



# **ĐIỀU KHIỂN TUẦN TỰ (SEQUENCE CONTROL)**

## CONTENTS

- ❖ Definitions and Examples
- ❖ Components in Sequence Control :
  - Programmable Logic Controller (PLC)
  - Relay
  - Input signal
  - Ladder diagram



## DEFINITIONS AND EXAMPLES

- ❖ Sequence control is the controlled operation of a machine according to a predefined sequence
- ❖ Example: car washing

1. Insert payment and Press Start



2. Car washing process



Water rinse



Cleaning



Brushing



Another water rinse



Jet drying

## DEFINITIONS AND EXAMPLES

- ❖ Sequence control is defined as control for successively advancing each step of a control procedure according to a predetermined order or an order determined according to a fixed logic
- ❖ Examples: parts conveyed belt conveyor
  - The part goes to the end of part
  - Sensor detects the part and responses to the robot
  - The robot grabs the part
  - Move the part to the desk

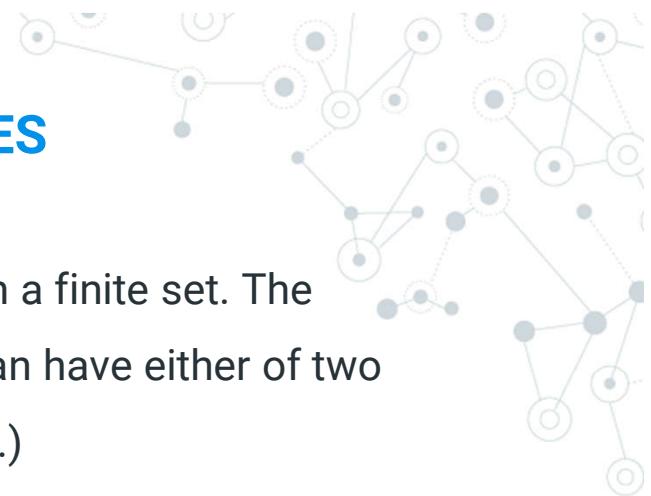
## DEFINITIONS AND EXAMPLES

- ❖ Examples: the operation of elevator
  - Press the button
  - Elevator comes
  - The door opens
  - Press destination button
  - Elevator moves
  - The door open when you arrive



## DEFINITIONS AND EXAMPLES

- ❖ Many control applications use process variables with a finite set. The simplest of such variable are binary variables, that can have either of two possible values (1 or 0, on or off, open and close, etc.)
- ❖ These control system operate by turning on and off switches, motors, valves, and other device in response to operating condition and as a function of time. Such system are referred to as sequence/logic control systems



## DEFINITIONS AND EXAMPLES

- ❖ Some of these can also be operated using analog control methods. In specific applications they may be viewed as discrete control or sensing devices

Type	Signal	Remark
Switch	<b>Binary Command</b>	External Input Device
Limit switch	<b>Position</b>	Feedback Sensor Device
Thumbwheel switch	<b>Set valued Command</b>	External Input Device
Thermostat	<b>Temperature Level</b>	Feedback Sensor Device
Photo cell	<b>Position of objects</b>	Feedback Sensor Device
Proximity detector	<b>Position of objects</b>	Feedback Sensor Device
Push button	<b>Command (unlatched)</b>	External Input Device

Tab: Discrete Sensors

Type	Output Quantity	Energy Source
Relay, Contactor	<b>voltage</b>	electrical
Motor Starter	<b>motion</b>	electrical
Lamp	<b>indication</b>	electrical
Solenoid	<b>motion</b>	electrical
On-off Flow Control valve	<b>Flow</b>	pneumatic, hydraulic
<b>Directional Valves</b>	<b>Hydraulic Pressure</b>	pneumatic, hydraulic

Tab: Industrial Discrete Output and Actuation Devices

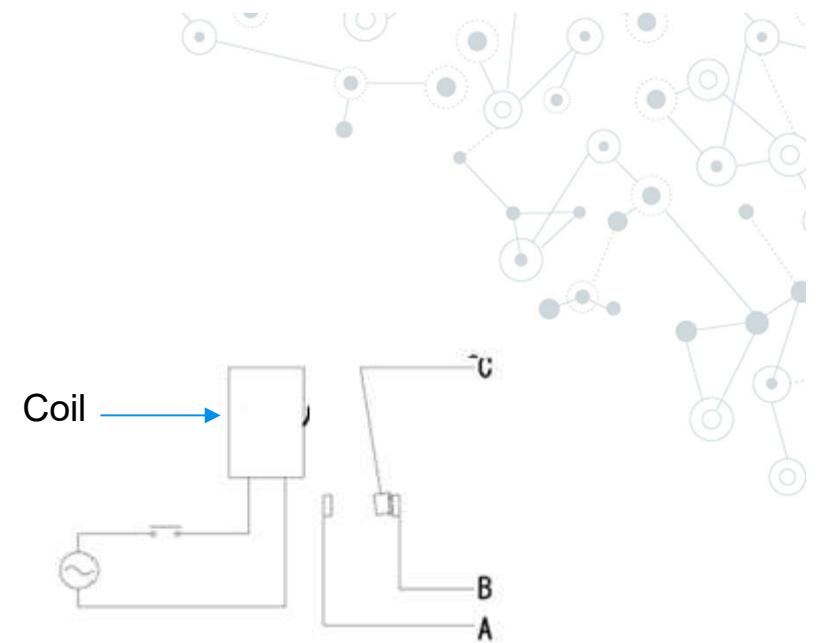
## DEFINITIONS AND EXAMPLES

- ❖ Comparing sequence control with analog control

Issue	Logic/Sequence Control	Analog Control
Model	Logical State-Transition	Numerical Differential/Difference Eqn
	Simple Model/Easy to build	Complex Model/Hard to build
	Infrequent	Liable to change
Signal <b>Temporal Property</b>	Signal range/status	Signal value
	(Timed) sequence	(Timed)Function/Trajectory
Control Redesign/Tuning	On-off/logical	linear/non linear analog
	Supervisory	automatic
	Open/Closed Loop	Open/Closed Loop
	Infrequent	Tuning needed

## Relay

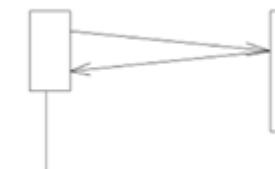
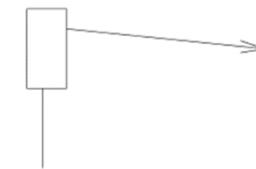
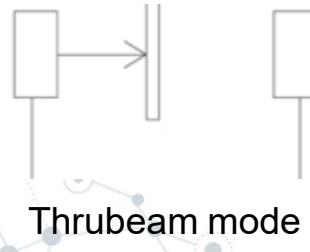
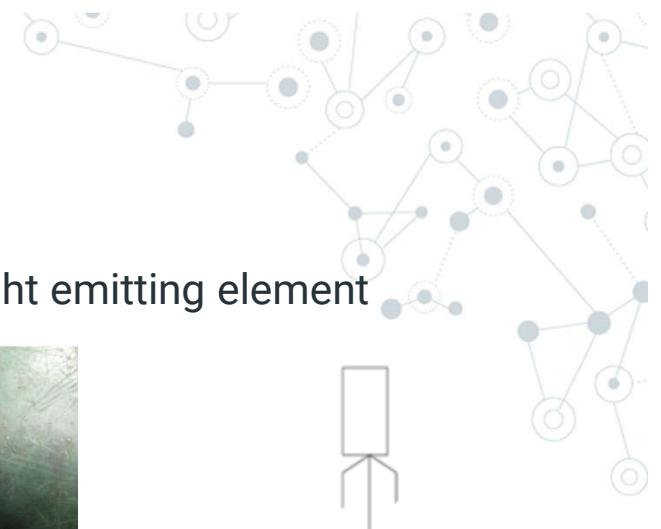
- ❖ Relay control is basic of sequence control
- ❖ When a voltage (AC voltage) is applied to the coil, contacts A and C are connected



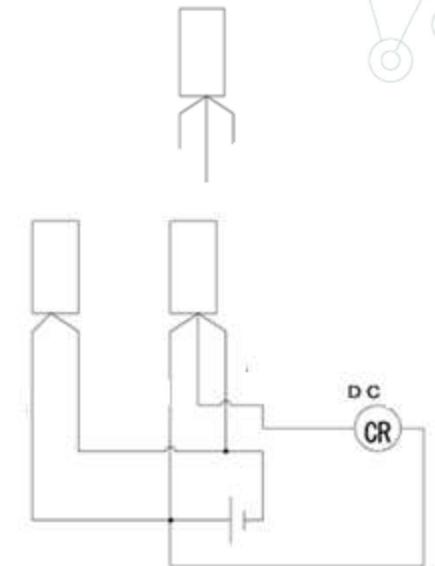
- A, B, C: terminals
- C: common
- C and A: normal open (NO)
- C and B: normal close (NC)

## Input Signal

- ❖ Photoelectric sensor: emits a light beam from its light emitting element



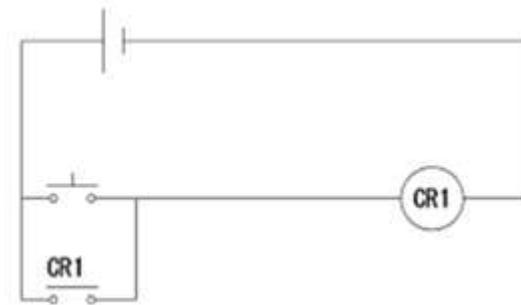
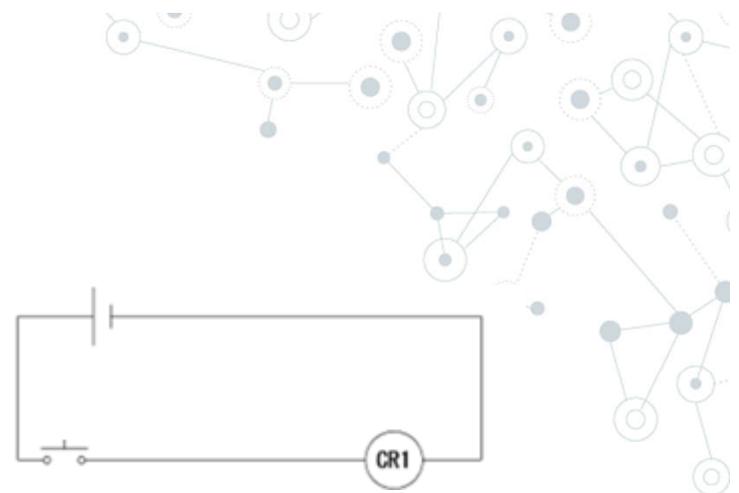
Reflective mode



Wiring

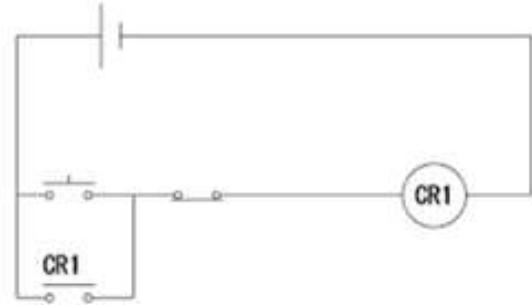
## Relay Control

- ❖ Connect the switch and the coil
  - ❖ Press the pushbutton switch to active the delay
  - ❖ Release the pushbutton, the relay returns to its original state.
- 
- ❖ A contact of the relay and the contact of the pushbutton switch are connected in parallel.
  - ❖ The “soft-holding” state



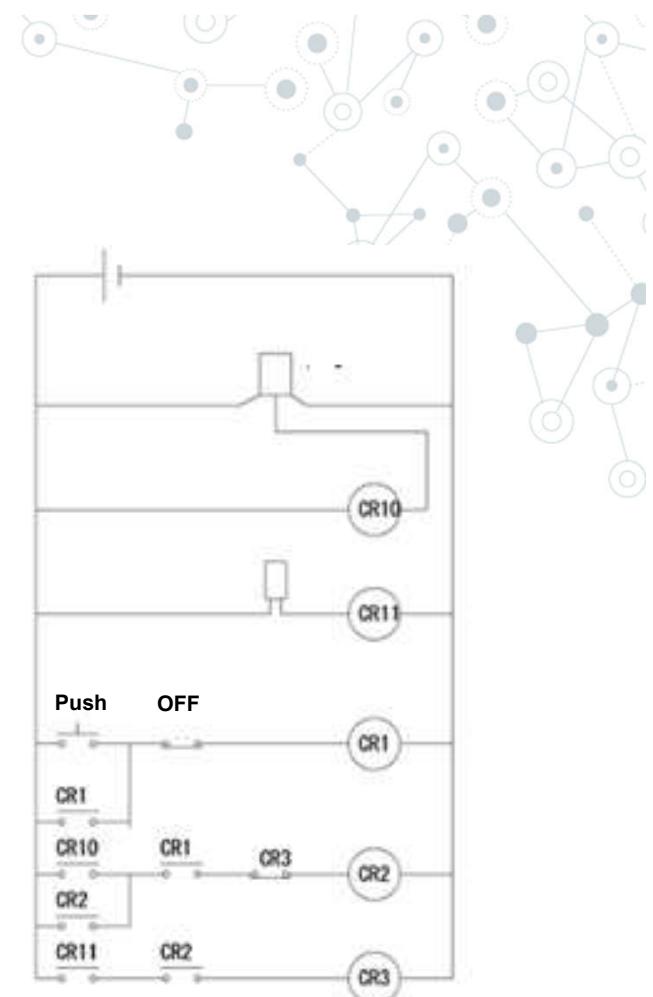
## Relay Control

- ❖ The “OFF” switch to release the self hold



# Relay Control

- ❖ Example
  - CR10: photoelectronic
  - CR11: cylinder
  - Press the push button: automation operation
  - Press the OFF button: stop



13

## Programmable Logic Controller (PLC)

- ❖ A modern controller device used extensively for sequence control in robotics, process control, and many other automated systems
- ❖ Programmable Logic Controller or Programmable Controllers
- ❖ It is a special purpose industrial microprocessor based real –time computing system, which performs the following functions in the context of industrial operations:
  - Monitor Input/sensors
  - Execute logic, sequencing, timing, counting functions for control
  - Drives actuators/indicators
  - Communications with other components

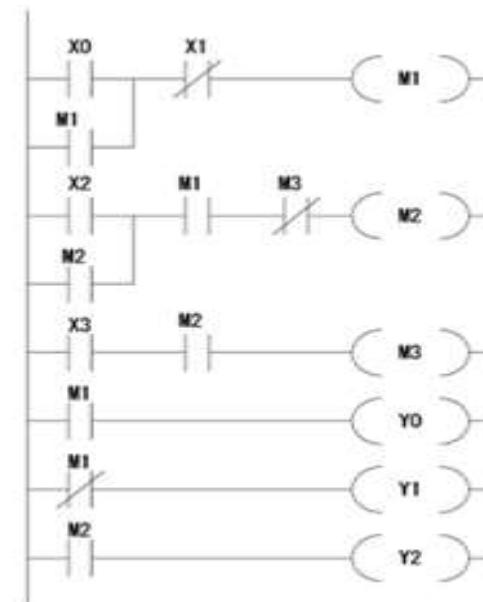
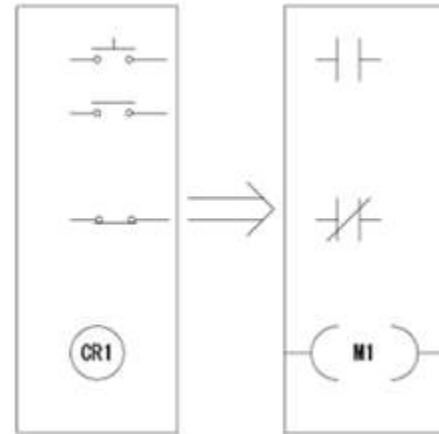
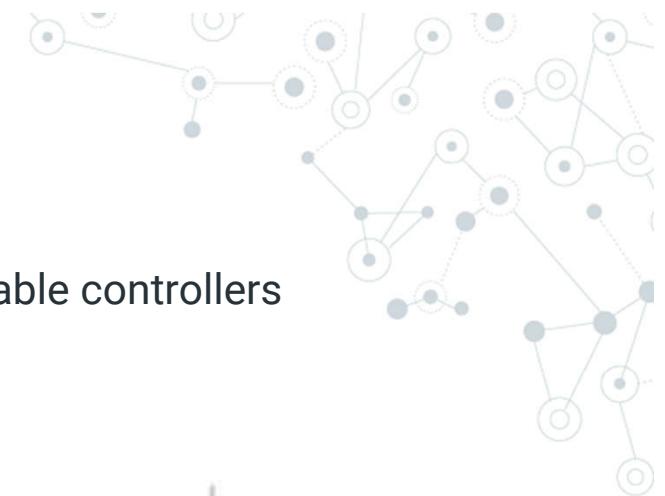
## Programmable Logic Controller (PLC)

- ❖ Advantages of PLC:
  - Easy to use to simple modular assembly and connection
  - Modular expansion capacity of the input, outputs and memory
  - Simply programming environments and the use of standardized task libraries and degging aids
  - Communication capacity with other programmable controllers and computers
- ❖ Applications: variety of automation tasks



## Ladder diagram

- ❖ Ladder diagram is the language used for programmable controllers
- ❖ It looks like a relay circuit
- ❖ It is essential for learning sequence control





# SƠ ĐỒ TRẠNG THÁI (STATE DIAGRAM)



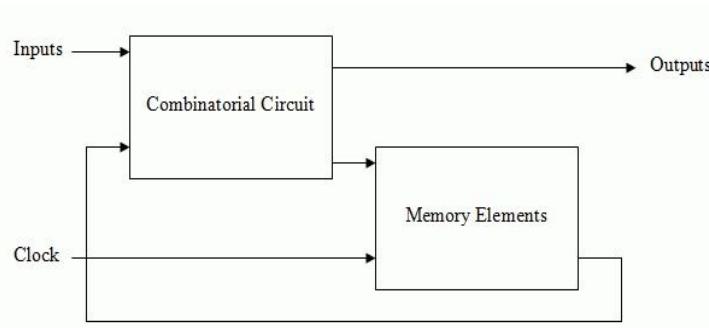
# CONTENTS

- ❖ Definitions
  - Sequencial circuits
  - State stable
  - State diagram
- ❖ Process control and state diagram
- ❖ Converting the state diagram to ladder diagram
  - State equations
  - State – Transition equations



## DEFINITIONS

- ❖ **Sequential Circuits:** the output is a function of both **current** and **past** inputs
  - Memory elements: store the past inputs (flip-flop)
  - Solve more complex computational problems
- ❖ The effect of all previous inputs on the output is represented by a **state** of the circuit.  
The output of the circuit at anytime depends upon its current state and the input
- ❖ The relationship among the inputs, outputs, present states and next states can be specified by either the **state table** or the **state diagram**



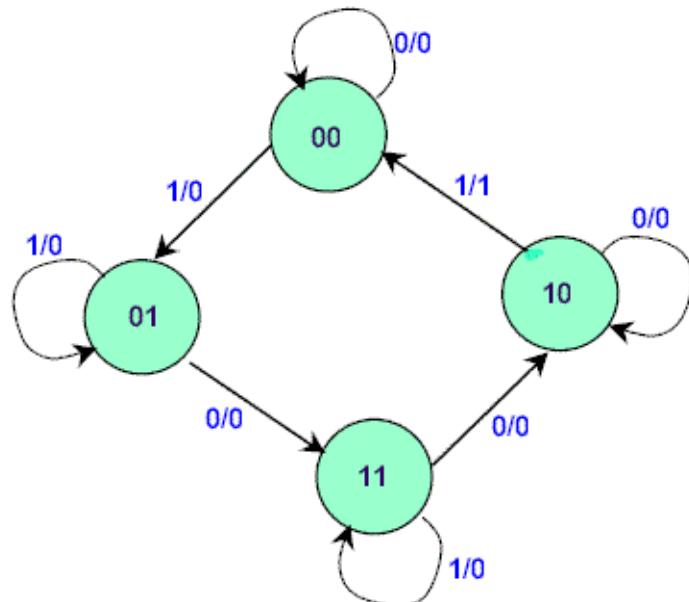
## DEFINITIONS

- ❖ **State table:** consists of three sections labelled *present state*, *next state* and *output*.  
The present state designates the state of flip-flop before the occurrence of a clock pulse. The next state shows the states of flip-flop after the clock pulse and the output section lists the value of the output variables during the present state

Current State		Input I	Next State		Outputs Y
A	B		A <sub>next</sub>	B <sub>next</sub>	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	X	X	X
1	1	1	X	X	X

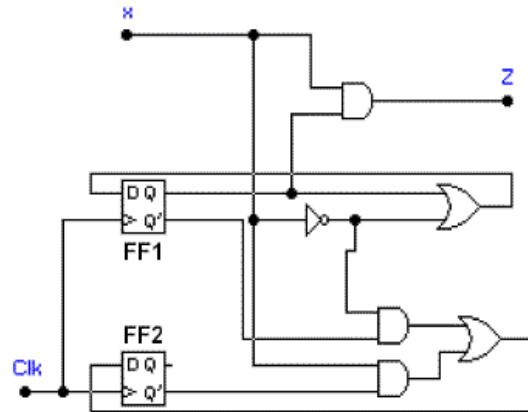
## DEFINITIONS

- ❖ **State diagram:** represents the flip flop by graphical symbols
  - A state: circle
  - The transition between states: lines (or arcs) connecting the circles
  - The binary number inside each circle: the state the circle
  - The binary in the directed lines ( $1/0$ ): the first is the input value, the second is value of the output



# DEFINITIONS

- ❖ Example:



$$Z = x * Q_1$$

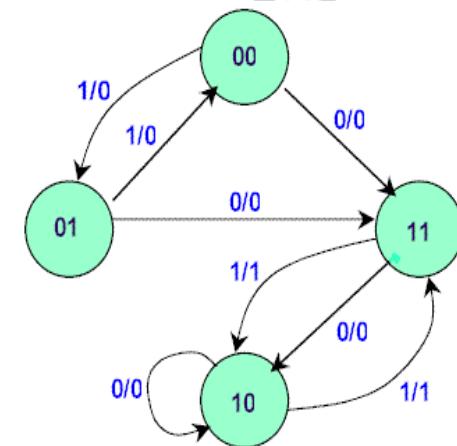
$$D_1 = x' + Q_1$$

$$D_2 = x * Q_2' + x' * Q_1'$$

Sequential circuit

Present State Q <sub>1</sub> Q <sub>2</sub>	Next State		Output	
	x = 0	x = 1	x = 0	x = 1
0 0	1 1	0 1	0	0
0 1	1 1	0 0	0	0
1 0	1 0	1 1	0	1
1 1	1 0	1 0	0	1

State table



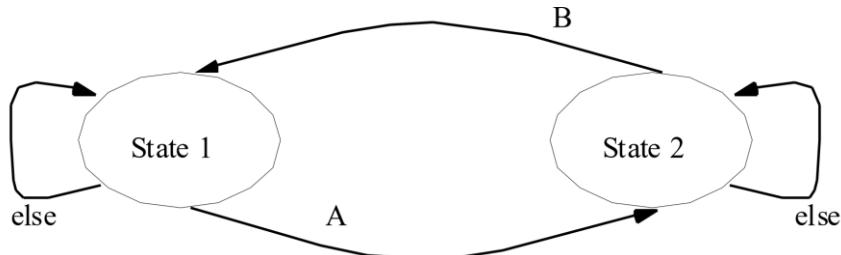
State diagram

## PROCESS CONTROL AND STATE DIAGRAM

- ❖ A system state is a mode of operation
- ❖ A state based system can be described with system states and the transitions between those states.
- ❖ Example: a bank machine with these states: idle, scan card, get secret number, select transaction type, ask for amount of cash, count cash, deliver cash/return card, idle.

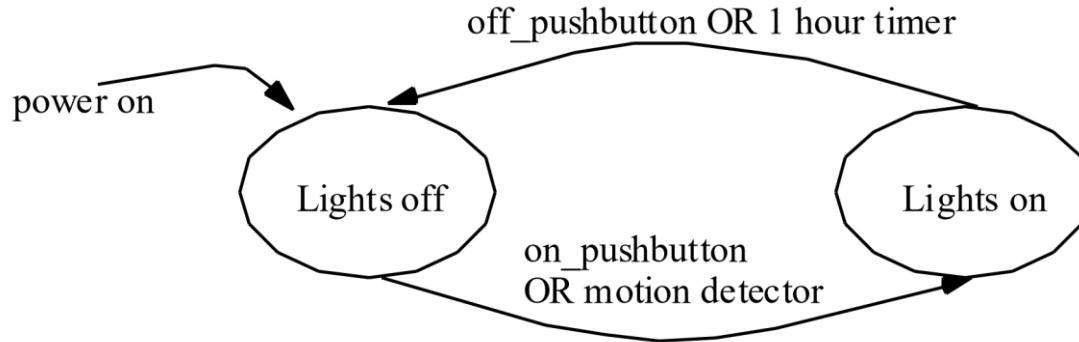
## PROCESS CONTROL AND STATE DIAGRAM

- ❖ Example: a state based system with State 1 and State 2.
  - If the system is in State 1 and A happens, the system will go into State 2, otherwise it will remain in State 1
  - If the system is in State 2 and B happens, the system will return State 1, otherwise it will remain in State 2



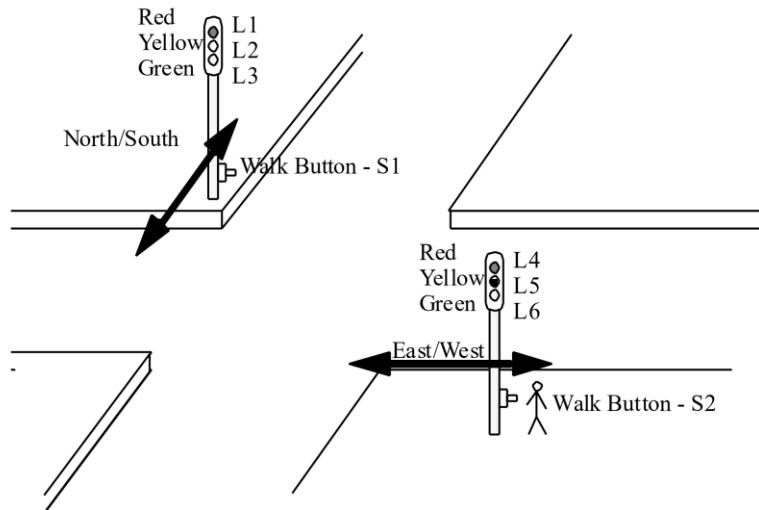
# PROCESS CONTROL AND STATE DIAGRAM

- ❖ Example: an automation light controller



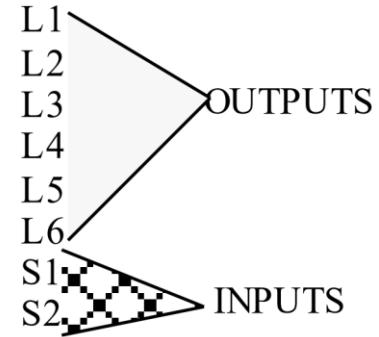
# PROCESS CONTROL AND STATE DIAGRAM

- ❖ Example: the traffic lights
  - Green light: in one direction for a long period of time (>10 s)
  - Yellow light: 4 s
  - A green or yellow light in one direction implies a red light in the other direction
  - When the cross walk light is turned on then the duration of the green light increases



# PROCESS CONTROL AND STATE DIAGRAM

- ❖ Step 1: define the inputs and outputs of the system
  - 8 items are ON or OFF
  - The outputs can be used to define the system state (each state will lead to a different set of outputs)
  - The inputs are used when defining the transitions
  - The state table:



System State						
	L1	L2	L3	L4	L5	L6
A binary number						
	0 = light off					
	1 = light on					
State Table						
State Description	#	L1	L2	L3	L4	L6
Green East/West	1	1	0	0	0	1
Yellow East/West	2	1	0	0	0	0
Green North/South	3	0	0	1	1	0
Yellow North/South	4	0	1	0	1	0

Here the four states determine how the 6 outputs are switched on/off.

## PROCESS CONTROL AND STATE DIAGRAM

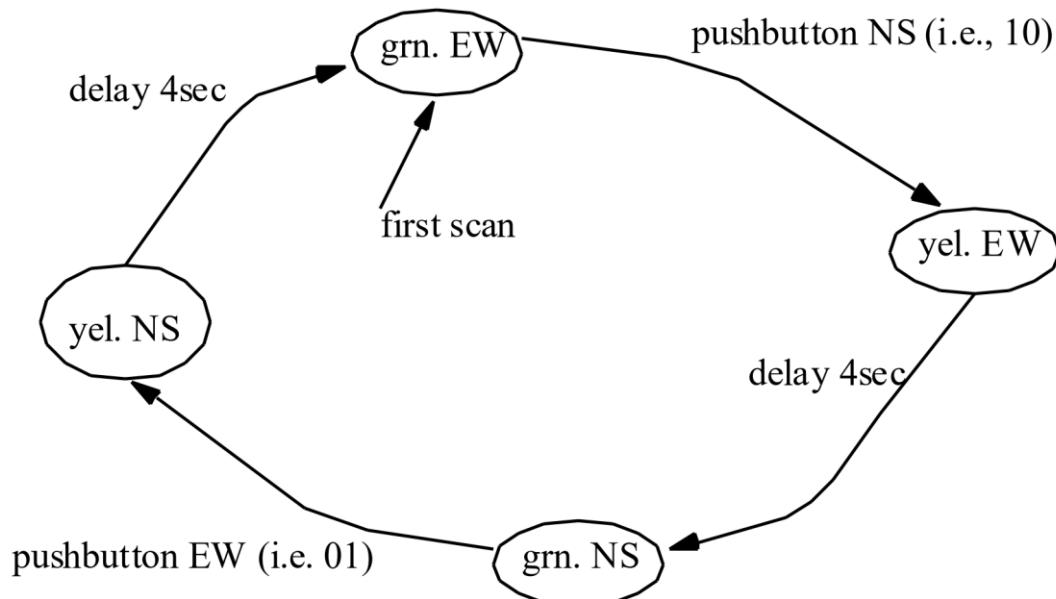
- ❖ Step 2: define state transitions
  - Transition 1: Green E/W to Yellow E/W (S1) (abnormal)
  - Transition 2: Yellow E/W to Green N/S (normal)
  - Transition 3: Green N/S to Yellow N/S (abnormal)

Description	#	L1	L2	L3	L4	L5	L6	transition
Green East/West	1	1	0	0	0	0	1	S1
Yellow East/West	2	1	0	0	0	1	0	delay 4sec
Green North/South	3	0	0	1	1	0	0	S2
Yellow North/South	4	0	1	0	1	0	0	

The diagram illustrates the state transitions for a process control system. It shows four states represented by numbers 1 through 4 in a table. State 1 (Green E/W) has a transition to state 2 (Yellow E/W) labeled 'S1'. State 2 (Yellow E/W) has a transition back to state 1 (Green E/W) labeled 'delay 4 sec'. State 3 (Green N/S) has a transition to state 4 (Yellow N/S) labeled 'S2'. State 4 (Yellow N/S) has a transition back to state 3 (Green N/S) labeled 'delay 4sec'.

# PROCESS CONTROL AND STATE DIAGRAM

- ❖ Step 3: Draw the state transition diagram



## CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ **The state equations:** State diagram can be converted to Boolean equations and then to Ladder logic using state equations.

