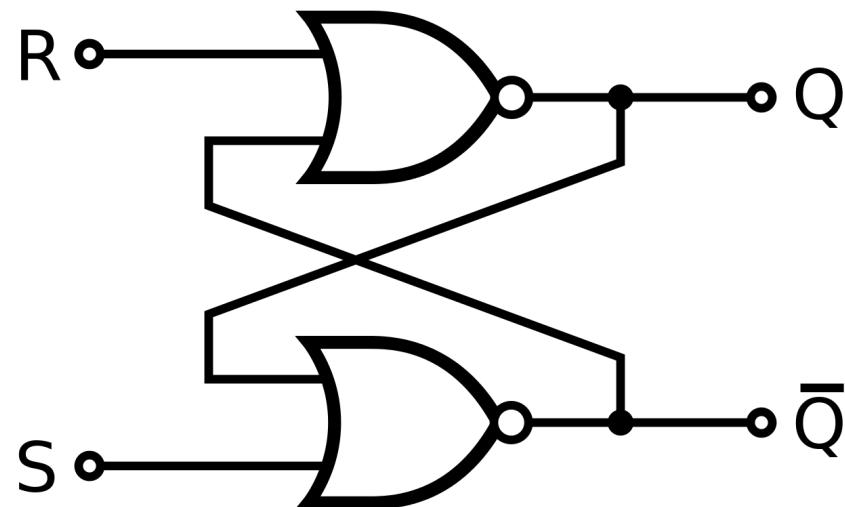
- Se añaden puertas adicionales para poder escribir el valor que se quiere almacenar.
- **Secuenciales asíncronos:** se actualiza el estado del biestable al cambiar la entrada.
- **Secuenciales síncronos:** se actualiza el estado del biestable en el momento en el que se produce un flanco o cambio de nivel en una señal de control (trigger o clk).
- **Biestables (Flip-flop):** RS (ó SR), JK, D, T



Niveles: 0 ó 1  
Flancos: subida o bajada

# Controladores lógicos sin unidad operativa. Secuencial

- Biestables RS



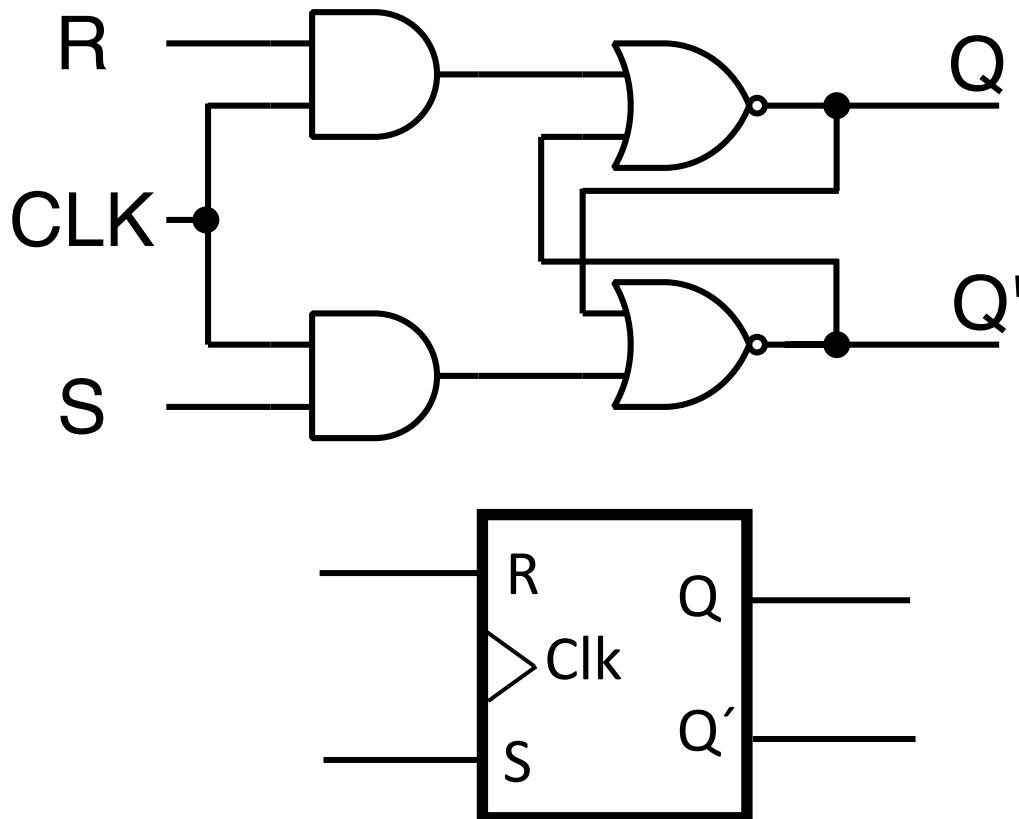
R	S	$Q(t+1)$	$Q'(t+1)$	Función
0	0	$Q(t)$	$Q'(t)$	Mantener $Q(t)$
0	1	1	0	Set
1	0	0	1	Reset
1	1	-	-	Prohibida

BIESTABLE ASÍNCRONO

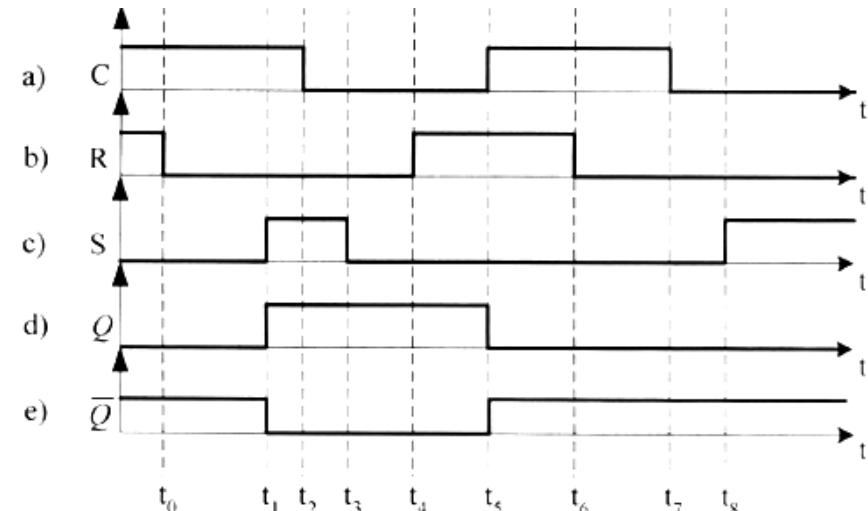
# Controladores lógicos sin unidad operativa.

## Secuencial

- Biestables RS. Disparado por nivel



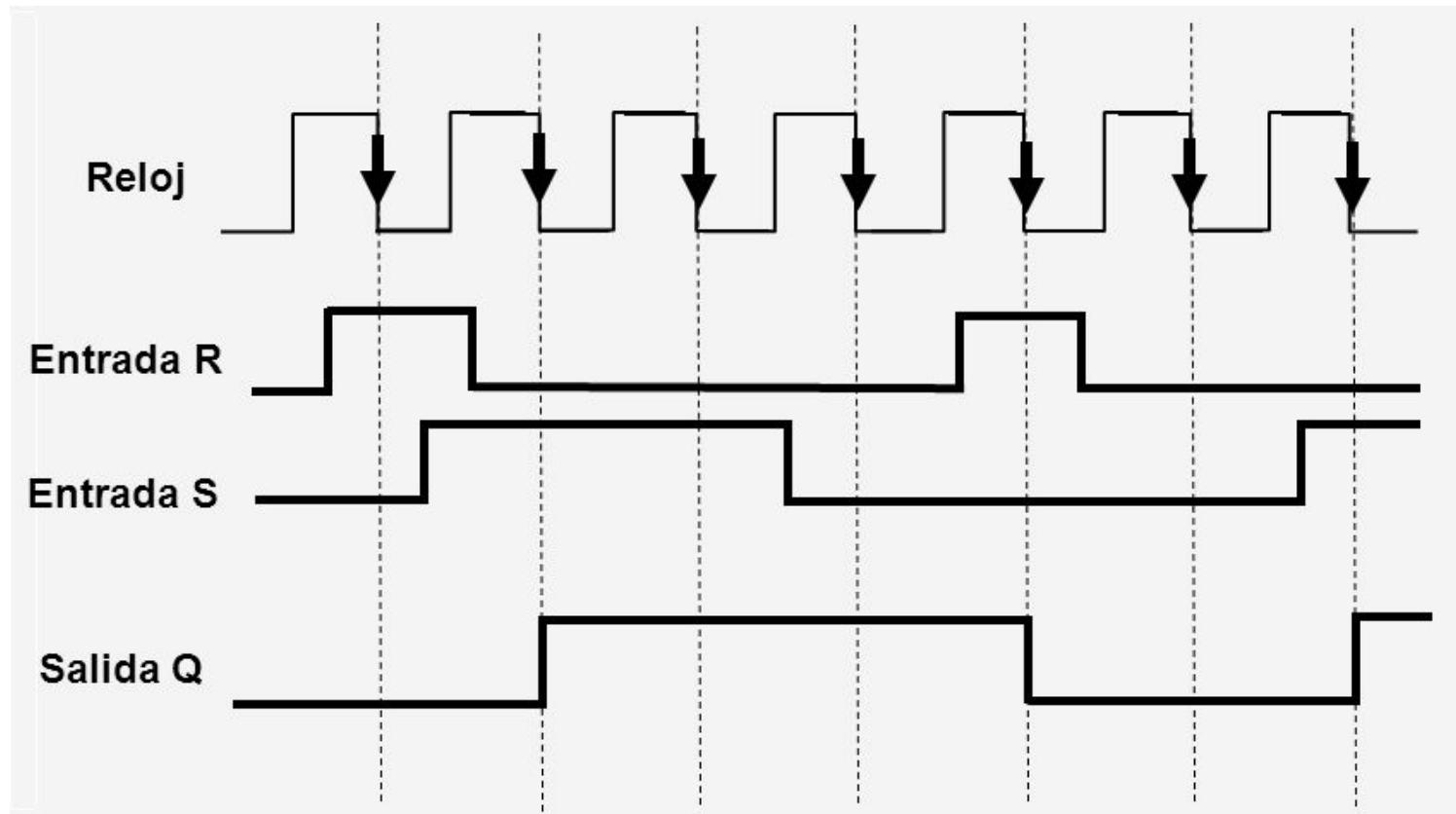
Clk	R	S	$Q(t+1)$	$Q'(t+1)$	Función
0	x	x	$Q(t)$	$Q'(t)$	Mantener $Q(t)$
1	0	0	1	0	Mantener $Q(t)$
1	0	1	1	0	Set
1	1	0	0	1	Reset
1	1	1	-	-	Prohibida



BIESTABLE SÍNCRONO RS DISPARADO POR NIVEL ALTO

# Controladores lógicos sin unidad operativa. Secuencial

- Biestables RS. Disparado por flanco de bajada

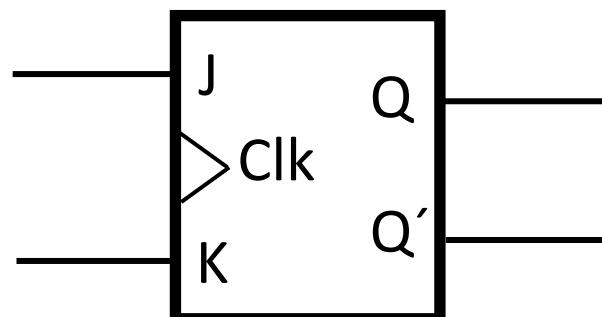
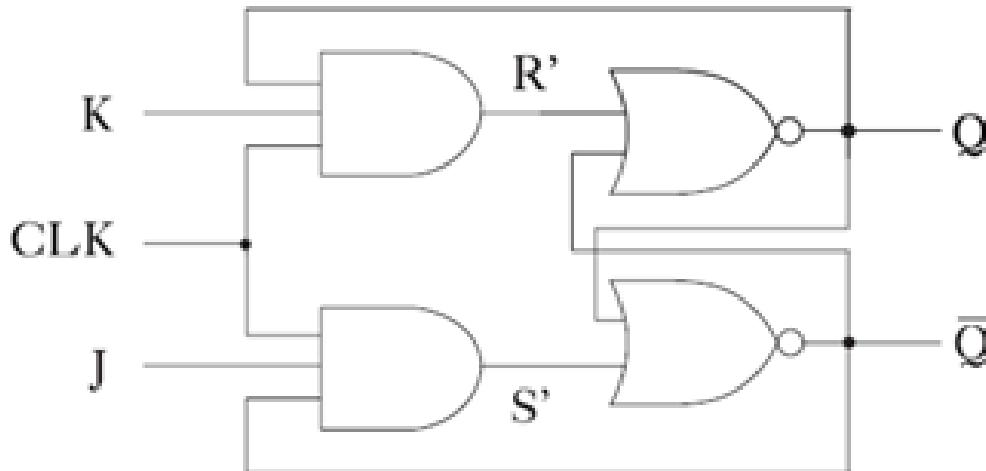


BIESTABLE SÍNCRONO RS DISPARADO POR FLANCO DE BAJADA

# Controladores lógicos sin unidad operativa.

## Secuencial

- Biestables JK. Disparado por nivel



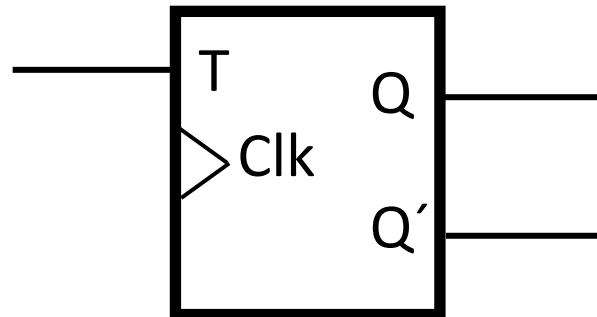
Clk J K	Q(t+1)	Q'(t+1)	Función
0 x x	Q(t)	Q'(t)	Mantener Q(t)
1 0 0	1	0	Mantener Q(t)
1 0 1	0	1	Reset
1 1 0	1	0	Set
1 1 1	-	-	Conmuta Q(t)

Q <sub>nJK</sub>	Q <sub>n+1</sub>
000	0
001	0
010	1
011	1
100	1
101	0
110	1
111	0

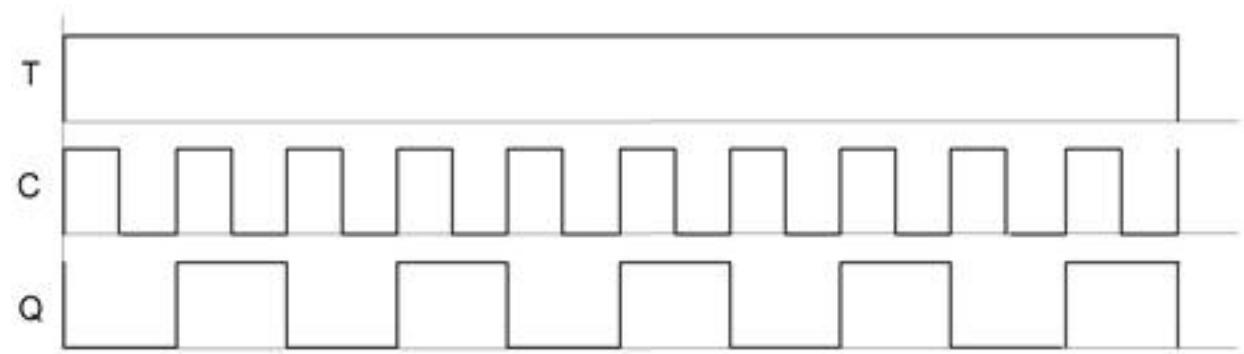
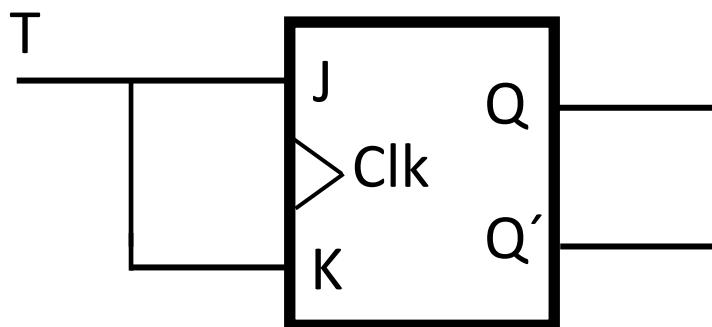
BIESTABLE SÍNCRONO JK DISPARADO POR NIVEL ALTO

# Controladores lógicos sin unidad operativa. Secuencial

- **Biestables T (trigger):** Biestable JK donde  $J=K=T$



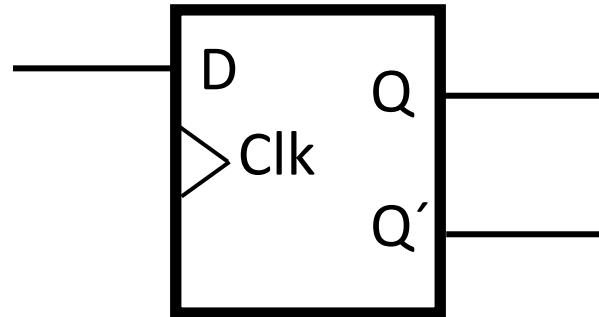
Clk T	Q(t+1)	Q'(t+1)	Función
0 x	Q(t)	Q'(t)	Mantener Q(t)
1 0	1	0	Mantener Q(t)
1 1	0	1	Conmuta Q(t)



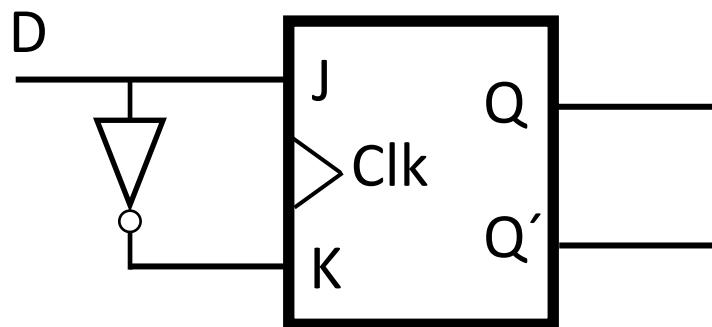
BIESTABLE SÍNCRONO T DISPARADO POR FLANCO