State  $X = (\text{State } X + \text{just arrived from another state}) \text{ and has not left for another state}$

$$STATE_i = \left( STATE_i + \sum_{j=1}^n (T_{j,i} \bullet STATE_j) \right) \bullet \prod_{k=1}^m \overline{(T_{i,k} \bullet STATE_i)}$$

where,  $STATE_i$  = A variable that will reflect if state i is on

$n$  = the number of transitions to state i

$m$  = the number of transitions out of state i

$T_{j,i}$  = The logical condition of a transition from state j to i

$T_{i,k}$  = The logical condition of a transition out of state i to k

# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Example: traffic lights

$ST1$  = state 1 - green NS

$ST2$  = state 2 - yellow NS

$ST3$  = state 3 - green EW

$ST4$  = state 4 - yellow EW

$$ST1 = (ST1 + ST4 \cdot TON_2(ST4, 4s)) \cdot \overline{ST1 \cdot S1 \cdot \overline{S2}} + FS$$

$$ST2 = (ST2 + ST1 \cdot S1 \cdot \overline{S2}) \cdot \overline{ST2 \cdot TON_1(ST2, 4s)}$$

$$ST3 = (ST3 + ST2 \cdot TON_1(ST2, 4s)) \cdot \overline{ST3 \cdot \overline{S1} \cdot S2}$$

$$ST4 = (ST4 + ST3 \cdot \overline{S1} \cdot S2) \cdot \overline{ST4 \cdot TON_2(ST4, 4s)}$$

Timer:  $TON_i(A, \text{delay})$ : TON-an on delay timer, A is the input to the timer, delay is the timer delay value

## CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Example: traffic lights
- The simplified Boolean equations:

$$ST1 = (ST1 + ST4 \cdot TON_2(ST4, 4)) \cdot (\overline{ST1} + \overline{S1} + S2) + FS$$

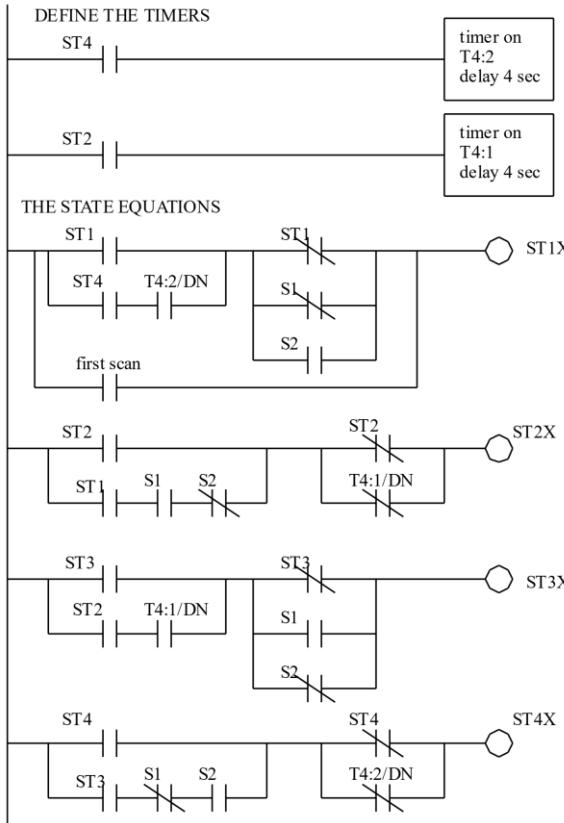
$$ST2 = (ST2 + ST1 \cdot S1 \cdot \overline{S2}) \cdot (\overline{ST2} + \overline{TON_1(ST2, 4)})$$

$$ST3 = (ST3 + ST2 \cdot TON_1(ST2, 4)) \cdot (\overline{ST3} + S1 + \overline{S2})$$

$$ST4 = (ST4 + ST3 \cdot \overline{S1} \cdot S2) \cdot (\overline{ST4} + \overline{TON_2(ST4, 4)})$$

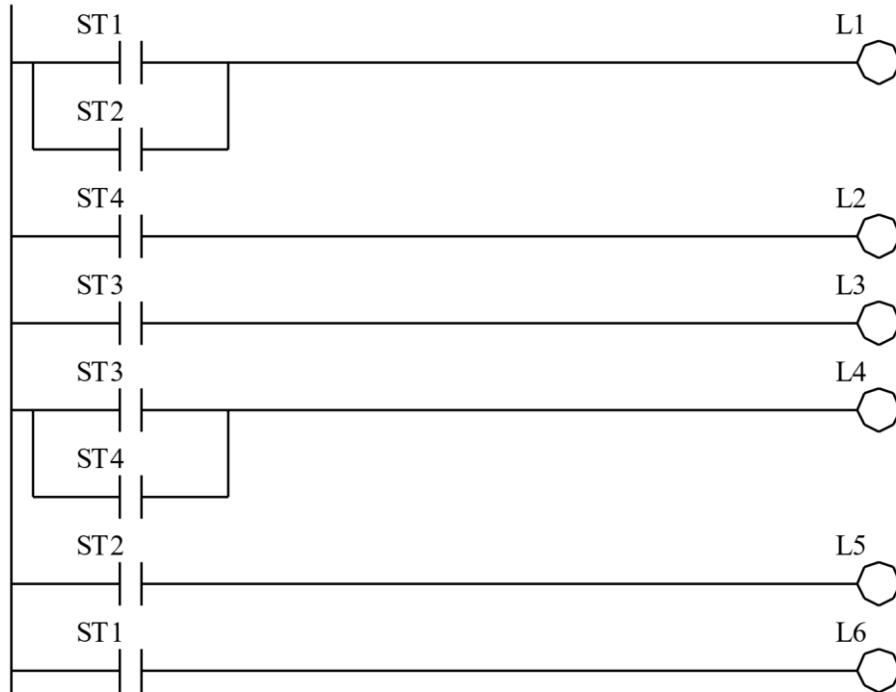
# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Example: traffic lights
  - Ladder Logic for state equations:



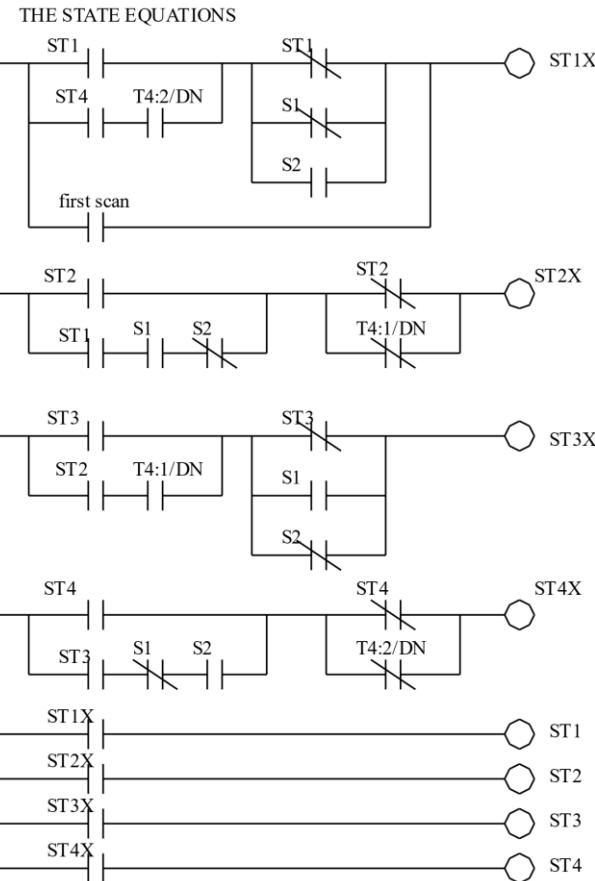
# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Example: traffic lights
  - Output logic for the lights:



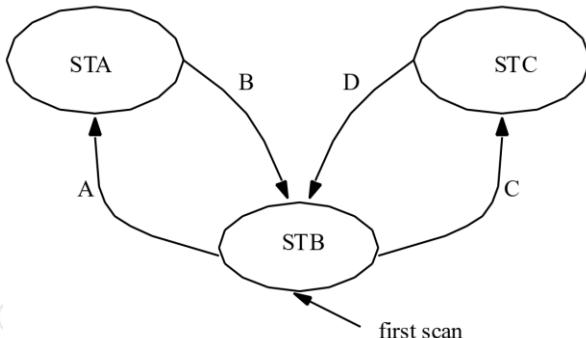
# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Example: traffic lights
  - State updating



# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ State equations with prioritization:
  - Each of the alternate transitions out of a state should be given a priority, from highest to lowest
  - The state equations can then be written to suppress transitions of lower priority when one or more occur simultaneously



$$STA = (STA + STB \cdot A) \cdot \overline{STA \cdot B}$$

$$STB = (STB + STA \cdot B + STC \cdot D) \cdot \overline{STB \cdot A} \cdot \overline{STB \cdot C} + FS$$

$$STC = (STC + STB \cdot C \cdot \bar{A}) \cdot \overline{STC \cdot D}$$

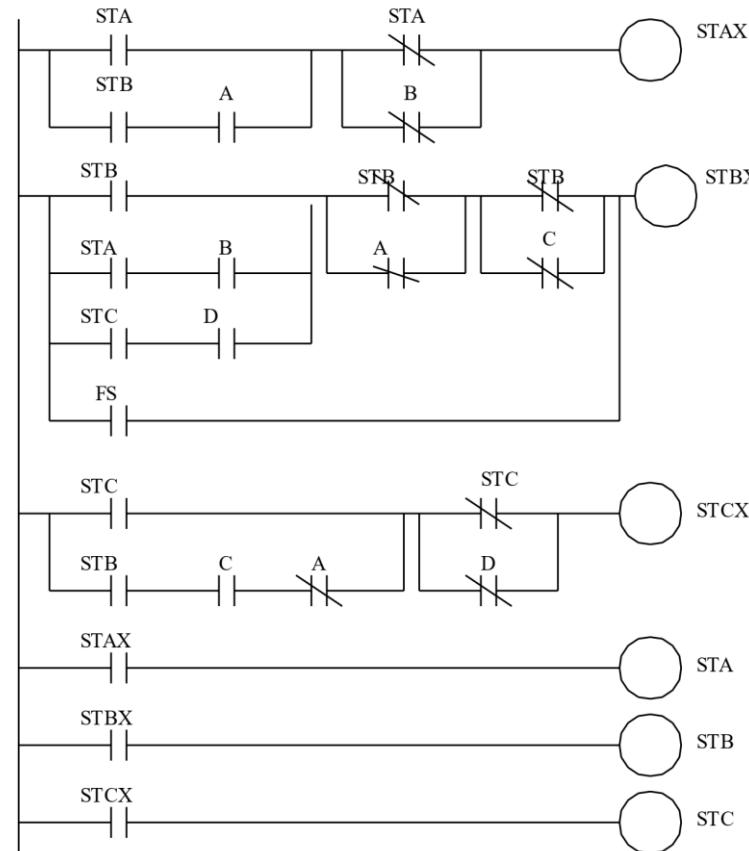
# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- Ladder logic with prioritization:

$$STA = (STA + STB \cdot A) \cdot \overline{STA \cdot B}$$

$$STB = (STB + STA \cdot B + STC \cdot D) \cdot \overline{STB \cdot A} \cdot \overline{STB \cdot C} + FS$$

$$STC = (STC + STB \cdot C \cdot \bar{A}) \cdot \overline{STC \cdot D}$$



# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

## ❖ The state - transition equations:

- Write an equation for each state and each transition. Each state and transition needs to be assigned a unique variable name
- The transition equations are written by looking at the each state and then determining which transitions will end the state
- The state equations are similar to the state equations in the previous state equations method, except they now only refer to the transitions

# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

## ❖ Example: traffic light

defined state and transition variables:

$ST1$  = state 1 - green NS

$ST2$  = state 2 - yellow NS

$ST3$  = state 3 - green EW

$ST4$  = state 4 - yellow EW

$T1$  = transition from  $ST1$  to  $ST2$

$T2$  = transition from  $ST2$  to  $ST3$

$T3$  = transition from  $ST3$  to  $ST4$

$T4$  = transition from  $ST4$  to  $ST1$

$T5$  = transition to  $ST1$  for first scan

state and transition equations:

$$T4 = ST4 \cdot TON_2(ST4, 4)$$

$$ST1 = (ST1 + T4 + T5) \cdot \overline{T1}$$

$$T1 = ST1 \cdot S1 \cdot \overline{S2}$$

$$ST2 = (ST2 + T1) \cdot \overline{T2}$$

$$T2 = ST2 \cdot TON_1(ST2, 4)$$

$$ST3 = (ST3 + T2) \cdot \overline{T3}$$

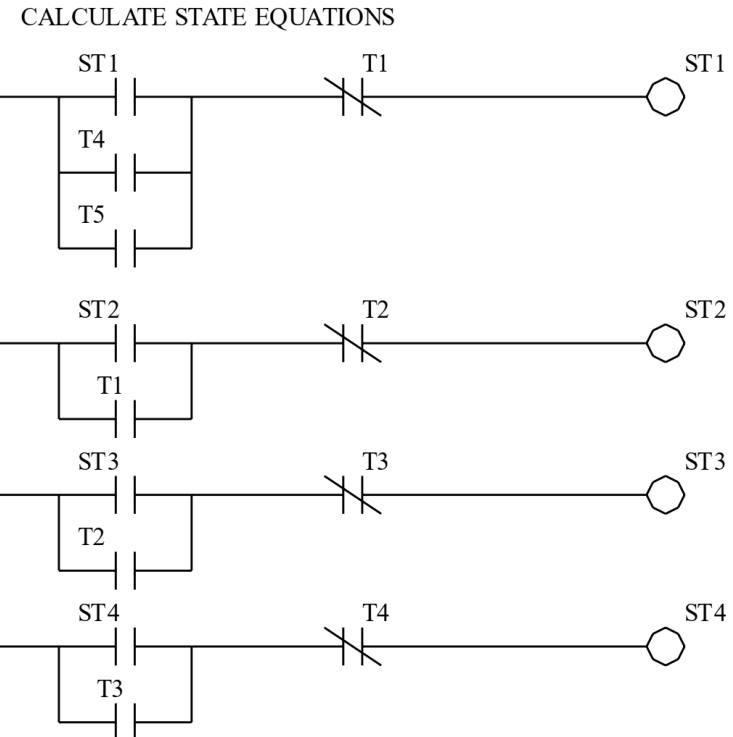
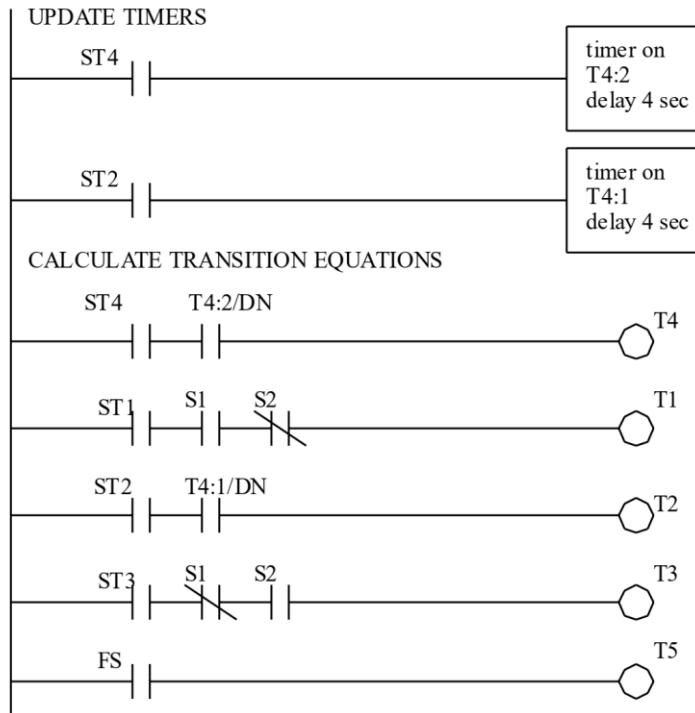
$$T3 = ST3 \cdot \overline{S1} \cdot S2$$

$$ST4 = (ST4 + T3) \cdot \overline{T4}$$

$$T5 = FS$$

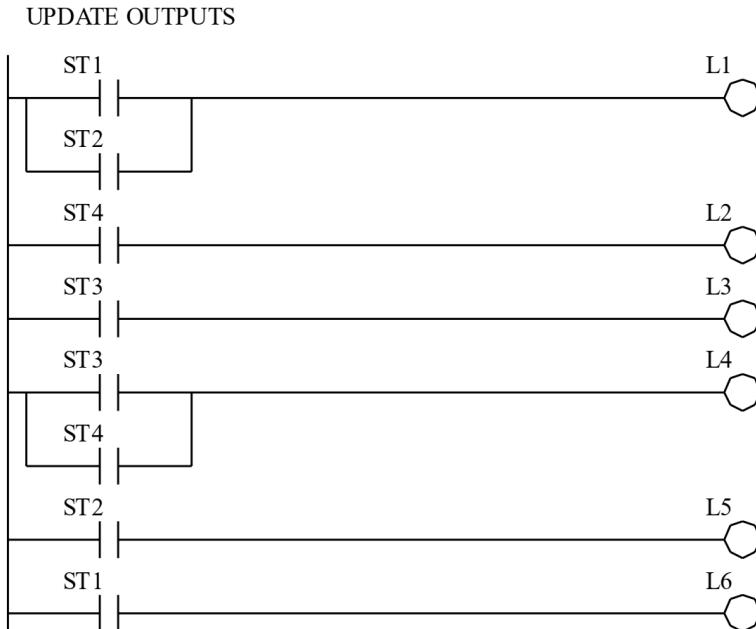
# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Example: traffic light
  - Ladder logic for state transition equations



# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Example: traffic light
  - Ladder logic for state transition equations



## CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

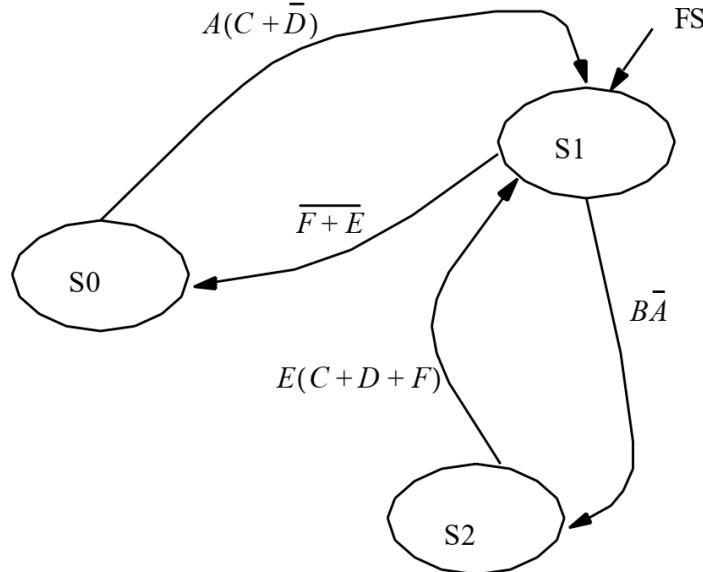
- ❖ Exercise 1: take an example about automatic application and then draw the state diagram of this system

# CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Exercise 2: convert the following state diagram to equations

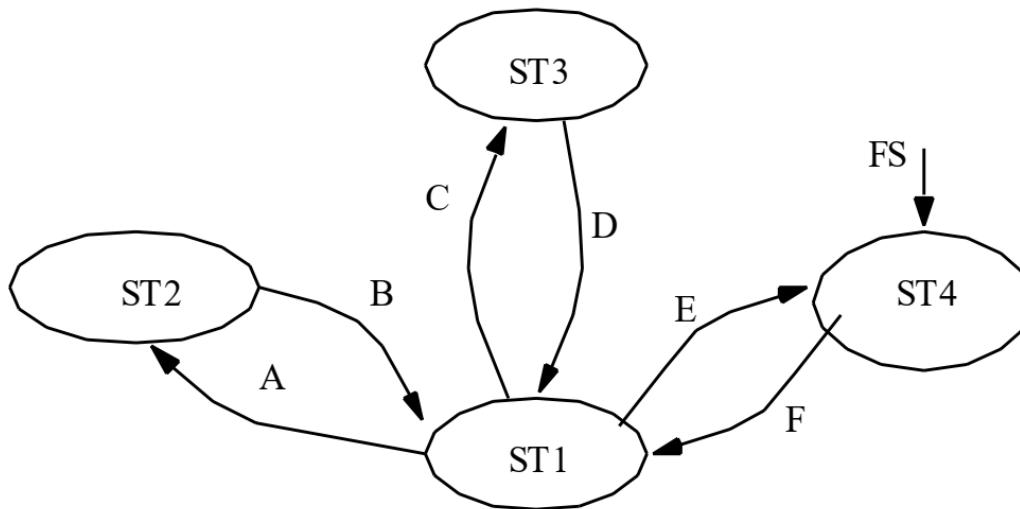
Inputs	Outputs
A	P
B	Q
C	R
D	
E	
F	

state	P	Q	R
S0	0	1	1
S1	1	0	1
S2	1	1	0



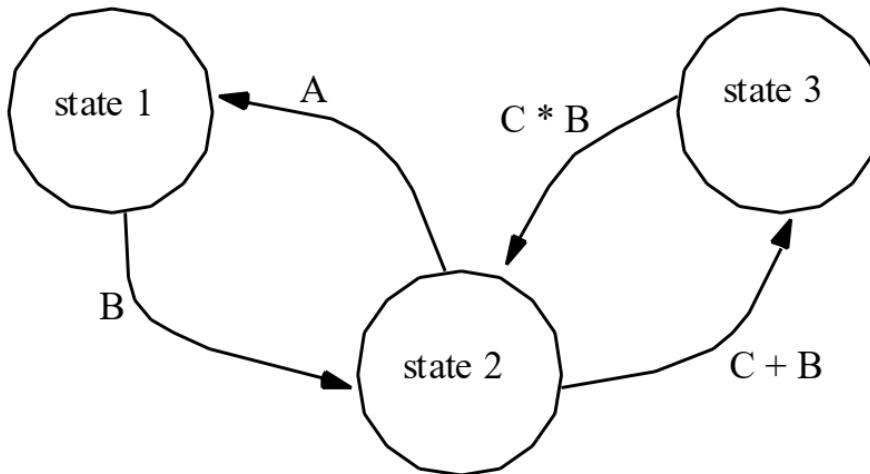
## CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Exercise 2: convert the following state diagram to equations



## CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Exercise 3: given the following state diagram, use equations to implement ladder logic



## CONVERSION FROM STATE DIAGRAM TO LADDER LOGIC

- ❖ Exercise 4: designing a garage door controller using state transition equations. The behavior of garage door controller is as follows:
  - there is a single button in the garage, and a single button remote control.
  - when the button is pushed the door will move up or down.
  - if the button is pushed once while moving, the door will stop, a second push will start motion again in the opposite direction.
  - there are top/bottom limit switches to stop the motion of the door.
  - there is a light beam across the bottom of the door. If the beam is cut while the door is closing the door will stop and reverse.
  - there is a garage light that will be on for 5 minutes after the door opens or closes.



# **BÀI 1:**

# **TỔNG QUAN VỀ**

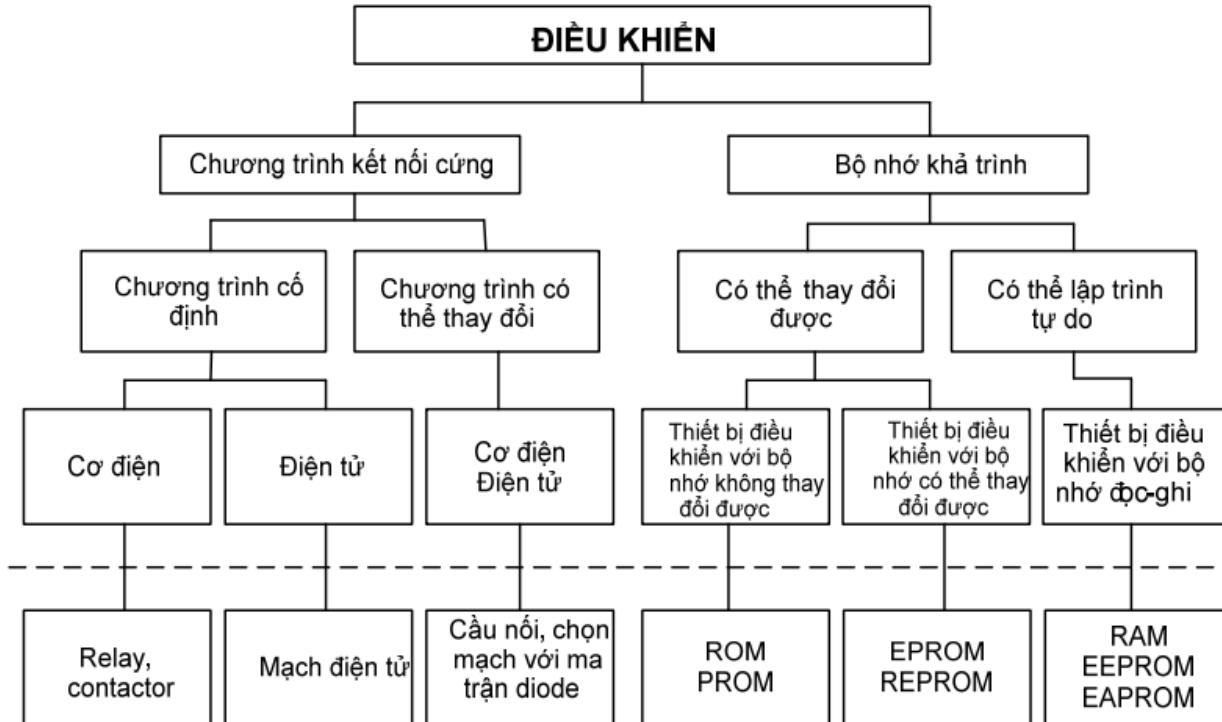
# **PLC S7 1200**

# NỘI DUNG

- ❖ Tổng quan về PLC.
- ❖ Giới thiệu PLC S7 1200.

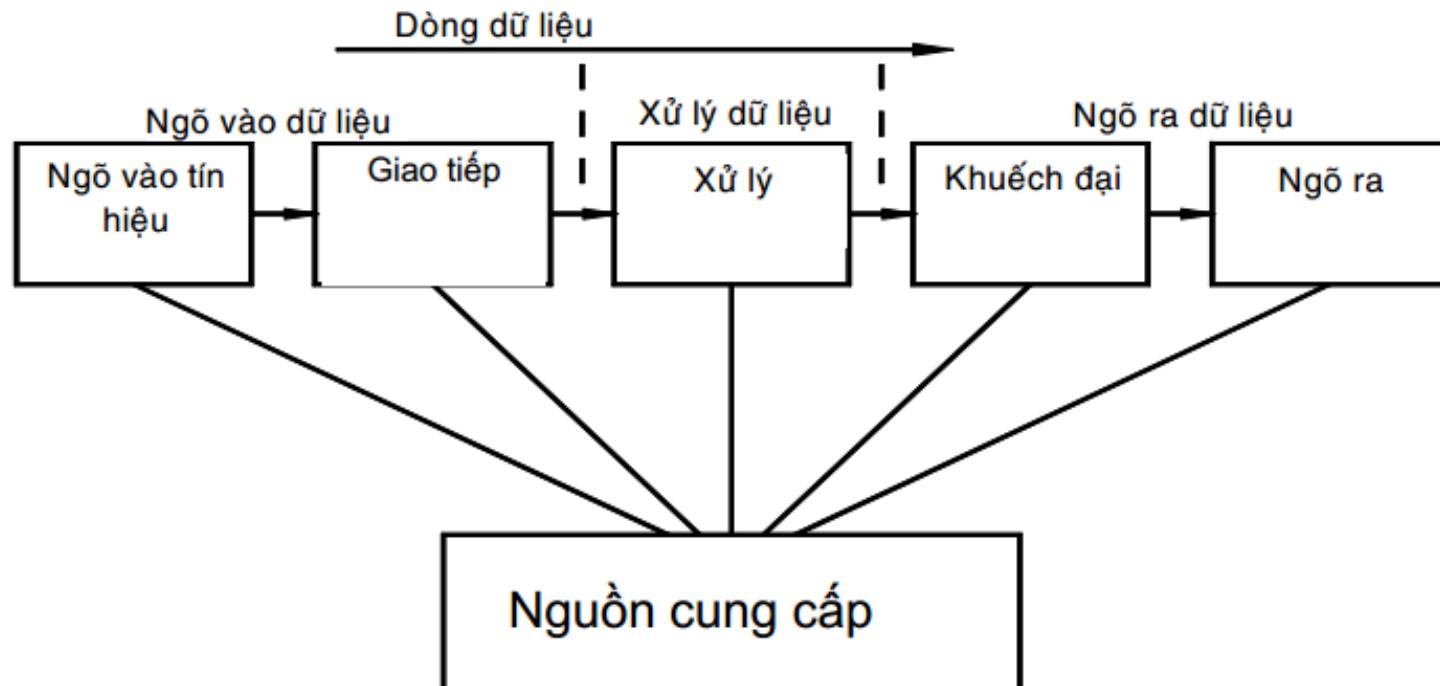
# TỔNG QUAN VỀ ĐIỀU KHIỂN

- Điều khiển có nhiệm vụ **thực hiện các chức năng riêng** của một **máy móc** hay **thiết bị** theo **một trình tự hoạt động định trước** phụ thuộc vào **trạng thái** của **máy** hay **bộ phát tín hiệu**.



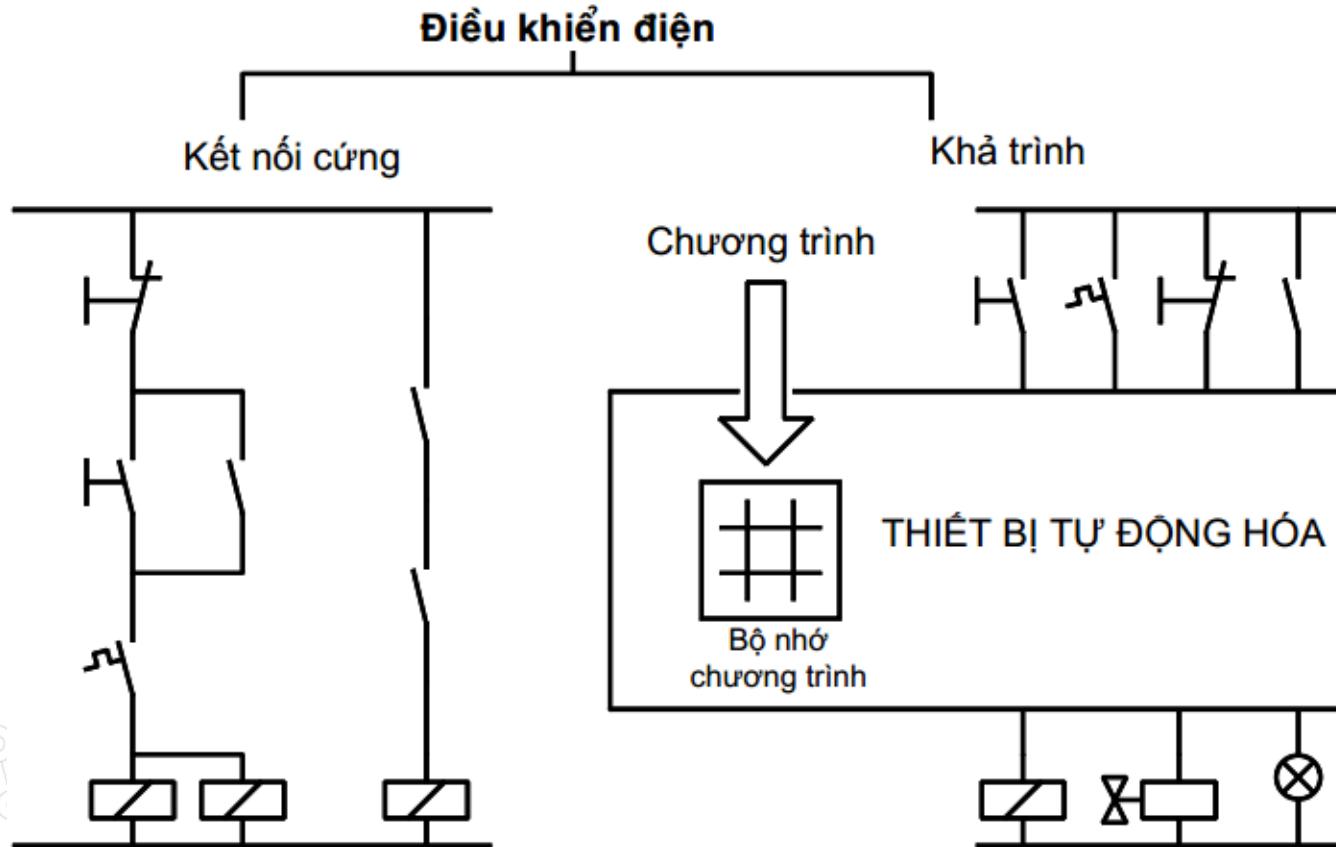
# TỔNG QUAN VỀ ĐIỀU KHIỂN

## □ Cấu trúc một quy trình điều khiển



# TỔNG QUAN VỀ ĐIỀU KHIỂN

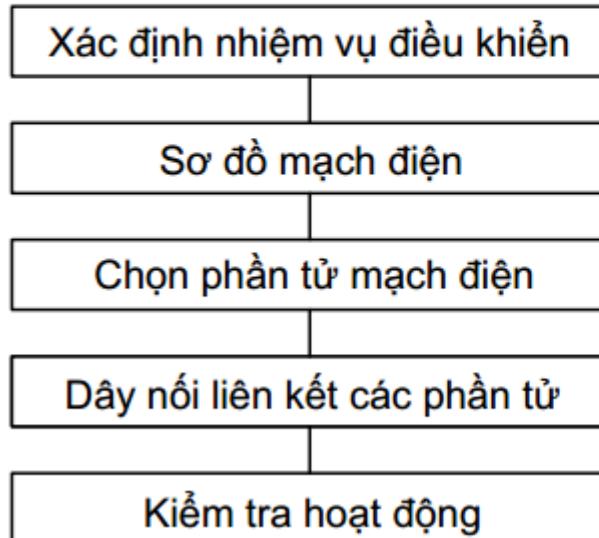
## ☐ Các loại điều khiển



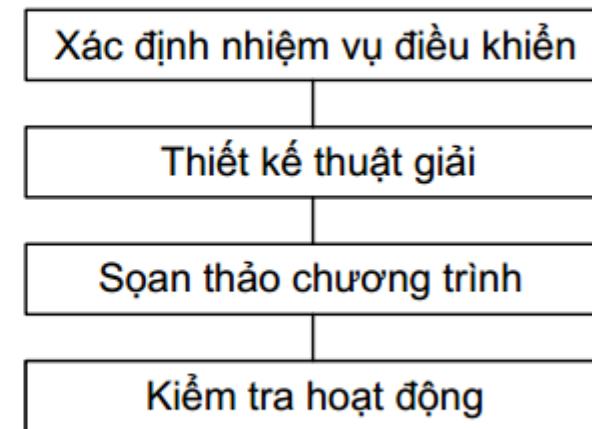
# TỔNG QUAN VỀ PLC

- PLC viết tắt là **Programmable Logic Controller**: là thiết bị điều khiển lập trình được (khả trình) cho phép thực hiện linh hoạt các thuật toán điều khiển logic thông qua ngôn ngữ lập trình.

## *Các bước thiết lập hệ điều khiển bằng relay điện*



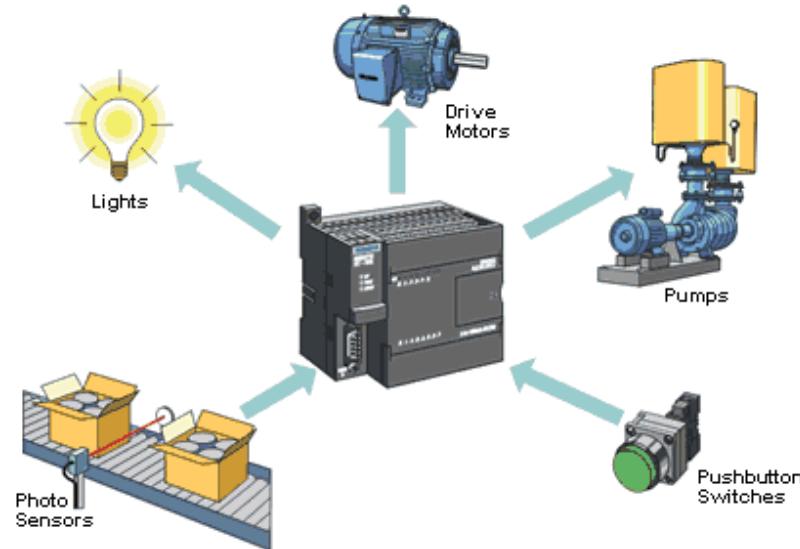
## *Các bước thiết lập hệ điều khiển bằng PLC*



# TỔNG QUAN VỀ PLC

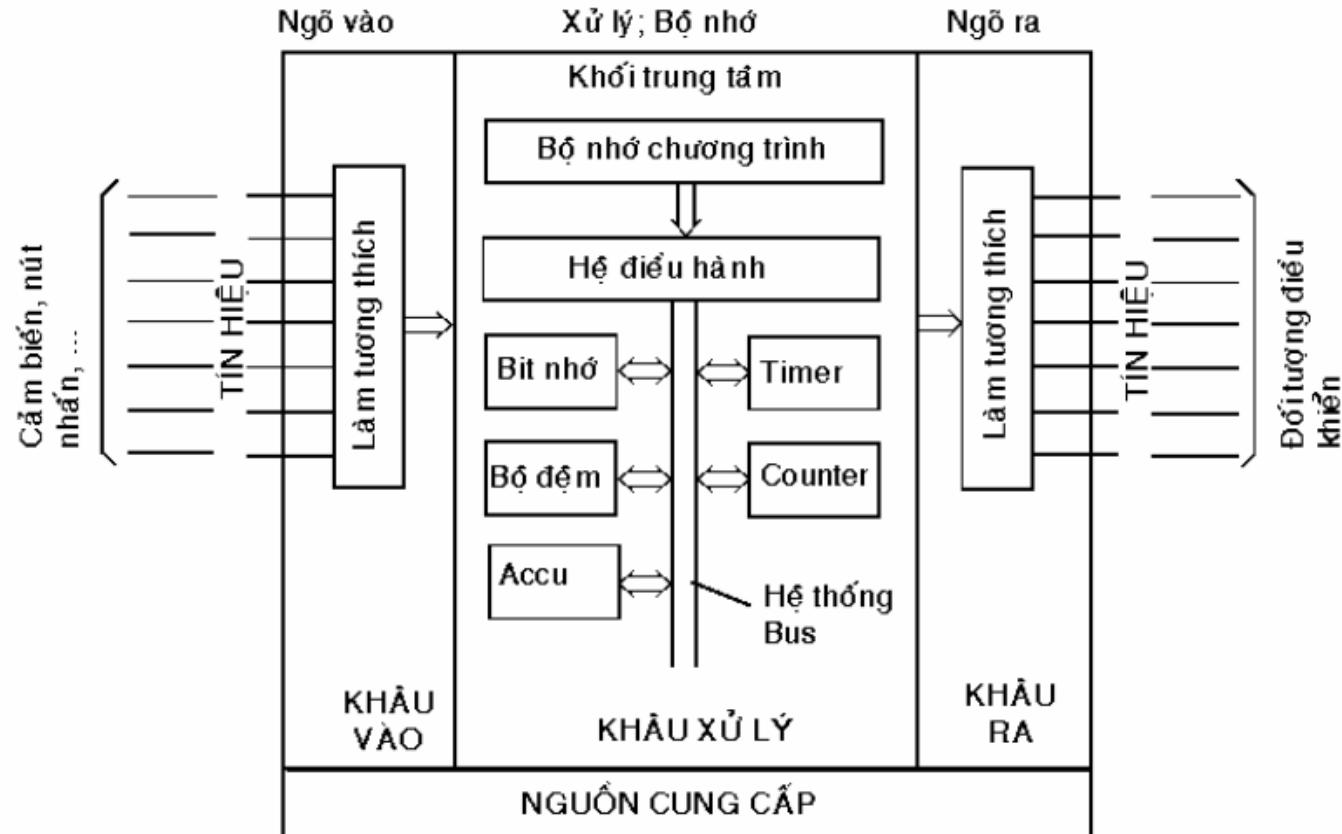
## □ Ưu điểm:

- Thích ứng với những nhiệm vụ điều khiển khác nhau.
- Khả năng thay đổi đơn giản trong quá trình đưa thiết bị vào sử dụng.
- Tiết kiệm không gian lắp đặt.
- Tiết kiệm thời gian trong quá trình mở rộng và phát triển nhiệm vụ điều khiển.
- Các thiết bị điều khiển theo chuẩn.
- Không cần các tiếp điểm.



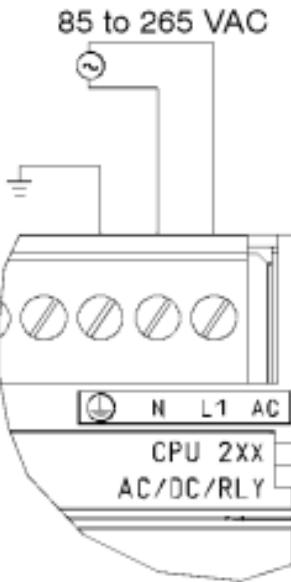
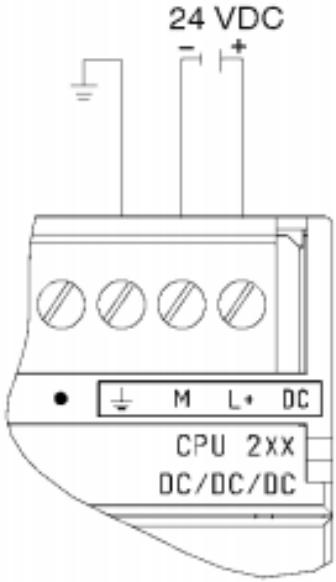
# TỔNG QUAN VỀ PLC

## □ Cấu trúc của một PLC



# KẾT NỐI NGUỒN TRONG PLC

## ☐ Kết nối nguồn

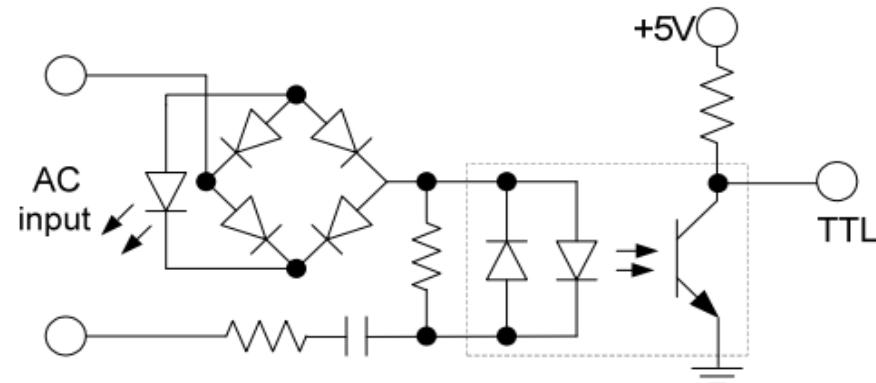
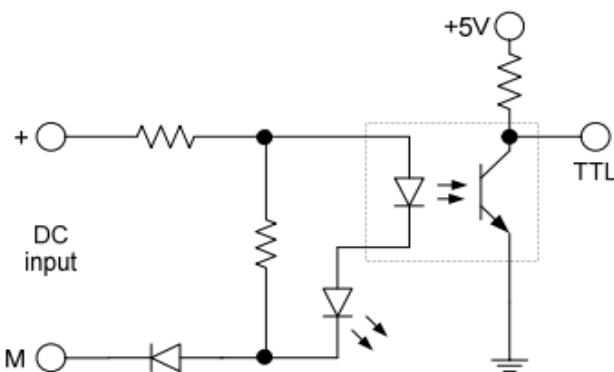


*Kết nối nguồn cho CPU loại DC/DC/DC và AC/DC/RLY*

# KẾT NỐI NGÕ VÀO SỐ VỚI NGOẠI VI

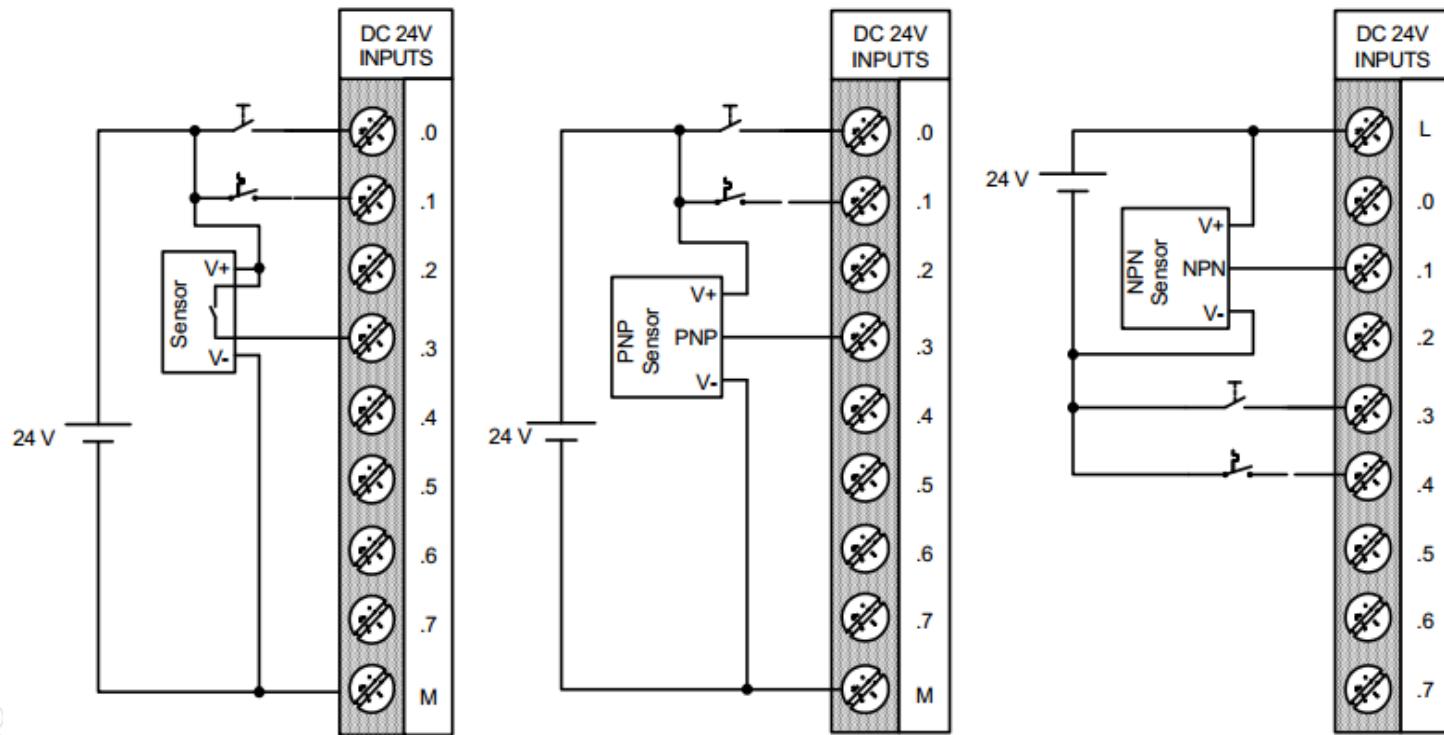
❑ Nguồn cung cấp cho các khối vào của họ S7-200 có thể là:

- Xoay chiều: 15...35 VAC, f = 47...63 Hz; dòng cần thiết nhỏ nhất 4mA và 79...135 VAC, f = 47...63 Hz; dòng cần thiết nhỏ nhất 4mA.
- Một chiều: 15 ... 30 VDC; dòng cần thiết nhỏ nhất 4mA.



Sơ đồ mạch điện bên trong của một ngõ vào sử dụng nguồn cung cấp DC và AC

# KẾT NỐI NGÕ VÀO SỐ VỚI NGOẠI VI

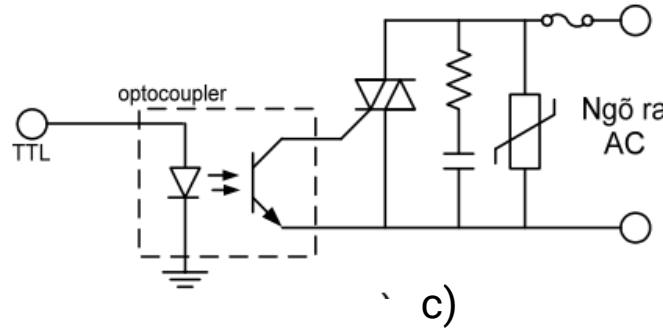
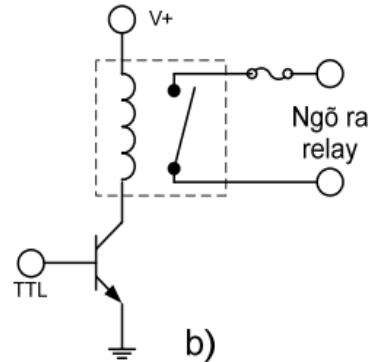
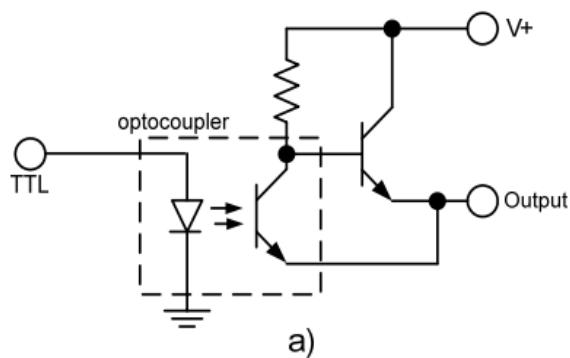


Kết nối ngõ vào với ngoại vi

# KẾT NỐI NGÕ RA SỐ VỚI NGOẠI VI

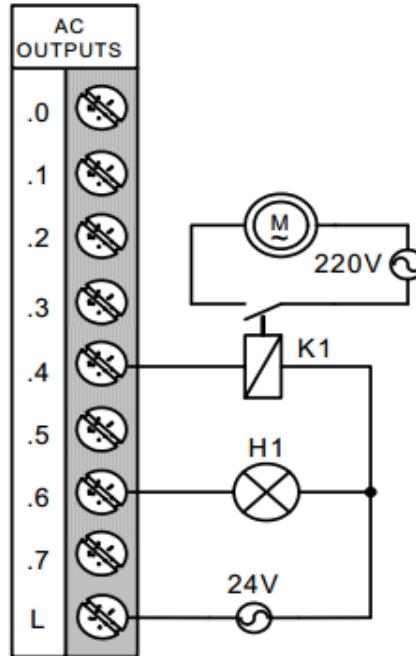
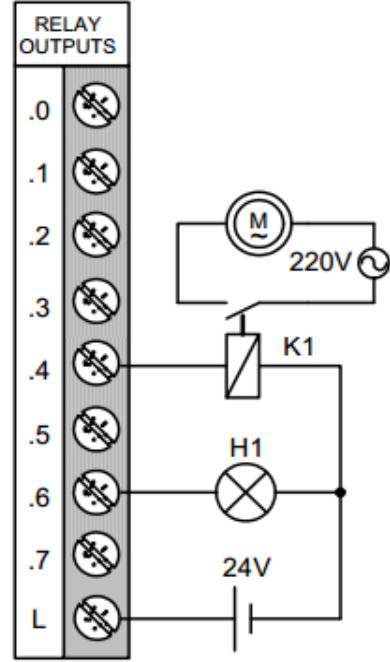
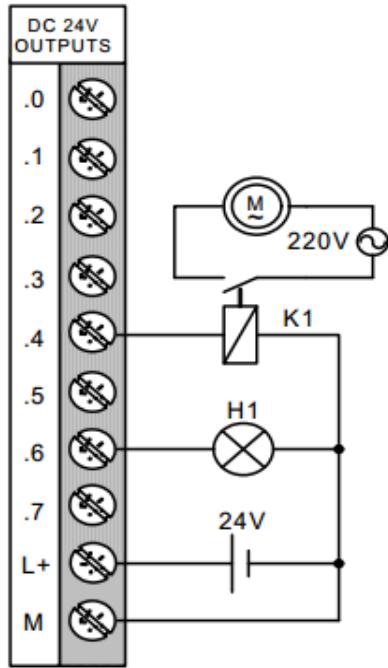
□ Nguồn cung cấp cho các khối ra có thể là:

- Xoay chiều: 20...264 VAC , f = 47...63 Hz;
- Một chiều: 5...30 VDC đối với ngõ ra rơ le; 20.4 ... 28.8 VDC đối với ngõ ra transistor.



Mạch điện bên trong của các loại ngõ ra khác nhau.  
a) Ngõ ra transistor; b) Ngõ ra relay; c) Ngõ ra triac

# KẾT NỐI NGÕ RA SỐ VỚI NGOẠI VI



Kết nối ngõ ra PLC với cơ cấu chấp hành

# GIỚI THIỆU VỀ PLC S7 1200

- **S7-1200** ra đời **năm 2009** dùng để thay thế dần cho S7-200. So với S7-200 thì S7-1200 có những **tính năng nổi trội hơn**.
- **S7-1200** được thiết kế **nhỏ gọn, chi phí thấp**, và **một tập lệnh mạnh** giúp những **giải pháp hoàn hảo hơn** cho ứng dụng sử dụng với S7-1200.
- **S7-1200** cung cấp **một cổng PROFINET**, hỗ trợ chuẩn **Ethernet** và **TCP/IP**.



# GIỚI THIỆU VỀ PLC S7 1200

---

❑ S7-1200 có 5 dòng CPU:

- PLC S7-1200 CPU **1211C** có bộ nhớ làm việc **50KB** work memory.
- PLC S7-1200 CPU **1212C** có bộ nhớ làm việc **75KB** work memory.
- PLC S7-1200 CPU **1214C** có bộ nhớ làm việc **100KB** work memory.
- **PLC S7-1200 CPU 1215C có bộ nhớ làm việc 125KB work memory.**
- PLC S7-1200 CPU **1217C** có bộ nhớ làm việc **150KB** work memory.

# GIỚI THIỆU VỀ PLC S7 1200

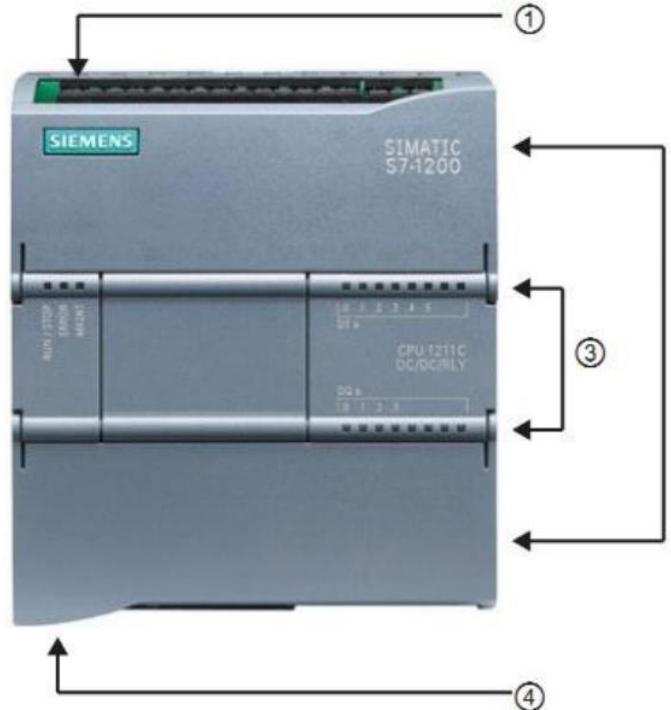
Chức năng	CPU 1211C	CPU 1212C	CPU 1214C	
Kích thước vật lý (mm)	90 x 100 x 75		110 x 100 x 75	
Bộ nhớ người dùng:	<ul style="list-style-type: none"><li>Bộ nhớ làm việc</li><li>Bộ nhớ nạp</li><li>Bộ nhớ giữ lại</li></ul>		<ul style="list-style-type: none"><li>25 kB</li><li>1 MB</li><li>2 kB</li><li>50 kB</li><li>2 MB</li><li>2 kB</li></ul>	
I/O tích hợp cục bộ	<ul style="list-style-type: none"><li>Kiểu số</li><li>Kiểu tương tự</li></ul>	<ul style="list-style-type: none"><li>6 ngõ vào / 4 ngõ ra</li><li>2 ngõ ra</li></ul>	<ul style="list-style-type: none"><li>8 ngõ vào / 6 ngõ ra</li><li>2 ngõ ra</li></ul>	<ul style="list-style-type: none"><li>14 ngõ vào / 10 ngõ ra</li><li>2 ngõ ra</li></ul>
Kích thước ảnh tiến trình	1024 byte ngõ vào (I) và 1024 byte ngõ ra (Q)			
Bộ nhớ bit (M)	4096 byte		8192 byte	
Độ mở rộng các <i>module</i> tín hiệu	Không	2	8	
Bảng tín hiệu	1			
Các <i>module</i> truyền thông	3 (mở rộng về bên trái)			

# GIỚI THIỆU VỀ PLC S7 1200

Các bộ đếm tốc độ cao	3	4	6
• Đơn pha	• 3 tại 100 kHz	• 3 tại 100 kHz 1 tại 30 kHz	• 3 tại 100 kHz 3 tại 30 kHz
• Vuông pha	• 3 tại 80 kHz	• 3 tại 80 kHz 1 tại 20 kHz	• 3 tại 80 kHz 3 tại 20 kHz
Các ngõ ra xung	2		
Thẻ nhớ	Thẻ nhớ SIMATIC (tùy chọn)		
Thời gian lưu giữ đồng hồ thời gian thực	Thông thường 10 ngày / ít nhất 6 ngày tại 40°C		
PROFINET	1 cổng truyền thông Ethernet		
Tốc độ thực thi tính toán thực	18 µs/lệnh		
Tốc độ thực thi Boolean	0,1 µs/lệnh		

# GIỚI THIỆU VỀ PLC S7 1200

## □ Cấu tạo của PLC S7 1200



- ① Bộ phận kết nối nguồn
- ② Các bộ phận kết nối nối dây của người dùng có thể tháo được (phía sau các nắp che)
- ③ Khe cắm thẻ nhớ nằm dưới cửa phía trên
- ④ Các LED trạng thái dành cho I/O tích hợp
- ④ Bộ phận kết nối PROFINET (phía trên của CPU).

# GIỚI THIỆU VỀ PLC S7 1200

---

## ❑ Tính năng nổi bật:

### ❖ Cổng truyền thông **Profinet (Ethernet)** được tích hợp sẵn:

- Dùng để kết nối máy tính, với màn hình HMI hay truyền thông PLC-PLC.
- Dùng kết nối với các thiết bị khác có hỗ trợ chuẩn Ethernet mở.
- Đầu nối RJ45 với tính năng tự động chuyển đổi đấu chéo.
- Tốc độ truyền 10/100 Mbits/s.
- Hỗ trợ 16 kết nối ethernet.
- TCP/IP, ISO on TCP, và S7 protocol.

# GIỚI THIỆU VỀ PLC S7 1200

---

## Tính năng nổi bật:

### ❖ Các tính năng về đo lường, điều khiển vị trí, điều khiển quá trình:

- 6 bộ đếm tốc độ cao (high speed counter) dùng cho các ứng dụng đếm và đo lường, trong đó có 3 bộ đếm 100kHz và 3 bộ đếm 30kHz.
- 2 ngõ ra PTO 100kHz để điều khiển tốc độ và vị trí động cơ bước hay bộ lái servo (servo drive).
- Ngõ ra điều rộng xung PWM, điều khiển tốc độ động cơ, vị trí valve, hay điều khiển nhiệt độ...
- 16 bộ điều khiển PID với tính năng tự động xác định thông số điều khiển (auto-tune functionality).

# GIỚI THIỆU VỀ PLC S7 1200

---

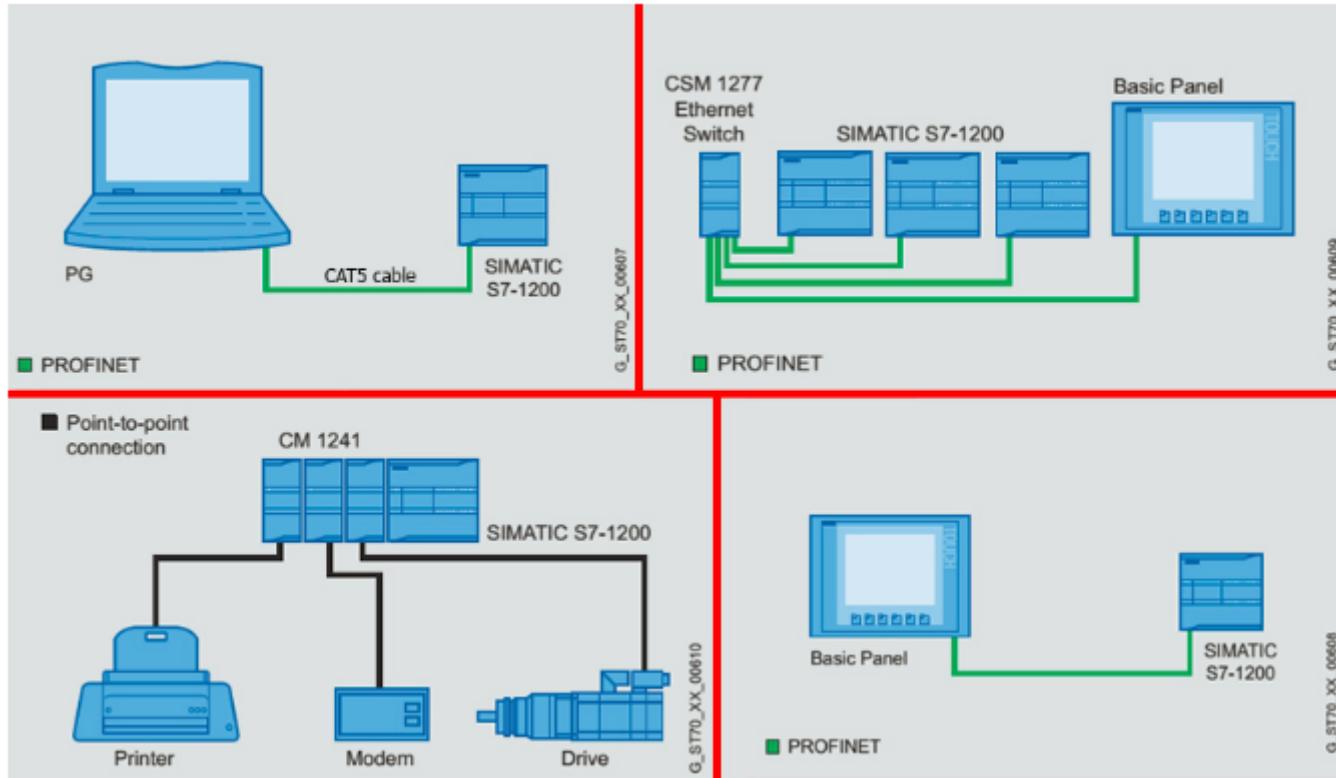
## ❑ Tính năng nổi bật:

### ❖ Thiết kế linh hoạt:

- Mở rộng tín hiệu vào/ra bằng board tín hiệu mở rộng (signal board), gắn trực tiếp phía trước CPU, giúp mở rộng tín hiệu vào/ra mà không thay đổi kích thước hệ điều khiển.
- Mỗi CPU có thể kết nối tối đa 8 module mở rộng tín hiệu vào/ra.
- Ngõ vào analog 0-10V được tích hợp trên CPU.
- 3 module truyền thông có thể kết nối vào CPU mở rộng khả năng truyền thông, vd module RS232 hay RS485.
- Card nhớ SIMATIC, dùng khi cần rộng bộ nhớ cho CPU, copy chương trình ứng dụng hay khi cập nhật firmware.
- Chẩn đoán lỗi online / offline.

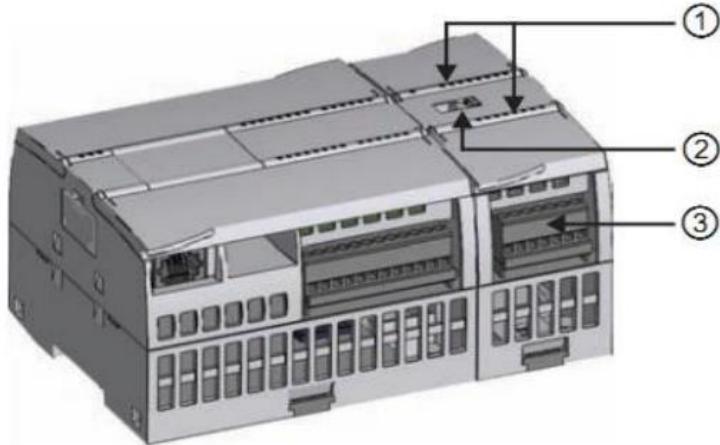
# GIỚI THIỆU VỀ PLC S7 1200

## □ Cấu hình giao tiếp của PLC S7 1200



# GIỚI THIỆU VỀ PLC S7 1200

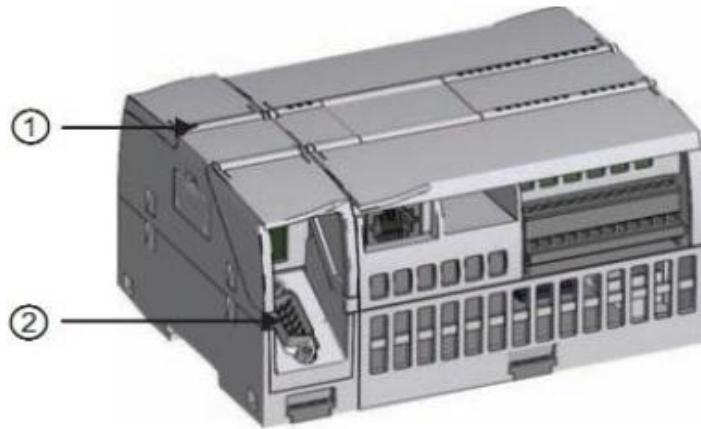
## □ Mở rộng module tín hiệu



- ① Các LED trạng thái dành cho I/O của module tín hiệu
- ② Bộ phận kết nối đường dẫn
- ③ Bộ phận kết nối nối dây của người dùng có thể tháo ra

# GIỚI THIỆU VỀ PLC S7 1200

## □ Mở rộng module truyền thông



- ① Các LED trạng thái dành cho module truyền thông
- ② Bộ phận kết nối truyền thông



# PLC 1200



## PLC 1200

- ❖ Quá trình thực thi chương trình.
- ❖ Lưu trữ dữ liệu và các vùng nhớ.
- ❖ Các kiểu dữ liệu.
- ❖ Giới thiệu ngôn ngữ lập trình LAD.
- ❖ Các lệnh IO cơ bản.
- ❖ Các lệnh so sánh.
- ❖ Các lệnh toán học.



# **QUÁ TRÌNH THỰC THI CHƯƠNG TRÌNH**

---

## **❑ Khối mã trong CPU:**

- Khối tổ chức (OB) xác định cấu trúc chương trình.
- Hàm (FC) và khối hàm (FB) chứa mã chương trình tương ứng với các nhiệm vụ riêng biệt hay với sự kết hợp các thông số.
- Khối dữ liệu (DB) lưu trữ dữ liệu mà có thể được sử dụng bởi các khối chương trình.

Sự thực thi chương trình người dùng bắt đầu với một hay nhiều hơn các khối tổ chức (OB) khởi động tùy chọn, được thực thi một lần trong lúc đi vào chế độ RUN, và được sau bởi một hay nhiều hơn các OB chu kỳ chương trình được thực thi một cách tuần hoàn.

# QUÁ TRÌNH THỰC THI CHƯƠNG TRÌNH

---

- Chu kỳ quét:** Ghi các ngõ ra, việc đọc các ngõ vào, việc thực thi các lệnh của chương trình người dùng, và việc thực hiện bảo trì hệ thống hay tiến trình xử lý nền sau.
- Ảnh tiến trình:** Vùng nhớ bên trong chứa một sự chụp nhanh các ngõ vào và ngõ ra vật lý (các điểm I/O trên CPU, trên bảng tín hiệu và trên các module tín hiệu), cập nhật đồng bộ với chu kỳ quét.
- Các tác vụ CPU thực hiện:**
  - Ghi các ngõ ra từ vùng ngõ ra ảnh tiến trình đến các ngõ ra vật lý.
  - Đọc các ngõ vào chỉ ưu tiên cho sự thực thi chương trình người dùng và lưu trữ các giá trị ngõ vào trong vùng ngõ vào ảnh tiến trình.
  - Thực thi logic của các lệnh người dùng và cập nhật các giá trị ngõ ra trong vùng ngõ ra ảnh tiến trình.

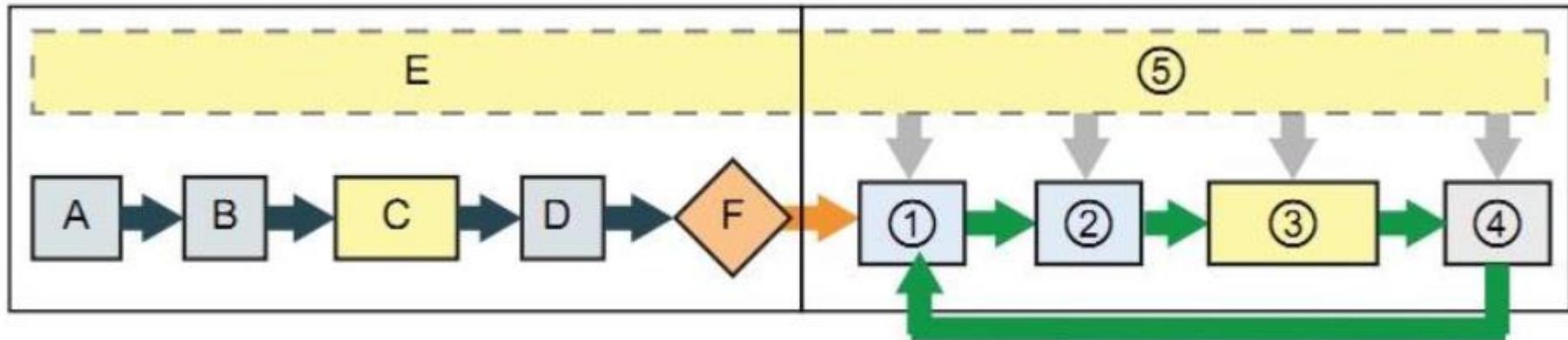
# CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA CPU

❑ CPU có 3 chế độ hoạt động: STOP, STARTUP và RUN.

- Trong chế độ STOP, CPU không thực thi chương trình nào, và ta có thể tải xuống một dự án.
- Trong chế độ STARTUP, các OB khởi động (nếu có) được thực thi một lần. Các sự kiện ngắt không được xử lý cho đến pha khởi động của chế độ RUN
- Trong chế độ RUN, chu kỳ quét được thực thi một cách lặp lại. Các sự kiện ngắt có thể xuất hiện và được thực thi tại bất kỳ điểm nào nằm trong pha chu kỳ chương trình.

**LƯU Ý:** *Không thể tải xuống một dự án trong khi đang ở chế độ RUN mà chỉ khi CPU ở trong chế độ STOP.*

# CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA CPU

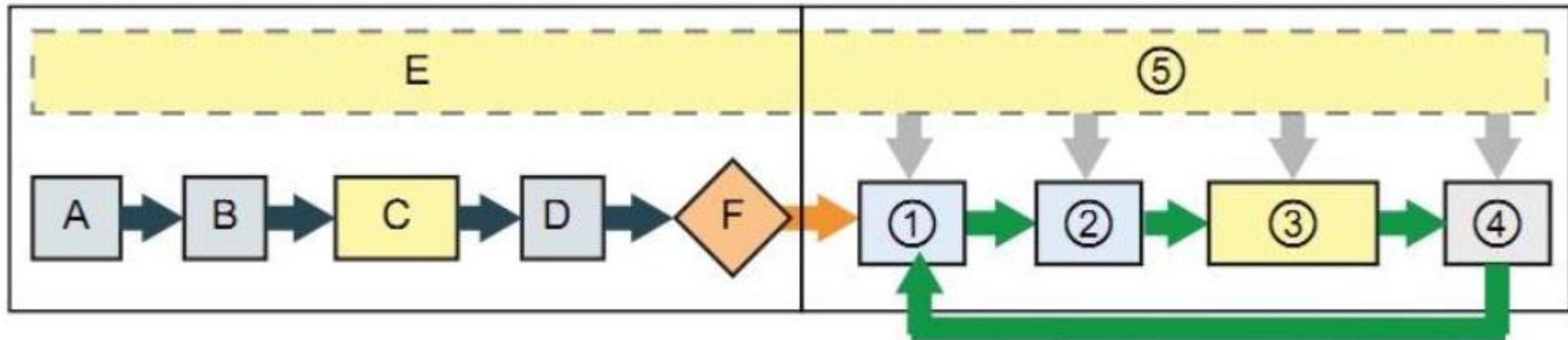


## ❑ STARTUP

Các tác vụ mà CPU thực hiện trong chế độ RUN

- A Xóa vùng nhớ I.
- B Khởi chạy các ngõ ra cả với giá trị cuối cùng hay giá trị thay thế.
- C Thực thi các OB khởi động.
- D Sao chép trạng thái của các ngõ vào vật lý đến vùng nhớ I.
- E Lưu trữ bất kỳ các sự kiện ngắn nào vào trong thứ tự để xử lý trong chế độ RUN.
- F Kích hoạt việc ghi vùng nhớ Q đến các ngõ ra vật lý.

# CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA CPU



## RUN

- (1) Ghi bộ nhớ Q đến các ngõ ra vật lý.
- (2) Sao chép trạng thái các ngõ vào vật lý đến vùng nhớ I.
- (3) Thực thi các OB chu kỳ chương trình.
- (4) Thực hiện các chẩn đoán tự kiểm tra.
- (5) Xử lý các ngắt và truyền thông trong suốt bất kỳ phần nào của chu kỳ quét.

# LƯU TRỮ DỮ LIỆU VÀ CÁC VÙNG NHỚ

## ❑ Các vùng nhớ

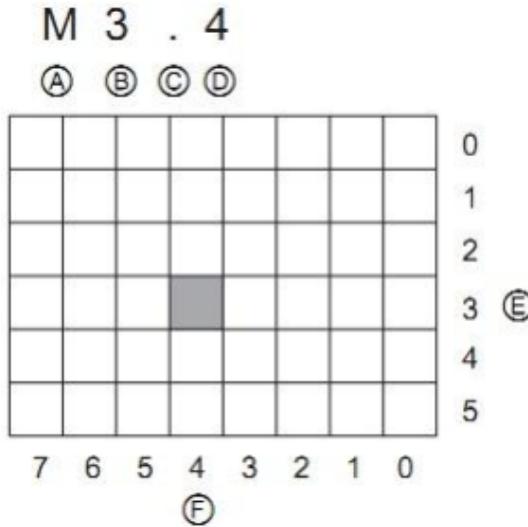
- **Global memory (Bộ nhớ toàn cục):** Ngõ vào (I), các ngõ ra (Q) và bộ nhớ bit (M). Bộ nhớ này là có thể truy xuất bởi tất cả các khối mã mà không có sự hạn chế nào.
- **Data block (DB - khối dữ liệu):** Lưu trữ dữ liệu cho các khối mã. Dữ liệu được lưu trữ vẫn duy trì khi sự thực thi của một khối mã có liên quan dần kết thúc.
- **Temp memory (Bộ nhớ tạm thời):** Khi một khối mã được gọi, hệ điều hành của CPU phân bổ bộ nhớ tạm thời hay cục bộ (L) để sử dụng trong suốt sự thực thi của khối. Khi sự thực thi của khối hoàn thành, CPU sẽ phân bổ lại bộ nhớ cục bộ dành cho việc thực thi các khối mã khác.