# Controladores lógicos sin unidad operativa. Secuencial

- **Biestables D:**



Clk	D	Q(t+1)	Q'(t+1)	Función
0	x	Q(t)	Q'(t)	Mantener Q(t)
1	0	0	1	0
1	1	0	1	1



BIESTABLE SÍNCRONO D DISPARADO POR FLANCO

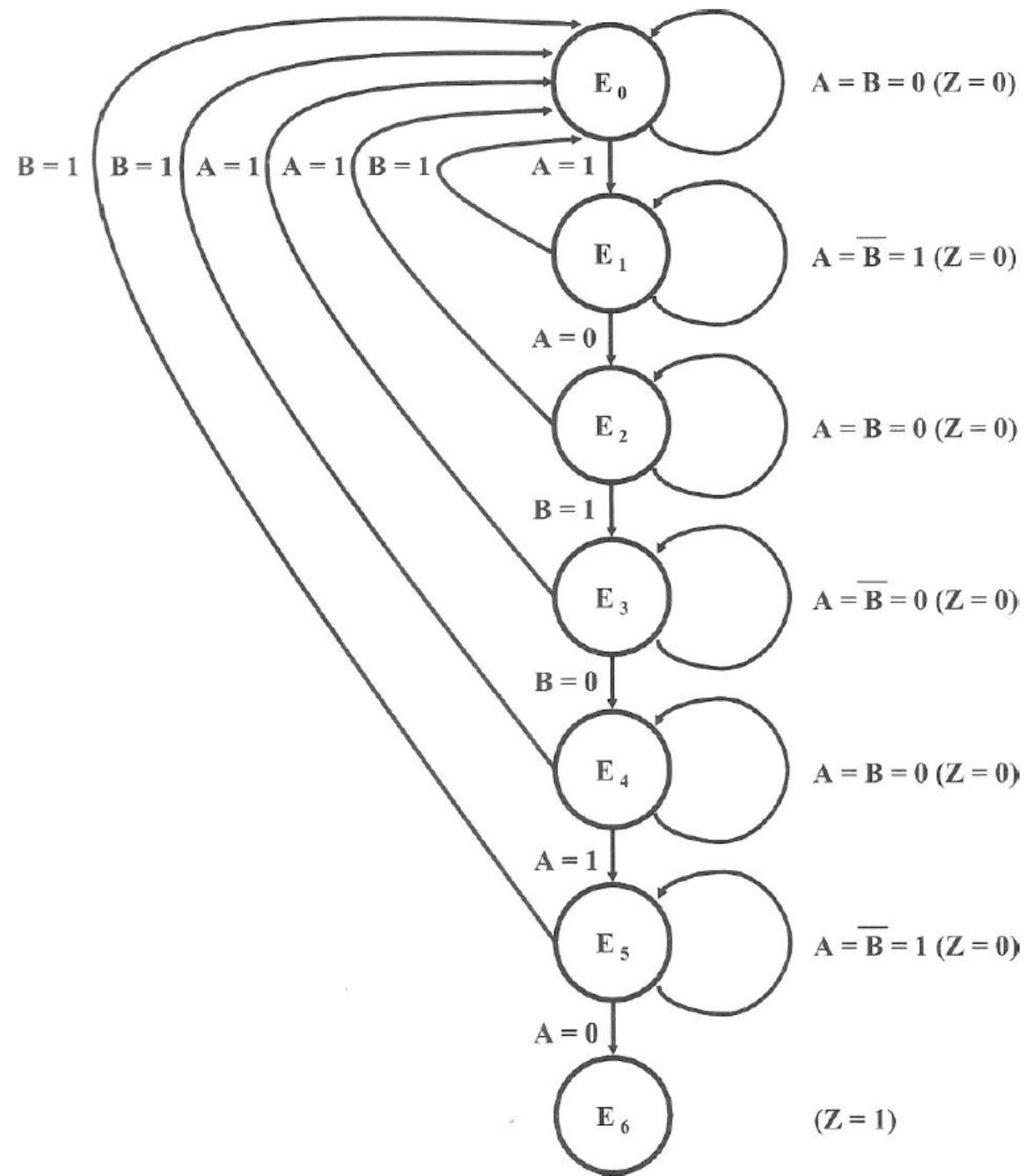
$Q_n D$	$Q_{n+1}$
00	0
01	1
10	0
11	1

# Controladores lógicos sin unidad operativa. Secuencial. Niveles

- **Ejercicio:** Vamos a implementar una cerradura electrónica mediante un controlador lógico que tiene dos entradas digitales, A y B, de 1 bit. Al iniciar el controlador se inicializa al estado inicial y evoluciona en función de los valores de A y B. La cerradura se abre si se da la siguiente secuencia:

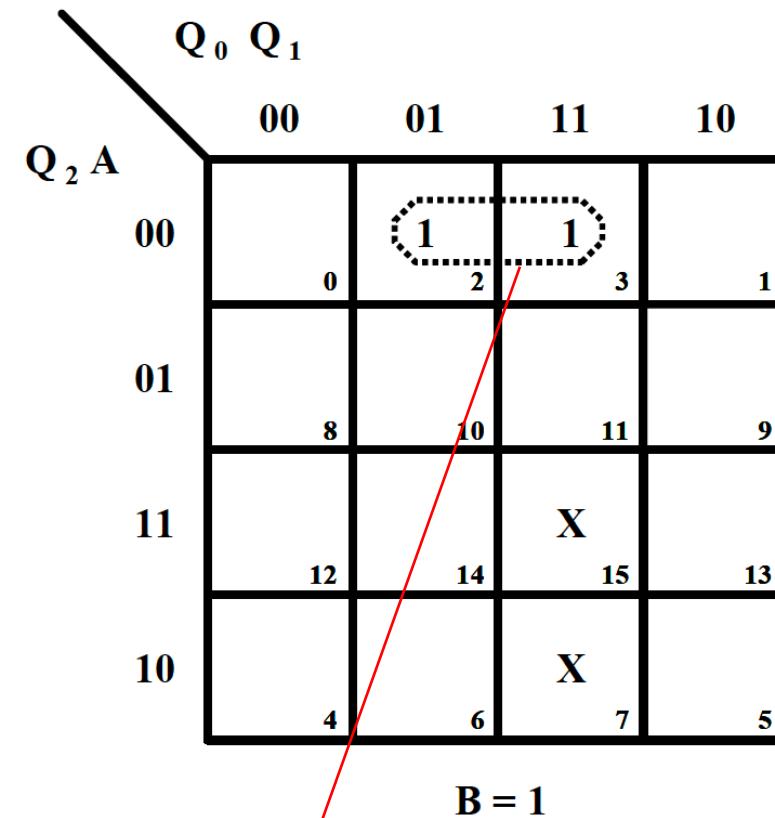
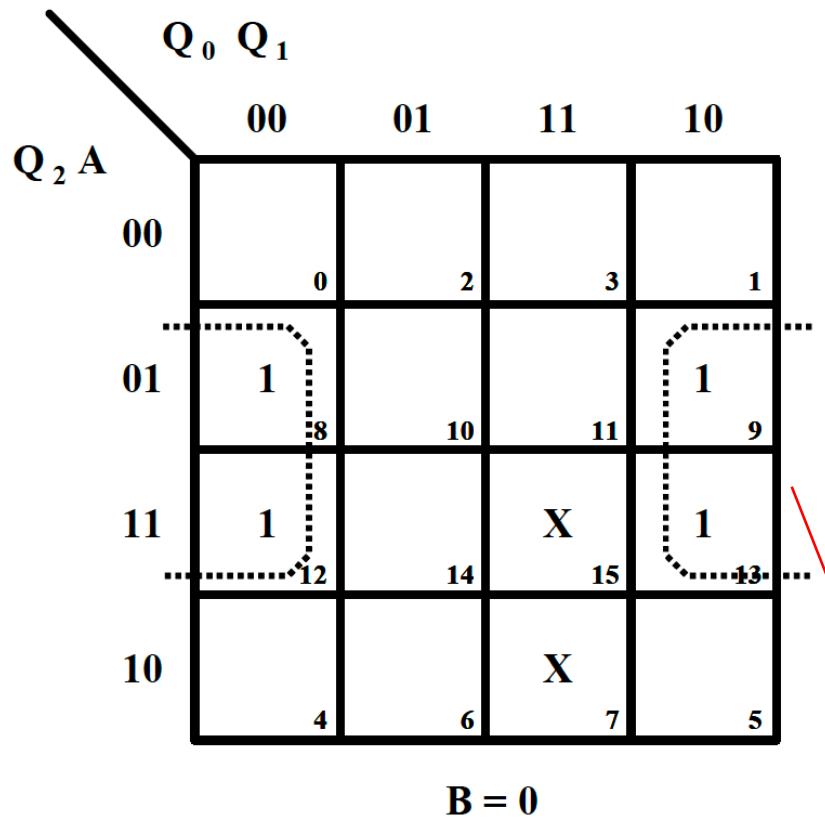
1. Se activa A y luego se desactiva
2. A continuación se activa B y se desactiva
3. Se vuelve a activar y desactivar A

Si la secuencia no es esta la cerradura permanece cerrada y el sistema vuelve al estado inicial.

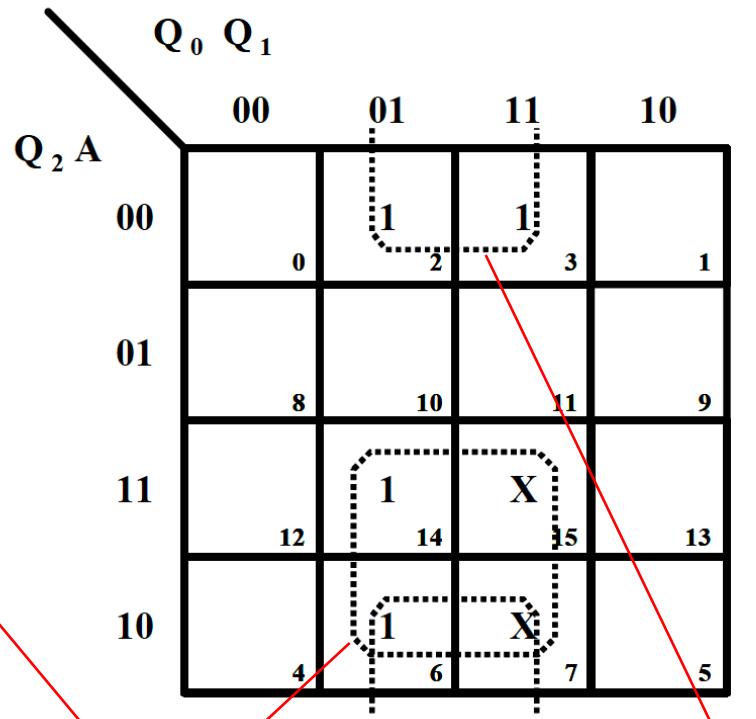
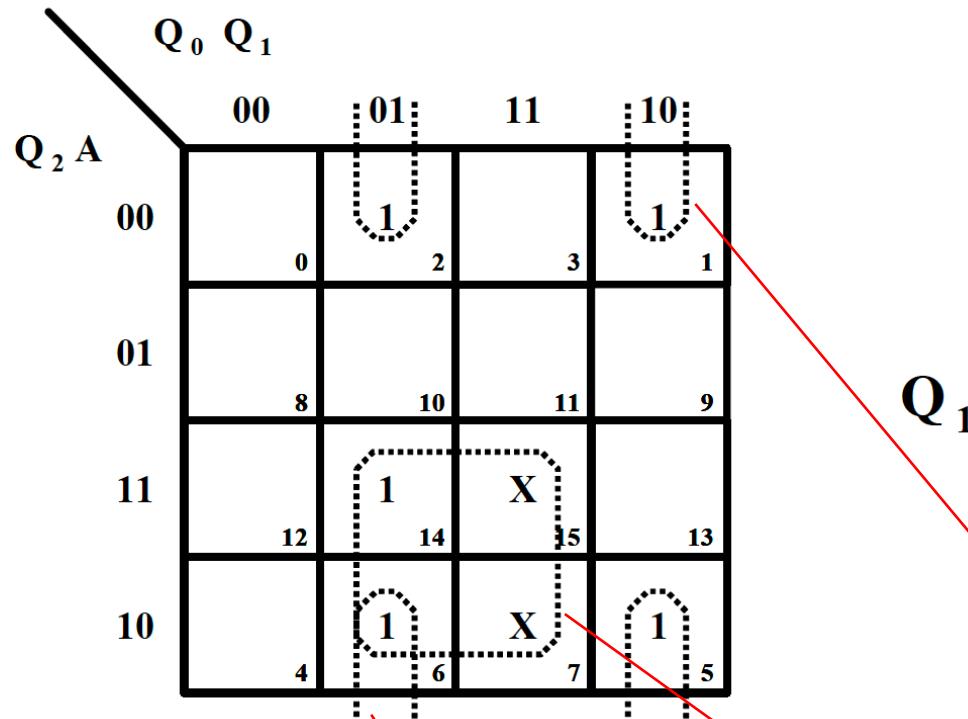


B	A	$\overbrace{Q_2 \quad Q_1 \quad Q_0}^t$			$\overbrace{Q_2 \quad Q_1 \quad Q_0}^{t+1}$			$Z = 1$
		$Q_2$	$Q_1$	$Q_0$	$Q_2$	$Q_1$	$Q_0$	
X	0	0	0	0	0	0	0	$E_0$
0	1	0	0	0	0	0	1	$E_1$
1	1	0	0	0	0	0	0	$E_2$
0	0	0	0	1	0	1	0	$E_3$
0	1	0	0	1	0	0	1	$E_4$
1	X	0	0	1	0	0	0	$E_5$
0	0	0	1	0	0	1	0	$E_6$
X	1	0	1	0	0	0	0	
1	0	0	1	0	0	1	1	
0	0	0	1	1	1	0	0	
X	1	0	1	1	1	0	0	
1	0	0	1	1	1	1	0	
0	0	1	0	0	0	0	1	
1	X	1	0	0	0	0	0	
0	1	1	0	0	0	0	1	
1	X	1	0	1	0	0	0	
0	1	1	0	1	0	1	1	
0	0	1	0	1	1	0	1	
X	X	1	1	0	1	1	1	
X	X	1	1	1	1	1	X	

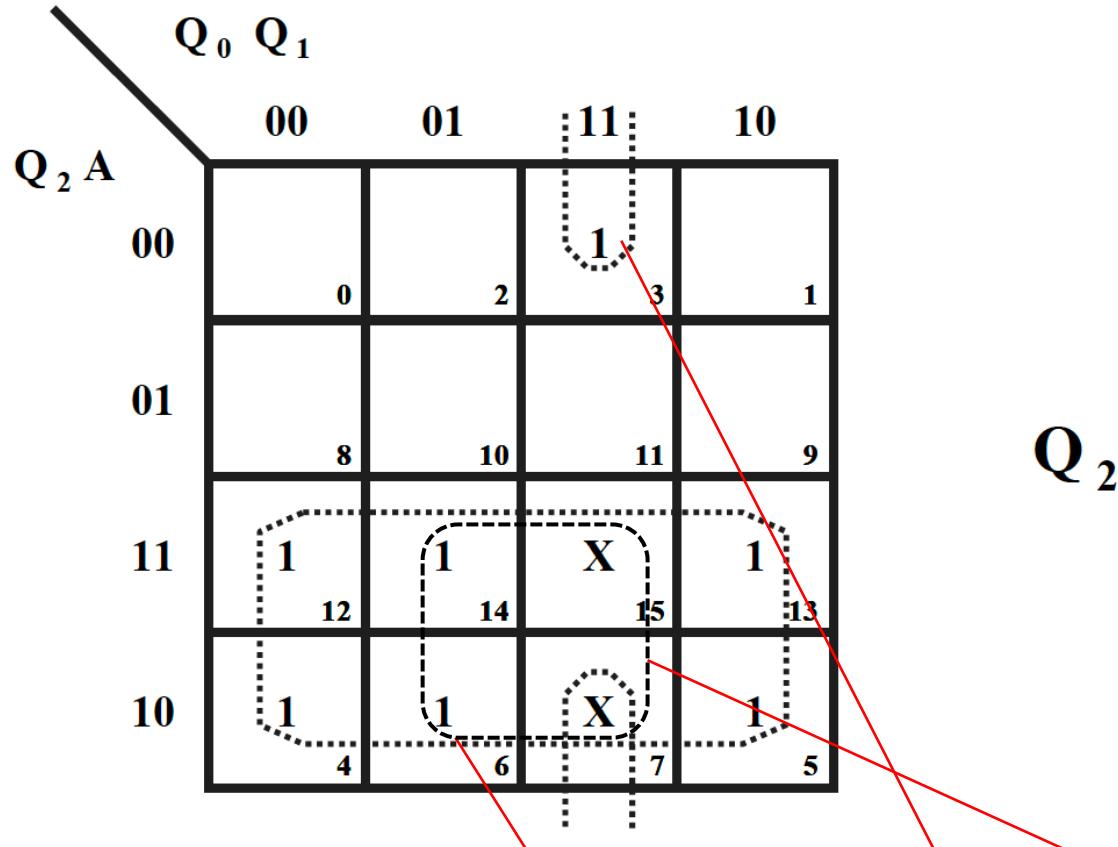
Fuente: Sistemas de automatización y autómatas programables



$$Q_0 = \bar{B} A \bar{Q}_1 + B \bar{Q}_2 \bar{A} Q_1$$

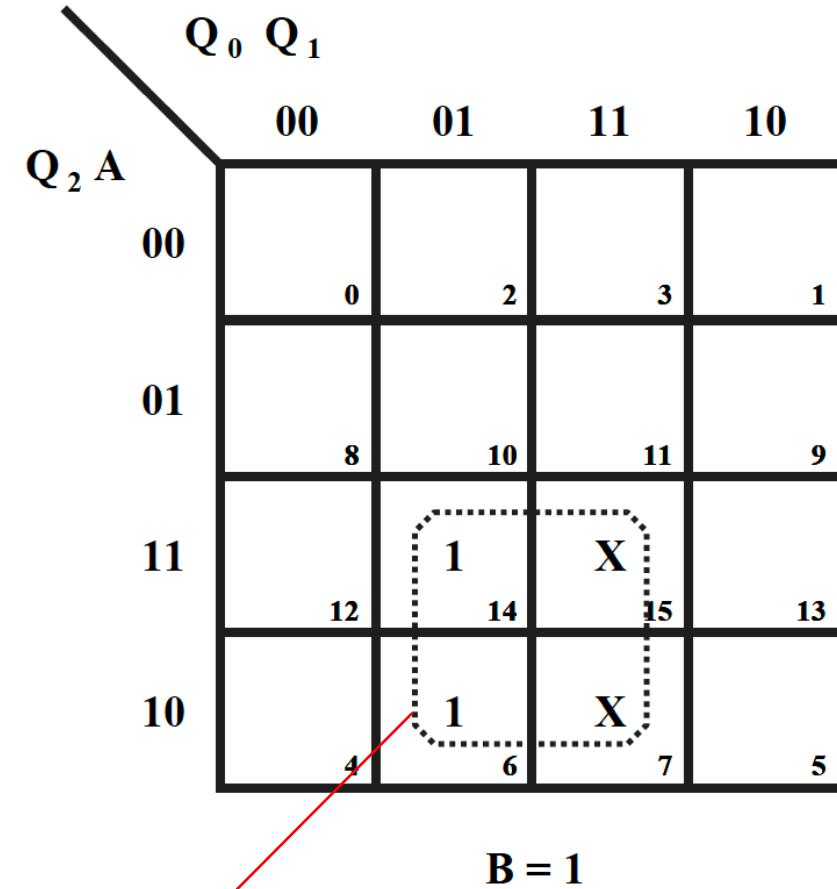


$$Q_1 = \bar{B} \bar{A} \bar{Q}_0 Q_1 + Q_2 Q_1 + \bar{B} \bar{A} \bar{Q}_1 Q_0 + B \bar{A} Q_1$$