# LƯU TRỮ DỮ LIỆU VÀ CÁC VÙNG NHỚ

Vùng nhớ	Miêu tả	Ép buộc	Lưu giữ
I Ngõ vào ảnh tiến trình	Được sao chép từ các ngõ vào vật lý tại điểm bắt đầu của chu trình quét	Không	Không
I_:P Ngõ vào vật lý	Việc đọc ngay lập tức của các điểm ngõ vào trên CPU, SB và SM	Có	Không
Q Ngõ ra ảnh tiến trình	Được sao chép đến các ngõ ra vật lý tại điểm bắt đầu của chu trình quét	Không	Không
Q_:P Ngõ ra vật lý	Việc ghi ngay lập tức đến các điểm ngõ ra vật lý trên CPU, SB và SM	Có	Không
M Bộ nhớ bit	Bộ nhớ dữ liệu và điều khiển	Không	Có
L Bộ nhớ tạm thời	Dữ liệu tạm thời cho một khối, một bộ phận của khối đó	Không	Không
DB Khối dữ liệu	Bộ nhớ dữ liệu và còn là bộ nhớ thông số dành cho các FB	Không	Có

# LƯU TRỮ DỮ LIỆU VÀ CÁC VÙNG NHỚ

## Cách thức truy xuất một bit



A Bộ định danh vùng nhớ

B Địa chỉ byte: byte 3

C Dấu ngăn cách

D Vị trí bit của byte (bit 4 trong số 8 bit)

E Các byte của vùng nhớ

F Các bit của byte được chọn

Để truy xuất ngõ vào hay ngõ ra vật lý, ta cộng thêm tham chiếu với ký tự “:P” (như là IO.3:P, Q1.7:P hay “Stop:P”)

# LƯU TRỮ DỮ LIỆU VÀ CÁC VÙNG NHỚ

## I (ngõ vào ảnh tiến trình):

- CPU tiến hành lấy mẫu các điểm ngõ vào vật lý ngoại vi vừa trước khi thực thi OB chu trình của mỗi chu trình quét và ghi các giá trị này đến ảnh tiến trình ngõ vào.
- Có thể truy xuất đến ảnh tiến trình ngõ vào theo bit, byte, word hay double word.
- Cả truy xuất đọc và ghi đều được cho phép, nhưng thông thường, các ngõ vào ảnh tiến trình là chỉ đọc.
- Các truy xuất sử dụng I:\_P không ảnh hưởng đến giá trị tương ứng được lưu trữ trong ảnh tiến trình ngõ vào.

Bit	I [địa chỉ byte].[địa chỉ bit]	I0.1
Byte, Word hay Double Word	I [kích thước].[địa chỉ byte khởi đầu]	IB4, IW5 hay ID12

# LƯU TRỮ DỮ LIỆU VÀ CÁC VÙNG NHỚ

## Q (ngõ ra ảnh tiến trình):

- CPU sao chép các giá trị được lưu trữ trong ảnh tiến trình ngõ ra đến các điểm ngõ ra vật lý.
- Có thể truy xuất ảnh tiến trình ngõ ra theo bit, byte, word hay double word.
- Cả truy xuất đọc và ghi đều được cho phép đổi với các ngõ ra ảnh tiến trình.
- Các truy xuất sử dụng Q\_.P ảnh hưởng đến cả ngõ ra vật lý cũng như giá trị tương ứng được lưu trữ trong ảnh tiến trình ngõ ra.

Bit	Q [địa chỉ byte].[địa chỉ bit]	Q0.1
Byte, Word hay Double Word	Q [kích thước].[địa chỉ byte khởi đầu]	QB5, QW10 hay QB40

# LƯU TRỮ DỮ LIỆU VÀ CÁC VÙNG NHỚ

## M (vùng nhớ bit):

- Dữ liệu dùng để lưu trữ trạng thái tức thời của một sự vận hành hay của các thông tin điều khiển khác.
- Có thể truy xuất vùng bộ nhớ *bit* theo *bit*, *byte*, *word* hay *double word*.
- Truy xuất đọc và ghi đều được cho phép đối với bộ nhớ M.

Bit	M [địa chỉ <i>byte</i> ].[địa chỉ <i>bit</i> ]	M26.7
Byte, Word hay Double Word	M [kích thước].[địa chỉ <i>byte</i> khởi đầu]	MB20, MW30, MD50

# LƯU TRỮ DỮ LIỆU VÀ CÁC VÙNG NHỚ

## Temp (bộ nhớ tạm):

- CPU phân bổ bộ nhớ tạm thời trên một nền tảng theo yêu cầu.
- CPU phân bổ bộ nhớ tạm thời cho khối mã tại thời điểm khi khối mã được bắt đầu (đối với một OB) hay được gọi (đối với một FC hay một FB).
- Sự phân bổ bộ nhớ tạm thời cho một khối mã có thể sử dụng lại cùng một vị trí bộ nhớ Temp được sử dụng trước đó bởi một OB, FC hay FB khác.
- Sự phân bổ bộ nhớ tạm thời cho một khối mã có thể sử dụng lại cùng một vị trí bộ nhớ Temp được sử dụng trước đó bởi một OB, FC hay FB khác.
- CPU không thiết lập giá trị ban đầu cho bộ nhớ tạm thời tại thời điểm phân bổ và do đó bộ nhớ tạm thời có thể chứa bất kỳ giá trị nào.

# LƯU TRỮ DỮ LIỆU VÀ CÁC VÙNG NHỚ

- Bộ nhớ tạm thời giống với bộ nhớ M ngoại trừ một điểm chính: bộ nhớ M có một dải hợp lệ là “**global**” (tổng thể) còn bộ nhớ tạm thời có dải hợp lệ là “**local**” (cục bộ).

## □ DB (Khối dữ liệu):

- Lưu trữ các kiểu dữ liệu khác nhau, bao gồm trạng thái trung gian của một hoạt động hay các thông số về thông tin điều khiển khác cho các FB, và các cấu trúc dữ liệu cần thiết cho nhiều lệnh như các bộ định thì hay các bộ đếm.

Bit	DB [số hiệu khối dữ liệu].DBX [địa chỉ byte].[địa chỉ bit]	DB1.DBX2.3
Byte, Word hay Double Word	DB [số hiệu khối dữ liệu].DB [kích cỡ].[địa chỉ byte bắt đầu]	DB1.DBB4, DB10.DBW2, DB20.DBD8

# CÁC KIỂU DỮ LIỆU

Kiểu dữ liệu	Kích thước (bit)	Phạm vi	Các ví dụ mục nhập vào cố định
Bool	1	0 đến 1	TRUE, FALSE, 0, 1
Byte	8	16#00 đến 16#FF	16#12, 16#AB
Word	16	16#0000 đến 16#FFFF	16#ABCD, 16#0001
DWord	32	16#00000000 đến 16#FFFFFF	16#02468ACE
Char	8	16#00 đến 16#FF	'A', 't', '@'
SInt	16	-128 đến 127	123, -123
Int	16	-32768 đến 32767	123, -123
DInt	32	-2147483648 đến 2147483647	123, -123

# CÁC KIỂU DỮ LIỆU

USInt	8	0 đến 255	123
UInt	16	0 đến 65535	123
UDInt	32	0 đến 4294967295	123
Real	32	$+- 1.18 \times 10^{-38}$ đến $+/- 3.40 \times 10^{38}$	123456, -3.4, -1.2E+12, 3.4E-3
LReal	32	$+- 2.23 \times 10^{-308}$ đến $+/- 1.79 \times 10^{308}$	12345.123456789, - 1.2E+40
Time	32	T#-24d_20h_31m_23s_648ms T#24d_20h_31m_23s_647ms Được lưu trữ dưới dạng : -2147483648 ms đến +2147483647 ms	T#5m_30s 5#-2d T#1d_2h_15m_30s_45ms
String	Thay đổi	Các ký tự có kích thước 0 đến 254 byte	'ABC'

# CÁC KIỂU DỮ LIỆU



## ❑ Mảng:

- Chứa nhiều phần tử của một kiểu cơ bản.
- Được tạo ra trong trình soạn thảo giao diện khối dành cho OB, FC, FB và DB.

Tên	Kiểu dữ liệu	Chú giải
My_Bits	Array [1 .. 10] đối với kiểu Bool	Mảng này chứa 10 giá trị Boolean
My_Data	Array [-5 .. 5] đối với kiểu SInt	Mảng này chứa 11 giá trị SInt, bao gồm chỉ mục 0

- Tham chiếu các phần tử của mảng:

**Array\_name[i]**, trong đó i là chỉ mục mong muốn

Ví dụ: #My\_Bits[3] - tham chiếu bit thứ 3 của mảng "My\_Bits".



# CÁC KIỂU DỮ LIỆU

## ❑ Kiểu dữ liệu DTL (Data and Time Long):

- Một cấu trúc gồm 12 byte lưu trữ thông tin trên ngày và giờ trong một cấu trúc được xác định trước.
- Có thể xác định một DTL cả trong bộ nhớ tạm Temp của khối hay trong một DB.
- Mỗi phần của DTL chứa một kiểu dữ liệu khác nhau và phạm vi của các giá trị.

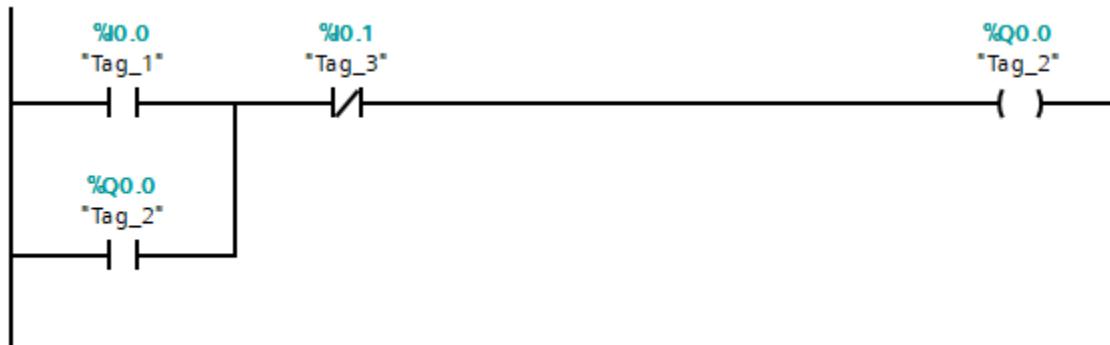
Độ dài (byte)	Định dạng	Phạm vi giá trị	Ví dụ về ngõ vào giá trị
12	Đếm thời gian và lịch (Year-Month:Hour:Minute:Second.Nanosecond)	Tối thiểu: DTL#1970-01-01-:00:00:00.0 Tối đa: DTL#2554-12-31-:23:59:59.999999999	DTL#2008-12-16-20:30:20.250

# CÁC KIỂU DỮ LIỆU

Byte	Thành phần	Kiểu dữ liệu	Phạm vi giá trị
0	Year	UInt	1970 đến 2554
1			
2	Month	USInt	1 đến 12
3	Day	USInt	1 đến 31
4	Day of week	USInt	1 (Chủ nhật) đến 2 (thứ bảy) Ngày trong tuần không cần chú ý trong mục nhập giá trị
5	Hour	USInt	0 đến 23
6	Minute	USInt	0 đến 59
7	Second	USInt	0 đến 59
8	Nanosecond	UDInt	0 đến 999 999 999
9			
10			
11			

# **GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH LAD**

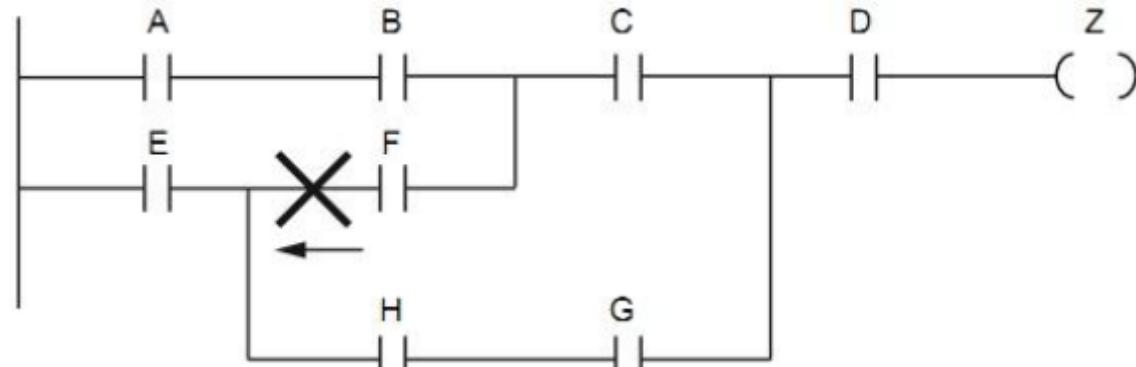
- ❑ **LAD** (ladder logic) là một ngôn ngữ lập trình kiểu đồ họa. Sự hiển thị được dựa trên các sơ đồ mạch điện.
  - ❑ Các phần tử của một sơ đồ mạch điện, như các tiếp điểm thường đóng hay thường mở, và các cuộn dây được nối với nhau để tạo thành các mạng.
  - ❑ Để tạo ra sơ đồ logic cho các thực thi phức tạp, ta có thể chèn vào các nhánh để tạo ra các mạch logic song song.



# GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH LAD

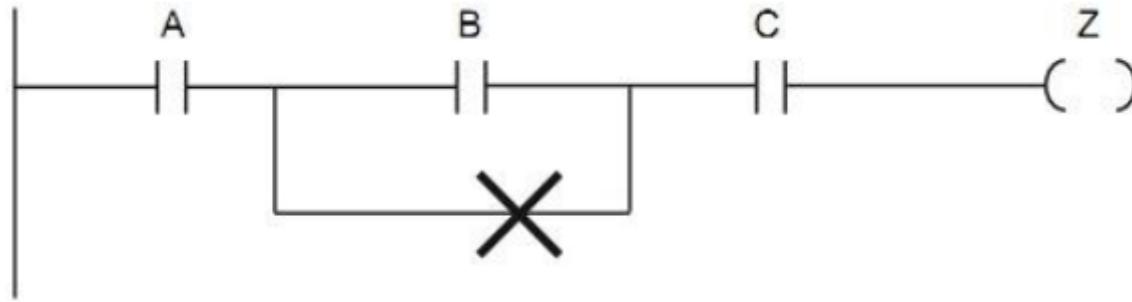
## ❑ Các quy tắc khi tạo mạng LAD

- Mỗi mạng LAD phải kết thúc bằng một cuộn dây hay một lệnh dạng hộp.
- Không được kết thúc một mạng với cả lệnh so sánh (Compare) hay lệnh phát hiện ngưỡng (ngưỡng dương hay ngưỡng âm).
- Không thể tạo ra một nhánh mà có thể đưa lại kết quả là một dòng tín hiệu theo chiều ngược lại.



# GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH LAD

- Không thể tạo ra một nhánh mà có thể gây nên ngắn mạch.



# CÁC LỆNH IO CƠ BẢN

## ❑ Bit logic:

### ❖ Các tiếp điểm ladder (LAD)

- Có thể kết nối các tiếp điểm với nhau và tạo ra mạch logic kết nối.
- Nếu bit ngõ vào mà ta chỉ rõ sử dụng bộ định danh I (ngõ vào) hay Q (ngõ ra), giá trị bit sẽ được đọc từ một thanh ghi ảnh tiến trình.
- CPU quét các tín hiệu ngõ vào được nối và cập nhật liên tục các giá trị tương ứng trong thanh ghi ngõ vào ảnh tiến trình.



NO (thường mở) NC (thường đóng)

# CÁC LỆNH IO CƠ BẢN

- Có thể ghi rõ một kết quả tức thời của một ngõ vào vật lý bằng cách sử dụng “:P” theo sau sự dịch chỉnh I (ví dụ: “%I3.4:P”).
- Đối với một kết quả tức thời, các giá trị dữ liệu bit được đọc một cách trực tiếp từ ngõ vào vật lý thay vì từ ảnh tiến trình.
- Một kết quả tức thời thì không cập nhật ảnh tiến trình

Thông số	Kiểu dữ liệu	Miêu tả
IN	Bool	Bit được gán giá trị

# CÁC LỆNH IO CƠ BẢN

## □ Lưu ý:

- Tiếp điểm thường hở NO (Normally Open) được đóng lại (ON) khi giá trị bit được gán bằng 1.
- Tiếp điểm thường đóng NC (Normally Closed) được đóng lại (ON) khi giá trị bit được gán bằng 0.
- Các tiếp điểm được nối nối tiếp sẽ tạo ra mạch logic AND.
- Các tiếp điểm được nối song song sẽ tạo ra mạch logic OR.

# CÁC LỆNH IO CƠ BẢN

## ❑ Bộ đảo logic NOT:

- Tiếp điểm NOT (LAD) chuyển đổi trạng thái logic của đầu vào dòng tín hiệu:
  - Nếu không có dòng tín hiệu vào trong tiếp điểm NOT, sẽ có dòng tín hiệu đi ra.
  - Nếu có dòng tín hiệu vào trong tiếp điểm NOT, sẽ không có dòng tín hiệu đi ra.

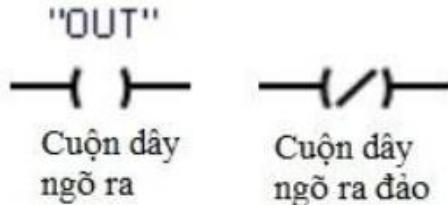
→ NOT ←

LAD: bộ đảo  
tiếp điểm NOT

# CÁC LỆNH IO CƠ BẢN

## □ Cuộn dây ngõ ra:

- Lệnh xuất cuộn dây sẽ ghi một giá trị cho một bit ngõ ra.
- Trong chế độ RUN, hệ thống CPU quét một cách liên tục các tín hiệu ngõ vào, xử lý các trạng thái ngõ vào theo chương trình logic, và sau đó tác động trở lại bằng cách thiết lập các giá trị trạng thái ngõ ra mới trong thanh ghi ngõ ra ảnh tiến trình.
- Sau mỗi chu trình thực thi chương trình, hệ thống CPU chuyển phản ứng trạng thái ngõ ra mới được lưu trữ trong thanh ghi ảnh tiến trình đến các đầu cực nối dây ngõ ra.



# CÁC LỆNH IO CƠ BẢN

- Có thể xác định một kết quả ghi tức thời của một ngõ ra vật lý bằng cách sử dụng “:P” sau độ dịch chuyển Q (ví dụ “%Q3.4:P”).
- Đối với một kết quả ghi tức thời, các giá trị dữ liệu bit **được ghi** đến ngõ ra ảnh tiến trình và trực tiếp đến ngõ ra vật lý.

Thông số	Kiểu dữ liệu	Miêu tả
OUT	Bool	Bit được gán giá trị

# CÁC LỆNH IO CƠ BẢN

## □ LƯU Ý:

- Nếu có luồng tín hiệu chạy qua một cuộn dây ngõ ra, bit ngõ ra được đặt lên 1.
- Nếu không có luồng tín hiệu chạy qua một cuộn dây ngõ ra, bit ngõ ra được đặt về 0.
- Nếu có luồng tín hiệu chạy qua một cuộn dây ngõ ra đảo, bit ngõ ra được đặt về 0.
- Nếu không có luồng tín hiệu chạy qua một cuộn dây ngõ ra đảo, bit ngõ ra được đặt lên 1.

# CÁC LỆNH IO CƠ BẢN

## ❑ Các lệnh Set (Đặt) và Reset (Đặt lại): 1 bit

- Khi lệnh S (Set) được kích hoạt, giá trị dữ liệu ở địa chỉ OUT được đặt lên 1. Khi lệnh S không được kích hoạt, ngõ ra OUT không bị thay đổi.
- Khi lệnh R (Reset) được kích hoạt, giá trị dữ liệu ở địa chỉ OUT được đặt về 0. Khi lệnh R không được kích hoạt, ngõ ra OUT không bị thay đổi.
- Những lệnh này có thể được đặt tại bất cứ vị trí nào trong mạch.

LAD: Set

"OUT"

—(S)—

LAD: Reset

"OUT"

—(R)—

Thông số	Kiểu dữ liệu	Miêu tả
IN	Bool	Vị trí bit được giám sát
OUT	Bool	Vị trí bit được đặt hoặc đặt lại

# CÁC LỆNH IO CƠ BẢN

## □ SET\_BF và RESET\_BF: Set và Reset một trường bit

- Khi SET\_BF được kích hoạt, một giá trị dữ liệu bằng 1 được gán cho “n” bit bắt đầu tại địa chỉ OUT. Khi SET\_BF không được kích hoạt, địa chỉ OUT không bị thay đổi.
- RESET\_BF ghi một giá trị dữ liệu bằng 0 đến “n” bit bắt đầu tại địa chỉ OUT. Khi RESET\_BF không được kích hoạt, địa chỉ OUT không bị thay đổi.
- Những lệnh này phải là lệnh nằm về bên phải trong một nhánh.

LAD: SET\_BF

"OUT"

—(SET\_BF)

"n"

LAD: RESET\_BF

"OUT"

—(RESET\_BF)

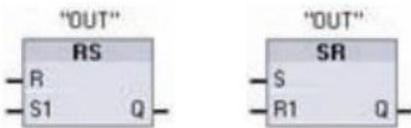
"n"

Thông số	Kiểu dữ liệu	Miêu tả
n	Constant	Số lượng các bit để ghi
OUT	Phần tử bắt đầu của một mảng Boolean	Phần tử bắt đầu của một trường bit được đặt hay đặt lại. Ví dụ #MyArray[3]

# CÁC LỆNH IO CƠ BẢN

## □ RS và SR: Các mạch chốt của bit set trội và reset trội

- RS là một mạch chốt set trội mà set chiếm ưu thế. Nếu tín hiệu set (S1) và reset (R) đều là đúng, địa chỉ ngõ ra OUT sẽ bằng 1.
- SR là một mạch chốt reset trội mà reset chiếm ưu thế. Nếu tín hiệu set (S) và reset (R1) đều là đúng thì địa chỉ ngõ ra OUT sẽ là 0.
- Thông số OUT định rõ địa chỉ *bit* được set hay reset. Ngõ ra OUT tùy chọn (Q) phản ánh trạng thái tín hiệu của địa chỉ OUT.



Thông số	Kiểu dữ liệu	Miêu tả
S, S1	Bool	Ngõ vào set; số “1” biểu thị sự ưu thế
R, R1	Bool	Ngõ vào reset; số “1” biểu thị sự ưu thế
OUT	Bool	Ngõ ra của bit được gán “OUT”
Q	Bool	Trạng thái kèm theo của bit “OUT”

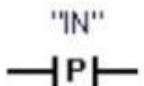
# CÁC LỆNH IO CƠ BẢN

Lệnh	S1	R	Bit “OUT”
RS	0	0	Trạng thái kè trước
	0	1	0
	1	0	1
	1	1	1
	S	R1	Bit “OUT”
SR	0	0	Trạng thái kè trước
	0	1	0
	1	0	1
	1	1	0

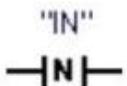
# CÁC LỆNH IO CƠ BẢN

## ☐ Các lệnh ngưỡng dương và âm

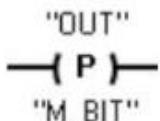
P contact: LAD



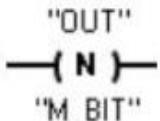
N contact: LAD



P coil: LAD



N coil: LAD



P\_TRIG: LAD\FBD



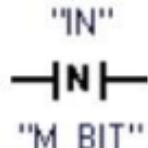
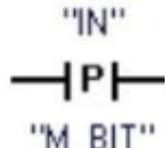
N\_TRIG: LAD\FBD



Thông số	Kiểu dữ liệu	Miêu tả
M_BIT	Bool	Bit nhớ trong đó trạng thái kề trước của ngõ vào được lưu trữ
IN	Bool	Bit ngõ vào mà ngưỡng quá độ của nó là dùng để phát hiện
OUT	Bool	Bit ngõ ra, cho biết một ngưỡng quá độ đã được phát hiện
CLK	Bool	Luồng tín hiệu hay bit ngõ vào mà ngưỡng quá độ của chúng là dùng để phát hiện
Q	Bool	Ngõ ra biểu thị một ngưỡng đã được phát hiện

# CÁC LỆNH IO CƠ BẢN

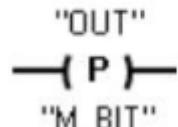
- Tiếp điểm P (LAD): trạng thái của tiếp điểm là “TRUE” khi một sự quá độ dương (từ OFF sang ON) được phát hiện trên bit “IN” được gán.
- Tiếp điểm P có thể được định vị tại bất kỳ vị trí nào trong mạch, ngoại trừ vị trí kết thúc của một nhánh.
- Tiếp điểm N (LAD): trạng thái của tiếp điểm này là “TRUE” khi một sự quá độ âm (từ ON sang OFF) được phát hiện trên bit được gán.
- Tiếp điểm N có thể được định vị tại bất kỳ vị trí nào trong mạch, ngoại trừ vị trí kết thúc của một nhánh.



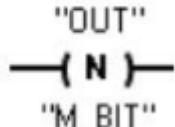
# CÁC LỆNH IO CƠ BẢN

- Cuộn dây P (LAD): bit được gán “OUT” là “TRUE” khi một sự quá độ dương (từ OFF sang ON) được phát hiện trên dòng tín hiệu đi vào cuộn dây.
- Dòng tín hiệu trong mạch luôn chạy xuyên qua cuộn dây, đóng vai trò như trạng thái ngõ ra dòng tín hiệu.
- Cuộn dây P có thể được định vị tại bất kỳ vị trí nào trong mạch.
- Cuộn dây N (LAD): bit được gán “OUT” là “TRUE” khi một sự quá độ âm (từ ON sang OFF) được phát hiện trên dòng tín hiệu đi vào cuộn dây.
- Cuộn dây N có thể được định vị tại bất kỳ vị trí nào trong mạch.

P coil: LAD



N coil: LAD



# CÁC LỆNH IO CƠ BẢN

- P\_TRIG (LAD/FBD): dòng tín hiệu ngõ ra Q hoặc trạng thái logic là “TRUE” khi một sự quá độ dương (từ OFF sang ON) được phát hiện trên dòng tín hiệu CLK (LAD).
- Lệnh P\_TRIG không thể được định vị tại vị trí khởi đầu hay kết thúc của một mạch.
- N\_TRIG (LAD/FBD): dòng tín hiệu ngõ ra Q hoặc trạng thái logic là “TRUE” khi một sự quá độ âm (từ ON sang OFF) được phát hiện trên dòng tín hiệu CLK (LAD).
- Lệnh N\_TRIG không thể được định vị tại vị trí khởi đầu hay kết thúc của một mạch.

P\_TRIG: LAD\FBD

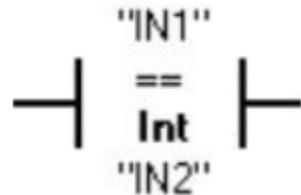


N\_TRIG: LAD\FBD



# CÁC LỆNH SO SÁNH

- Sử dụng các lệnh so sánh để so sánh hai giá trị của cùng một kiểu dữ liệu.
- Khi việc so sánh tiếp điểm LAD là “TRUE”, tiếp điểm này được kích hoạt.
- Có thể lựa chọn kiểu so sánh và kiểu dữ liệu từ các trình đơn thả xuống



Thông số	Kiểu dữ liệu	Miêu tả
IN1, IN2	SInt, Int, Dint, USInt, UInt, UDInt, Real, LReal, String, Char, Time, DTL, Constant	Các giá trị để so sánh

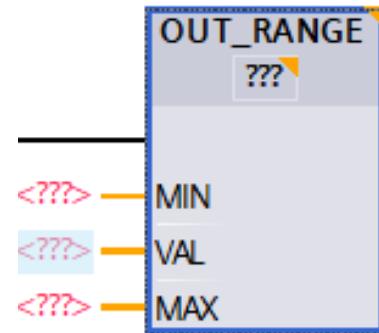
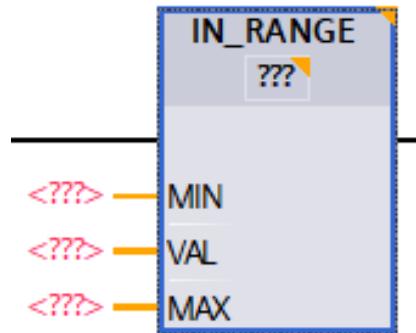
# CÁC LỆNH SO SÁNH

Kiểu quan hệ	Sự so sánh là đúng nếu:
$= =$	IN1 bằng IN2
$\diamond$	IN1 không bằng IN2
$>=$	IN1 lớn hơn hay bằng IN2
$<=$	IN1 nhỏ hơn hay bằng IN2
$>$	IN1 lớn hơn IN2
$<$	IN1 nhỏ hơn IN2

# CÁC LỆNH SO SÁNH

## ❑ Các lệnh “IN\_RANGE” và “OUT\_RANGE”

- Sử dụng các lệnh IN\_RANGE và OUT\_RANGE để kiểm tra trong trường hợp một giá trị ngõ vào nằm trong hay nằm ngoài mức giá trị được định sẵn.
- Nếu sự so sánh là “TRUE” thì ngõ ra của hộp là “TRUE”.
- Các thông số ngõ vào MIN, VAL và MAX phải có cùng kiểu dữ liệu.



# CÁC LỆNH SO SÁNH

- Sau khi nhấp chuột lên lệnh trong trình soạn thảo chương trình, ta có thể lựa chọn kiểu dữ liệu từ các trình đơn thả xuống.

Kiểu quan hệ	Sự so sánh là đúng nếu:
IN_RANGE	$\text{MIN} \leq \text{VAL} \leq \text{MAX}$
OUT_RANGE	$\text{VAL} < \text{MIN}$ hoặc $\text{VAL} > \text{MAX}$

Thông số	Kiểu dữ liệu	Miêu tả
MIN, VAL, MAX	SInt, Int, DInt, USInt, UDInt, Real, Constant	Các ngoặc vào phần tử so sánh

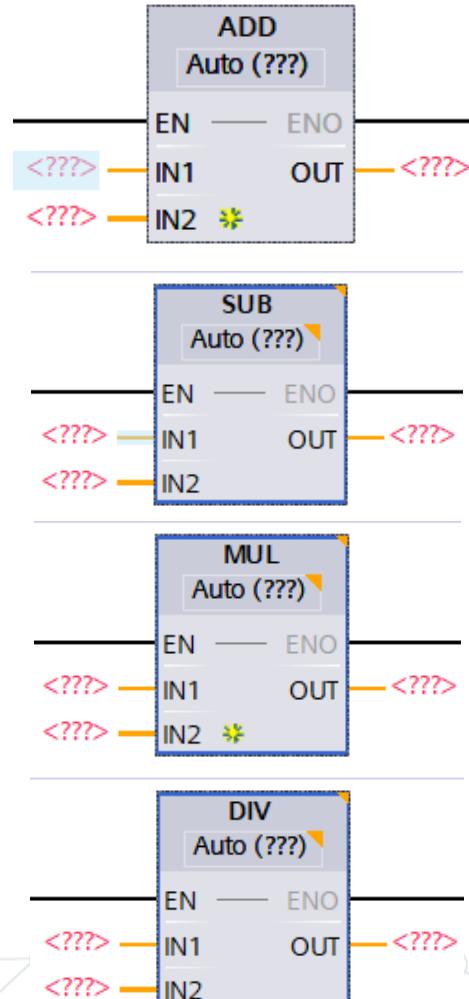
# CÁC LỆNH TOÁN HỌC

## ❑ Cách lệnh cộng, trừ, nhân, chia

- ADD : phép cộng ( $IN1 + IN2 = OUT$ )
- SUB : phép trừ ( $IN1 - IN2 = OUT$ )
- MUL : phép nhân ( $IN1 * IN2 = OUT$ )
- DIV : phép chia ( $IN1 / IN2 = OUT$ )

### ❖ Lưu ý:

- Một hoạt động chia số nguyên sẽ cắt bỏ phần phân số của thương số để tạo ra một tín hiệu ra số nguyên.
- Các thông số lệnh phép toán cơ bản IN1, IN2 và OUT phải có kiểu dữ liệu giống nhau.



# CÁC LỆNH TOÁN HỌC

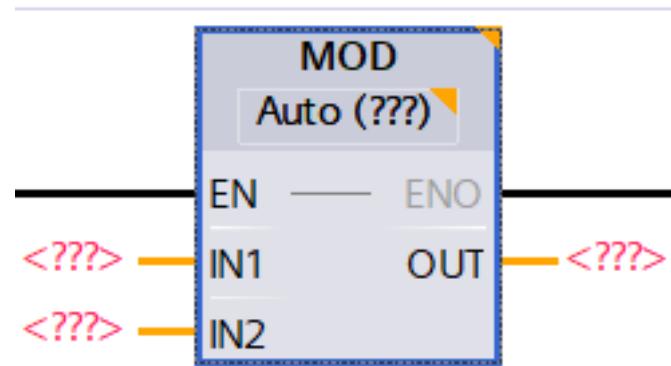
Thông số	Kiểu dữ liệu	Miêu tả
IN1, IN2	SInt, Int, DInt, UInt, UDInt, Real, LReal, Constant	Các ngõ vào phép toán
OUT	SInt, Int, DInt, UInt, UDInt, Real, LReal	Ngõ ra phép toán

- Khi được cho phép (EN = 1), lệnh phép toán thực hiện hoạt động được định rõ trên các giá trị ngõ vào (IN1 và IN2) và lưu trữ kết quả trong địa chỉ nhớ được xác định bởi thông số ngõ ra (OUT).
- Sau một sự hoàn tất thành công phép toán, lệnh sẽ đặt ENO = 1.

# CÁC LỆNH TOÁN HỌC

## ❑ Lệnh chia lấy phần dư (MOD)

- Phép toán  $IN1 \text{ MOD } IN2 = IN1 - (IN1/IN2) = \text{thông số OUT}$ .
- Nhấp vào phía dưới tên hộp và chọn một kiểu dữ liệu từ trình đơn thả xuống.
- Các thông số IN1, IN2 và OUT phải có kiểu dữ liệu giống nhau.



# CÁC LỆNH TOÁN HỌC

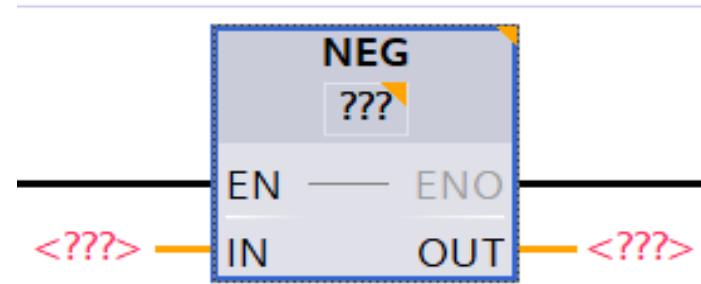
Thông số	Kiểu dữ liệu	Miêu tả
IN1 và IN2	Int, DInt, USInt, UInt, UDInt, Constant	Các ngõ vào <i>modulo</i>
OUT	Int, DInt, USInt, UInt, UDInt	Ngõ ra <i>modulo</i>

Trạng thái ENO	Miêu tả
1	Không có lỗi
0	Giá trị IN2 = 0, OUT được gán giá trị bằng 0

# CÁC LỆNH TOÁN HỌC

## ❑ Lệnh NEG

- Sử dụng lệnh NEG (phép đảo) để đảo ngược dấu số học của giá trị tại thông số IN và lưu trữ kết quả trong thông số OUT.
- Nhấp vào phía dưới tên hộp và chọn một kiểu dữ liệu từ trình đơn thả xuống.
- Các thông số IN và OUT phải có kiểu dữ liệu giống nhau.



# CÁC LỆNH TOÁN HỌC

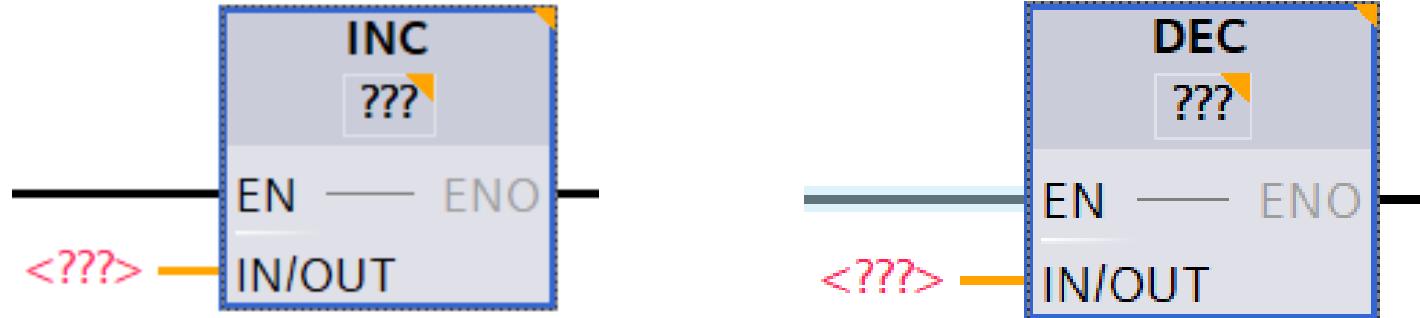
Thông số	Kiểu dữ liệu	Miêu tả
IN	SInt, Int, DInt, Real, LReal, Constant	Ngõ vào phép toán
OUT	SInt, Int, DInt, Real, LReal	Ngõ ra phép toán

Trạng thái ENO	Miêu tả
1	Không có lỗi
0	Giá trị kết quả vượt quá phạm vi hợp lệ của kiểu dữ liệu được chọn. Ví dụ đối với SInt: NEG (- 128) cho kết quả + 128 vượt quá giá trị tối đa của kiểu dữ liệu này.

# CÁC LỆNH TOÁN HỌC

## ❑ Các lệnh Tăng (INC) và giảm (DEC)

- Tăng giá trị một số nguyên có dấu hoặc không dấu. INC: giá trị thông số IN/OUT + 1 = giá trị thông số IN/OUT.
- Giảm giá trị một số nguyên có dấu hoặc không dấu. DEC: giá trị thông số IN/OUT – 1 = giá trị thông số IN/OUT.
- Nhấp vào phía dưới tên hộp và chọn một kiểu dữ liệu từ trình đơn thả xuống.



# CÁC LỆNH TOÁN HỌC

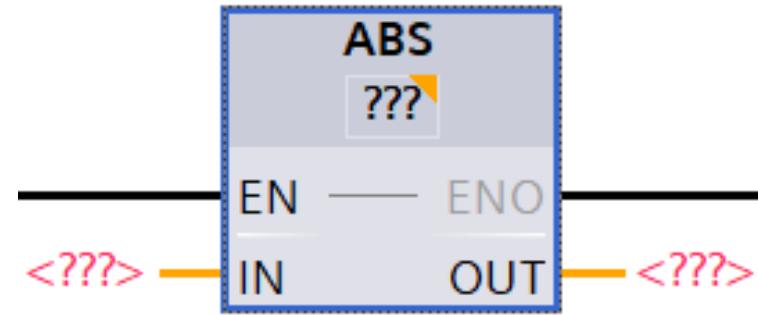
Thông số	Kiểu dữ liệu	Miêu tả
IN/OUT	SInt, Int, DInt, USInt, UDInt	Ngõ vào và ngõ ra phép toán

Trạng thái ENO	Miêu tả
1	Không có lỗi
0	Giá trị kết quả vượt quá phạm vi hợp lệ của kiểu dữ liệu được chọn. Ví dụ đối với SInt: INC (127) cho kết quả + 128 vượt quá giá trị tối đa của kiểu dữ liệu này.

# CÁC LỆNH TOÁN HỌC

## ❑ Lệnh giá trị tuyệt đối

- Sử dụng lệnh ABS để nhận được giá trị tuyệt đối của một số nguyên có dấu hoặc một số thực tại thông số IN và lưu trữ kết quả trong thông số OUT.
- Nhấp vào phía dưới tên hộp và chọn một kiểu dữ liệu từ trình đơn thả xuống.
- Các thông số IN và OUT phải có kiểu dữ liệu giống nhau.



# CÁC LỆNH TOÁN HỌC

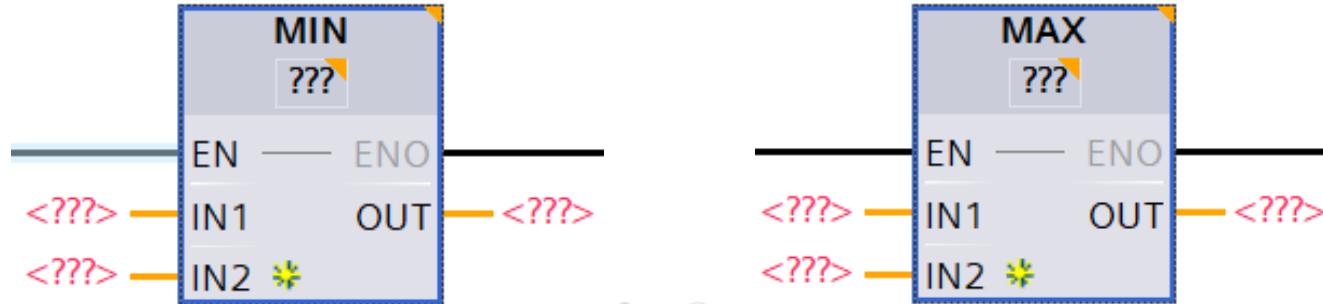
Thông số	Kiểu dữ liệu	Miêu tả
IN	SInt, Int, DInt, Real, LReal	Ngõ vào phép toán
OUT	SInt, Int, DInt, Real, LReal	Ngõ ra phép toán

Trạng thái ENO	Miêu tả
1	Không có lỗi
0	Giá trị kết quả phép toán vượt quá phạm vi hợp lệ của kiểu dữ liệu được chọn. Ví dụ đối với SInt: ABS (- 128) cho kết quả + 128 vượt quá giá trị tối đa của kiểu dữ liệu này.

# CÁC LỆNH TOÁN HỌC

## ❑ Lệnh MIN và MAX

- Lệnh MIN so sánh giá trị của hai thông số IN1 và IN2 và gán giá trị cực tiểu (nhỏ hơn) cho thông số OUT.
- Lệnh MAX so sánh giá trị của hai thông số IN1 và IN2 và gán giá trị cực đại (lớn hơn) cho thông số OUT.
- Nhấp vào phía dưới tên hộp và chọn một kiểu dữ liệu từ trình đơn thả xuống.
- Các thông số IN và OUT phải có kiểu giá trị giống nhau.



# CÁC LỆNH TOÁN HỌC

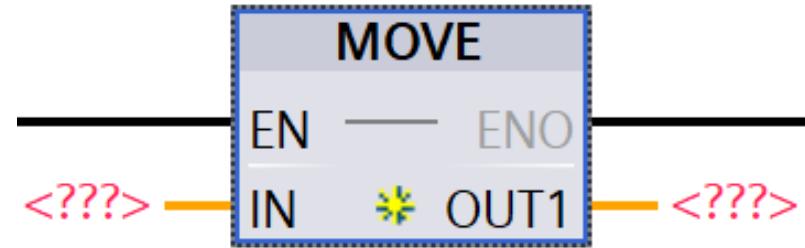
Thông số	Kiểu dữ liệu	Miêu tả
MIN, IN và MAX	SInt, Int, DInt, USInt, UInt, UDInt, Real, Constant	Các ngõ vào phép toán
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real	Ngõ ra phép toán

Trạng thái ENO	Miêu tả
1	Không có lỗi
0	Real: nếu một hay nhiều hơn các giá trị của MIN, IN và MAX là NaN (không phải một số) thì NaN sẽ được trả về.
0	Nếu MIN lớn hơn MAX, giá trị IN được gán đến OUT.

# CÁC LỆNH TOÁN HỌC

## ❑ Lệnh di chuyển (MOVE)

- Sử dụng các lệnh di chuyển để sao chép các phần tử dữ liệu đến một địa chỉ nhớ mới và chuyển đổi từ một kiểu dữ liệu này sang kiểu khác.
- Dữ liệu nguồn không bị thay đổi trong quá trình di chuyển.
- MOVE: sao chép một phần tử dữ liệu được lưu trữ tại một địa chỉ xác định đến một địa chỉ mới.



# CÁC LỆNH TOÁN HỌC

MOVE		
Thông số	Kiểu dữ liệu	Miêu tả
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Char, Array, Struct, DTL, Time	Địa chỉ nguồn
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Char, Array, Struct, DTL, Time	Địa chỉ đích



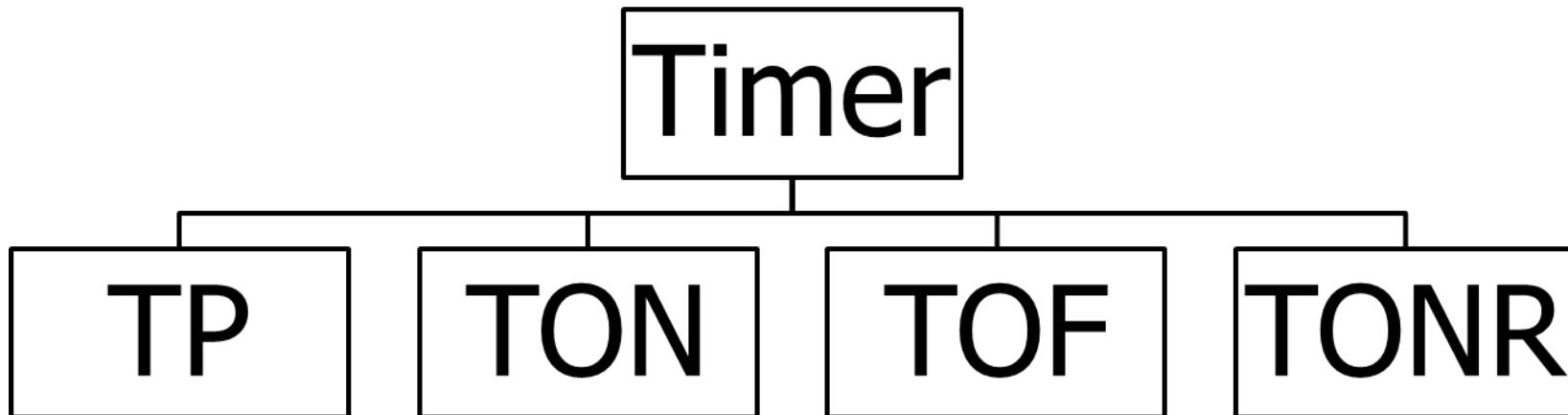
# BỘ ĐỊNH THỜI VÀ BỘ ĐÊM

# NỘI DUNG

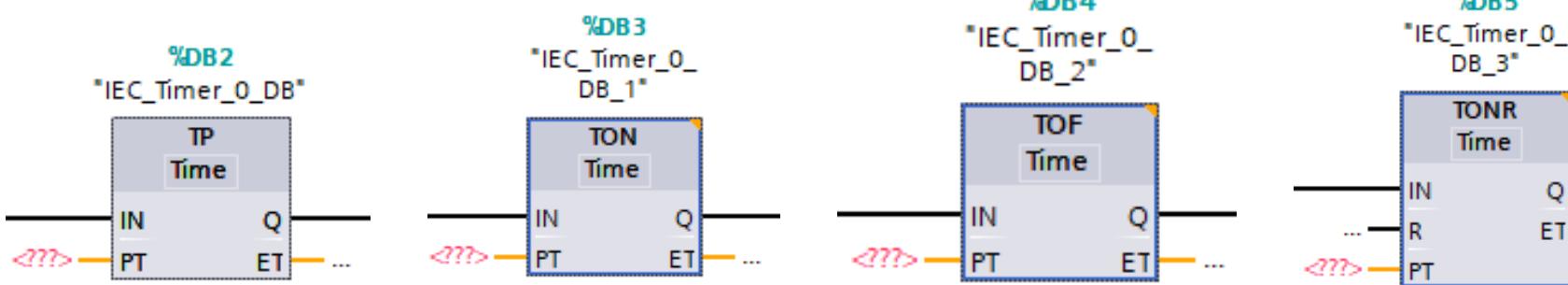
- ❖ Bộ định thời (Timer)
- ❖ Bộ đếm (Counter)

## BỘ ĐỊNH THỜI (TIMER)

- ❑ Bộ định thời: Tạo ra các trì hoãn thời gian được lập trình.



# BỘ ĐỊNH THỜI (TIMER)



Thông số	Kiểu dữ liệu	Miêu tả
IN	Bool	Ngõ vào bộ định giờ cho phép
R	Bool	Đặt lại thời gian trôi qua của TONR về 0
PT	Bool	Ngõ vào giá trị thời gian đặt trước
Q	Bool	Ngõ ra bộ định giờ
ET	Time	Ngõ ra giá trị thời gian trôi qua
Khối dữ liệu định giờ	DB	Chi ra bộ định giờ nào để đặt lại với lệnh RT

# BỘ ĐỊNH THỜI (TIMER)

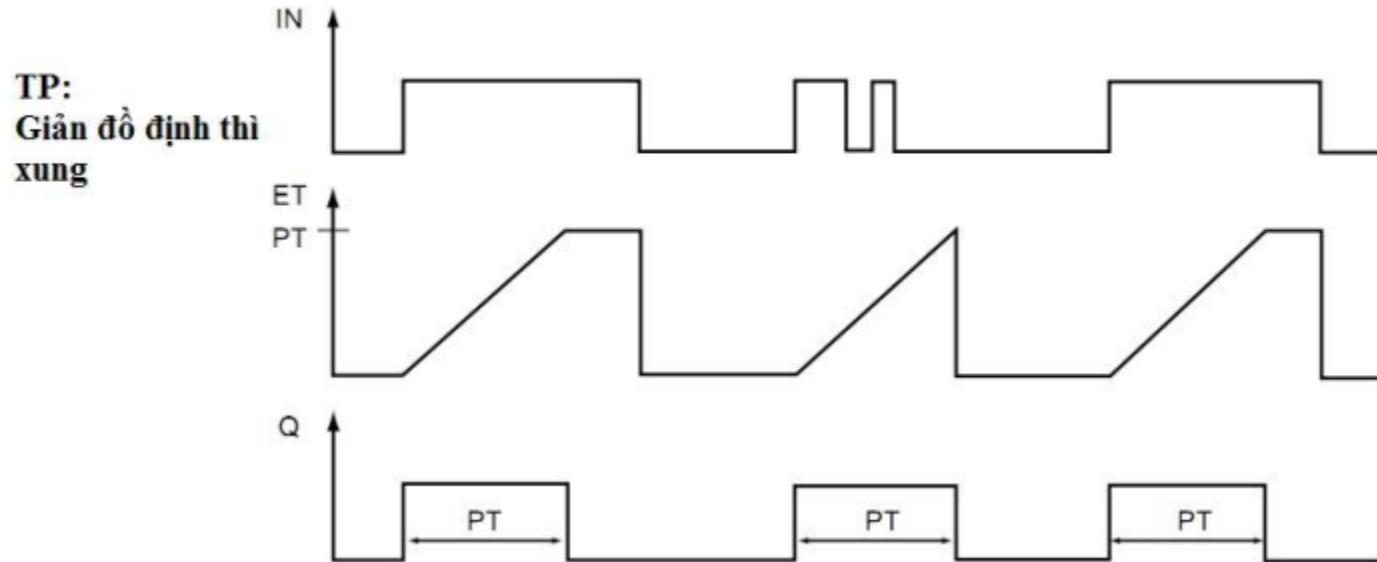
## ❑ Các giá trị TIME:

- Các giá trị PT (preset time – thời gian đặt trước) và ET (elapsed time – thời gian đã trôi qua) được lưu trữ trong bộ nhớ như các số nguyên **double** có dấu, tương trưng cho những mili giây thời gian.
- Dữ liệu TIME sử dụng bộ định danh T# và có thể được nhập vào như một đơn vị thời gian thuần túy “T#200ms” hay như các đơn vị thời gian phức hợp “T#2s\_200ms”.

Kiểu dữ liệu	Kích cỡ	Phạm vi số hợp lệ
TIME	32 bit	T#-24d_20h_31m_23s_648ms T#24d_20h_31m_23s_647ms – 2.147.483.648 ms đến + 2.147.483.647 ms

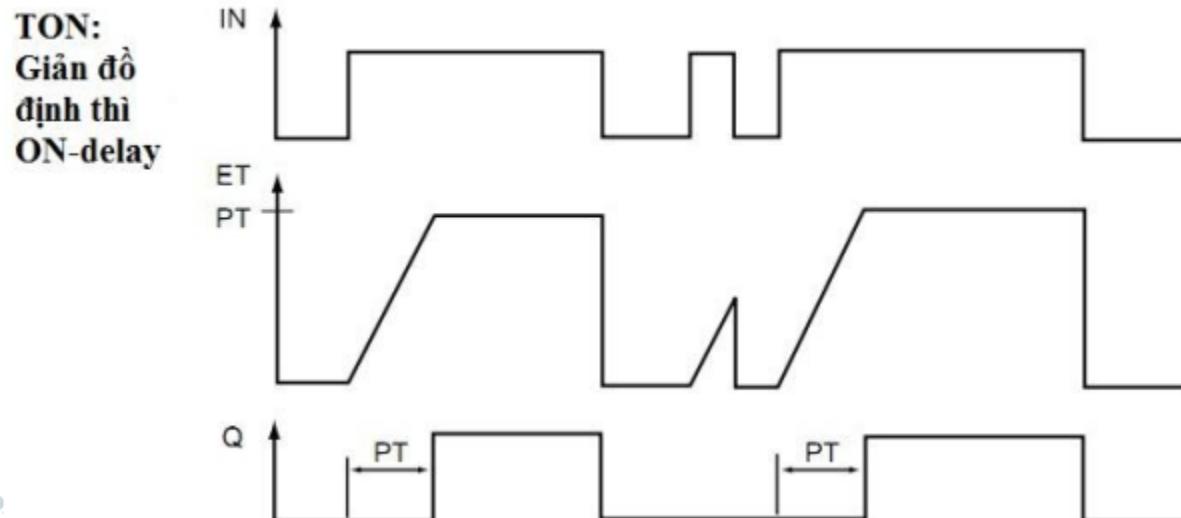
## BỘ ĐỊNH THỜI TP

- ☐ Bộ định thời xung phát ra một xung với bề rộng xung được đặt trước.
- ☐ Thay đổi PT không có ảnh hưởng trong khi bộ định thời vận hành.
- ☐ Thay đổi IN không có ảnh hưởng trong khi bộ định thời vận hành.



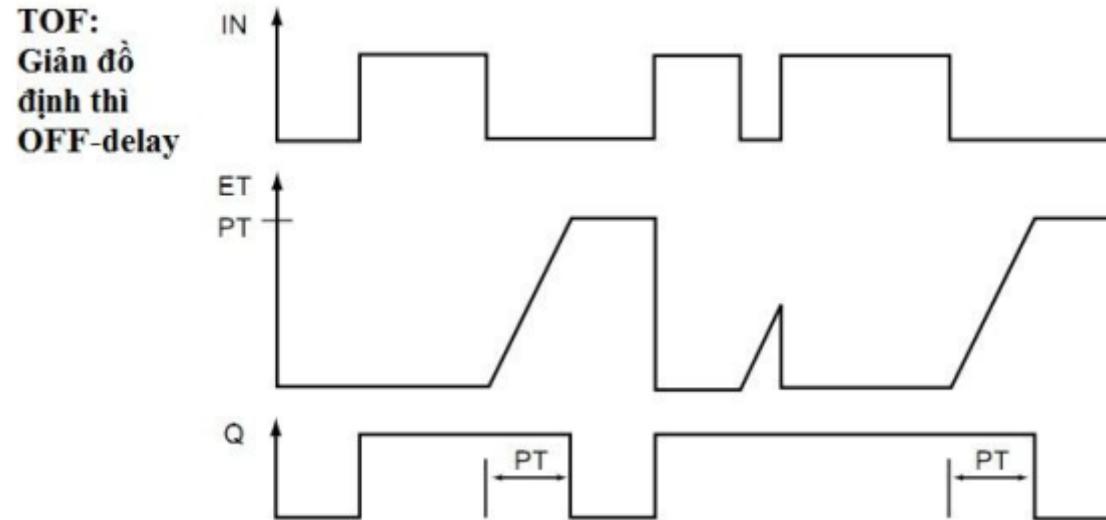
## BỘ ĐỊNH THỜI TON

- ☐ Ngõ ra của bộ định thì *ON – delay* Q được đặt lên ON sau một sự trì hoãn thời gian đặt trước.
- ☐ Thay đổi PT không có ảnh hưởng trong khi bộ định thì vận hành.
- ☐ Thay đổi IN sang “FALSE”, trong khi bộ định thì vận hành, sẽ đặt lại và dừng bộ định thì.



## BỘ ĐỊNH THỜI TOF

- ☐ Ngõ ra Q của bộ định thi *OFF – delay* được đặt lại về OFF sau một sự trì hoãn thời gian đặt trước.
- ☐ Thay đổi PT không có ảnh hưởng trong khi bộ định thi vận hành.
- ☐ Thay đổi IN sang “TRUE”, trong khi bộ định thi vận hành, sẽ đặt lại và dừng bộ định thi.



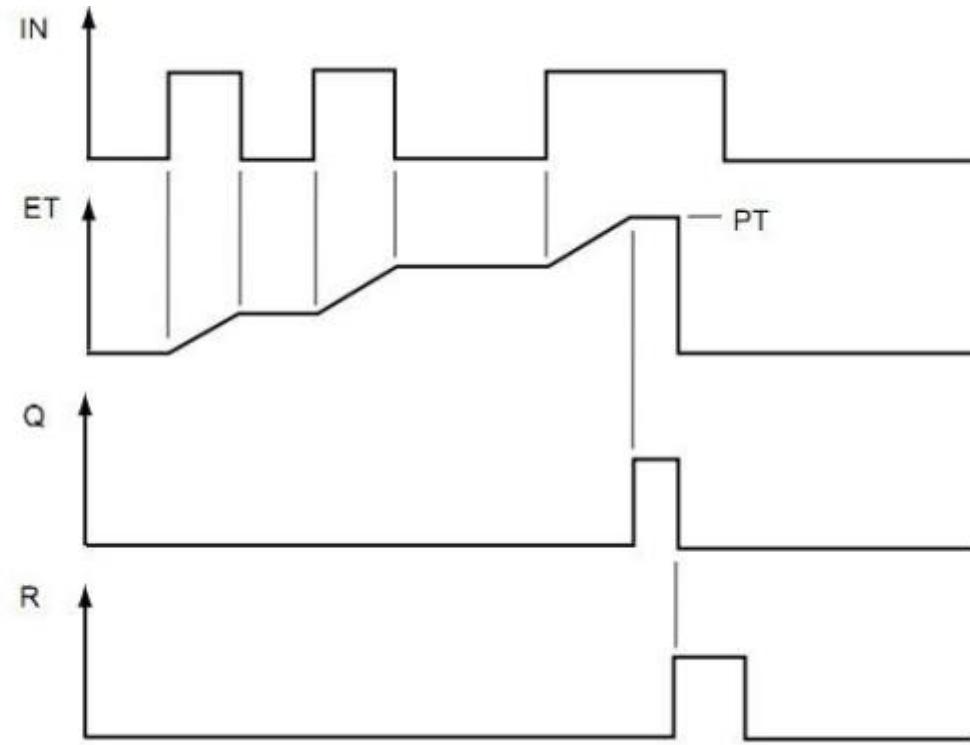
## BỘ ĐỊNH THỜI TONR

---

- Ngõ ra bộ định thì có khả năng nhớ *ON – delay* được đặt lên ON sau một trì hoãn thời gian đặt trước.
- Thời gian trôi qua được tích lũy qua nhiều giai đoạn định thì cho đến khi ngõ vào R được sử dụng để đặt lại thời gian trôi qua.
- Thay đổi PT không có ảnh hưởng trong khi bộ định thì vận hành, nhưng có ảnh hưởng khi định thì khôi phục lại.
- Thay đổi IN sang “FALSE”, trong khi bộ định thì vận hành, sẽ dừng bộ định thì nhưng không đặt lại bộ định thì.
- Thay đổi IN trở lại sang “TRUE” sẽ làm bộ định thì bắt đầu tính toán thời gian từ giá trị thời gian được tích lũy.

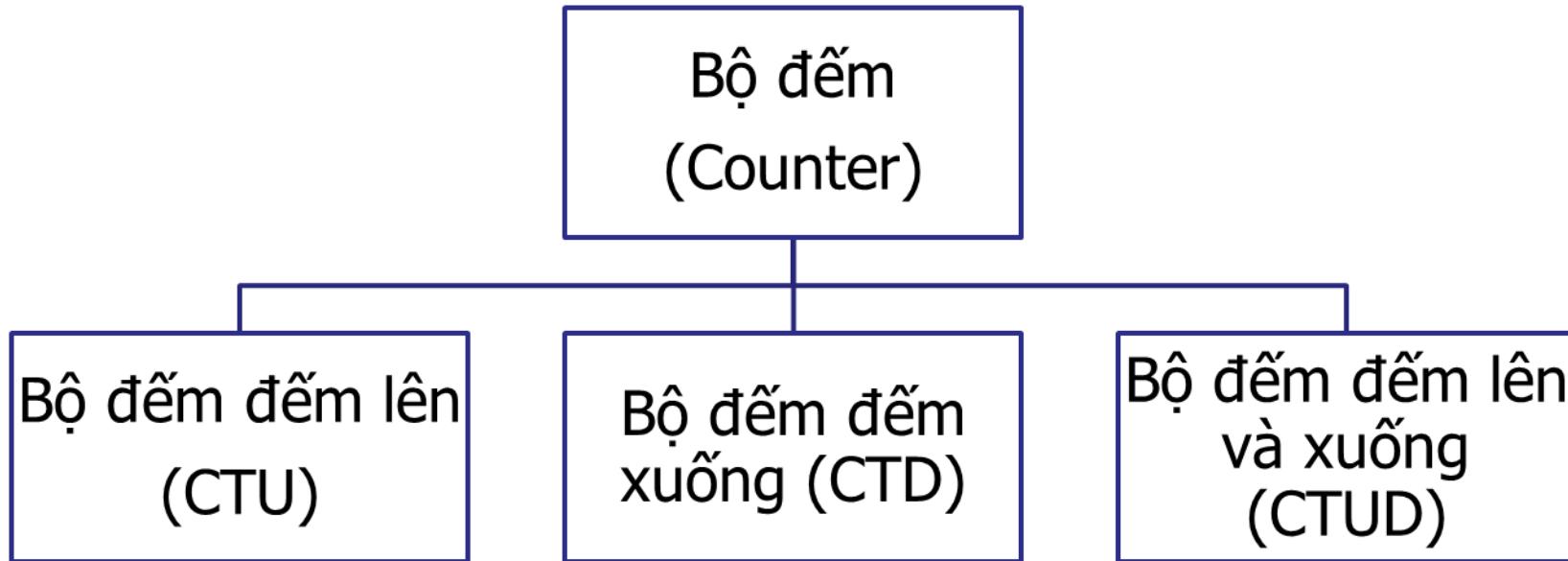
# BỘ ĐỊNH THỜI TONR

**TONR:**  
Giản đồ định  
thì có nhớ  
ON-delay

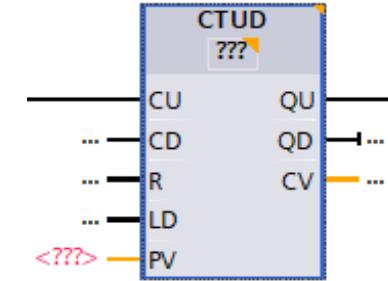
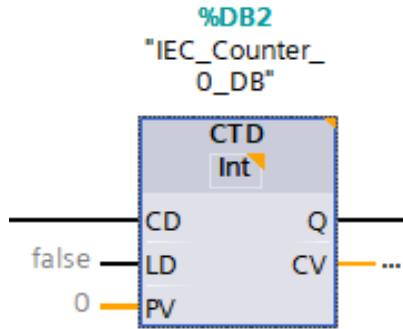
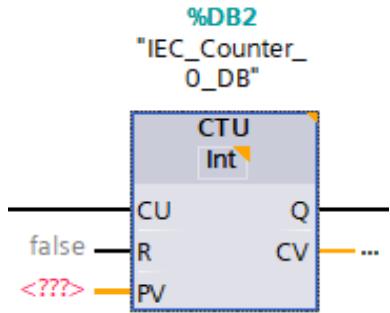


# BỘ ĐẾM (COUNTER)

- Sử dụng các lệnh bộ đếm để đếm các sự kiện chương trình bên trong và các sự kiện xử lý bên ngoài.



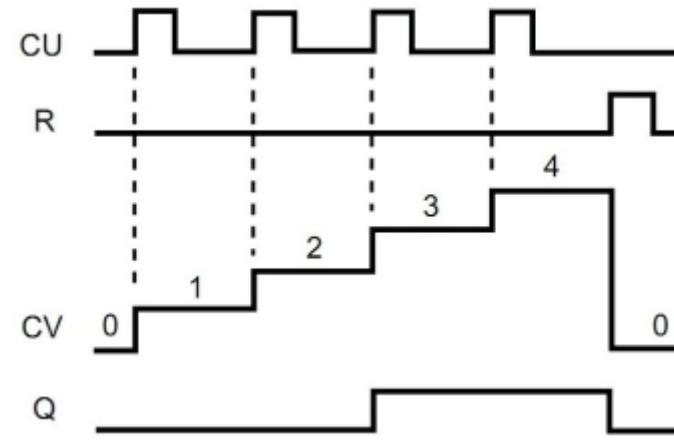
# BỘ ĐẾM (COUNTER)



Thông số	Kiểu dữ liệu	Miêu tả
CU, CD	Bool	Đếm lên hay đếm xuống, bởi một lần đếm
R (CTU, CTUD)	Bool	Đặt lại giá trị đếm về 0
LOAD (CTD, CTUD)	Bool	Nạp điều khiển cho giá trị đặt trước
PV	SInt, Int, DInt, USInt, UInt, UDInt	Giá trị đếm đặt trước
Q, QU	Bool	Đúng nếu CV $\geq$ PV
QD	Bool	Đúng nếu CV $\leq$ 0
CV	SInt, Int, DInt, USInt, UInt, UDInt	Giá trị đếm hiện thời

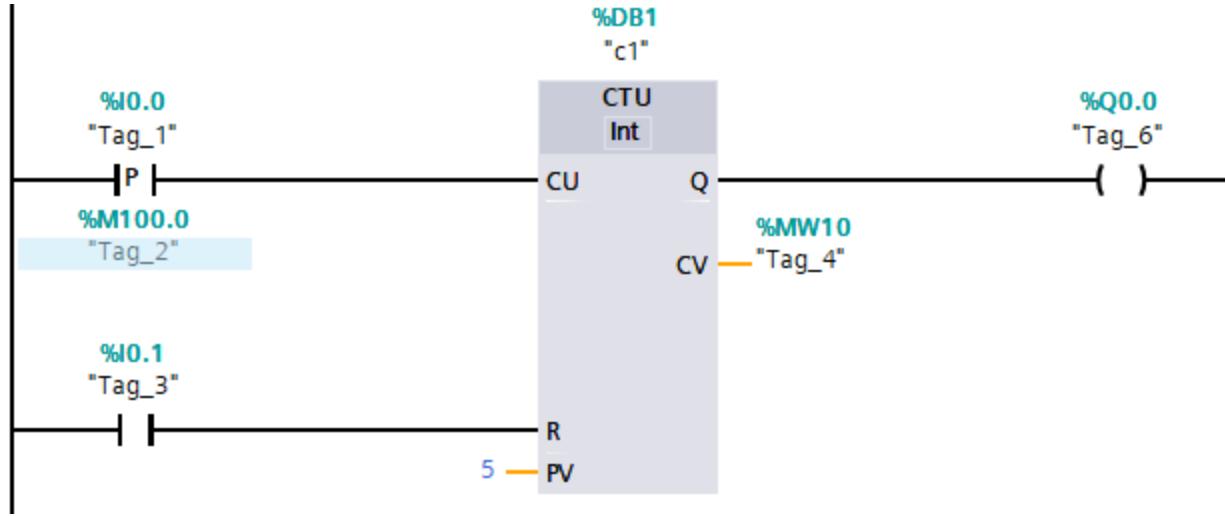
## BỘ ĐẾM ĐẾM LÊN (CTU)

- CTU đếm lên 1 đơn vị khi giá trị của thông số CU thay đổi từ 0 lên 1.
- Nếu giá trị của thông số CV (*Current count value* – giá trị đếm hiện thời) lớn hơn hoặc bằng giá trị thông số PV (*Preset count value* – giá trị đếm đặt trước) thì thông số ngõ ra của bộ đếm Q = 1.
- Nếu giá trị của thông số đặt lại R thay đổi từ 0 lên 1, giá trị đếm hiện thời được xóa về 0.



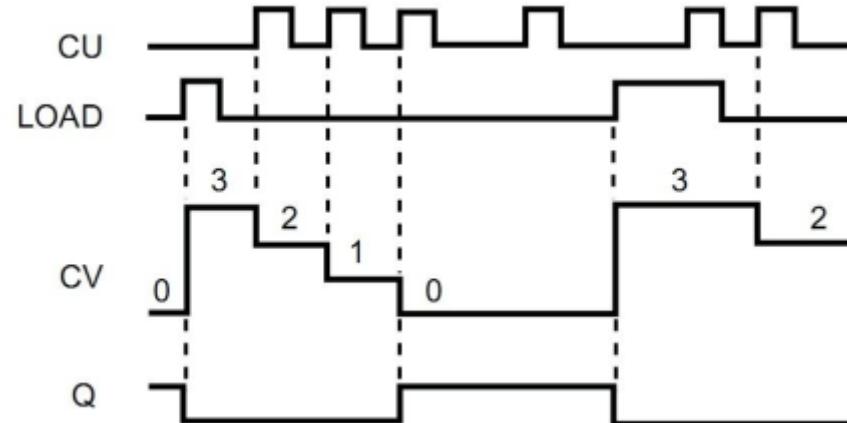
# BỘ ĐẾM ĐẾM LÊN (CTU)

❑ Ví dụ:



## BỘ ĐẾM ĐẾM XUỐNG (CTD)

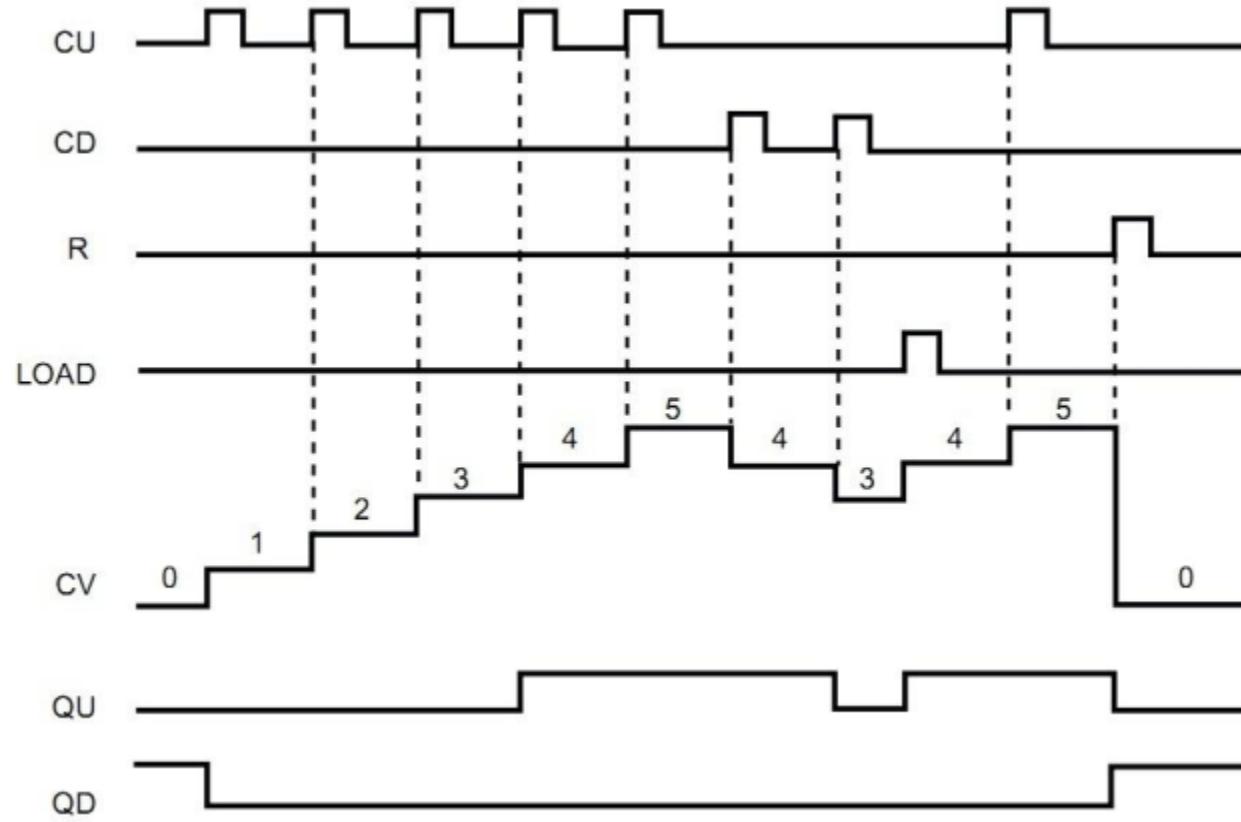
- CTD đếm xuống 1 đơn vị khi giá trị của thông số CD thay đổi từ 0 lên 1.
- Nếu giá trị của thông số CV (*Current count value* – giá trị đếm hiện thời) **nhỏ hơn hoặc bằng 0** thì thông số ngõ ra của bộ đếm Q = 1.
- Nếu giá trị của thông số LOAD thay đổi từ 0 lên 1, giá trị tại thông số PV (*Preset count value* – giá trị đặt trước) được nạp đến bộ đếm như một giá trị CV mới.