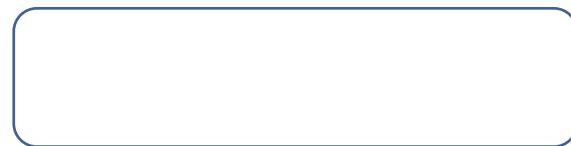
$Q_2$

$$Q_2 = \bar{B}Q_2 + \bar{B}\bar{A}Q_0Q_1 + Q_2Q_1$$

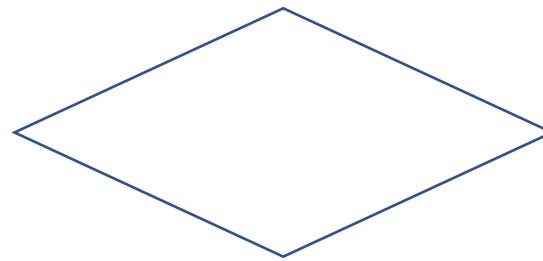


$B = 1$

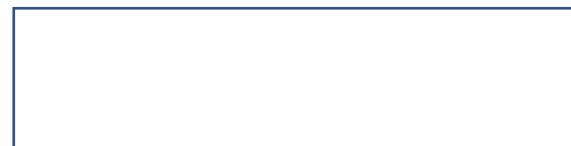
# Controladores lógicos sin unidad operativa. Secuencial. Flancos



Inicio o Fin



Toma decisión



Proceso

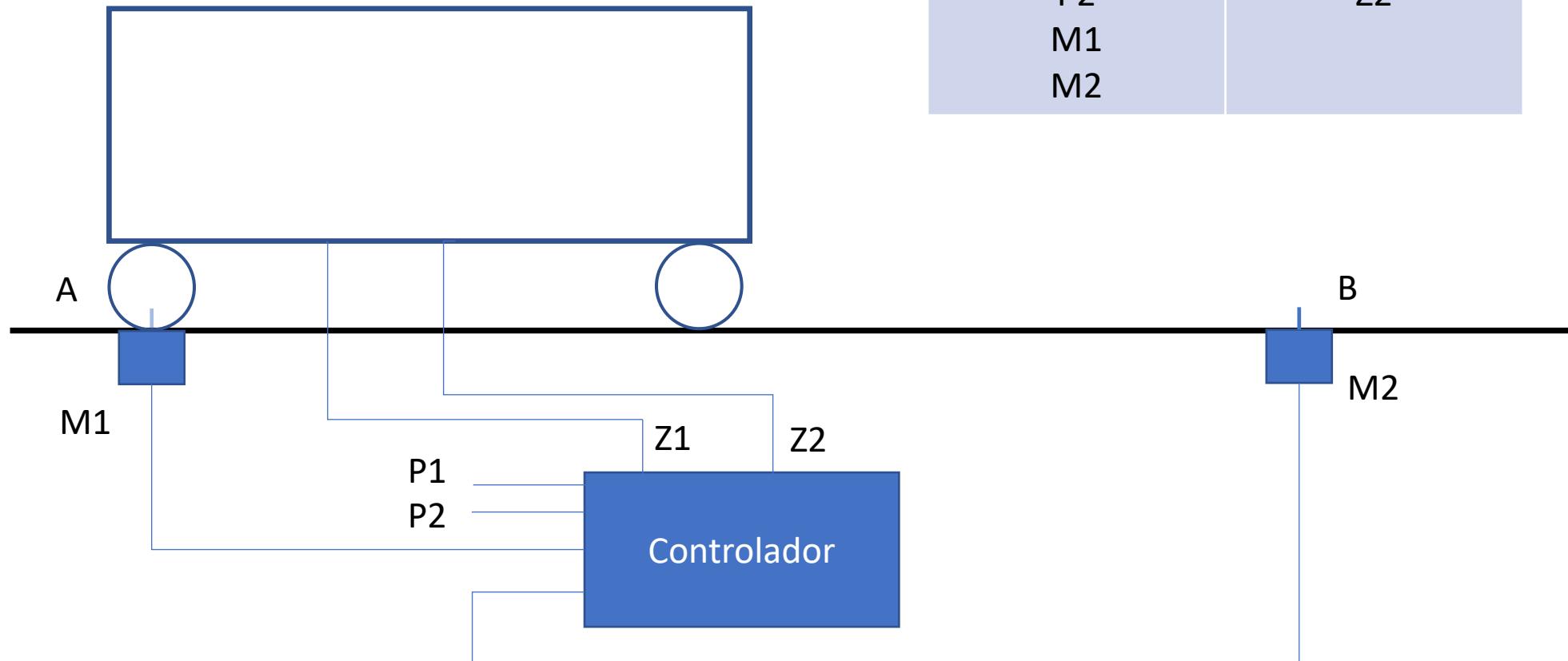


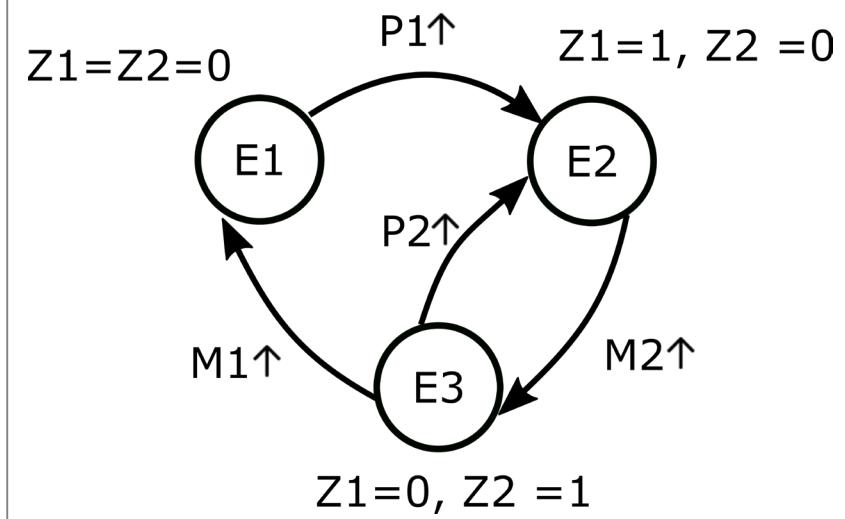
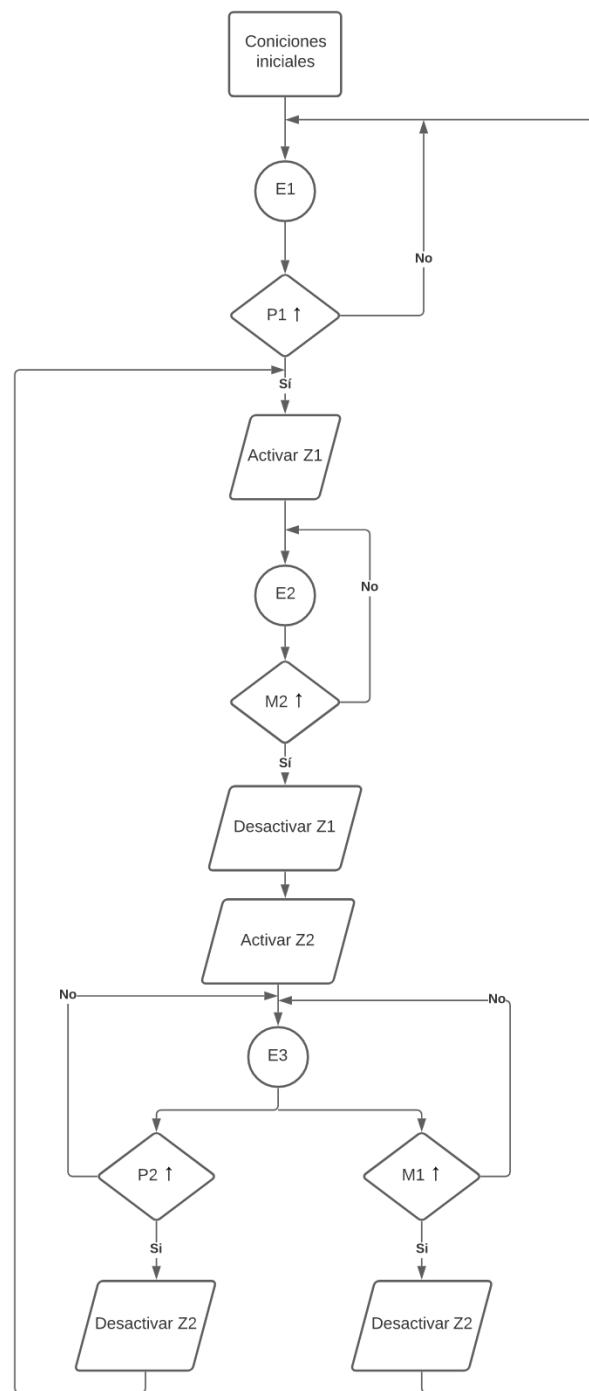
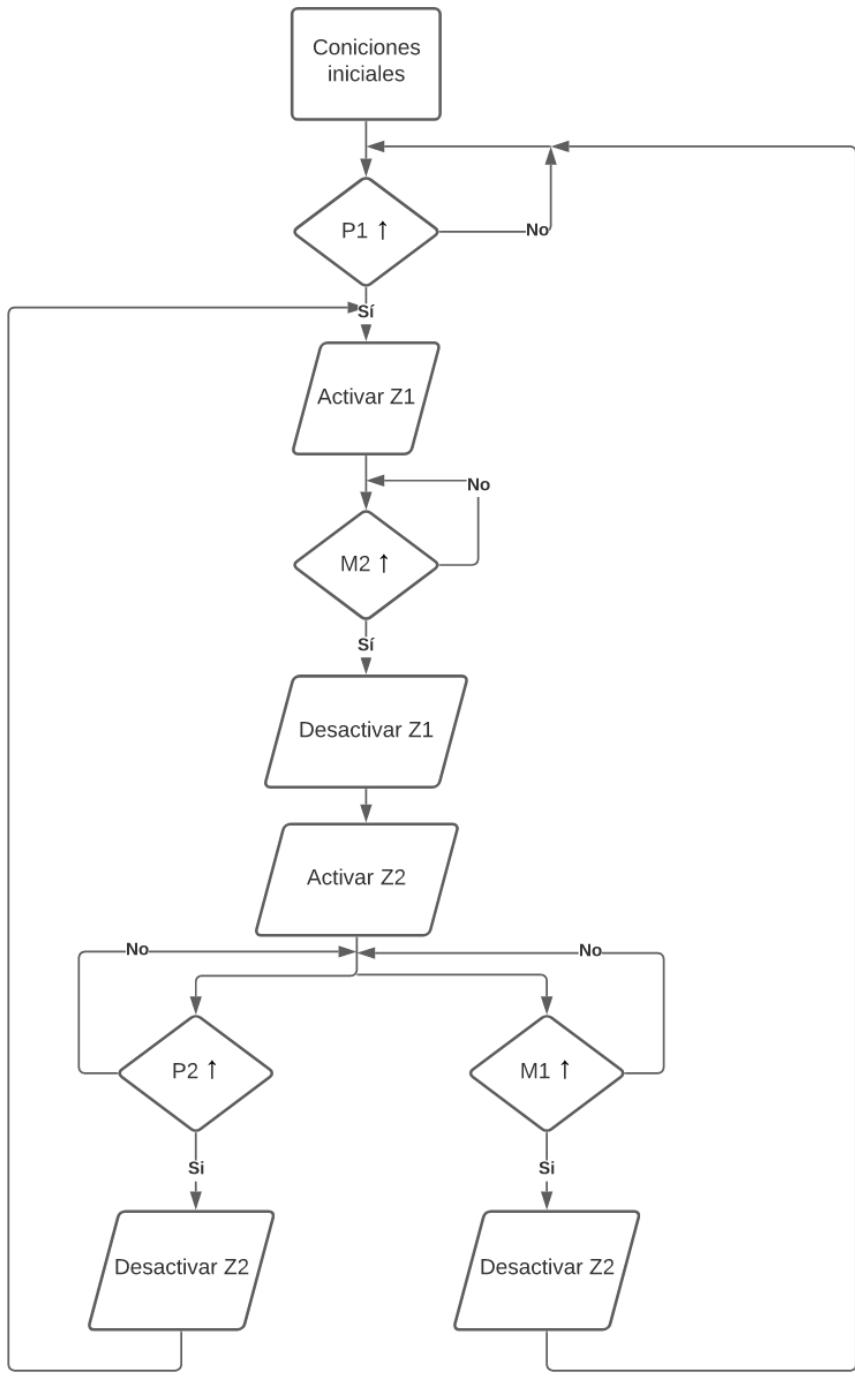
Activación o  
desactivación de  
variable

# Controladores lógicos sin unidad operativa. Secuencial. Flancos

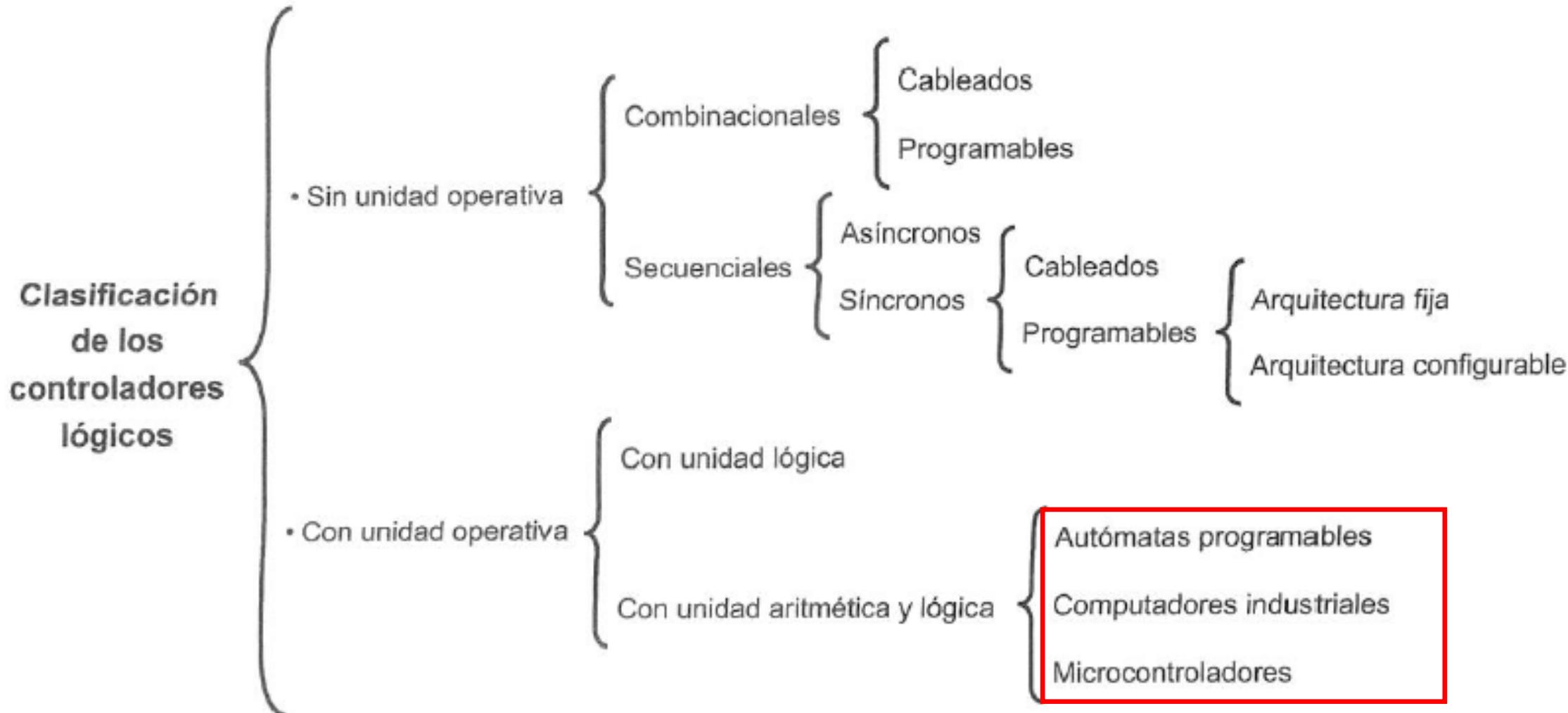
- **Ejercicio:** Obtén el diagrama de funcionamiento de un carro ha de moverse sobre unos carriles entre dos puntos A y B que vienen indicados por sendos finales de carrera M1 y M2, y puede ser controlado mediante dos pulsadores P1 y P2. En el instante inicial el carro está parado en el punto A y permanece en dicha posición hasta que se actúe sobre el pulsador P1, instante en el que debe activarse la salida Z1 que actúa sobre el motor del carro y hace que se mueva hacia el punto B. El carro continúa su movimiento hacia B aunque se actúe sobre cualquiera de los pulsadores P1 y P2. Cuando el carro alcanza el punto B, actúa sobre el final de carrera M2 lo cual hace que se active la variable Z2 y que se desactive la variable Z1 para iniciar el movimiento de retorno al punto A. Si durante dicho movimiento se actúa sobre el pulsador P2, el carro debe invertir el sentido, es decir, volver a desplazarse hacia el punto B para lo cual se vuelve a activar Z1 y se desactiva Z2. Si por el contrario no se acciona el pulsador P2, el carro continúa su movimiento hacia el punto A y se para al accionar el final de carrera M1.