## BỘ ĐẾM ĐẾM LÊN VÀ XUỐNG (CTUD)

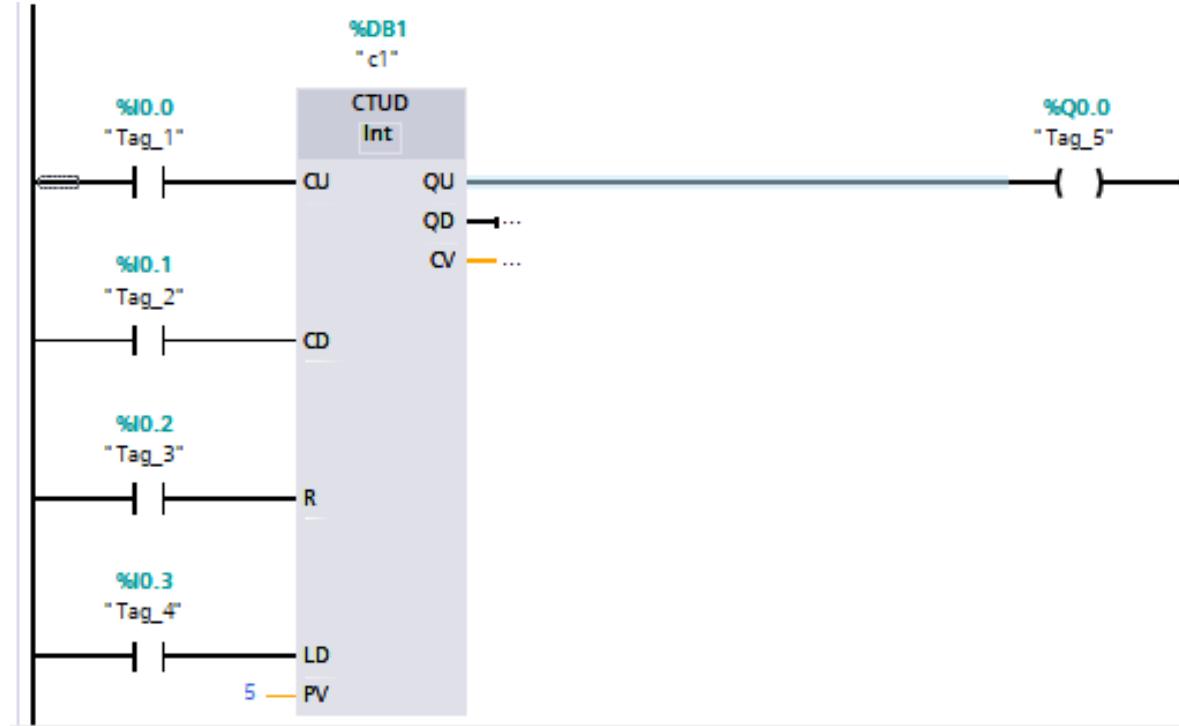
- CTUD đếm lên hay xuống 1 đơn vị theo sự quá độ từ 0 lên 1 của ngõ vào đếm lên (Count up – CU) hay đếm xuống (Count down – CD).
- Nếu giá trị của thông số CV (giá trị đếm hiện thời) lớn hơn hoặc bằng giá trị thông số PV (giá trị đếm đặt trước) thì thông số ngõ ra của bộ đếm QU = 1.
- Nếu giá trị của thông số CV nhỏ hơn hay bằng 0, thông số ngõ ra của bộ đếm QD = 1.
- Nếu giá trị của thông số LOAD thay đổi từ 0 lên 1, giá trị tại thông số PV được nạp đến bộ đếm như một giá trị CV mới.
- Nếu giá trị của thông số đặt lại R thay đổi từ 0 lên 1, giá trị đếm hiện thời sẽ được xóa về 0.

# BỘ ĐẾM ĐẾM LÊN VÀ XUỐNG (CTUD)



# BỘ ĐẾM ĐẾM LÊN VÀ XUỐNG (CTUD)

❑ Ví dụ:





NGẮT (INCORRUPT)

# NỘI DUNG

- ❖ Ngắt trong PLC S7 1200

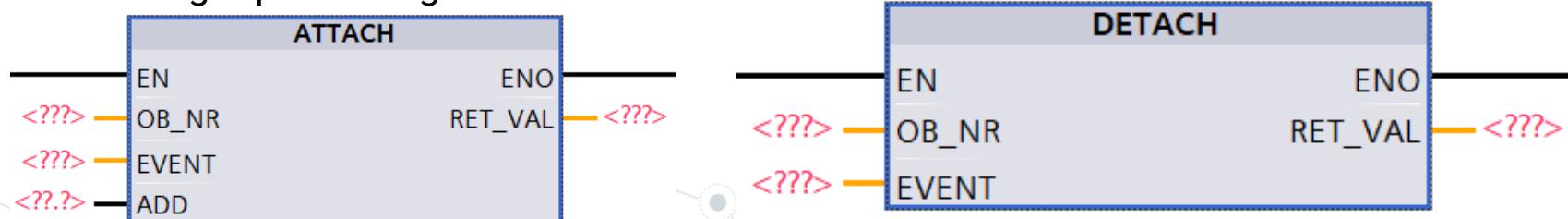
# NGẮT TRONG PLC S7 1200

---

- ❑ Hàm ngắt là một chương trình con nằm ngoài chương trình chính mà khi đạt điều kiện ngắt sẽ làm gián đoạn việc thực hiện lệnh trong chương trình chính để ưu tiên thực hiện lệnh trong hàm ngắt.
- ❑ Các khối OB ngắt:
  - Time delay interrupt: Khối ngắt thời gian trễ thực hiện sau một khoảng thời gian trễ định trước của một sự kiện (khối OB20 ).
  - Cyclic interrupt: Khối ngắt theo chu kỳ thực hiện cứ sau một khoảng thời gian nhất định (ví dụ: OB30).
  - Hardware interrupt: Khối ngắt phần cứng thực hiện khi có sự kiện ngắt đầu vào hoặc ngắt do Bộ đếm tốc độ cao (khối OB40).
  - Time error interrupt: Khối ngắt lỗi thời gian thực hiện khi có lỗi về thời gian thực hiện vòng quét của PLC hoặc khi xảy ra lỗi liên quan đến bộ định thời Timer (khối OB80).

# NGẮT TRONG PLC S7 1200

- Diagnostic interrupt: Khối ngắt chuẩn đoán thực hiện khi có một số lỗi (ví dụ đứt dây, ngắn mạch,...) xảy ra (khối OB82).
- Ngoài ra còn một số loại khối hàm OB khác như: Pull or plug of modules OB, Rack or station failure OB, Time of day OB, Status OB, Update OB, Profile OB.
- ❑ Có thể kích hoạt và làm vô hiệu các chương trình con điều khiển theo sự kiện ngắt với các lệnh ATTACH và DETACH.
- ❑ ATTACH cho phép sự thực thi chương trình con OB ngắt đối với một sự kiện ngắt phần cứng.
- ❑ DETACH làm vô hiệu sự thực thi chương trình con OB ngắt đối với một sự kiện ngắt phần cứng.



# NGẮT TRONG PLC S7 1200

Thông số	Kiểu thông số	Kiểu dữ liệu	Miêu tả
ON_NR	IN	Int	<b>Định danh khối tổ chức:</b> Lựa chọn từ các OB ngắt phần cứng có sẵn đã được tạo ra bằng chức năng “Add new block”. Nhấp đôi chuột vào trường thông số, sau đó nhấp lên biểu tượng trợ giúp để xem các OB có sẵn.
EVENT	IN	DWord	<b>Định danh sự kiện:</b> Lựa chọn từ các sự kiện ngắt phần cứng có sẵn mà đã được cho phép trong cấu hình thiết bị PLC đối với các ngõ vào số hay các bộ đếm tốc độ cao. Nhấp đôi chuột lên trường thông số, sau đó nhấp vào biểu tượng trợ giúp để xem các sự kiện có sẵn.
ADD (chỉ ATTACH)	IN	Bool	ADD = 0 (mặc định): sự kiện này làm thay thế tất cả những đính kèm sự kiện trước đó đối với OB này. ADD = 1: sự kiện này làm thêm vào tất cả những đính kèm sự kiện trước đó đối với OB này
RET_VAL	OUT	Int	Mã điều kiện thực thi

# NGẮT TRONG PLC S7 1200

## Các sự kiện ngắt phần cứng

- **Sự kiện ngưỡng tăng** (tất cả các ngõ vào số CPU tích hợp cùng với bất kỳ các ngõ vào số bảng tín hiệu): Một ngưỡng tăng xuất hiện khi ngõ vào số chuyển đổi từ OFF sang ON như một đáp ứng đến một sự thay đổi trong tín hiệu từ một trường thiết bị được kết nối đến ngõ vào.
- **Sự kiện ngưỡng giảm** (tất cả các ngõ vào số CPU tích hợp cùng với bất kỳ ngõ vào bảng tín hiệu): Một ngưỡng giảm xuất hiện khi ngõ vào số chuyển đổi từ ON sang OFF.
- **Các sự kiện giá trị hiện thời của bộ đếm tốc độ cao (HSC) = giá trị tham chiếu (CV = RV) với các HSC từ 1 đến 6:** Một ngắt CV = RV đối với một HSC được sinh ra khi giá trị đếm hiện thời chuyển đổi từ một giá trị gần kề đến giá trị mà đạt đến một giá trị tham chiếu một cách chính xác đã được ấn định trước đó.

# NGẮT TRONG PLC S7 1200

---

- **Các sự kiện thay đổi điều khiển HSC (HSC từ 1 đến 6):** Một sự kiện thay đổi điều khiển xảy ra khi HSC được phát hiện thay đổi từ tăng dần sang giảm dần, hay từ giảm dần sang tăng dần.
- **Các sự kiện đặt lại bên ngoài HSC (HSC từ 1 đến 6):** Chắc chắn các chế độ HSC cho phép sự gán của một ngõ vào số như một sự đặt lại bên ngoài mà được sử dụng để đặt lại giá trị đếm HSC về 0. Một sự kiện đặt lại bên ngoài xuất hiện đối với một vài HSC, khi tín hiệu vào này chuyển đổi từ OFF sang ON.

# CHO PHÉP CÁC SỰ KIỆN NGẮT PHẦN CỨNG TRONG CẤU HÌNH THIẾT BỊ

- Các ngắt phần cứng phải được cho phép trong cấu hình thiết bị
- Phải kiểm tra hộp sự kiện cho phép trong cấu hình thiết bị đối với một kênh ngõ vào số hay một HSC, nếu ta cần gắn kèm sự kiện này trong suốt việc cấu hình hay trong thời gian thực thi.
- Các tùy chọn hộp chọn bên trong phần cấu hình thiết bị PLC:
  - ❖ *Tín hiệu vào số*
    - Cho phép sự phát hiện ngưỡng tăng.
    - Cho phép sự phát hiện ngưỡng giảm.
  - ❖ *Bộ đếm tốc độ cao (HSC)*
    - Cho phép sử dụng bộ đếm tốc độ cao này
    - Sinh ra ngắt khi giá trị bộ đếm bằng với giá trị tham chiếu đếm
    - Sinh ra ngắt cho sự kiện đặt lại bên ngoài.
    - Sinh ra ngắt cho sự kiện thay đổi mệnh lệnh

# CHO PHÉP CÁC SỰ KIỆN NGẮT PHẦN CỨNG TRONG CẤU HÌNH THIẾT BỊ

- Các ngắt phần cứng phải được cho phép trong cấu hình thiết bị.
- Phải kiểm tra hộp sự kiện cho phép trong cấu hình thiết bị đối với một kênh ngõ vào số hay một HSC, nếu ta cần gắn kèm sự kiện này trong suốt việc cấu hình hay trong thời gian thực thi.
- Các tùy chọn hộp chọn bên trong phần cấu hình thiết bị PLC:
  - ❖ *Tín hiệu vào số*
    - Cho phép sự phát hiện ngưỡng tăng.
    - Cho phép sự phát hiện ngưỡng giảm.
  - ❖ *Bộ đếm tốc độ cao (HSC)*
    - Cho phép sử dụng bộ đếm tốc độ cao này
    - Sinh ra ngắt khi giá trị bộ đếm bằng với giá trị tham chiếu đếm.
    - Sinh ra ngắt cho sự kiện đặt lại bên ngoài.
    - Sinh ra ngắt cho sự kiện thay đổi mệnh lệnh

# THÊM CÁC KHỐI MÃ OB NGẮT PHẦN CỨNG MỚI VÀO CHƯƠNG TRÌNH

- Chỉ có các OB ngắt phần cứng mới có thể được gắn kèm đến một sự kiện ngắt phần cứng.
- Tất cả các OB ngắt phần cứng được tạo sẵn đều xuất hiện trong danh sách thả xuống “HW interrupt”:
- Nếu không có OB nào được liệt kê, khi đó ta phải tạo ra một OB hay gõ “Hardware interrupt”. Dưới nhánh “Program blocks” của cây đề án:
  - Nhấp đồi chuột vào “Add new block”, chọn “Organization block (OB)” và chọn “Hardware interrupt”.
  - Một cách tùy chọn, ta có thể đổi tên OB, lựa chọn ngôn ngữ, lựa chọn số hiệu khối.
  - Chỉnh sửa OB và thêm vào phản ứng được lập trình mà ta muốn thực thi khi sự kiện xuất hiện. Có thể gọi các FC và FB từ OB này, với một bề dày lồng vào nhau tối đa là 4.

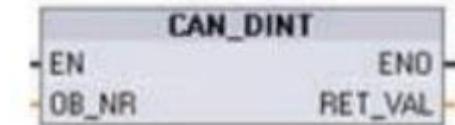
## VÍ DỤ NGẮT THEO CHU KỲ THỰC HIỆN

- Code trong OB1 và trong hàm ngắt. Sau chu kỳ 2s, chương trình chạy vào ngắt gán trạng thái cuộn Q0.0 cho cuộn Q0.1



# CÁC LỆNH KHỞI ĐỘNG VÀ BỎ QUA NGẮT TRÌ HOÃN THỜI GIAN

- ❑ Có thể khởi động và bỏ qua việc xử lý ngắt trì hoãn thời gian với các lệnh SRT\_DINT và CAN\_DINT.
- ❑ Mỗi ngắt trì hoãn thời gian là một sự kiện một lần mà nó xuất hiện sau một thời gian trì hoãn xác định.
- ❑ Nếu sự kiện trì hoãn thời gian bị bỏ qua trước khi trì hoãn thời gian hết hiệu lực, chương trình ngắt sẽ không xuất hiện.
- ❑ Lệnh SRT\_DINT khởi động một ngắt trì hoãn thời gian mà nó thực thi một chương trình con OB (organization block: khối tổ chức) khi thời gian trì hoãn được xác định bởi thông số DTIME đã trôi qua.
- ❑ Lệnh CAN\_DINT bỏ qua một ngắt trì hoãn thời gian mà nó vừa mới khởi động. OB ngắt trì hoãn thời gian sẽ không được thực thi trong trường hợp này.

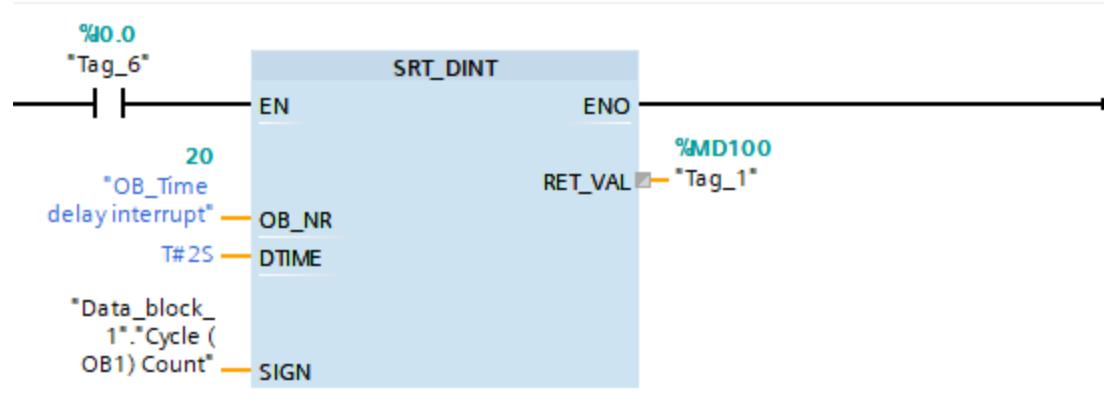


# CÁC LỆNH KHỞI ĐỘNG VÀ BỎ QUA NGẮT TRÌ HOÃN THỜI GIAN

Thông số	Kiểu thông số	Kiểu dữ liệu	Miêu tả
OB_NR	IN	Int	<p>Khởi tố chức (OB) dùng để khởi động sau một trì hoãn thời gian:</p> <p>Lựa chọn từ các OB ngắt trì hoãn thời gian có sẵn đã được tạo ra sử dụng chức năng “Add new block” ở cây đề án. Nhấp đôi chuột lên trường thông số, sau đó nhấp vào biểu tượng trợ giúp để xem các OB có sẵn.</p>
DTIME	IN	Time	<p>Giá trị trì hoãn thời gian (từ 1 đến 60000 ms)</p> <p>Ta có thể tạo ra các thời gian trì hoãn dài hơn, ví dụ bằng cách sử dụng một bộ đếm bên trong một OB ngắt trì hoãn thời gian.</p>
SIGN	IN	Word	Không được sử dụng bởi S7 – 1200; mọi giá trị đều được chấp nhận.
RET_VAL	OUT	Int	Mã điều kiện thực thi

# VÍ DỤ NGẮT TRÌ HOÃN THỜI GIAN

- Code trong OB1 và trong hàm ngắt. Khi I0.0 từ 1 về 0, chương trình sẽ chạy vào ngắt và trễ 2s hàm ngắt cho Q0.1 lên mức 1.



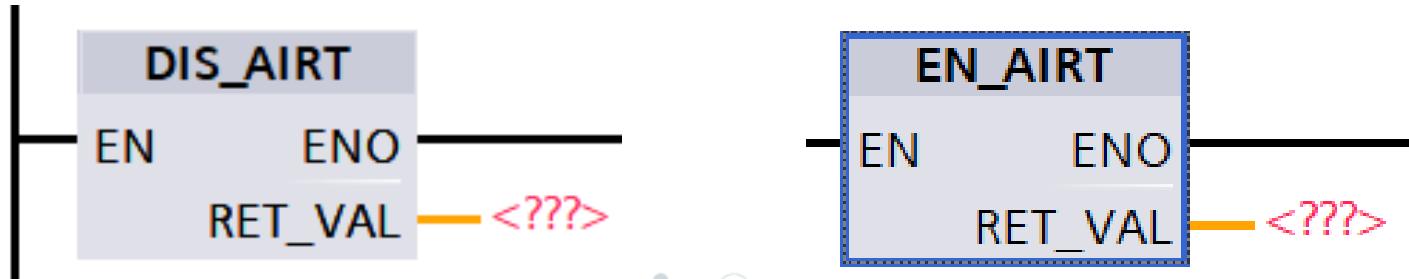
# CÁC LỆNH KHỞI ĐỘNG VÀ BỎ QUA NGẮT TRÌ HOÃN THỜI GIAN

## ❑ Các thông số CAN\_DINT

Thông số	Kiểu thông số	Kiểu dữ liệu	Miêu tả
OB_NR	IN	Int	Định danh OB ngắt trì hoãn thời gian. Ta có thể sử dụng một số hiệu OB hay tên gọi ký hiệu.
RET_VAL	OUT	Int	Mã điều kiện thực thi

# CÁC LỆNH VÔ HIỆU VÀ CHO PHÉP NGẮT CẢNH BÁO

- ❑ Sử dụng các lệnh DIS\_AIRT và EN\_AIRT để vô hiệu và cho phép sự xử lý ngắt cảnh báo.
- ❑ Lệnh DIS\_AIRT trì hoãn việc xử lý của các sự kiện ngắt mới.
- ❑ Có thể thực thi lệnh DIS\_AIRT nhiều lần trong một OB.
- ❑ Lệnh EN\_AIRT cho phép việc xử lý các sự kiện ngắt mà ta đã vô hiệu trước đó bằng lệnh DIS\_AIRT.
- ❑ Mỗi sự thực thi DIS\_AIRT phải được hủy bỏ bằng một sự thực thi EN\_AIRT.



## CÁC LỆNH VÔ HIỆU VÀ CHO PHÉP NGẮT CẢNH BÁO

- ❑ Các sự thực thi EN\_AIRT phải xuất hiện bên trong một OB giống nhau, hay bất kỳ FC hay FB nào được gọi từ một OB giống nhau, trước khi các ngắt được cho phép trở lại đối với OB này.
- ❑ Thông số RET\_VAL chỉ ra số lần mà việc xử lý ngắt đã bị vô hiệu, cũng là số lượng các thực thi DIS\_AIRT được xếp hàng. Việc xử lý ngắt chỉ được cho phép trở lại khi thông số RET\_VAL = 0.

Thông số	Kiểu thông số	Kiểu dữ liệu	Miêu tả
RET_VAL	OUT	Int	Số lượng các trì hoãn = số lượng các thực thi DIS_AIRT được xếp hàng.

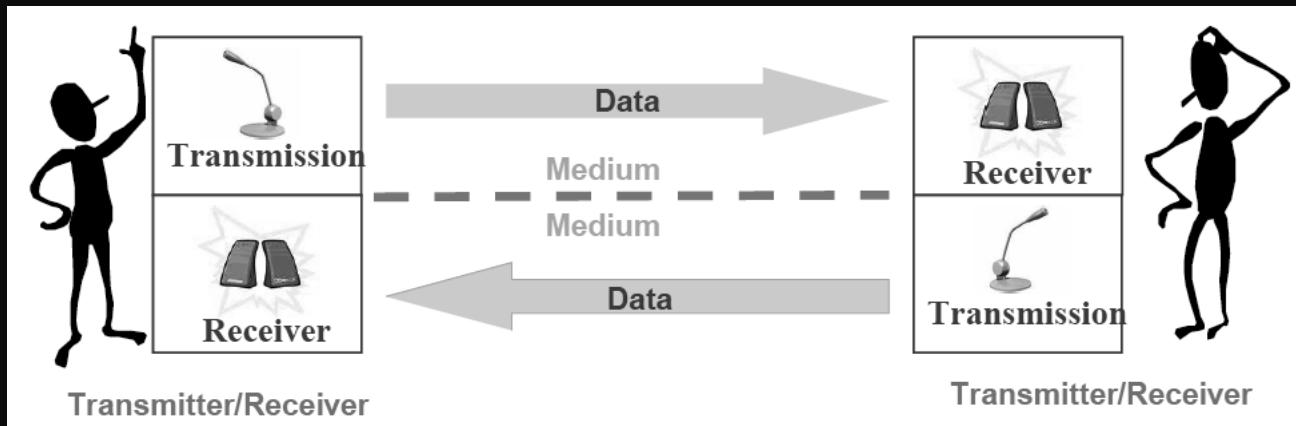
# Mạng truyền thông công nghiệp

## Nội dung

1. Các khái niệm cơ bản
2. Cấu trúc mạng
3. Kiến trúc giao thức
4. Truy nhập bus
5. Mã hóa và bảo toàn dữ liệu
6. Truyền dẫn tín hiệu
7. Các thành phần mạng
8. Các hệ thống bus tiêu biểu
9. Modbus
10. CAN
11. DeviceNet
12. PROFIBUS
13. Ethernet
14. ...

# Các khái niệm cơ bản

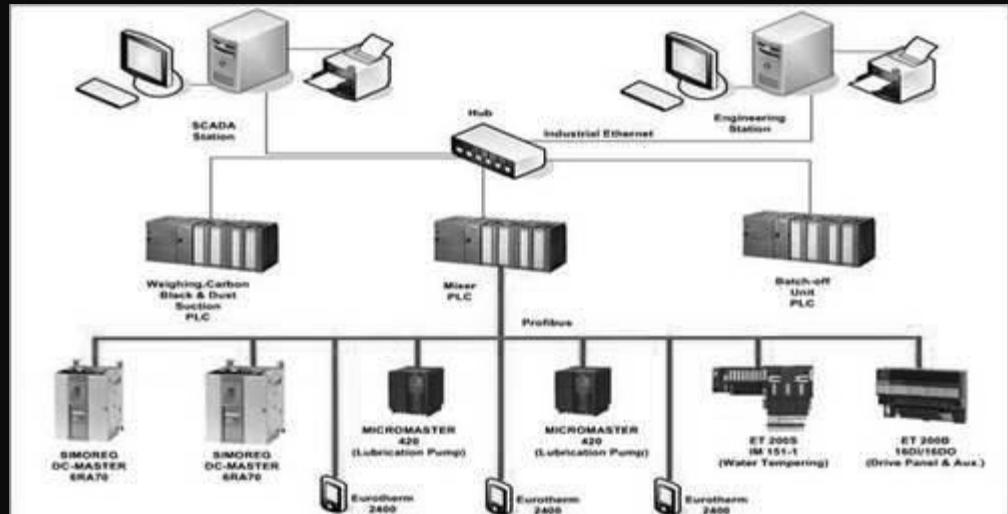
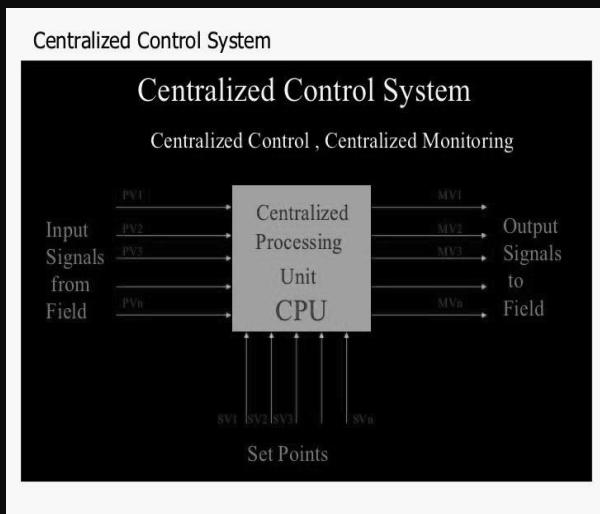
- Truyền thông là khái niệm dùng để chỉ sự trao đổi thông tin giữa các đối tác (communications partner) với nhau



- Mạng truyền thông công nghiệp (TTCN) chỉ các hệ thống mạng truyền thông số, truyền bit nối tiếp, sử dụng để ghép nối các thiết bị công nghiệp.
- Mạng TTCN yêu cầu độ tin cậy, tính năng thời gian thực và khả năng tương thích với môi trường công nghiệp

# Các khái niệm cơ bản

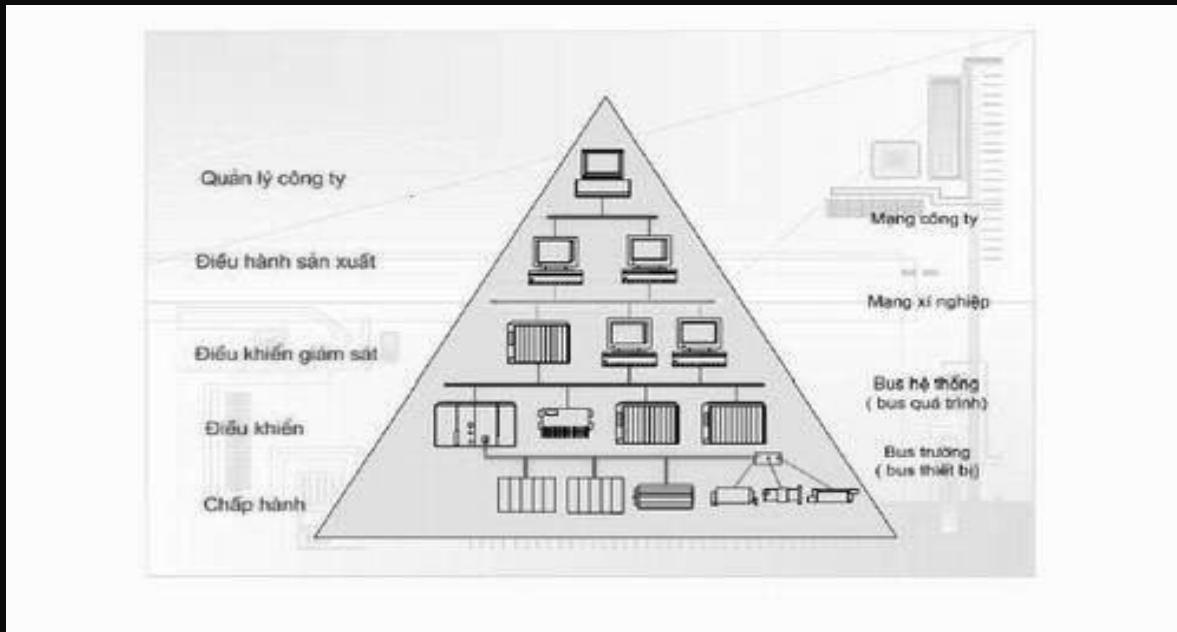
- Mạng TTCN cho phép kết nối bộ điều khiển với các thiết bị chấp hành, các cảm biến, và giữa các bộ điều khiển với nhau.



- Đơn giản hóa liên kết giữa các thiết bị, giảm dây nối, công lắp đặt và bảo trì hệ thống
- Nâng cao độ tin cậy và tính chính xác của thông tin
- Nâng cao độ linh hoạt, tính mở của hệ thống
- Cho phép phân tán hệ thống điều khiển

# Các khái niệm cơ bản

## Phân loại:



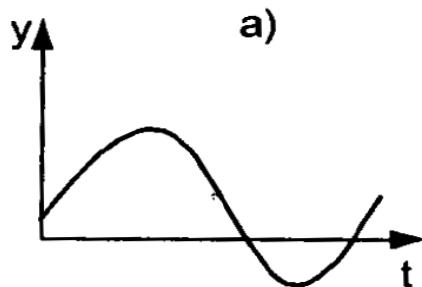
- Bus trường, bus thiết bị: kết nối các vào/ra phân tán, các cảm biến, hoặc cơ cấu chấp hành.
- Bus hệ thống, bus điều khiển: chủ yếu kết nối các bộ/hệ điều khiển quá trình
- Mạng xí nghiệp, mạng công ty: mạng LAN, internet, điều hành kế hoạch sản xuất, hoạt động của toàn bộ hệ thống

# Các khái niệm cơ bản

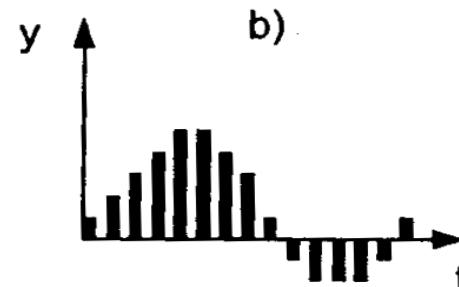
- Thông tin: mức độ nhận thức, sự hiểu biết về một vấn đề, một sự kiện, một hệ thống...
- Dữ liệu: lượng hóa thông tin, có thể dưới dạng mã nhị phân.
- Lượng thông tin: giá trị về sự hiểu biết một nguồn thông tin mang lại. Đơn vị dữ liệu có thể xử lý được: bit.
- Tín hiệu: diễn biến của một đại lượng vật lý chứa đựng tham số thông tin/dữ liệu, có thể truyền dẫn được (ánh sáng, âm thanh, hình ảnh, điện áp, dòng điện, tần số,...).
  - Tương tự: giá trị bất kỳ trong một khoảng nào đó
  - Rời rạc: thông tin chỉ có một số giá trị nhất định
  - Liên tục: tín hiệu có ý nghĩa tại bất kỳ thời điểm nào
  - Gián đoạn: tín hiệu chỉ có ý nghĩa tại những thời điểm nhất định

# Các khái niệm cơ bản

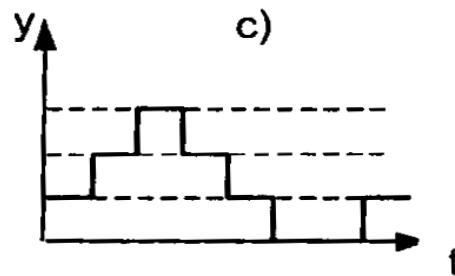
- Truyền tín hiệu



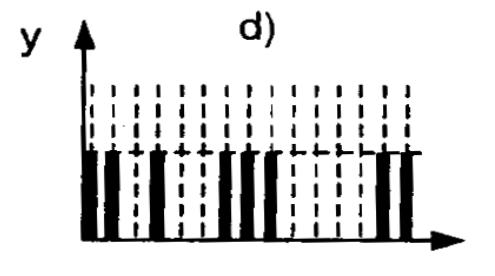
Dạng tín hiệu: **tương tự, liên tục**  
Tham số thông tin: **Biên độ**



Dạng tín hiệu: **tương tự, gián đoạn**  
Tham số thông tin: **Biên độ xung**



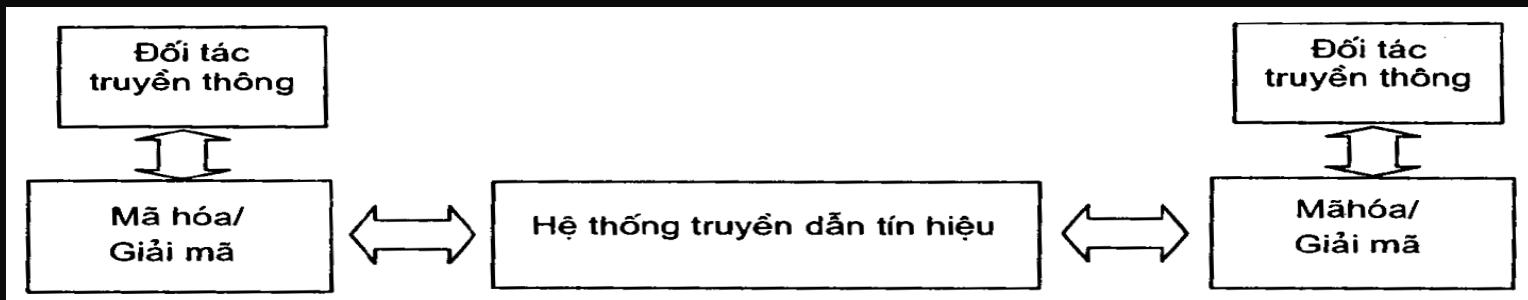
Dạng tín hiệu: **rời rạc, liên tục**  
Tham số thông tin: **Biên độ**



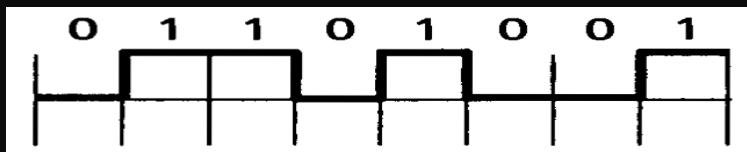
Dạng tín hiệu: **rời rạc (số), gián đoạn**  
Tham số thông tin: **Tần số xung**

# Các khái niệm cơ bản

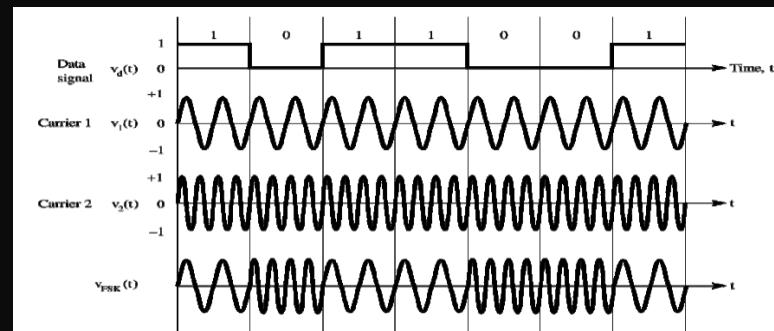
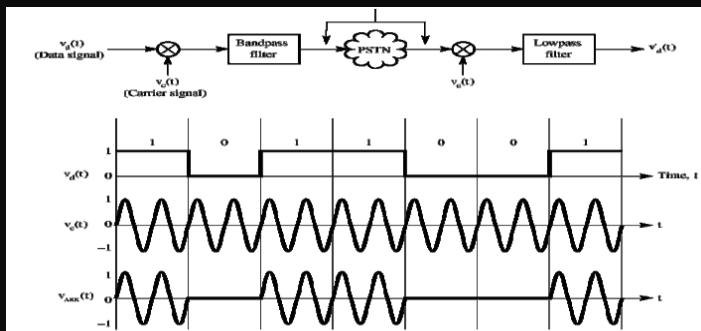
- Mã hóa/giải mã
  - Mã hóa nguồn, mã hóa đường truyền