# Controladores lógicos sin unidad operativa. Secuencial. Flancos





# Introducción. Controladores lógicos



# Introducción. Controladores lógicos

## LÓGICA CABLEADA

### Ventajas:

- Simplicidad
- Personal poco cualificado
- Baratos para aplicaciones sencillas

### Inconvenientes:

- Ocupan espacio
- Poca flexibilidad
- Mantenimiento costoso
- No funciones complejas

### Uso:

- Problemas de automatización sencillos

## LÓGICA PROGRAMADA

### Ventajas:

- Flexibilidad
- Ocupan poco espacio
- Mantenimiento sencillo

### Inconvenientes:

- Caros para aplicaciones sencillas
- Personal cualificado

### Uso:

- Aplicaciones con un grado de complejidad elevado

# Programmable Logic Controller

- Autómata Programable (AP) o PLC (Programmable Logic Controller): equipo electrónico diseñado para controlar un proceso secuencial en **tiempo real** y en un **ambiente industrial**.
- La lógica del autómata se hace a través de la programación del dispositivo (programa interno).

VENTAJAS	DESVENTAJAS
Modificaciones sencillas	Mayor coste inicial
Menor espacio	Personal más especializado
Mayores posibilidades de control	
Menor tiempo de puesta en marcha	

# Programmable Logic Controller

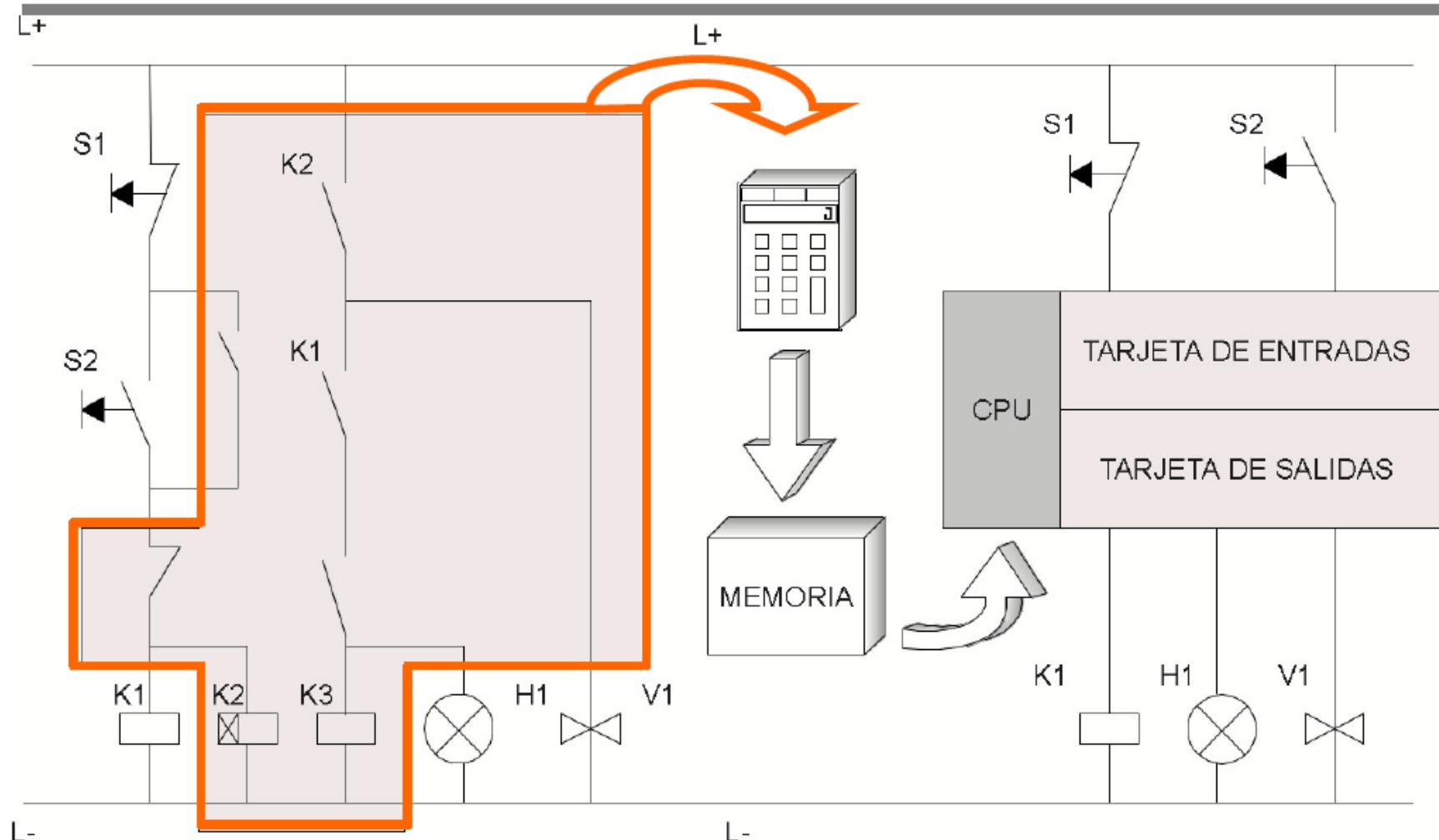
## Definición de autómata IEC 61131:

Un autómata programable (AP) es un **sistema electrónico programable** diseñado para ser utilizado en un **entorno industrial**, que utiliza una **memoria programable** para el almacenamiento interno de instrucciones orientadas al usuario, para implantar unas soluciones específicas tales como **funciones lógicas, secuencia, temporización, recuento y funciones aritméticas**, con el fin de controlar mediante entradas y salidas (**digitales y/o analógicas** – sistema híbrido) diversos tipos de máquinas y/o procesos.

**AP: Autómata programable**

**PLC: Programmable Logic Controller**

# Programmable Logic Controller



Automatismo eléctrico

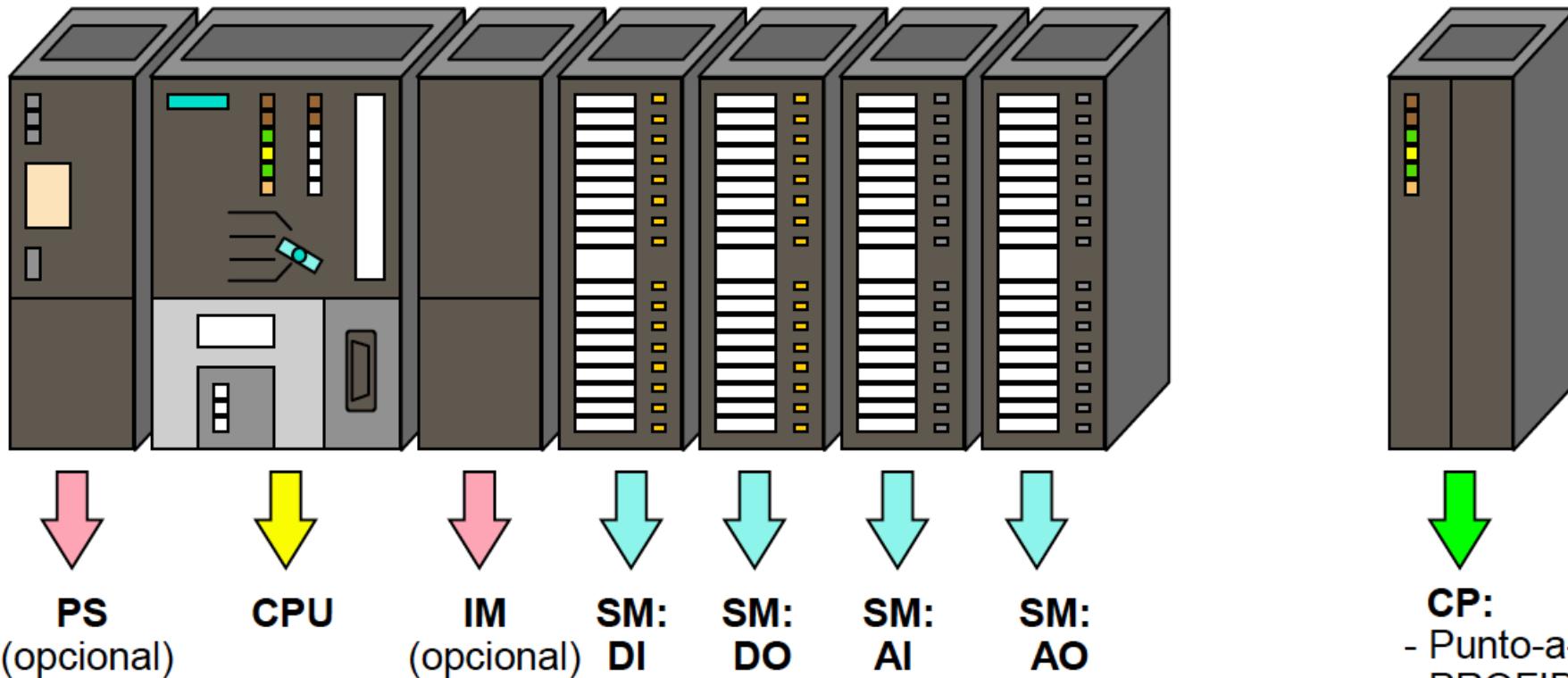
Vs.

Autómata programable

# Programmable Logic Controller

- CPU
- Fuente de alimentación (PS)
- Entradas digitales (SM-DI)
- Salidas digitales (SM-DO)
- Entradas analógicas (SM-AI)
- Salidas analógicas (SM-AO)
- Módulos de aplicaciones específicas:
  - Reguladores PID
  - Encoders
  - Buses de comunicación
  - ....

# Programmable Logic Controller



PS: Power Source

CPU: Central Process Unit

IM: Módulo interfaz

DI: Digital Input

DO: Digital Output

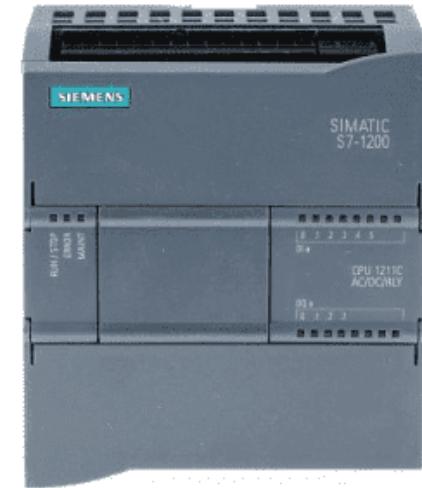
AI: Analog Input

AO: Analog Output

CP: Communications Processor

# Programmable Logic Controller

- Cuando los fabricantes hablan de CPU suelen incluir el microprocesador, memoria y un puerto de comunicación y E/S
- SIEMENS S7-1200:
  - Operación lógica:  $0.1 \mu s$
  - Transferencia de palabra de memoria:  $12 \mu s$
  - Operación punto flotante:  $18 \mu s$
- SIEMENS S7-1500:
  - Operación lógica:  $60 \eta s$
  - Transferencia de palabra de memoria:  $72 \eta s$
  - Operación punto flotante:  $384 \eta s$



200-500 €

SIEMENS CPU S7-1200

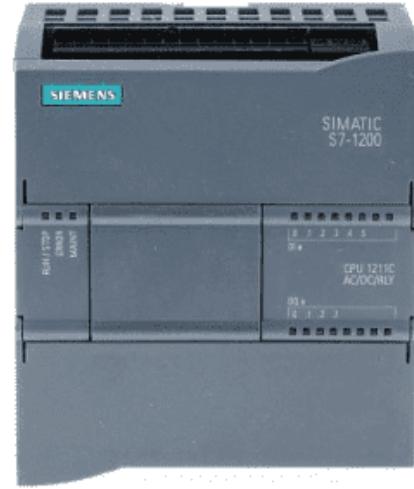


1500-  
3000 €

SIEMENS CPU S7-1500

# Programmable Logic Controller

- La capacidad de memoria del programa/datos indica el número máximo instrucciones/datos que se pueden almacenar.
  - SIEMENS S7-1200:
    - Programa: 4 KB – 8 KB
    - Datos: 25 KB – 50 KB
  - SIEMENS S7-1500:
    - Programa: 150 KB – 1 MB
    - Datos: 1 MB – 5 MB



## SIEMENS CPU S7-1200



# SIEMENS CPU S7-1500

# Programmable Logic Controller

- Configuración de las E/S:
  - Centralizadas
    - Autómatas compactos / Microautómatas
    - Autómatas modulares
  - Distribuidas:
    - Buses de campo: Costes de instalación más reducidos. Se transmite la información de forma digital (PROFIBUS, PROFINET, EHTERCAT,...)



SIEMENS S7-1500 1513



Logo! Siemens



Módulo E/S S7

# Programmable Logic Controller

- Comunicación: Enlace con sistemas externos (HMI, unidades E/S remotas,...)
  - Bus de campo: profibus, AS-i, modbus, ethercat...
  - Redes industriales: ethernet , wifi, bluetooth, zigbee
- Módulos especiales
  - Control PID
  - Posicionamiento de ejes
  - Lectura encoders



Controlador servomotor

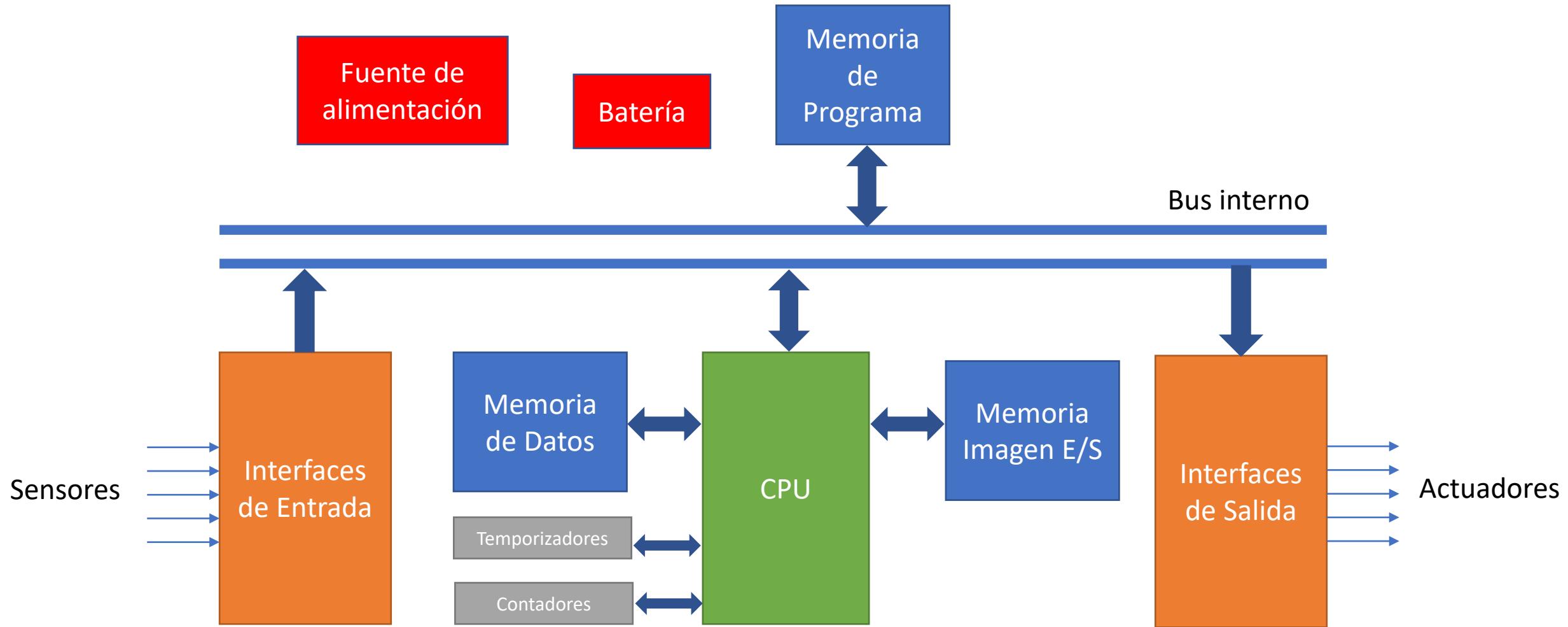
# Programmable Logic Controller

Precio

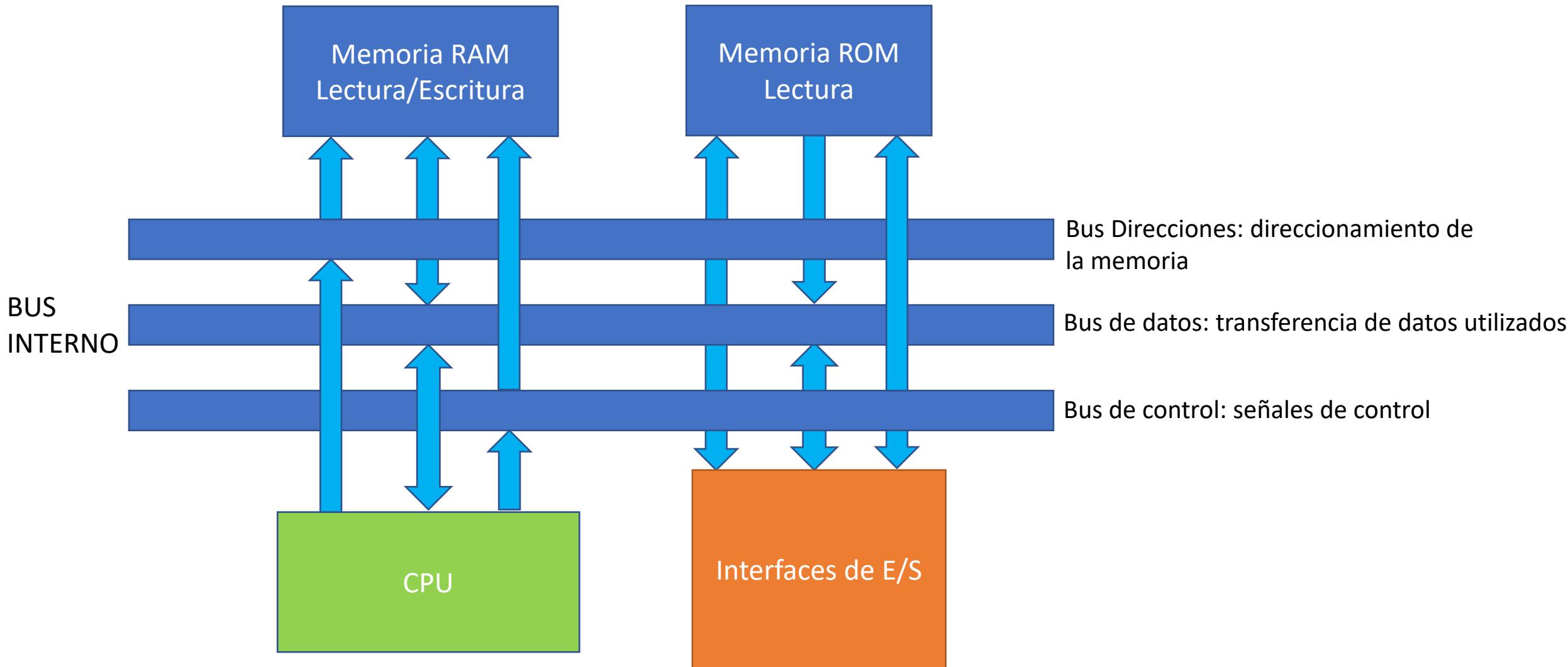


Functionalidad

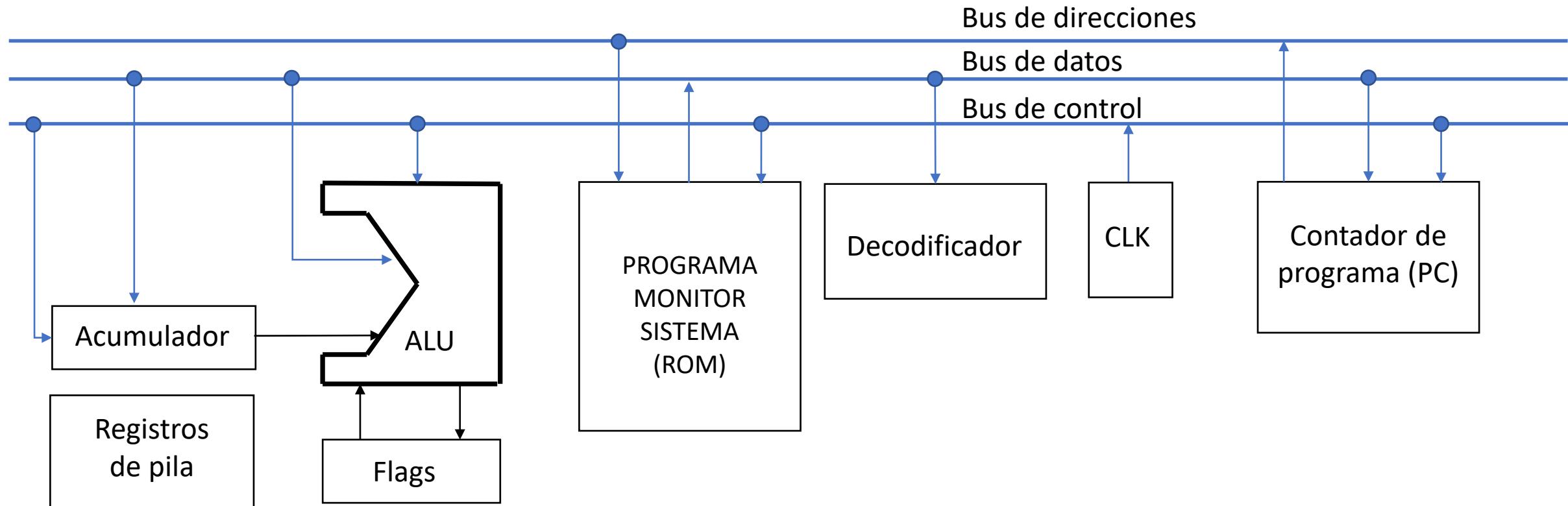
# Programmable Logic Controller. Bloques



# Programmable Logic Controller. Bloques

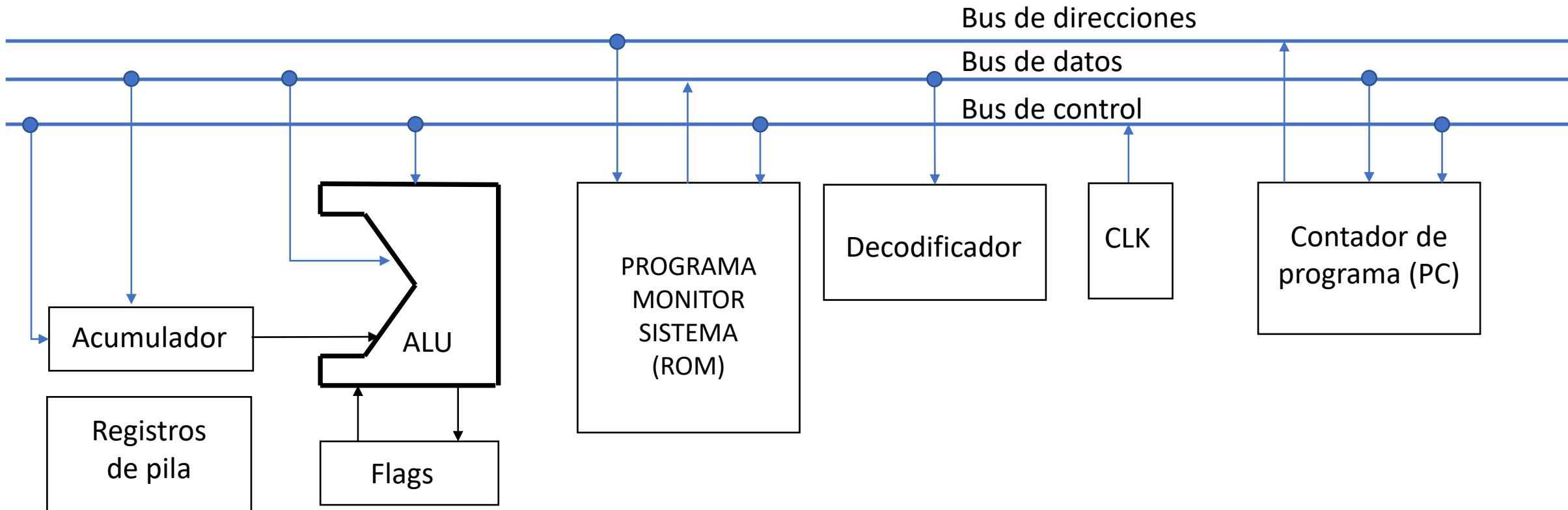


# Programmable Logic Controller. CPU



- La CPU es la encargada de ejecutar el programa del usuario y gestionar las entradas y las salidas (Microprocesador)
- Puede establecer comunicación con periféricos externos o la unidad de programación
- La CPU lee de memoria las instrucciones y realiza las operaciones indicadas

# Programmable Logic Controller. CPU

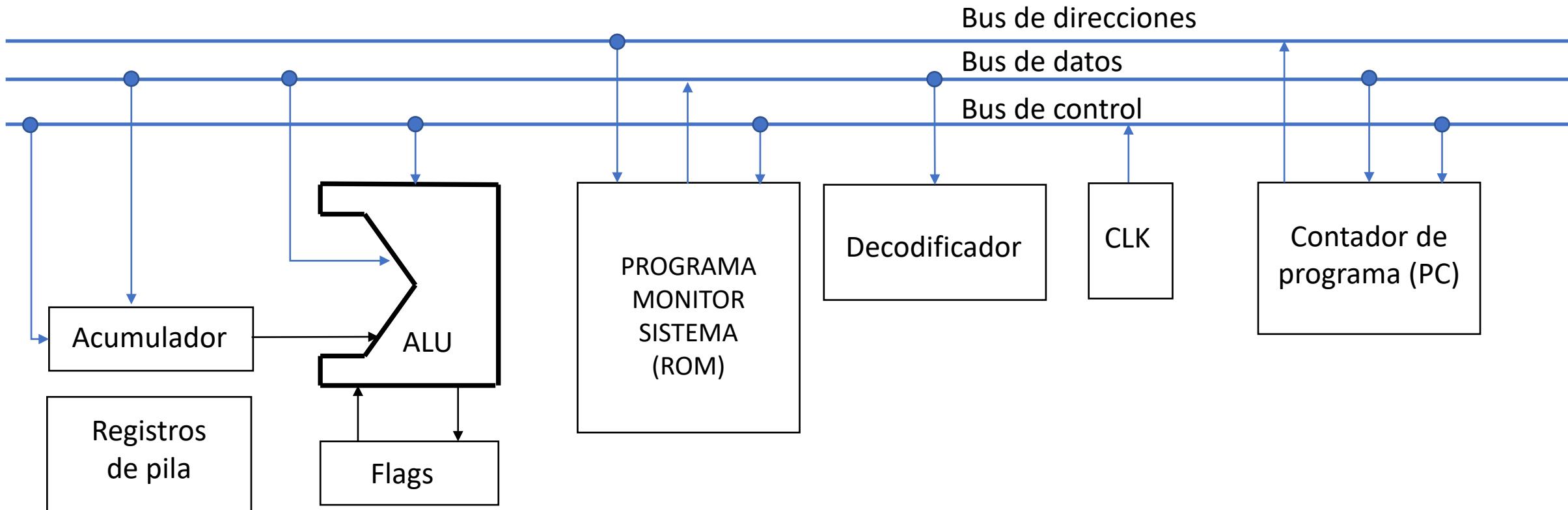


**ALU** (Arithmetic Logic Unit): Operaciones matemáticas, comparaciones, etc.

**Acumulador**: Almacena resultado ALU. // **FLAGS**: Indicadores de resultado ( $>$ ,  $<$ ,  $=0$ , ...)

**PC** (Program Counter): Encargado de lectura de instrucciones.

# Programmable Logic Controller. CPU



**Decodificador:** Decodifica las instrucciones leídas de memoria y genera las señales de control

**Programa Monitor Sistema:** instrucciones de puesta en marcha del PLC, test, ....

**CLK (Reloj):** Genera la señal de control de cambio de estados

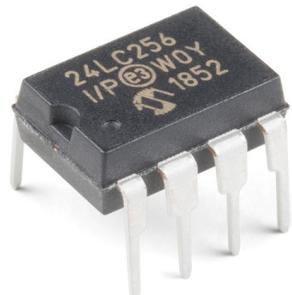
# Programmable Logic Controller. Memoria

- En la Memoria el PLC guarda los datos necesarios para realizar las tareas de control:
  - Datos del proceso: Señales de entrada, variables internas, datos numéricos.
  - Datos de control: Instrucciones del programa, configuración del PLC (número E/S, modo de funcionamiento, etc.)
- Se usan distintos tipos de memoria:
  - RAM (Random Access Memory): Memorias volátiles. La lectura y escritura tardan lo mismo.
  - ROM (Read Only Memory): Memorias establecidas por el fabricante, no se pueden borrar. Contienen el programa que chequea el estado del PLC, test, lectura escrita E/S, ...

# Programmable Logic Controller. Memoria

- Se usan distintos tipos de memoria:

- EPROM (Electrically Programmable Read Only Memory): Memoria no volátil que se puede reescribir mediante la aplicación de rayos UV a la memoria.
- EEPROM (Electrically Erasable Programmable Read Only Memory): Memoria no volátil que se puede reprogramar mediante medios eléctricos.
- FLASH: Memorias no volátiles que se pueden reprogramar por medios eléctricos (mayor integración que EEPROM).



Memoria EEPROM



Memoria FLASH



Memoria EPROM

# Programmable Logic Controller. Memoria

- Memoria de programa está dividida en dos partes:
  - Memoria que contiene el **programa ejecutivo o monitor** del PLC (“sistema operativo”): E/S del sistema, estado del PLC, etc. Su contenido lo establece el fabricante y no se puede modificar. Se implementa con EEPROM que puede ser actualizada por el usuario (nuevo firmware) o memoria ROM.
  - Memoria de control: contiene el **programa** diseñado por el **usuario** que es transferidos desde una unidad de programación (normalmente un ordenador) a la memoria. Se utilizan memorias EEPROM, RAM+BATERÍA o FLASH. Se puede reescribir durante la vida útil del PLC.

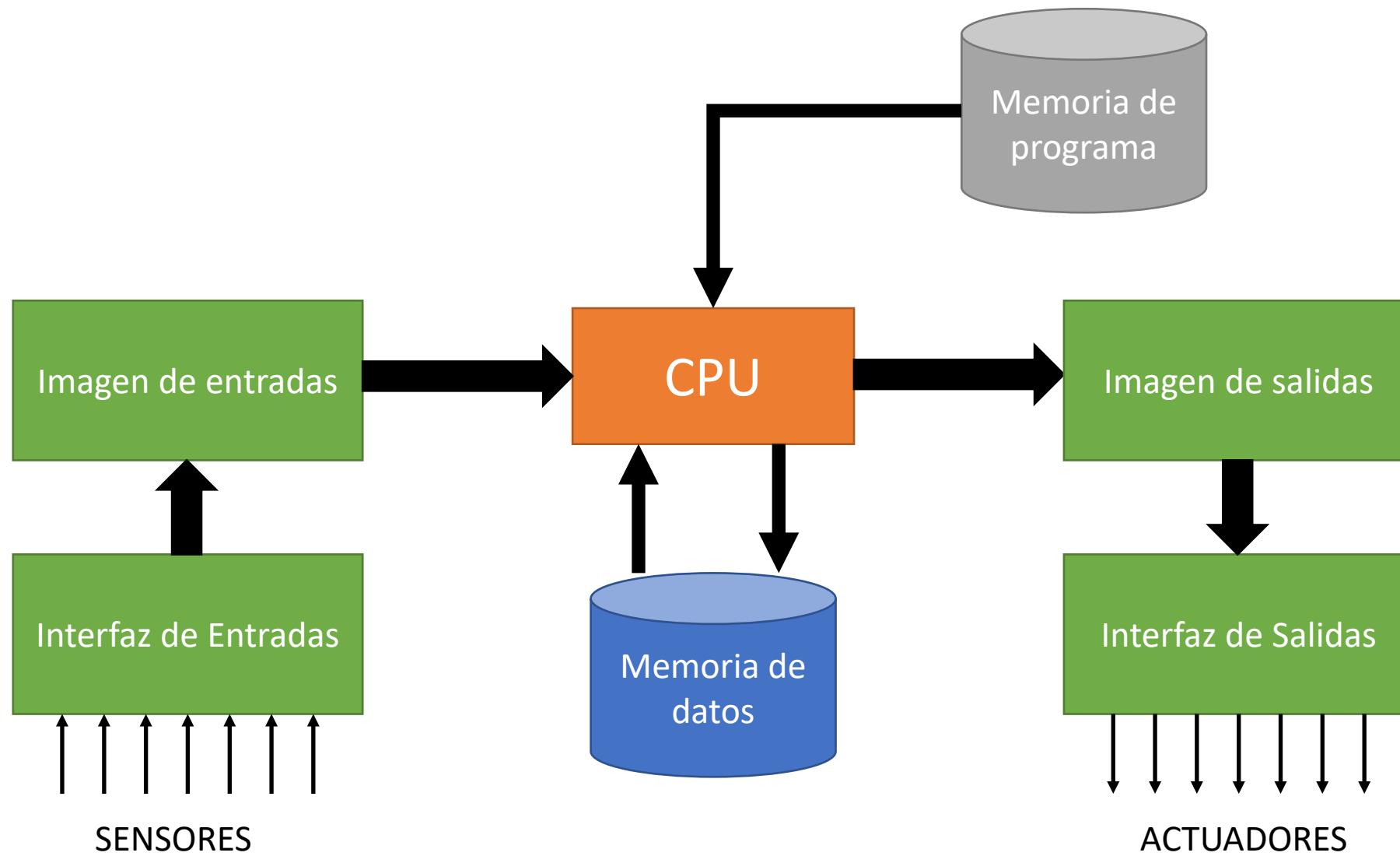
# Programmable Logic Controller. Memoria

- Memoria de datos es una memoria de acceso aleatorio que contiene:
  - Datos del programa ejecutivo
  - Memoria de E/S de las variables digitales
  - Datos numéricos de E/S analógicas
  - Variables internas

# Programmable Logic Controller. Interfaz E/S

- Las interfaces de E/S establecen comunicación entre la CPU y el proceso
- Adaptan y codifican las señales de entrada para que la CPU pueda trabajar con ellas
- Adapta las salidas del PLC para que puedan ser utilizadas por los actuadores
- Existen múltiples variantes (**TEMA 6**):
  - Digitales (1 o varios bits) / Analógicas
  - CC / CA
  - ....

# Programmable Logic Controller. Memoria



# Programmable Logic Controller. Fuente de alimentación

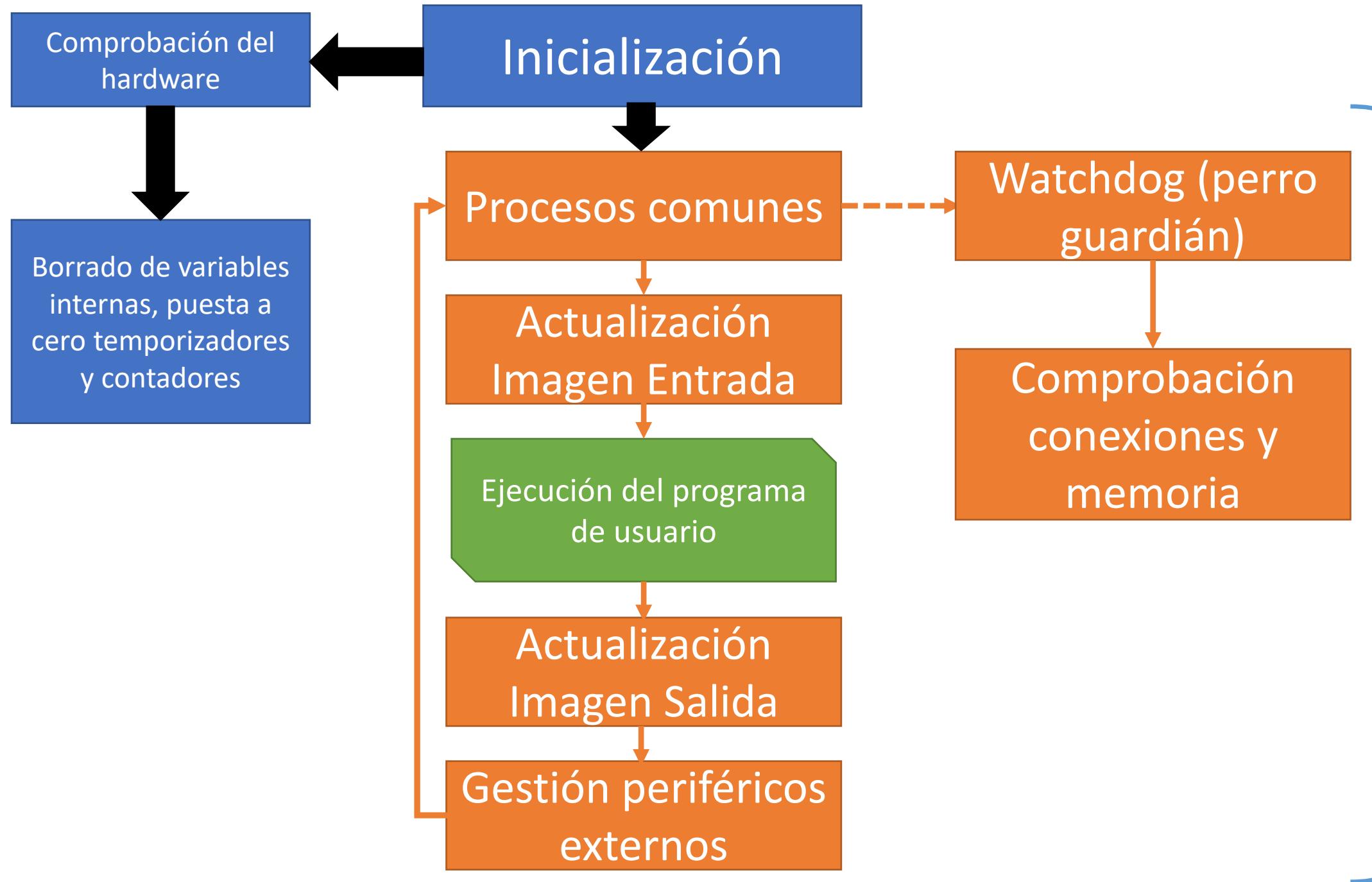
- La fuente de alimentación proporciona la energía para que funcionen los distintos componentes del PLC
- Se busca independizar la alimentación de CPU y E/S. Algunos PLCs disponen de varias fuentes de alimentación separadas para CPU y E/S.