- Ví dụ mã hóa bit



- Điều chế tín hiệu



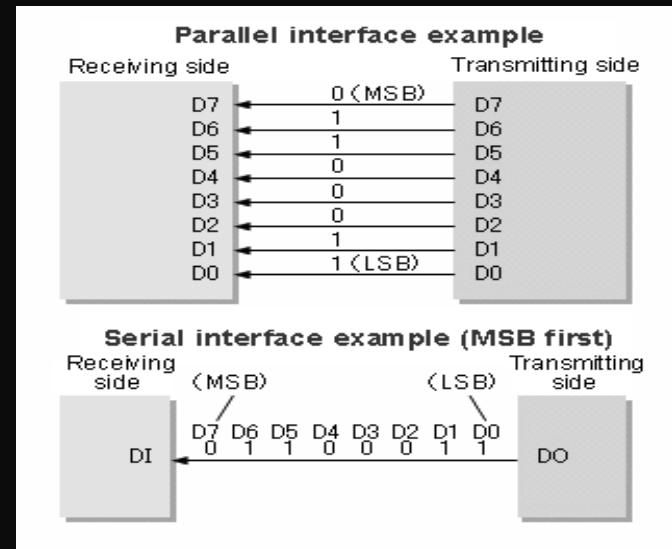
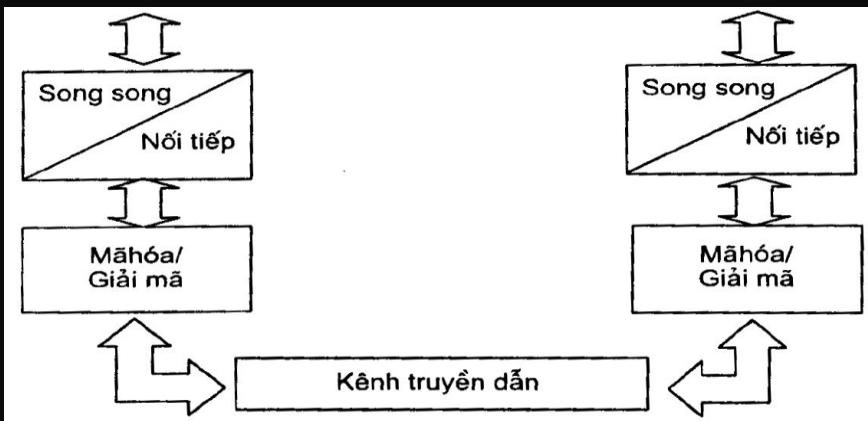
# Các khái niệm cơ bản

- Tốc độ truyền và tốc độ bit
  - Baud: số lần tín hiệu thay đổi thông tin/giây
  - Bps: số bit dữ liệu truyền đi/giây
- Tính năng thời gian thực
  - Độ nhanh nhạy: tốc độ truyền thông tin hữu ích đủ nhanh
  - Tính tiền định: Dự đoán được thời gian phản ứng tiêu biểu và thời gian phản ứng chậm nhất
  - Độ tin cậy, kịp thời: đảm bảo tổng thời gian vận chuyển dữ liệu tin cậy giữa các trạm trong khoảng cho phép
  - Tính bền vững: khả năng xử lý sự cố thích hợp

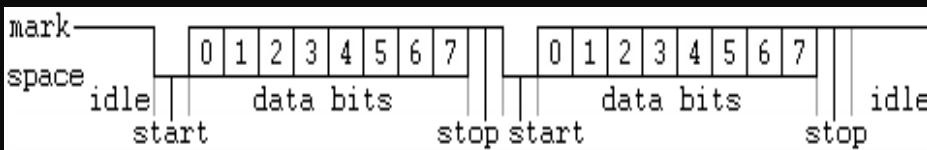
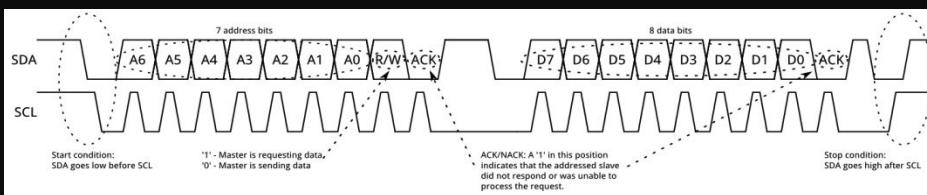
# Các khái niệm cơ bản

- Chế độ truyền tải

- Truyền bit song song/ nối tiếp



- Truyền đồng bộ và không đồng bộ



# Các khái niệm cơ bản

- Chế độ truyền tải

- Truyền một chiều và truyền hai chiều

- Truyền một chiều (simplex transmission)



- Truyền hai chiều gián đoạn (half duplex transmission)



- Truyền hai chiều toàn phần (full duplex transmission)

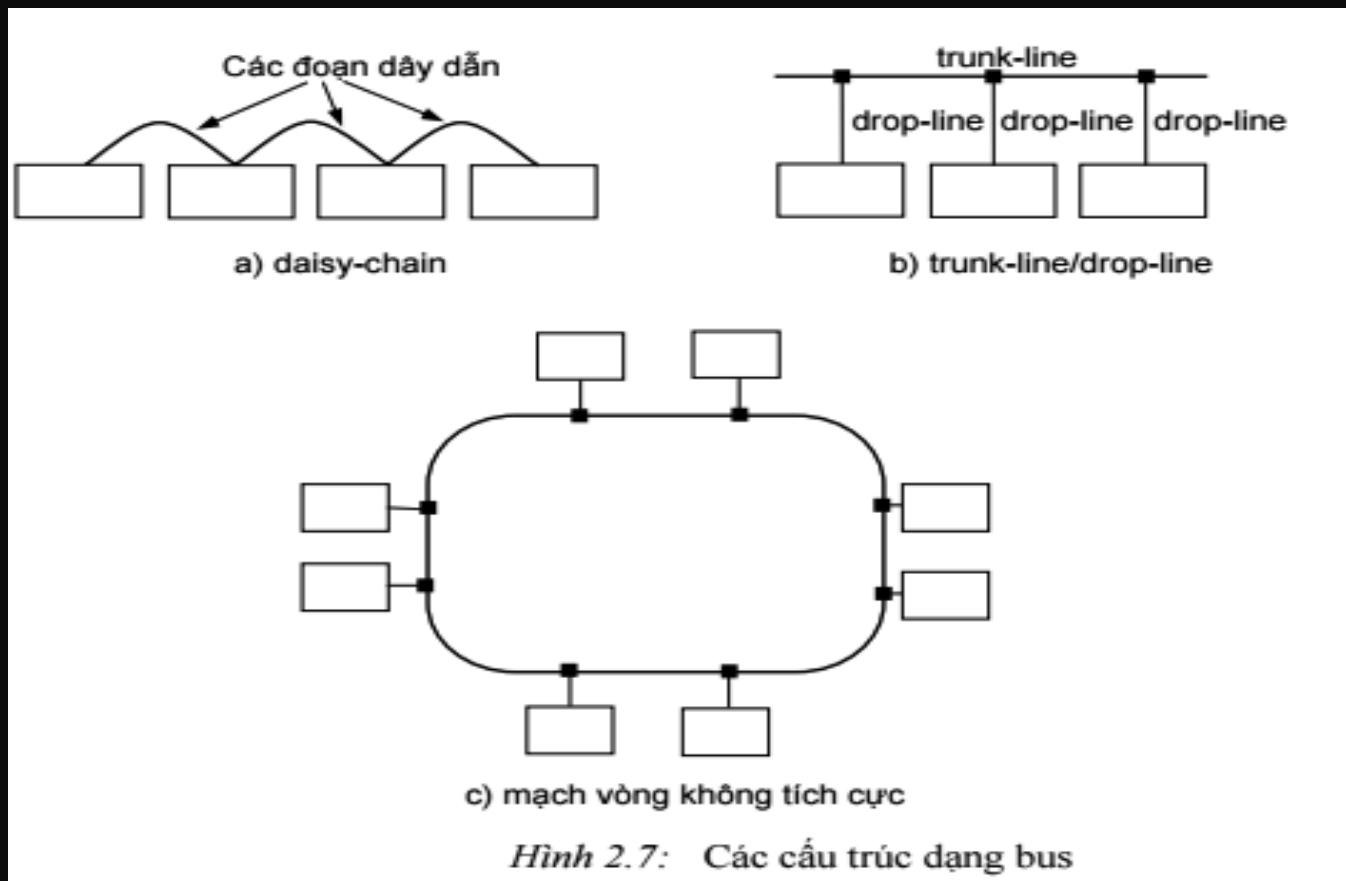


# Cấu trúc mạng

- Liên kết
  - Điểm-điểm (point to point)
  - Điểm-nhiều điểm (multi-drop): một trạm duy nhất phát trong khi các trạm còn lại thu thập thông tin cùng lúc
  - Nhiều điểm (multipoint): thông tin qua lại tự do
- Cấu trúc bus
  - Truyền đồng bộ và không đồng bộ

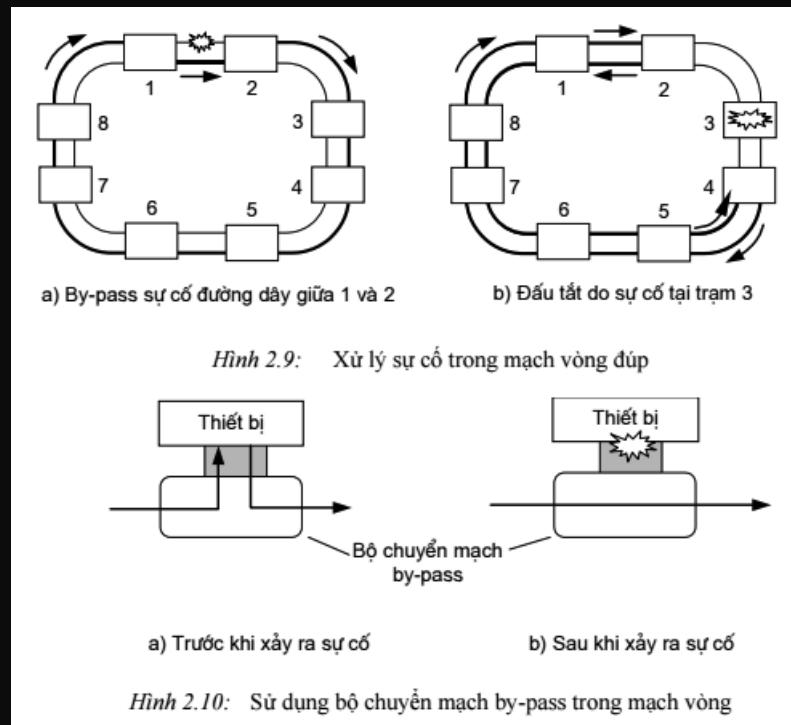
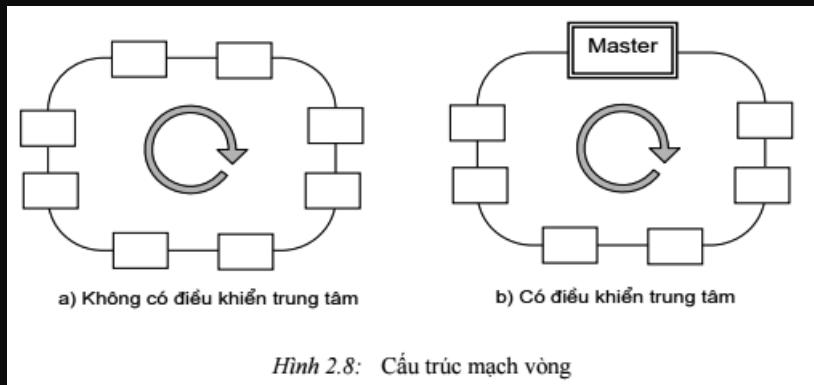
# Cấu trúc mạng

- Cấu trúc bus



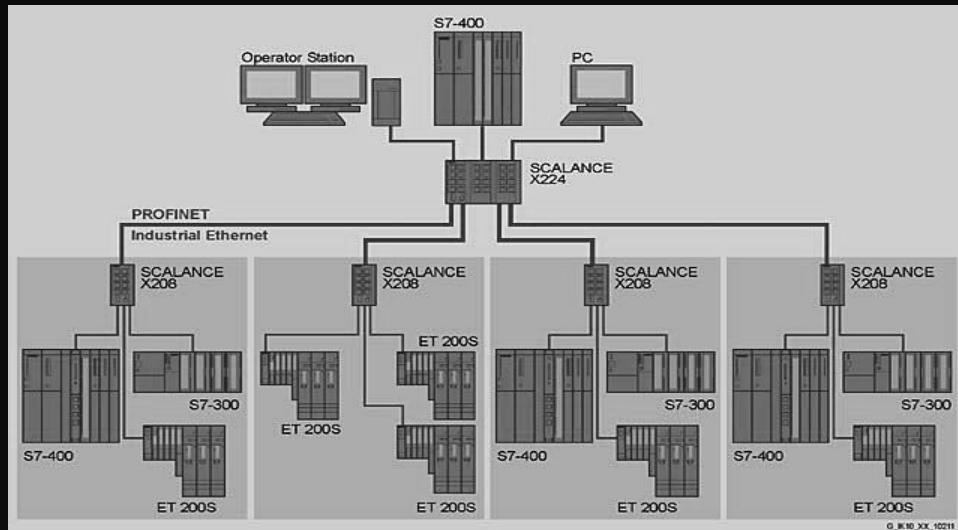
# Cấu trúc mạng

- Cấu trúc mạch vòng tích cực



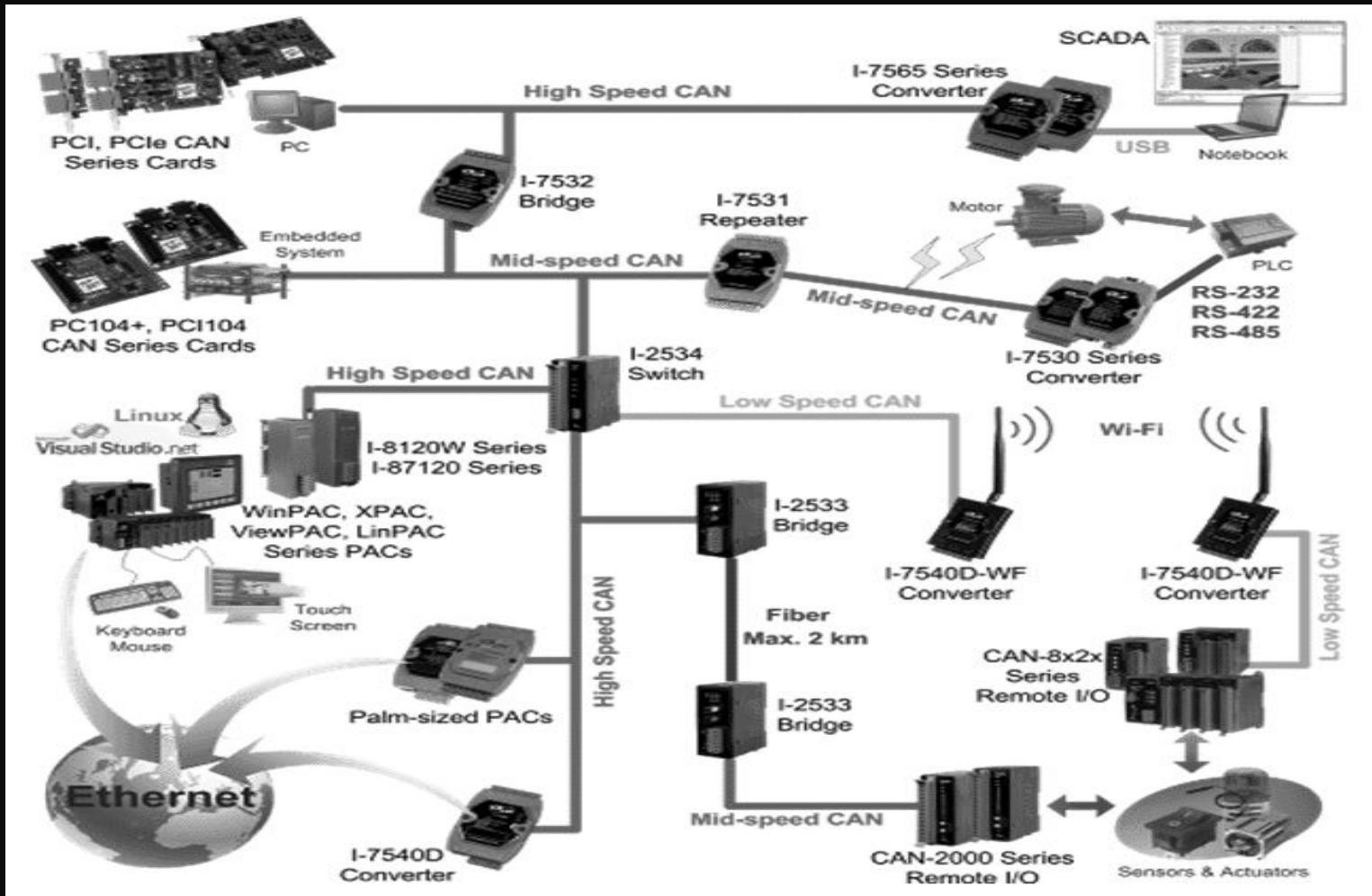
# Cấu trúc mạng

- Cấu trúc hình sao



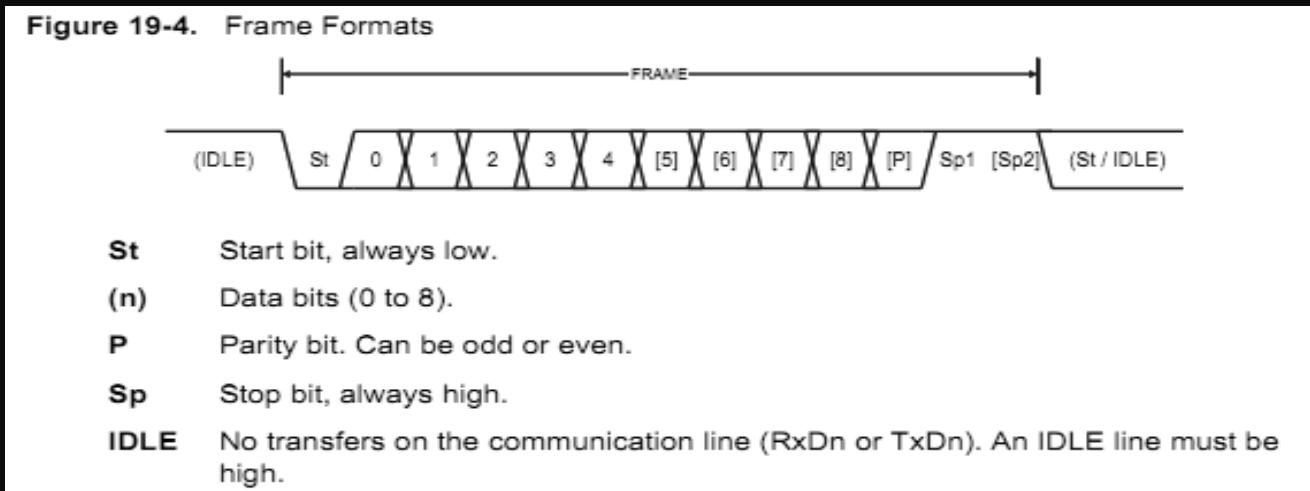
# Cấu trúc mạng

- Cấu trúc cây



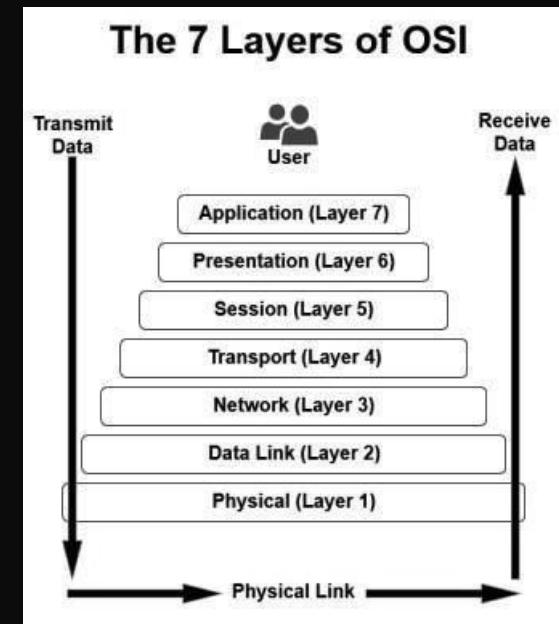
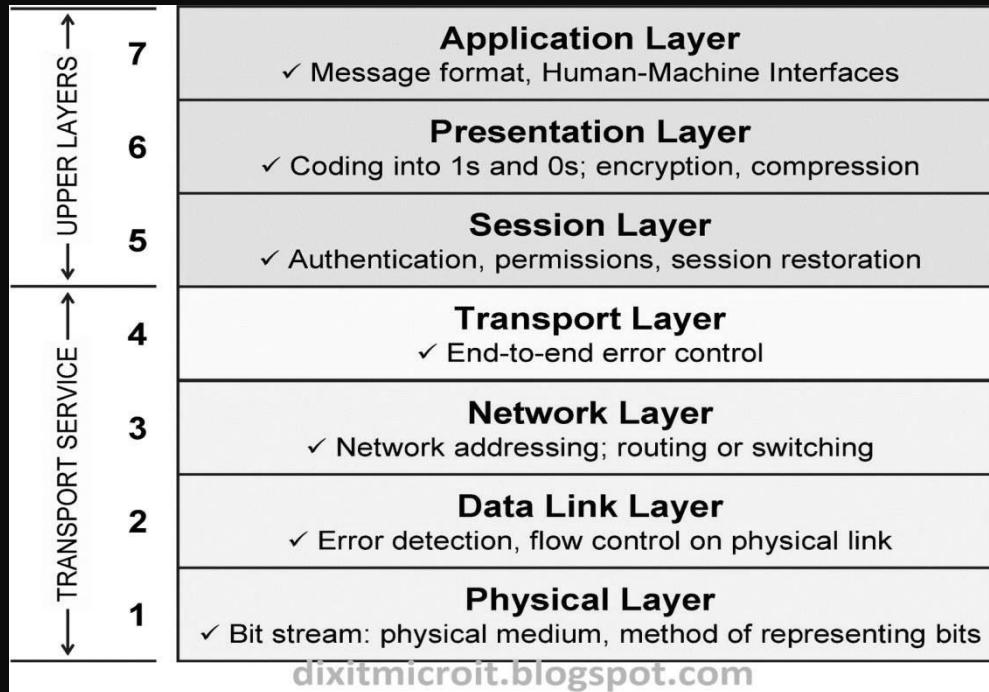
# Kiến trúc giao thức

- Giao thức: quy tắc, thủ tục cho việc giao tiếp.
  - Cú pháp: cấu trúc gói dữ liệu
  - Ý nghĩa: ý nghĩa cụ thể của từng thành phần trong gói dữ liệu
  - Định thời: trình tự giao tiếp, tốc độ truyền...
- UART (Universal Asynchronous Receiver/Transmitter)



# Kiến trúc giao thức

- Mô hình OSI (Open System Interconnect - Reference model)
  - ❑ Hệ thống thiết bị của các hãng công nghiệp rất đa dạng trong giao thức, chuẩn truyền dẫn, truy cập môi trường....
  - ❑ Năm 1984, tổ chức ISO (International Standardization Organization) đưa ra chuẩn ISO 7498, mô hình 7 lớp OSI



# Kiến trúc giao thức

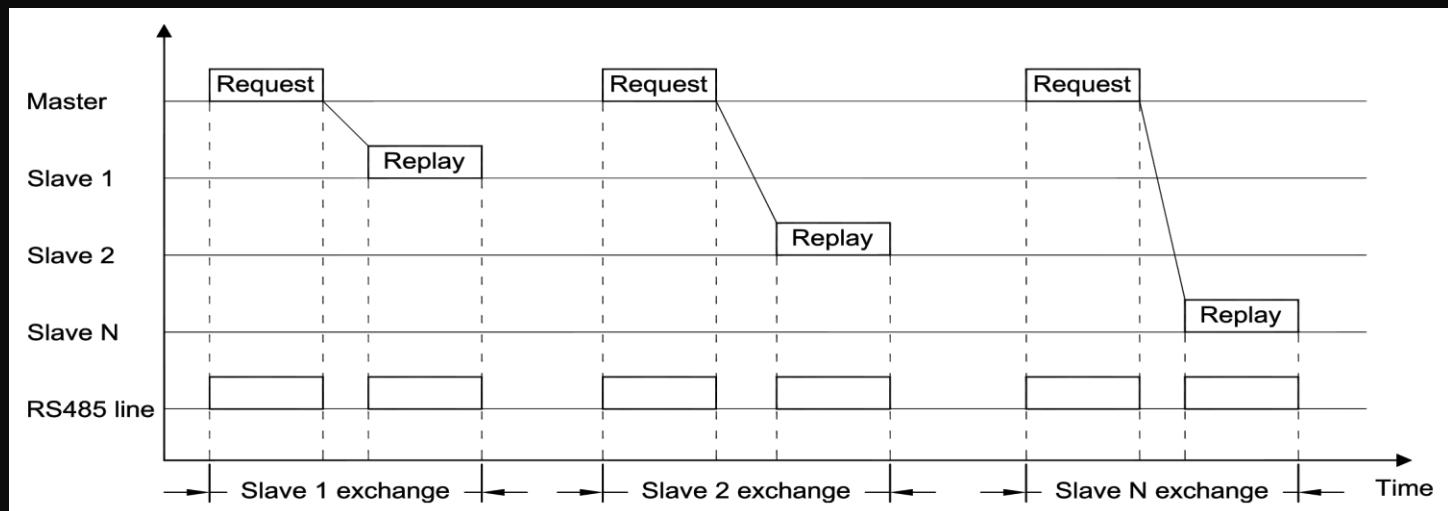
- Lớp ứng dụng
- Lớp biểu diễn dữ liệu
- Lớp kiểm soát nối
- Lớp vận chuyển
- Lớp mạng
- Lớp liên kết dữ liệu
- Lớp vật lý

# Truy nhập bus

- Các hệ thống có cấu trúc dạng bus
  - Linh hoạt, chi phí thấp
  - Thích hợp khoảng cách vừa và nhỏ
- Vấn đề với bus
  - Xung đột tín hiệu truyền đi
  - Phân chia quyền gửi thông tin trên đường truyền
- Tính năng kỹ thuật của các phương pháp truy nhập bus
  - Độ tin cậy
  - Tính năng thời gian thực
  - Hiệu suất sử dụng đường truyền

# Truy nhập bus

- Độ tin cậy
  - Bảo toàn dữ liệu
- Tính năng thời gian thực
  - Thời gian đáp ứng tối đa
  - Chu kỳ bus
  - Độ rung (jitter)
- Hiệu suất sử dụng đường truyền



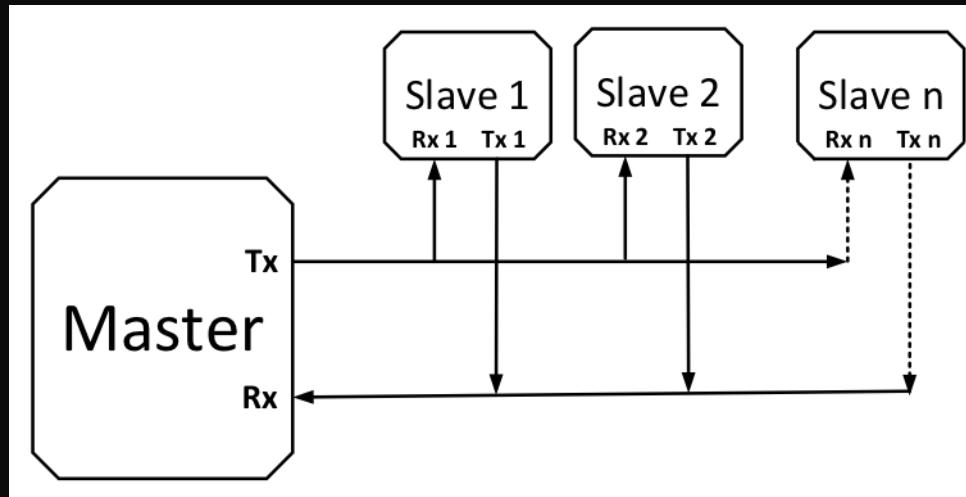
# Truy nhập bus

---

- Các phương pháp truy nhập bus cơ bản
  - Chủ/tớ (master/slave)
  - TDMA
  - Token Passing
  - CSMA/CD
  - CSMA/CA

# Truy nhập bus

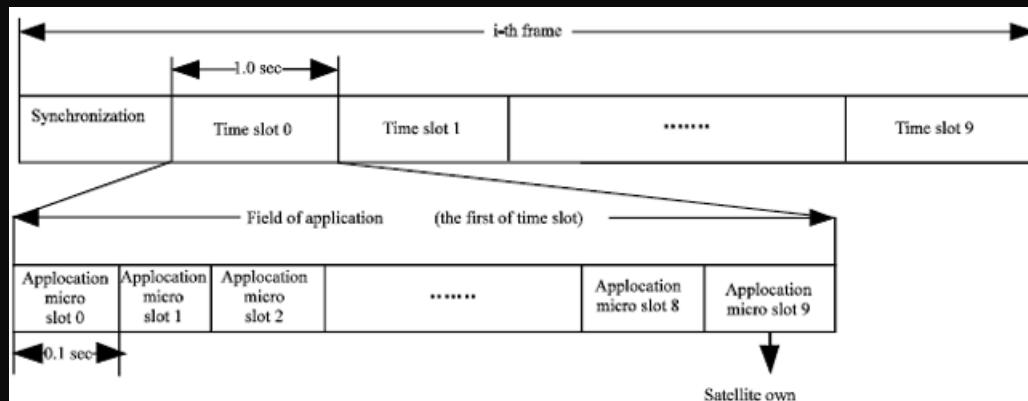
- Chủ/tớ (master/slave)



- Một trạm chủ, nhiều trạm tớ
- Trạm chủ kiểm soát toàn bộ truy nhập bus
- Có thể sử dụng hỏi tuần tự (polling)
- Độ tin cậy của hệ thống phụ thuộc hoàn toàn vào trạm chủ

# Truy nhập bus

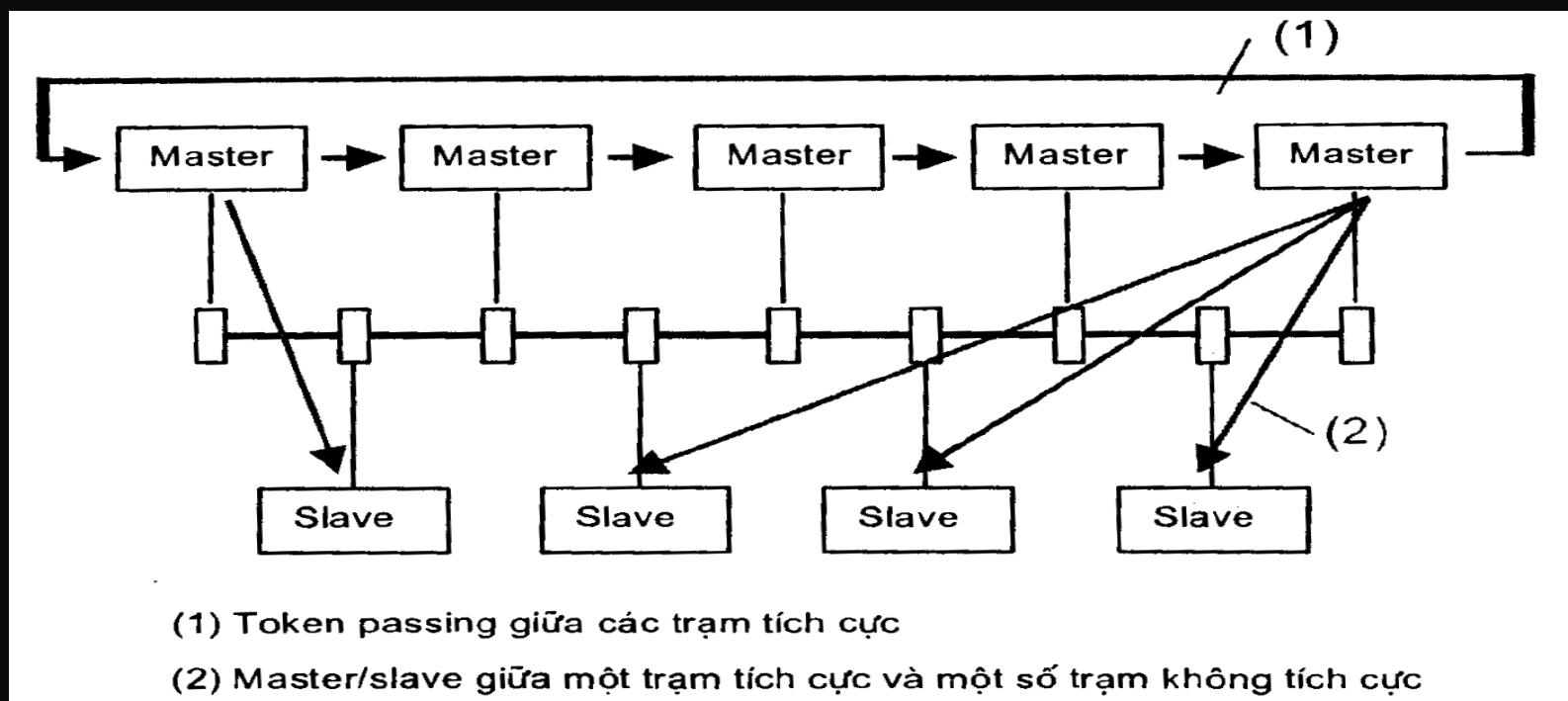
- TDMA (Time Division Multiple Access)
  - ❑ Mỗi trạm được phân một khoảng thời gian truy nhập bus xác định
  - ❑ Có thể có trạm chủ với vai trò kiểm soát lát thời gian



# Truy nhập bus

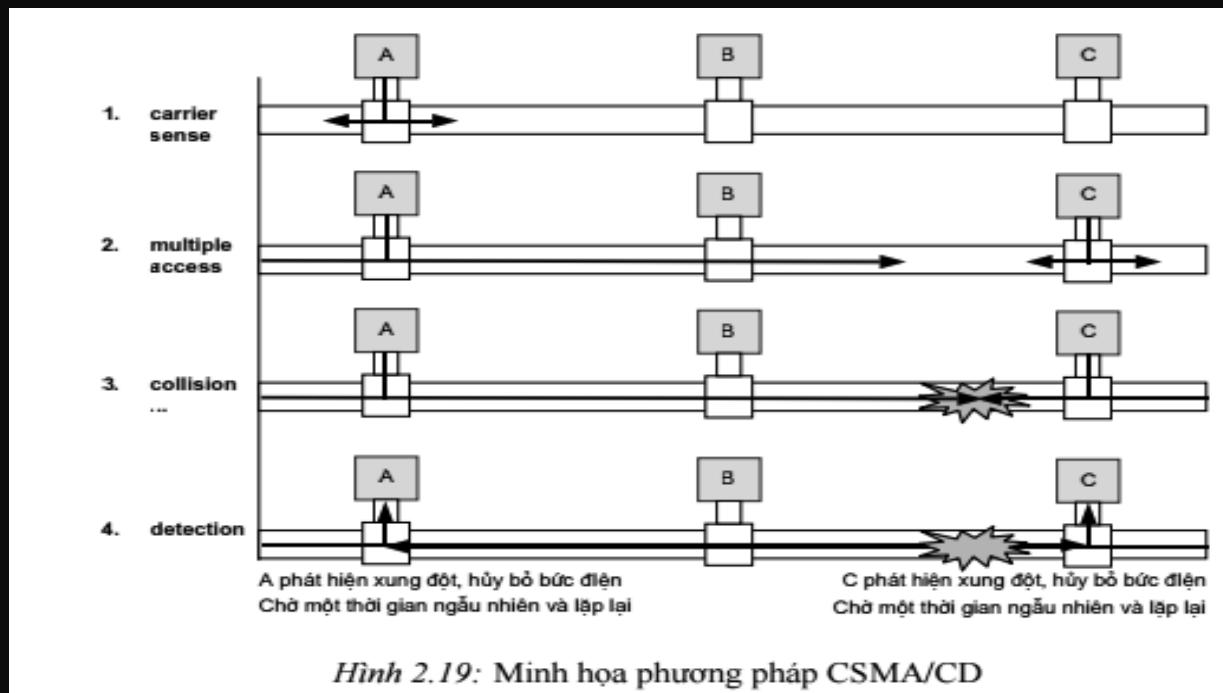
- Token Passing

- Bức điện đặc biệt: Token
- Trạm giữ Token được quyền truy nhập bus
- Chuyển Token sang trạm khác theo trình tự xác định
- Kết hợp Master/Slave thành Multi-Master



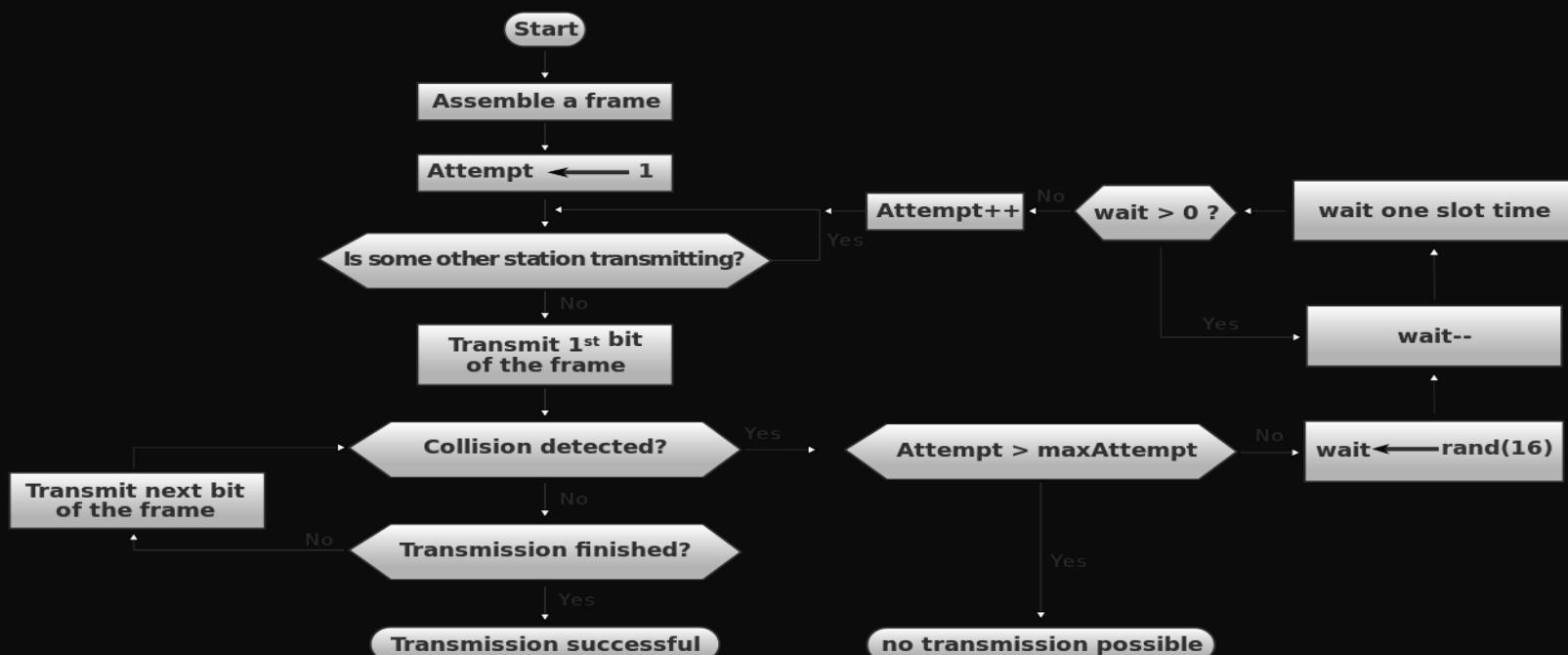
# Truy nhập bus

- CSMA/CD
  - Carrier Sense Multiple Access with Collision Detection
  - Mỗi trạm tự nghe đường dẫn, truyền tín hiệu khi đường dẫn rỗi
  - Nghe và so sánh tín hiệu trong lúc truyền, tránh xung đột
  - Khi xung đột, hủy bỏ bức điện, đợi một khoảng thời gian ngẫu nhiên, thử gửi lại.