# Programmable Logic Controller.

## Funcionamiento

- El PLC es un “ordenador” por lo que ejecuta un determinado programa almacenado en la memoria de programa
- Hay dos modos de funcionamiento del PLC:
  - Modo programación/STOP: El PLC (programa monitor) se comunica con el elemento de programación para transferir el programa (USB, ethernet). El PLC no está controlando el proceso.
  - Modo ejecución (RUN): Se ejecuta el programa del usuario (programa de control) de tal forma que el PLC controla el proceso. La ejecución del programa de control se realiza de forma cíclica. A cada ejecución del programa de control se le denomina **ciclo de scan o ciclo de trabajo**.

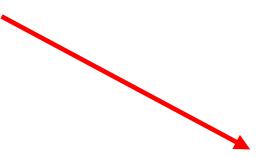


CICLO  
DE  
TRABAJO  
(CICLO DE  
SCAN)

# Programmable Logic Controller.

## Funcionamiento

- **Inicialización:** Se lleva a cabo cuando el autómata se pone en marcha. Se verifica que no haya ningún problema y en caso de que lo haya se enciende una señal luminosa. En este proceso se borran las variables internas y se ponen a cero contadores y temporizadores.
- **Procesos comunes:**
  - Watchdog (perro guardián): Inicia un contador descendente implementado en hardware que funciona en paralelo a la ejecución del programa de usuario. Si el watchdog llega a cero se detiene el PLC. Esto evita que el PLC caiga en bucles infinitos (100 ms)
  - Autodiagnóstico: Comprobación de la memoria, alimentación, buses de conexión...



```
a = 1  
do  
{....  
a =2  
}while(a!=2)
```

# Programmable Logic Controller.

## Funcionamiento

- **Actualización Imagen Entrada:** Se copian los valores de las entradas en una memoria. Estos serán los valores que se utilicen a la hora de ejecutar el programa. Los cambios en las entradas durante el ciclo de scan no se van a detectar.
- **Ejecución del programa de usuario:** Se ejecuta el programa de usuario de forma secuencial para implementar la lógica de control. Se utilizan los valores de entradas obtenidos en la etapa anterior. Se calculan las salidas.
- **Actualización Imagen Salida:** Se actualizan los valores de las salidas de los actuadores. El cambio en la salida no se produce cuando se calcula el nuevo valor (en la etapa anterior) sino cuando se actualiza la salida.

# Programmable Logic Controller.

## Funcionamiento

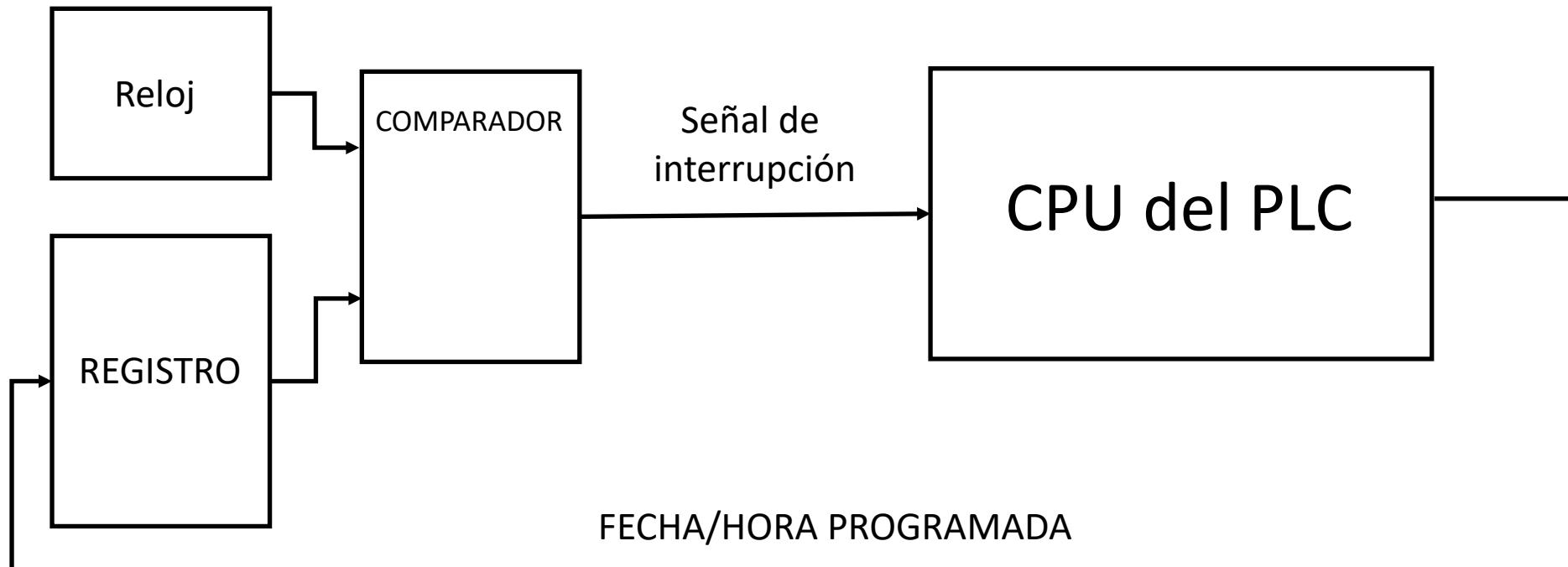
- **Gestión de periféricos:** Se produce la comunicación con otros PLC, con PCs, HMI, módulos especiales, etc.
- Este esquema de funcionamiento puede variar dependiendo del fabricante, por ejemplo algunos fabricantes actualizan al inicio del ciclo de las entradas y las salidas y luego ejecutan el programa, o la gestión de periféricos se realiza en otro instante de tiempo.

# Programmable Logic Controller. Capacidad de interrupción

- Hasta finales de los 80 la actualización de las E/S se realizaba durante el ciclo de scan, pero en determinados sistemas de en los que se necesite realizar una detección rápida de cambios en las entradas o una actuación rápida en las salidas esto no era admisible.
- A partir de la década de los 90 se introdujo la capacidad de interrupción en los PLC gracias a que las CPUs incorporaban dicha capacidad.
- Interrupción: Evento que hace que el autómata abandone el programa principal y ejecute una rutina especial que realiza unas determinadas tareas. Una vez realizadas dichas tareas el PLC sigue ejecutando el programa en el punto en el que se quedó (Procesado rápido de instrucciones)

# Programmable Logic Controller. Capacidad de interrupción

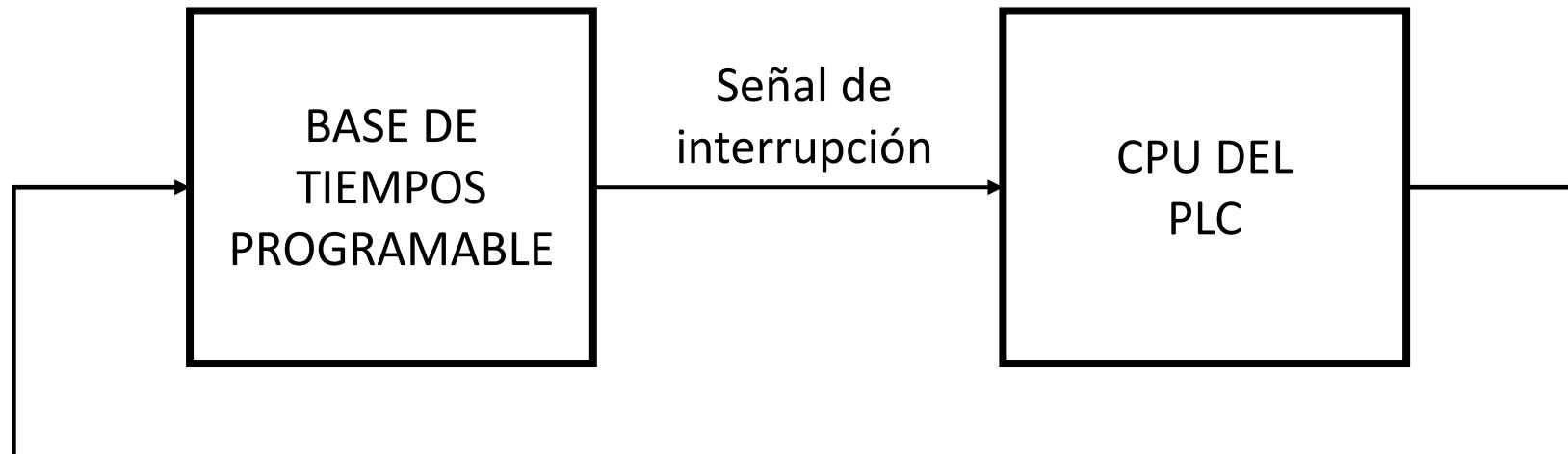
- Tipos interrupciones:
  - De reloj: Realizar determinadas tareas a una hora determinada gracias al reloj interno de la CPU.



# Programmable Logic Controller. Capacidad de interrupción

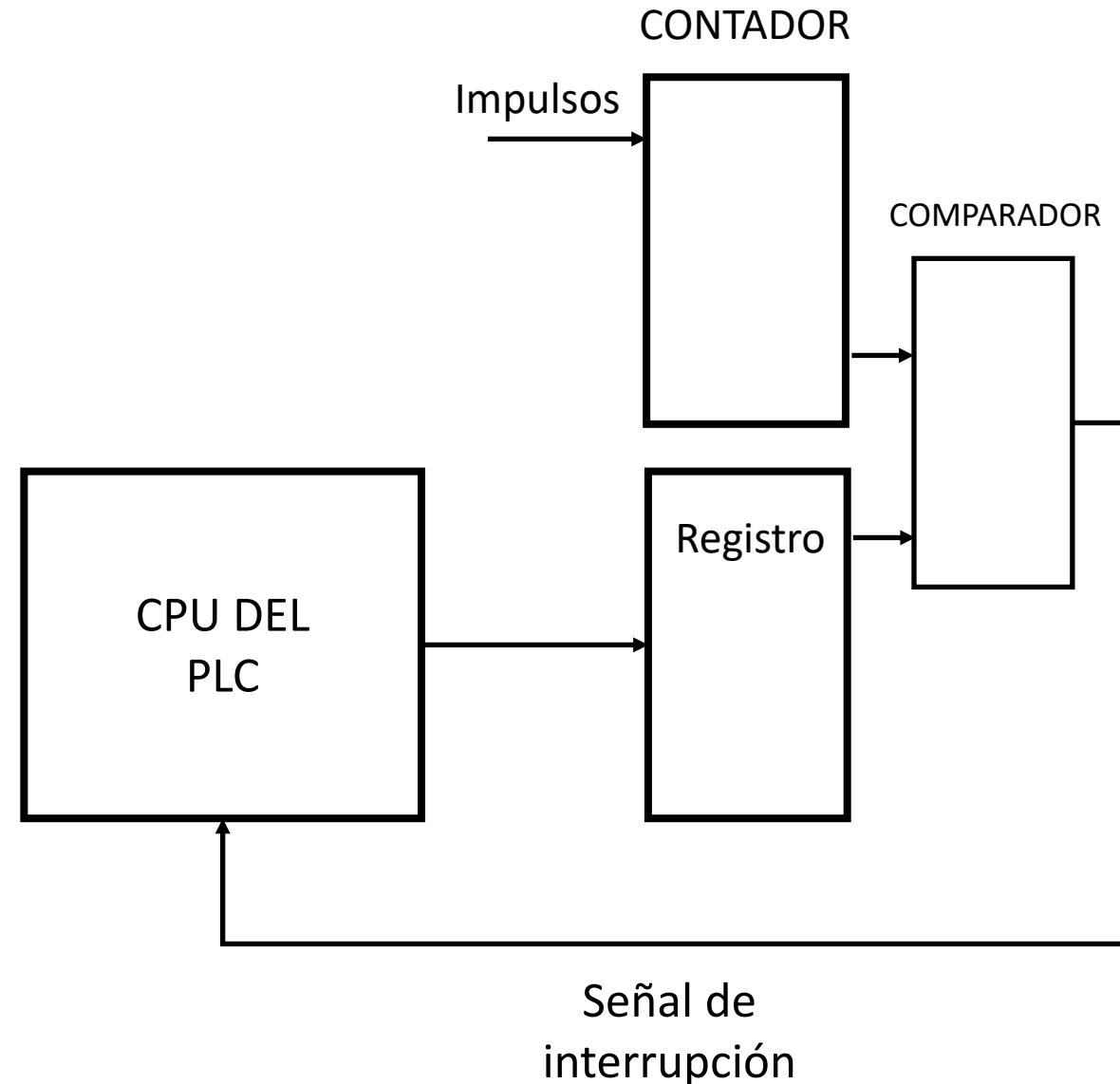
- Tipos interrupciones:

- De reloj: Realizar determinadas tareas a una hora determinada gracias al reloj interno de la CPU.
- Temporizadas: Se programa la interrupción a partir de una base de tiempos (divisor de frecuencia) para que ejecute la tarea de forma periódica.



# Programmable Logic Controller. Capacidad de interrupción

- Tipos interrupciones:
  - De reloj: Realizar determinadas tareas a una hora determinada gracias al reloj interno de la CPU.
  - Temporizadas: Se programa la interrupción a partir de una base de tiempos (divisor de frecuencia) para que ejecute la tarea de forma periódica.
  - De contador: Se lanza la interrupción cuando un contador alcanza un determinado valor.

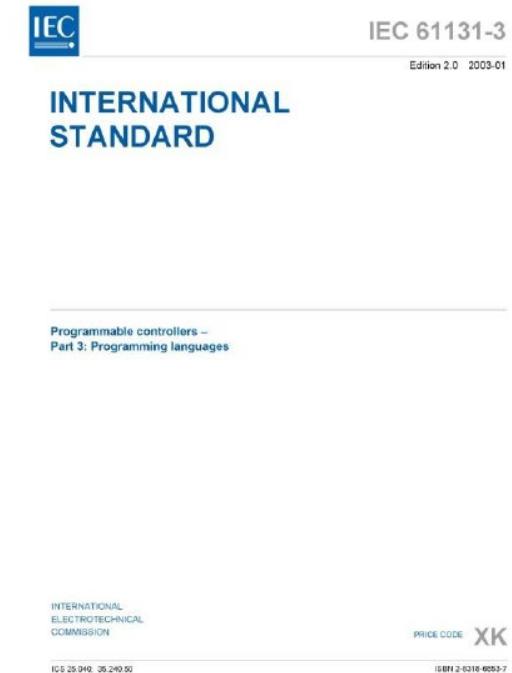


# Programmable Logic Controller. Capacidad de interrupción

- Tipos interrupciones:
  - De reloj: Realizar determinadas tareas a una hora determinada gracias al reloj interno de la CPU.
  - Temporizadas: Se programa la interrupción a partir de una base de tiempos (divisor de frecuencia) para que ejecute la tarea de forma periódica.
  - De contador: Se lanza la interrupción cuando un contador alcanza un determinado valor.
  - De comunicación: Se genera cuando un periférico genera una señal de entrada.
  - De terminal o borne: Se produce cuando una determinada entrada cambia de estado.

# ESTÁNDAR IEC 61131-3

- La Norma IEC61131 (International Electrotechnical Commission) especifica las funciones que tiene que tener una autómata programable, estandariza el modelo de software y los lenguajes de programación de los autómatas.
- Partes de la norma:
  - Parte 1: Información general
  - Parte 2: Especificaciones y ensayos de los equipos
  - Parte 3: Lenguajes de programación
  - Parte 4: Guías de usuario
  - Parte 5: Comunicaciones
  - Parte 6: Seguridad
  - Parte 7: Control borroso (fuzzy)
  - Parte 8: Guías de programación
  - Parte 9: Comunicación sensores y actuadores pequeños
  - Parte 10: PLC open XML



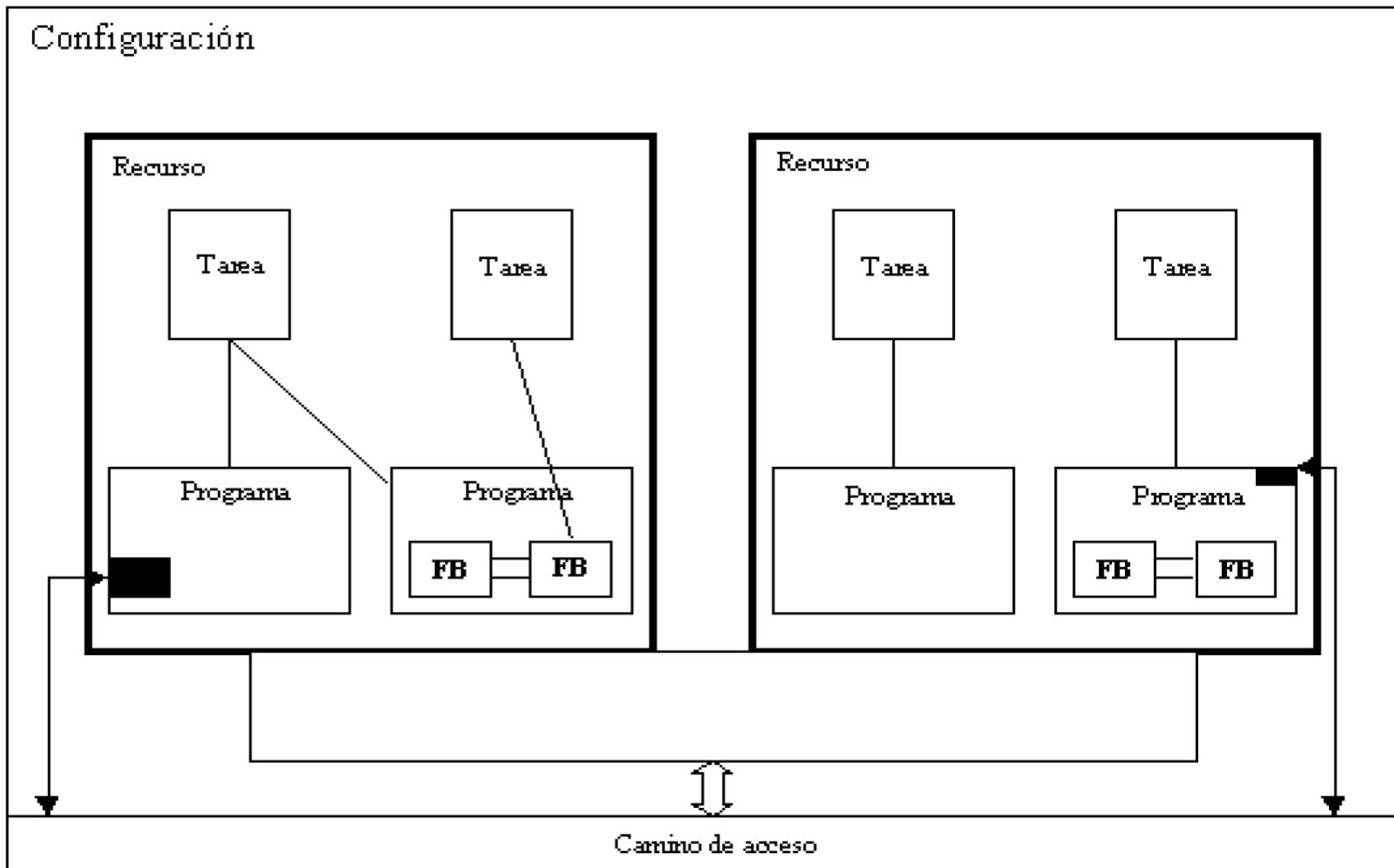
# ESTÁNDAR IEC 61131-3. Parte 3

- Define los lenguajes de programación y las reglas sintácticas y semánticas, juegos de instrucciones fundamentales
- Dividido en:
  - Elementos comunes:
    - Tipos de datos y variables
  - Lenguajes de programación

# ESTÁNDAR IEC 61131-3

- Elementos comunes:
  - Tipos de datos: booleanos, enteros, reales, byte (8 bits), palabra (word-16 bits), cadenas de caracteres, fecha, hora del día, canal analógico de entrada, ..., **tipos de datos derivados** (creados por el usuario)
  - Variables: Asignan direcciones del hardware: E/S, memoria y datos. **Locales o globales**. Hacen la programación independiente del hardware.

# ESTÁNDAR IEC 61131-3



**FB**  
BLOQUE FUNCIONAL

# ESTÁNDAR IEC 61131-3

- Una **configuración** es el elemento software requerido para solucionar un problema de control particular: hardware, procesadores, canales I/O...
- Dentro de una configuración se pueden definir uno o más **recursos** (procesadores que ejecutan programas): procesador capaz de ejecutar una tarea. En un recurso pueden estar definidas una o más **tareas**.
- Las tareas controlan la ejecución de un conjunto de programas y/o bloques funcionales (FB).

# ESTÁNDAR IEC 61131-3

- Un programa es una interacción de **funciones** y **bloques funcionales**.
- Una función es una unidad de organización que al ser ejecutada proporciona exactamente un elemento de datos (lenguajes literales):

$$X := \text{COS} (Z) + \text{SIN} (Y)$$

- Las funciones **no contienen información del estado interno** del sistema (para la misma entrada debe proporcionar la misma salida).

# ESTÁNDAR IEC 61131-3

- Funciones estándar: ADD, ABS, SQRT, SIN, COS, AND, OR,....
- Funciones definidas por el usuario:

*FUNCTION OperacionCombinada:REAL*

*Nombre*

*VAR\_INPUT*

*A, B, C : REAL*



*Variables*

*END\_VAR*

*OperacionCombinada := A\*B/C;*

*Cuerpo*

*END\_FUNCTION*

# ESTÁNDAR IEC 61131-3

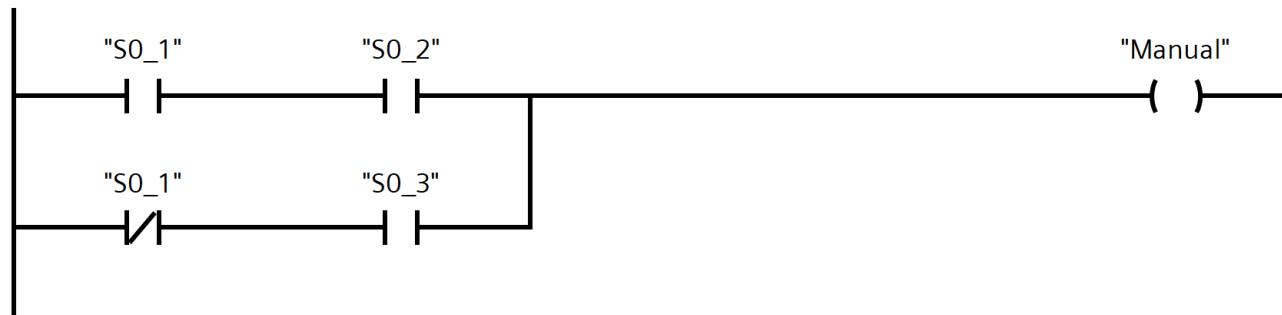
- Los bloques funcionales (FBs) representan funciones de control especializadas. Los FBs tienen datos e instrucciones y **pueden guardar el valor de las variables** (diferencia con respecto a funciones). Pueden ser estándar (biestables, contadores, temporizadores, PID,...) o definidos por el usuario
- Cada llamada de un FB se denomina instancia
- Dos instancias del mismo tipo pueden dar distinta salida ante la misma entrada (dependencia de la historia del sistema).

# ESTÁNDAR IEC 61131-3. Lenguajes de programación

- Lenguajes gráficos:
  - Diagrama de Escalera (Ladder Diagram - LD)
  - Diagrama de bloques funcionales (Function Block Diagram – FBD)
- Lenguajes literales:
  - Lista de instrucciones (Instruction List – IL)
  - Texto estructurado (Structured Text – ST)
- La elección del lenguaje depende de los conocimientos de la persona que programa, el problema a tratar, nivel de descripción del proceso, coordinación con otros departamento, etc.

# ESTÁNDAR IEC 61131-3. LD-KOP

- Proporciona un conjunto estandarizado de símbolos de programación tipo “relé en escalera” (LD: Ladder; KOP: Kontakt Plan).
- Nace para su uso por parte de los ingenieros de control en EE.UU (electricistas).

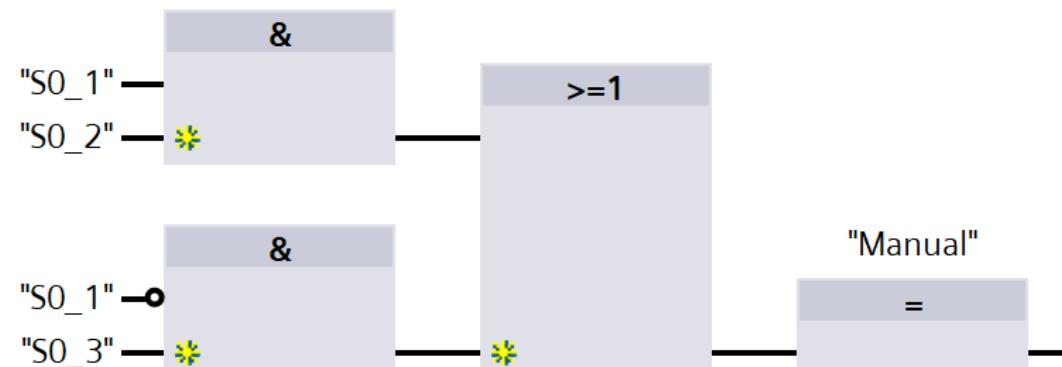


$$\text{Manual} = S_{01} \cdot S_{02} + S_{01}' \cdot S_{03}$$



# ESTÁNDAR IEC 61131-3. FBD o FUP

- Lenguaje usado en Europa (FBD: Function Block Diagram; FUP: Funktionplan).
- Conexión de bloques: similar a la representación de los circuitos electrónicos de puertas lógicas.
- Usado en aplicaciones que impliquen flujo de información o datos entre sistemas de control.



$$\text{Manual} = S_{01} \cdot S_{02} + S_{01}' \cdot S_{03}$$



# ESTÁNDAR IEC 61131-3. IL - AWL

- Basado en un acumulador que actúa como pila. (AWL: Anweisungliste, IL-Instruction List).
- Programación tipo ensamblador
- Cada línea implica una única operación sobre uno o dos operandos

1	A	"S0_1"
2	AN	"S0_2"
3	=	"Manual"



$$\text{Manual} = \text{S0\_1} \cdot \text{S0\_2}'$$

# ESTÁNDAR IEC 61131-3. IL

- Lenguaje de alto nivel estructurado en bloques (ST- Structured Text).
- Similar a PASCAL.
- Permite expresiones complejas y anidadas.
- Soporta: bucles (repeat-until, while-do), condicionales (IF-THEN-ELSE, CASE) y funciones (SQRT(), SIN(),....).

---

```
1 "Manual" := ("S0_1" AND "S0_2") OR (NOT "S0_1" AND "S0_3");
```

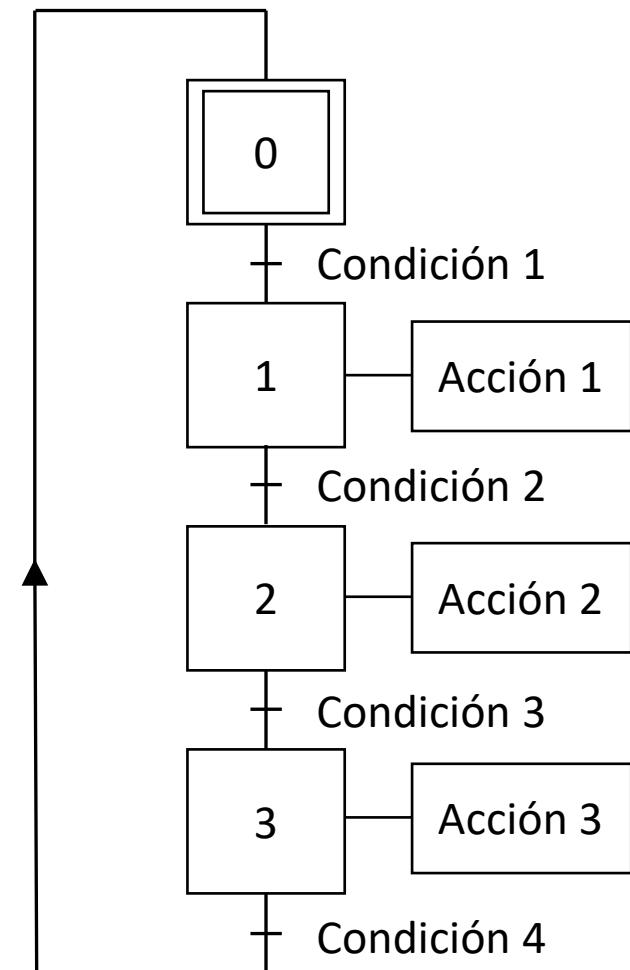
$$\text{Manual} = S_01 \cdot S_02 + S_01' \cdot S_03$$

---

```
1 IF ("S0_1" AND "S0_2") OR (NOT "S0_1" AND "S0_3") THEN
2   "Manual" := TRUE;
3 ELSE
4   "Manual" := FALSE;
5 END IF;
```

# ESTÁNDAR IEC 61131-3. Diagrama Funcional Secuencial (SFC)

- Representación gráfica del comportamiento secuencial del programa de control.
- Parte el problema de control.
- Facilita diagnóstico de problemas en el algoritmo de control.
- Elementos: Etapas con bloque de acción y transiciones.



# Organización de programas

- Cuando se crea un programa de usuario las instrucciones se incluyen en bloques lógicos (OB, FB o FC). Por defecto está disponible el bloque de organización Main [OB1]



FC (Función)  
No tiene datos de instancia



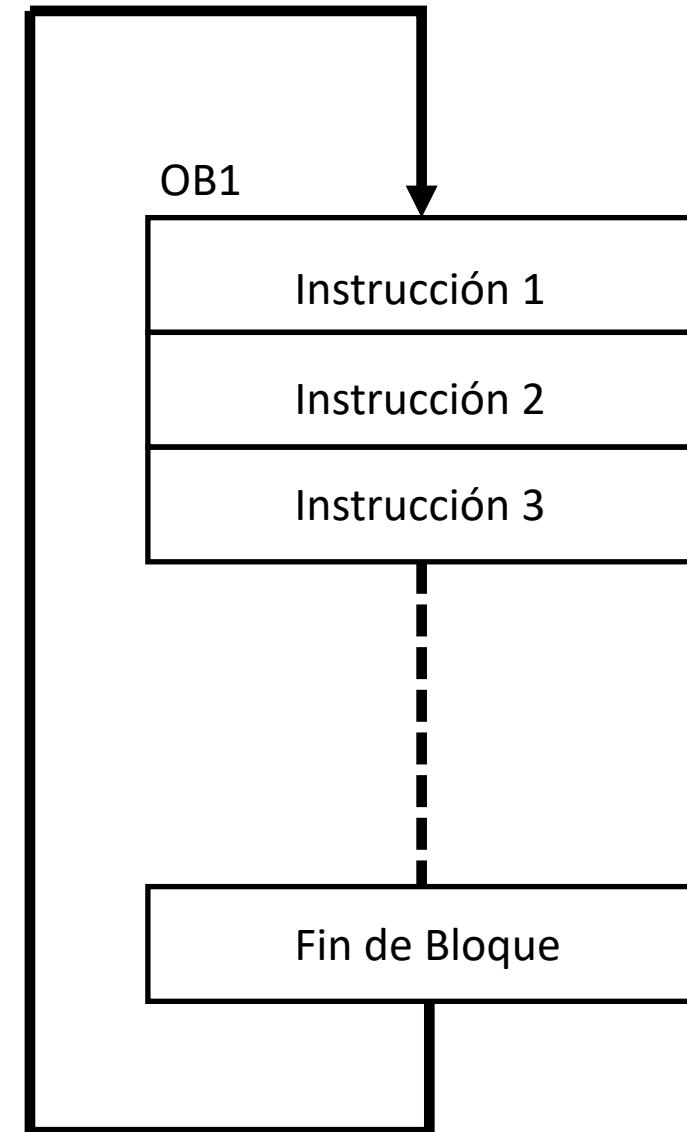
FB (Bloque de Función)  
Puede incluir un bloque de datos de instancia para almacenamiento temporal



DB (Bloque de Datos)  
Acceso global y almacenamiento permanente

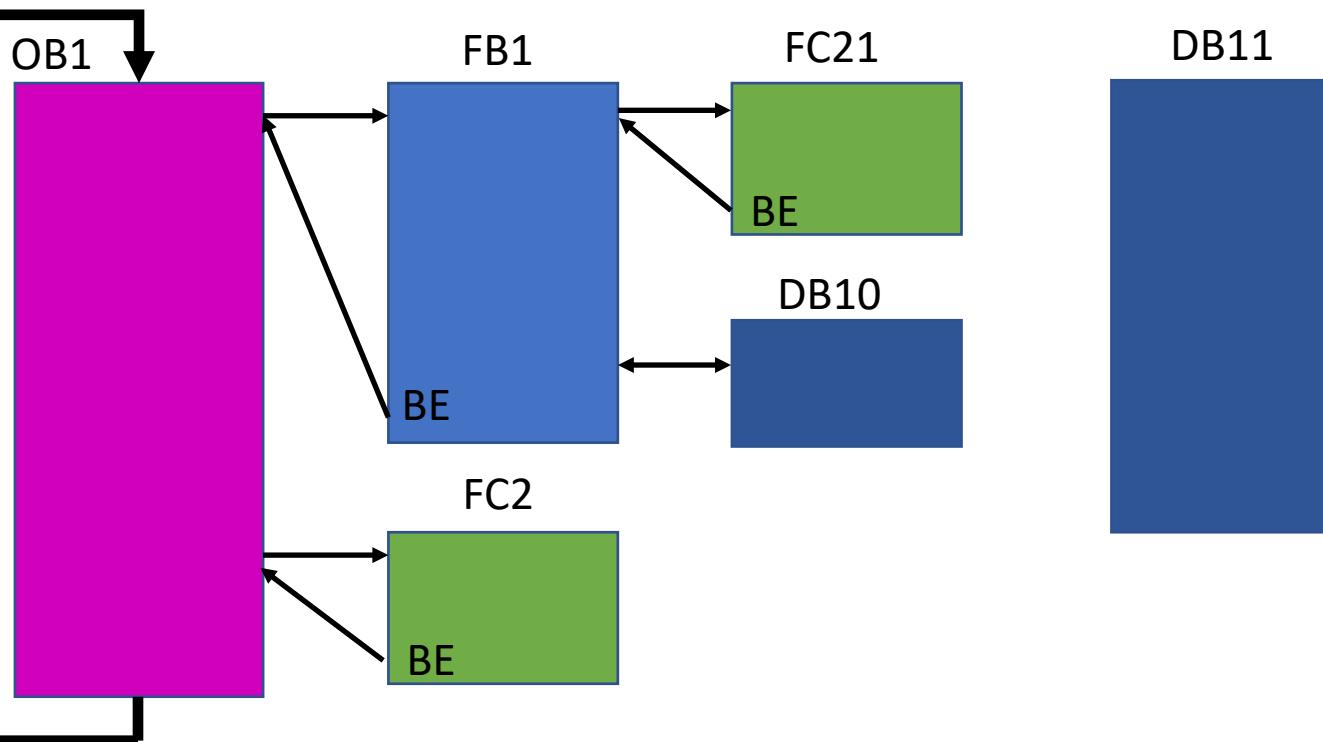
# Organización de programas

- **Programación lineal:** Un programa lineal ejecuta todas las instrucciones de la tarea de forma secuencial (una a continuación de otra). Todas las instrucciones están en un OB (Organization Block) que se encarga de ejecutar el programa. En SIEMENS este es el OB1 (= loop de ARDUINO)



# Organización de programas

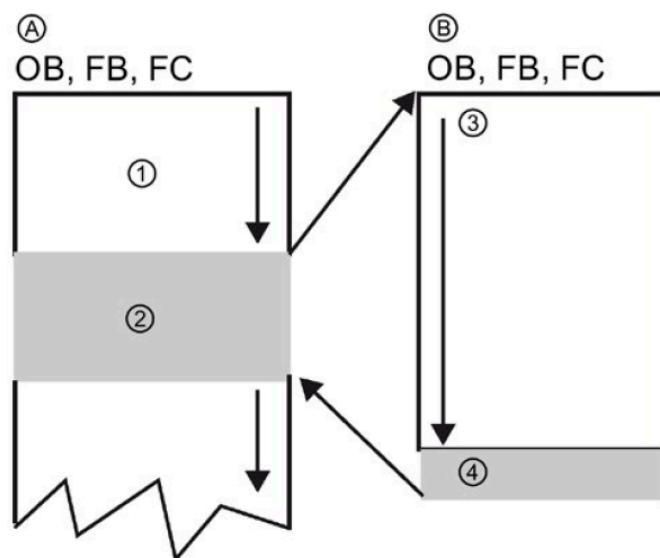
- **Programación estructurada:** Para abordar tareas complejas el programa se divide en bloques más pequeños. Los bloques de programa deben ser llamados desde un bloque de orden superior. Si se detecta un fin de bloque (BE – Block End) el programa continua desde el bloque que llamó, justo a continuación de la llamada.



OB: Organization Block  
FB: Function Block  
FC: Function  
DB: Data Block

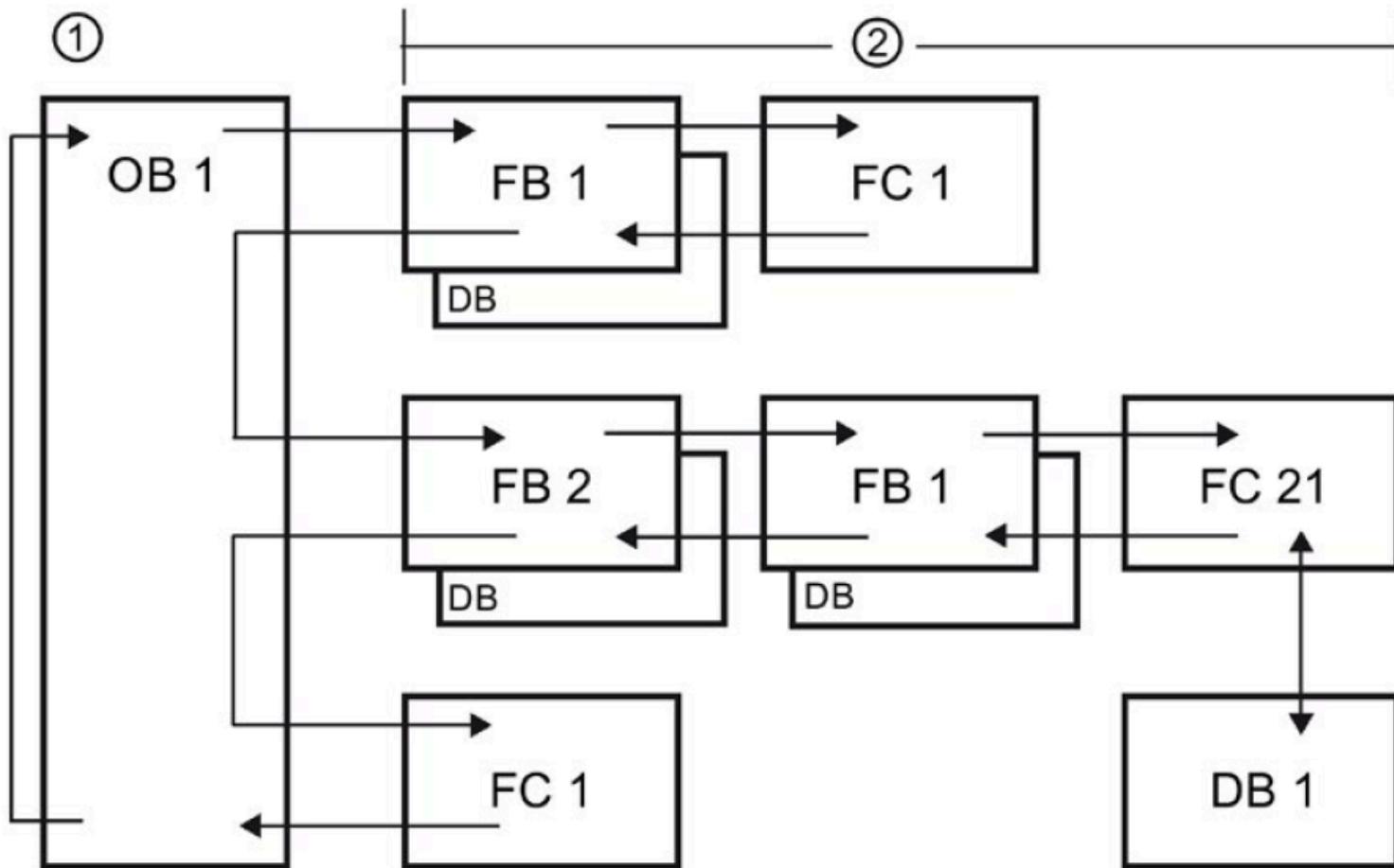
# Organización de programas

- **Programación estructurada:** Se diseñan FBs y FCs que ejecuten tareas genéricas (bloques lógicos modulares). El programa de usuario se estructura de tal forma que otros bloques llamen a estos bloques lógicos modulares. Cuando un bloque lógico llama a otro bloque lógico la CPU ejecuta la lógica del bloque llamado. Una vez finalizada la ejecución la CPU reanuda la ejecución del bloque que ha efectuado la llamada.



- A Bloque que llama
- B Bloque llamado (o que interrumpe)
- ① Ejecución del programa
- ② Instrucción o evento que inicia la ejecución de otro bloque
- ③ Ejecución del programa
- ④ Fin del bloque (regresa al bloque que llama)

# Organización de programas



**1 Inicio del ciclo**

**2 Profundidad de anidamiento**

# Organización de programas

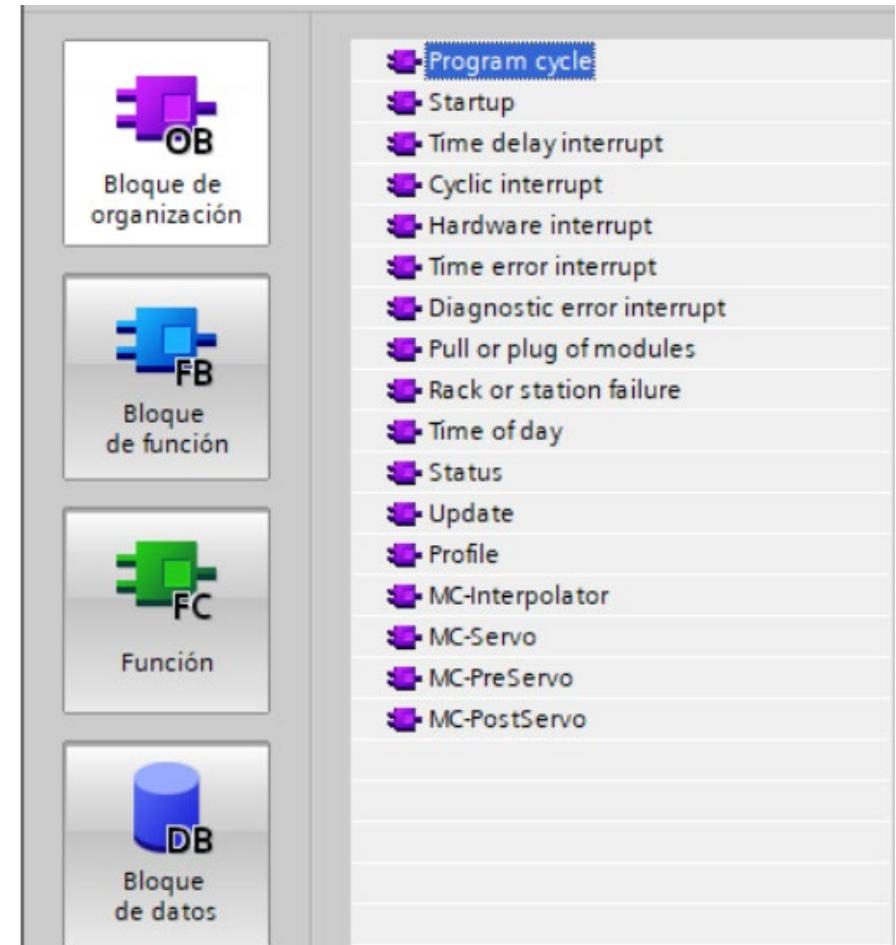
- **Programación estructurada:** Los bloques lógicos se pueden reutilizar, simplificando el diseño e implementación del sistema de control:
  - Bloques lógicos reutilizables para tareas estándar (control de bomba o motor). Se pueden crear librerías para poder llamar a estos bloques lógicos.
  - El programa de control se puede dividir en componentes modulares para las tareas funcionales, esto facilita la comprensión y gestión. Facilita modificaciones posteriores.
  - Se simplifica la depuración del programa de control.

# Organización de programas. Tipos de módulos

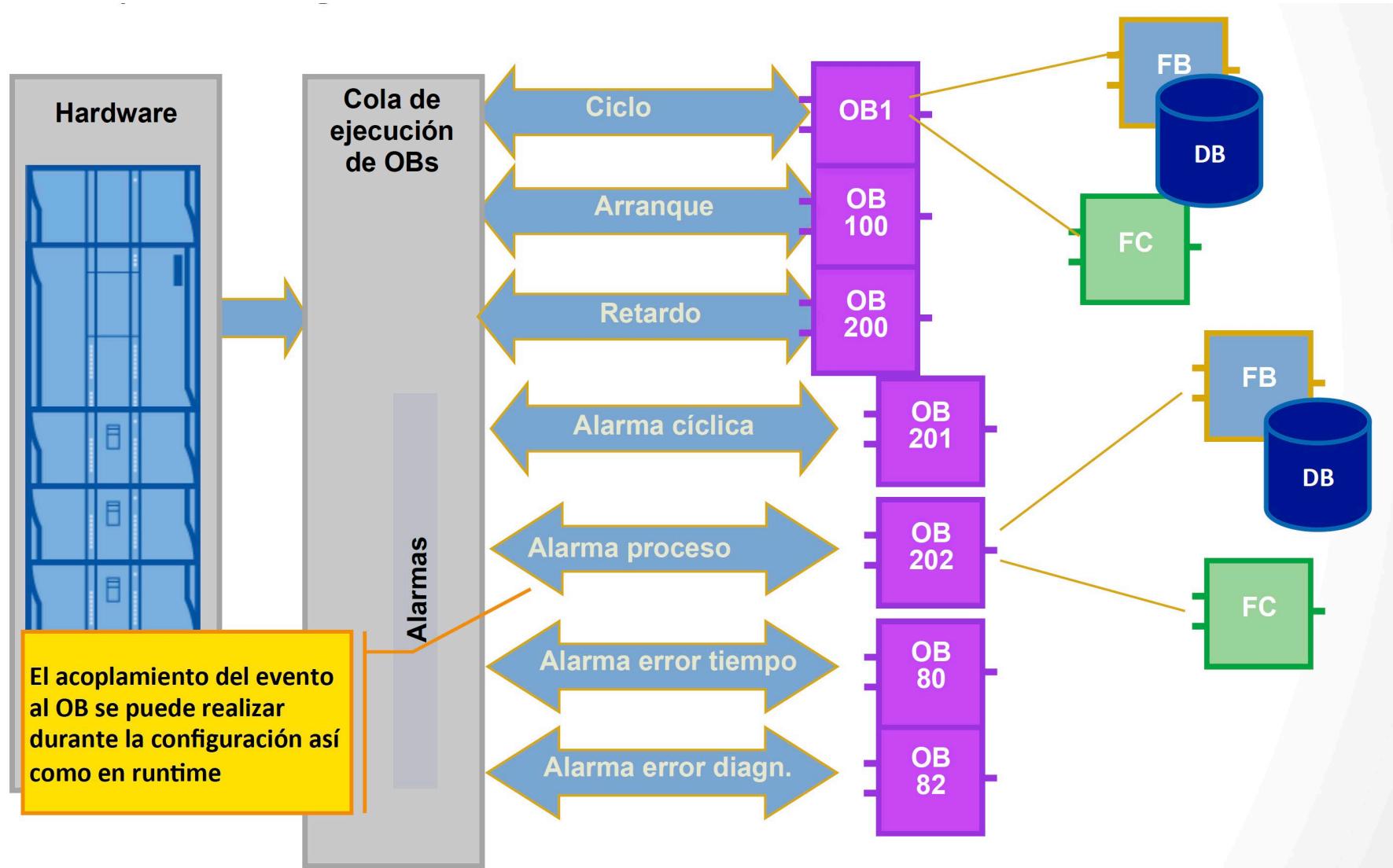
- **Bloque de organización (OB):** Un OB reacciona a un evento específico en la CPU y puede interrumpir la ejecución del programa de usuario. El OB predeterminado para el programa de usuario es el OB1, que se ejecuta de forma cíclica y es el único OB que se necesita para el programa de usuario. Otros OB puede interrumpir la ejecución del OB1. El resto de OBs ejecutan tareas específicas:
  - Tareas arranque
  - Procesamiento de alarmas
  - Tratamiento de errores

# Organización de programas. Tipos de módulos

- **Bloque de organización (OB):** Los OBs son controlados por eventos (p. ej. una alarma puede hacer que la CPU ejecute un OB. Hay algunos OBs predeterminados (p. ej. OB100 es el OB que se ejecuta en el arranque)
- La CPU determina el orden de ejecución del OB en función de su prioridad asignado al OB.



# Organización de programas. Tipos de módulos



# Organización de programas. Tipos de módulos

- **Función (FC):** Es un bloque que realiza una función específica, también se puede usar para agrupar código. La FC realiza una operación y guarda los resultados en memoria. Se usa la FC para:
  - Cálculos matemáticos
  - Controles individuales de bits

Una FC se puede llamar desde diferentes puntos, lo que facilita la programación de tareas repetitivas. FC no tiene bloque de datos de instancia (DB) asociado, usa datos locales para realizar los cálculos (no se almacenan los datos temporales).

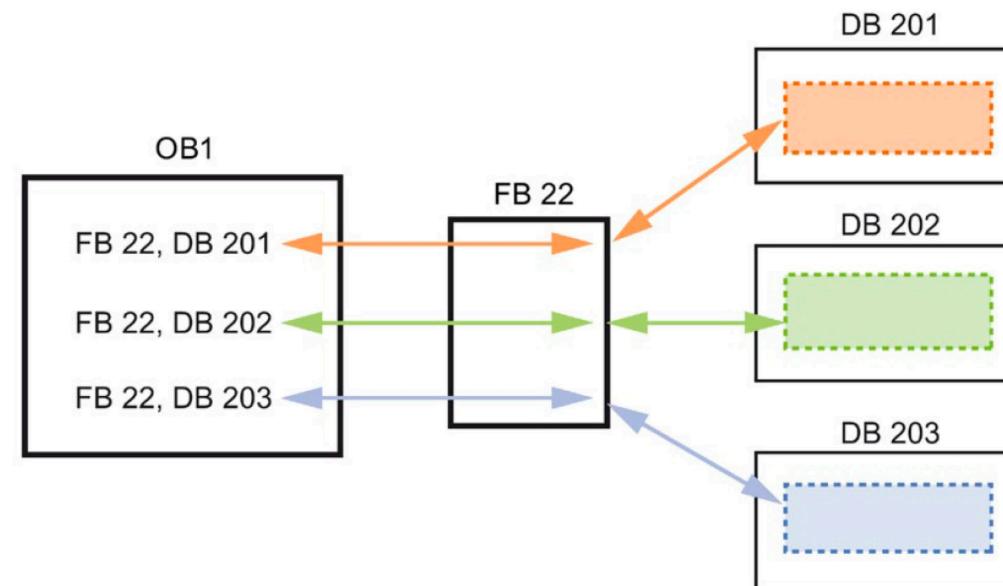
# Organización de programas. Tipos de módulos

- **Bloque de Función (FB):** Es un bloque lógico que utiliza un bloque de datos de instancia (DB) para almacenar parámetros y datos estáticos. Los FBs tienen una memoria variable ubicada en un bloque de datos (DB) o DB “instancia”. El DB instancia proporciona un bloque de memoria asociado a la instancia del FB y almacena datos una vez que el FB haya finalizado.

Se pueden asociar distintos DBs de instancia a diferentes llamadas del FB. Los DBs de instancia permiten usar un DB genérico para controlar varios dispositivos.

# Organización de programas. Tipos de módulos

**Bloque de Función (FB).** Ejemplo: OB que llama a un FB tres veces usando distintos bloques de datos en cada llamada. Así, con un FB genérico podemos controlar dispositivos similares (motor, válvula,...) asignando un bloque de datos de instancia diferente a cada llamada de los distintos dispositivos. Cada DB de instancia almacena los datos (p. ej. Velocidad de rotación, tiempo de aceleración, tiempo de operación total) de un dispositivo particular.



# Organización de programas. Tipos de módulos

- **Bloque de Datos (DB)**: Los DBs se crean para almacenar datos de los bloques lógicos. Todos los bloques del programa de usuario puede acceder a un DB global. En un DB instancia sólo el FB asociado puede acceder a ese DB.
- Los datos almacenados en un DB no se borran cuando finaliza la ejecución del bloque lógico asociado. Podemos tener dos tipos de DBs:
  - **DB global** que almacena datos de los bloques lógicos en el programa. Cualquier OB, FB o FC puede acceder a los datos de un DB global.
  - **DB instancia** que almacena los datos de un FB específico.

# BIBLIOGRAFÍA

- **Sistemas de automatización y autómatas programables.** Enrique Mandado, Jorge Acevedo, Celso Fernández, Ignacio Armesto, José Luis Rivas, José María Núñez. Ed. Marcombo.
- Autómatas Programables. Josep Barcells y José Luís Romeral. Ed. Marcombo.
- Automatización Industrial. Roberto Sanchís, Julio Ariel Romero y Carlos Vicente Ariño. Ed. Universitat Jaume I.
- [SIMATIC S7 Controlador programable S7-1200. Manual del Sistema.](#)  
[Siemens.](#)