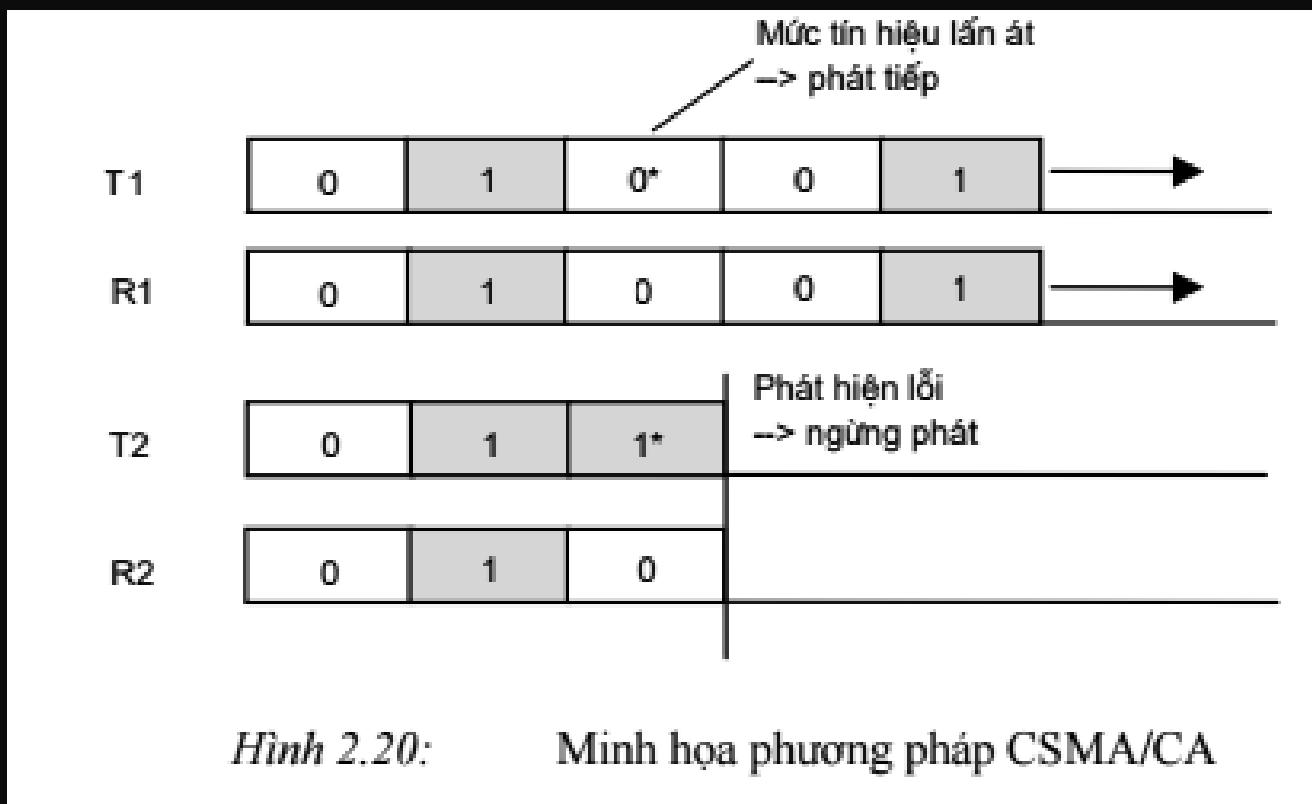
# Truy nhập bus

- CSMA/CD
  - Có điều kiện ràng buộc giữa tốc độ truyền, chiều dài bức điện và chiều dài dây dẫn
  - Thời gian gửi một bức điện lớn hơn hai lần thời gian lan truyền tín hiệu.
  - Hiệu suất đường truyền thấp, không xác định được tương đối chính xác thời gian phản ứng.



# Truy nhập bus

- CSMA/CA
  - Carrier Sense Multiple Access with Collision Avoidance
  - Mã hóa bit thích hợp, tín hiệu trội hoặc lặn

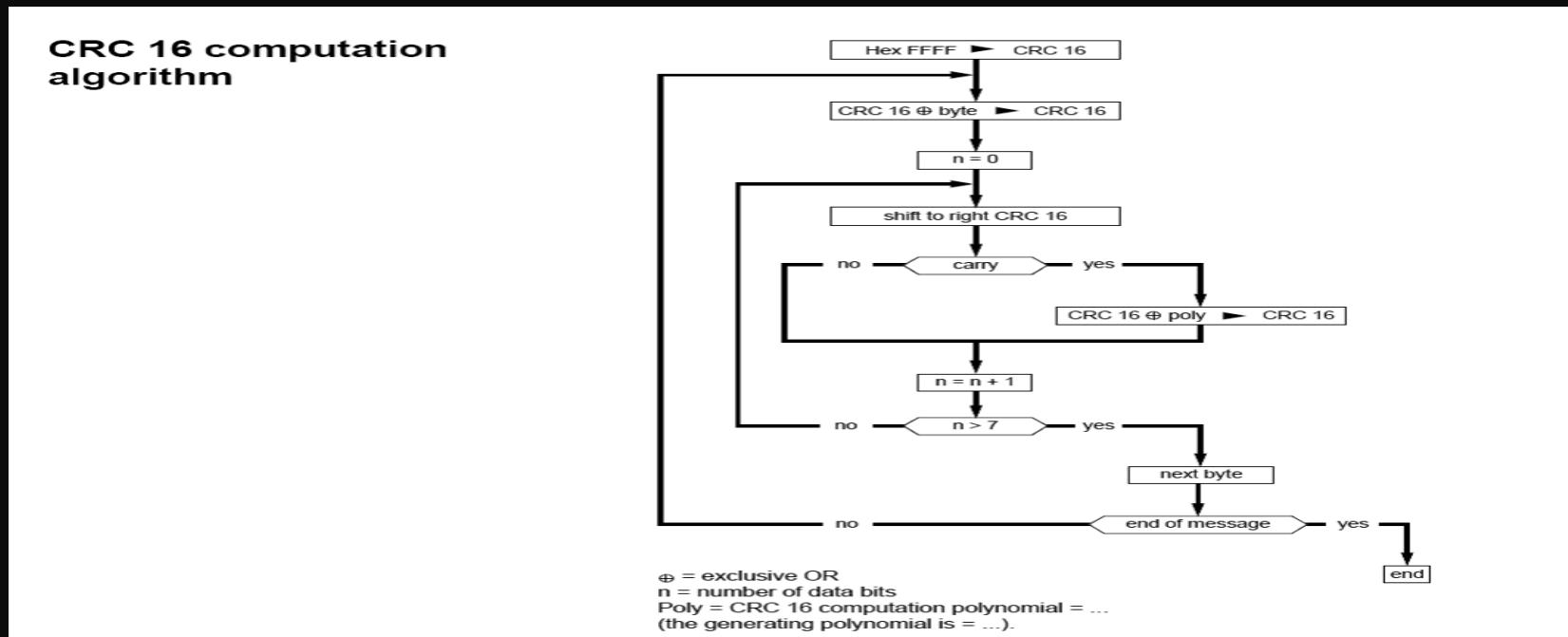


# Bảo toàn dữ liệu

- Xây dựng phần cứng chống lỗi
- Xử lý giao thức, phát hiện lỗi
  - Bit chẵn lẻ (Parity bit)
  - CRC (Cyclic Redundancy Check)
  - Nhồi bit (Bit stuffing)

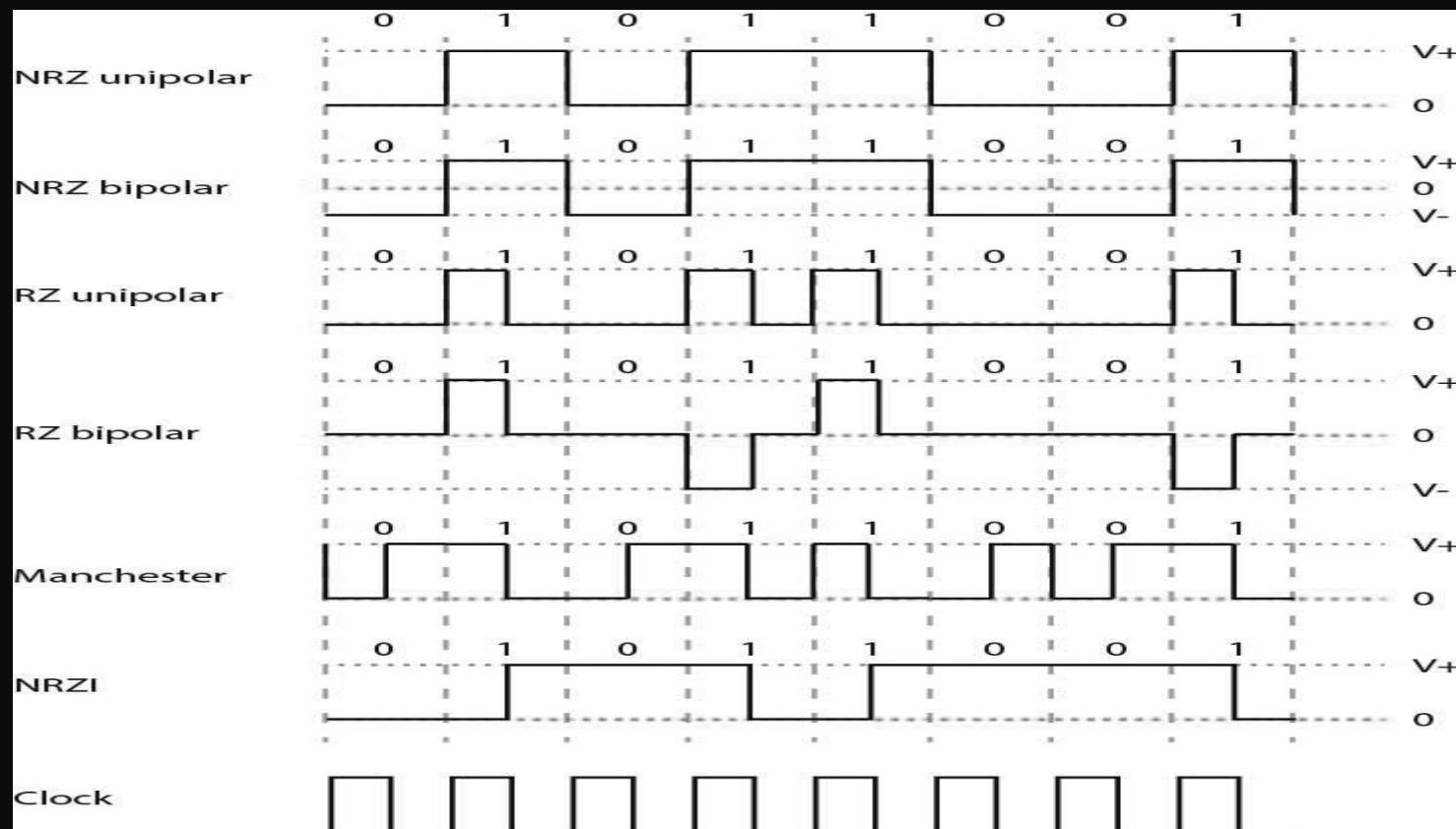
Parity bit examples		
sequence of seven bits	with eighth even parity bit:	with eighth odd parity bit:
0100010	01000100	01000101
1000000	10000001	10000000

ComputerHope.com



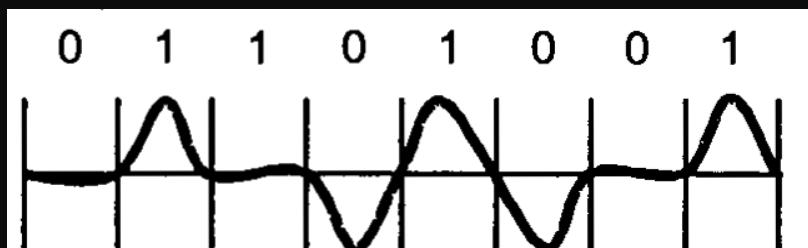
# Mã hóa bit

- NRZ, RZ
  - Non-Return to Zero, đơn giản, phô biến
  - Điều chế biên độ
  - Không hợp cho đồng bộ hóa, có dòng một chiều

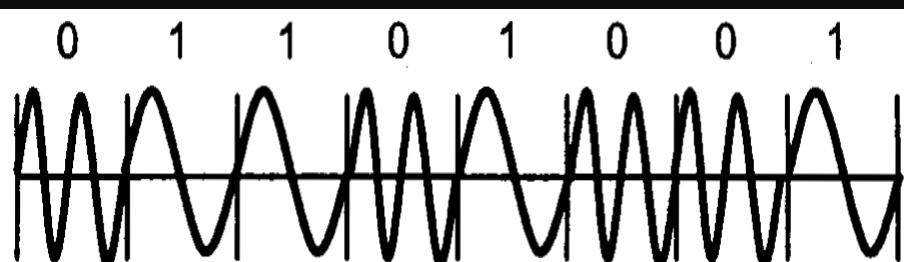


# Mã hóa bit

- Manchester
  - Điều chế pha xung
  - Có khả năng đồng tải nguồn
- AFP (Alternate Flanked Pulse)
  - Điều chế vị trí xung (xung xoay chiều)
- FSK (Frequency Shift Keying)
  - Điều chế tần số tín hiệu



AFP: Thay đổi giữa 0 và 1 được đánh dấu bằng một xung xoay chiều



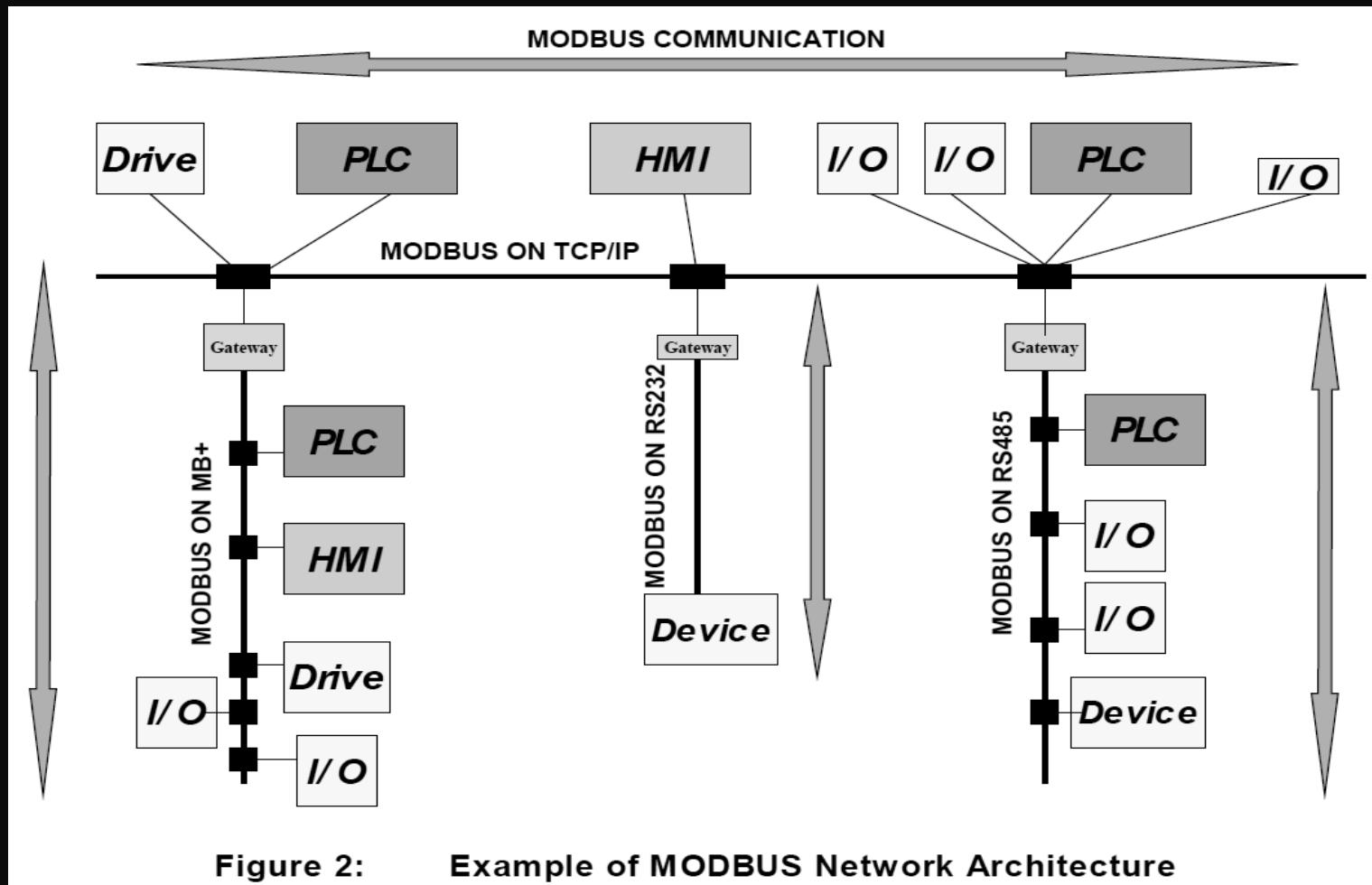
FSK: 0 và 1 ứng với các tần số khác nhau

# Modbus

- Lịch sử
  - Phát triển đầu tiên bởi Modicon (nay thuộc Schneider Electric)
  - Dựa trên các ứng dụng trong công nghiệp
  - Là một chuẩn mở, miễn phí bản quyền
  - Tiếp tục được phát triển và quản lý bởi Modbus Organization từ tháng 4 - 2004
- Ứng dụng
  - Có thể được sử dụng để kết nối máy tính giám sát với các thiết bị vào ra từ xa (remote terminal unit)
  - Được sử dụng trong các hệ thống giám sát và thu thập dữ liệu điều khiển SCADA (supervisory control and data acquisition)

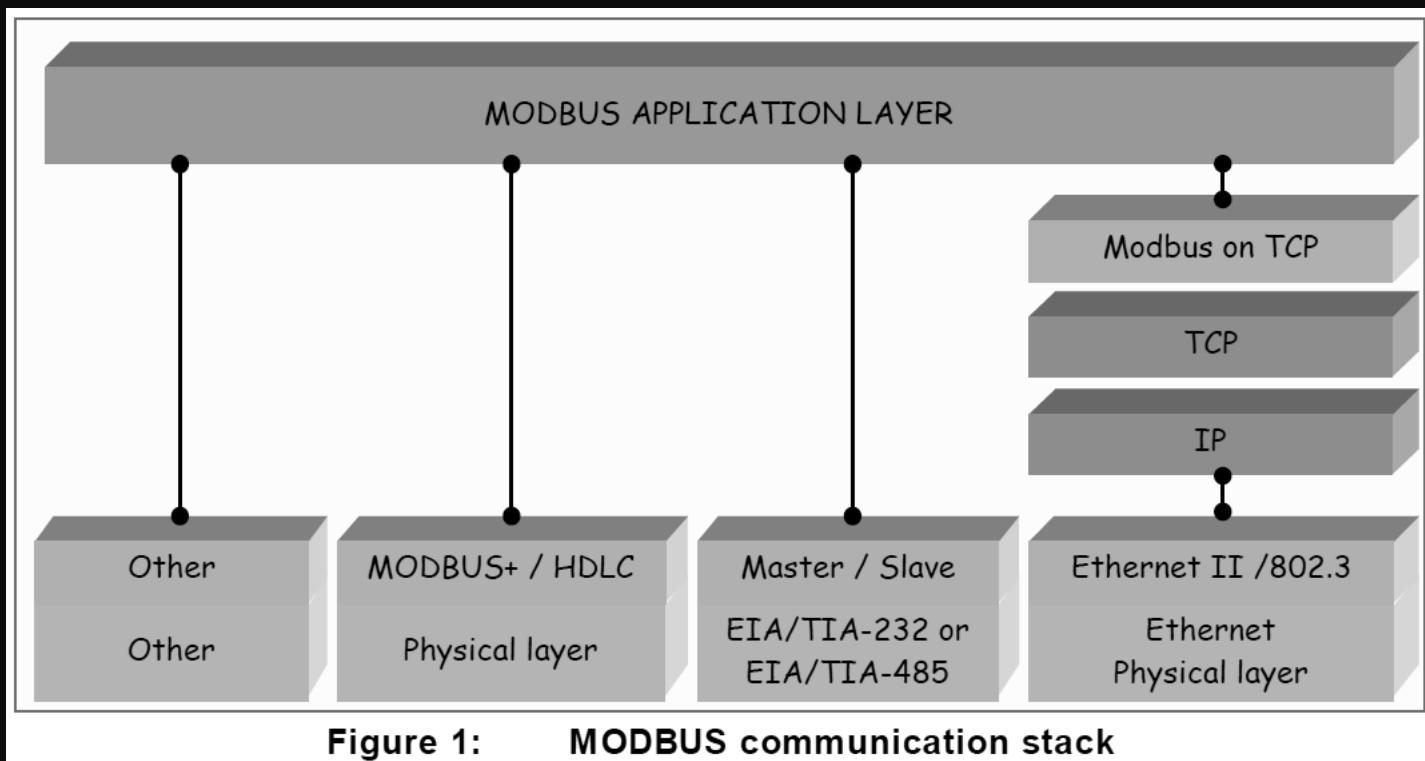
# Modbus

- Giới thiệu
  - Được sử dụng để kết nối nhiều loại thiết bị công nghiệp



# Modbus

- Giới thiệu
  - Modbus là một chuẩn giao thức và dịch vụ thuộc lớp ứng dụng

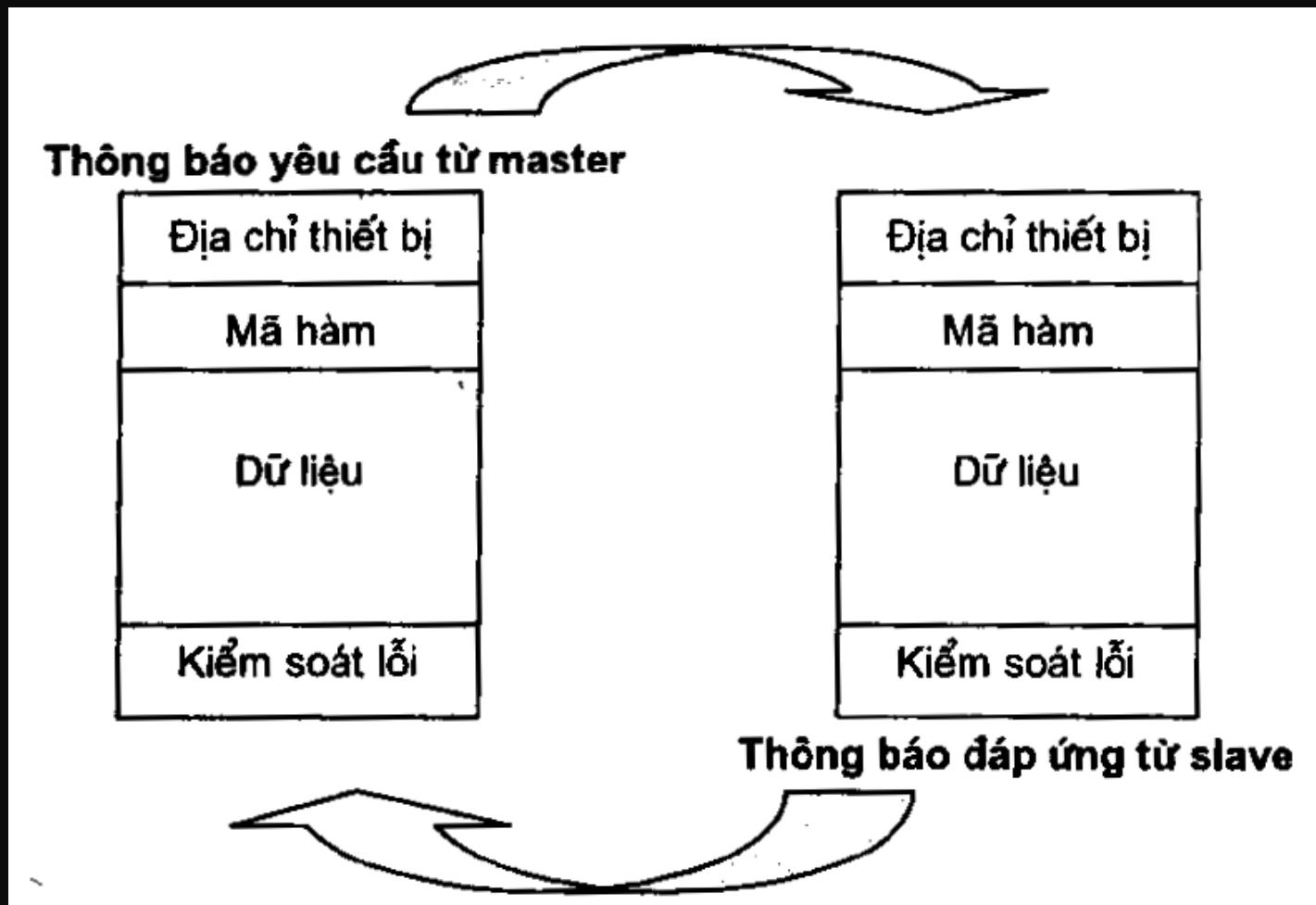


# Modbus – Cơ chế giao tiếp

- Modbus chuẩn
  - Gặp trên nhiều bộ điều khiển của Modicon, sử dụng RS232
  - Cơ chế giao tiếp: Master/Slave
  - Một thông báo yêu cầu có chứa địa chỉ trạm nhận, mã hàm dịch vụ, dữ liệu đi kèm và thông tin kiểm lỗi
- Chu trình yêu cầu – đáp ứng:  
thông báo yêu cầu bao gồm
  - Địa chỉ trạm nhận yêu cầu (0-247), 0 là địa chỉ gửi đồng loạt
  - Mã hàm yêu cầu trạm nhận thực hiện
  - Dữ liệu đi kèm mã hàm
  - Thông tin kiểm lỗi

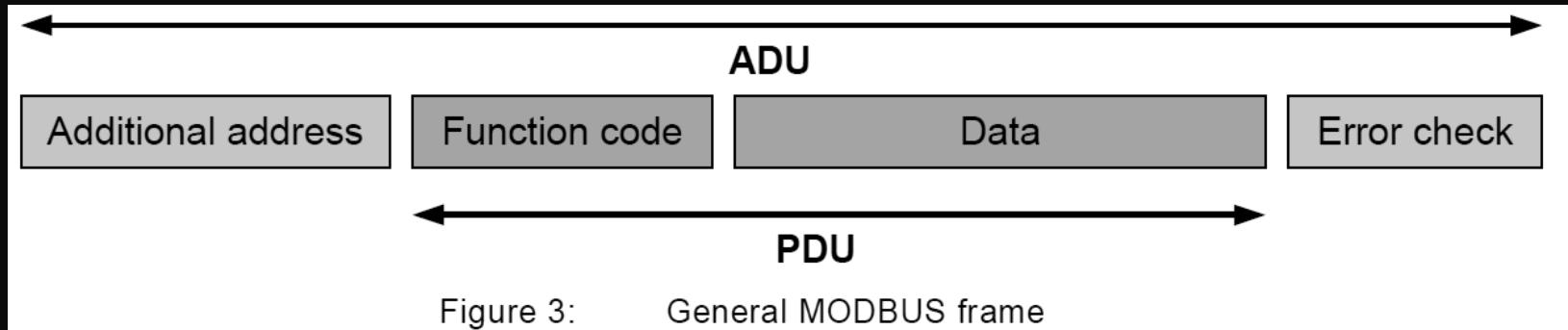
# Modbus – Cơ chế giao tiếp

- Chu trình yêu cầu – đáp ứng



# Modbus – Cấu trúc bức điện

- Cấu trúc bức điện

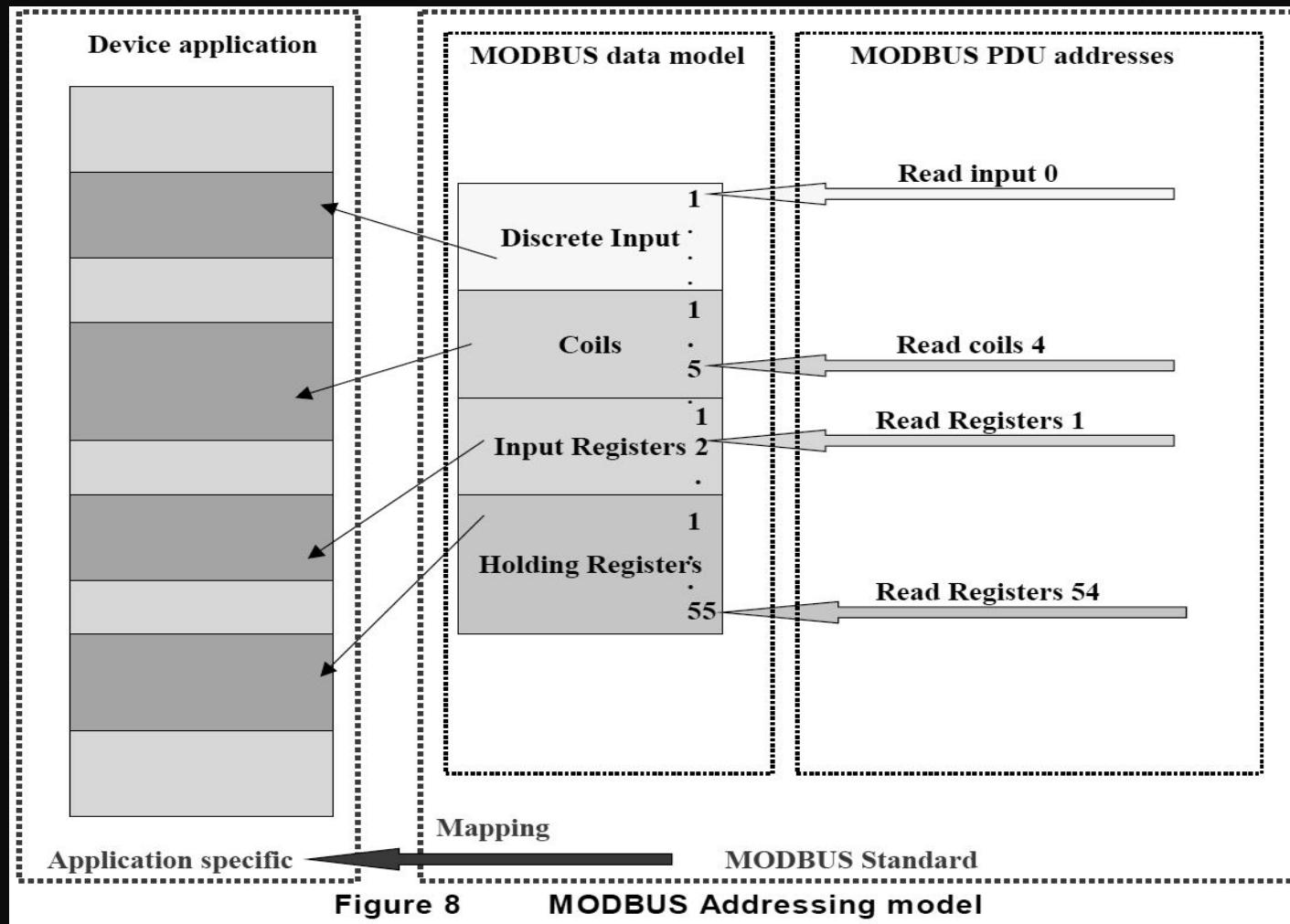


- Các đối tượng dữ liệu

Primary tables	Object type	Type of	Comments
Discretes Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system
Holding Registers	16-bit word	Read-Write	This type of data can be alterable by an application program.

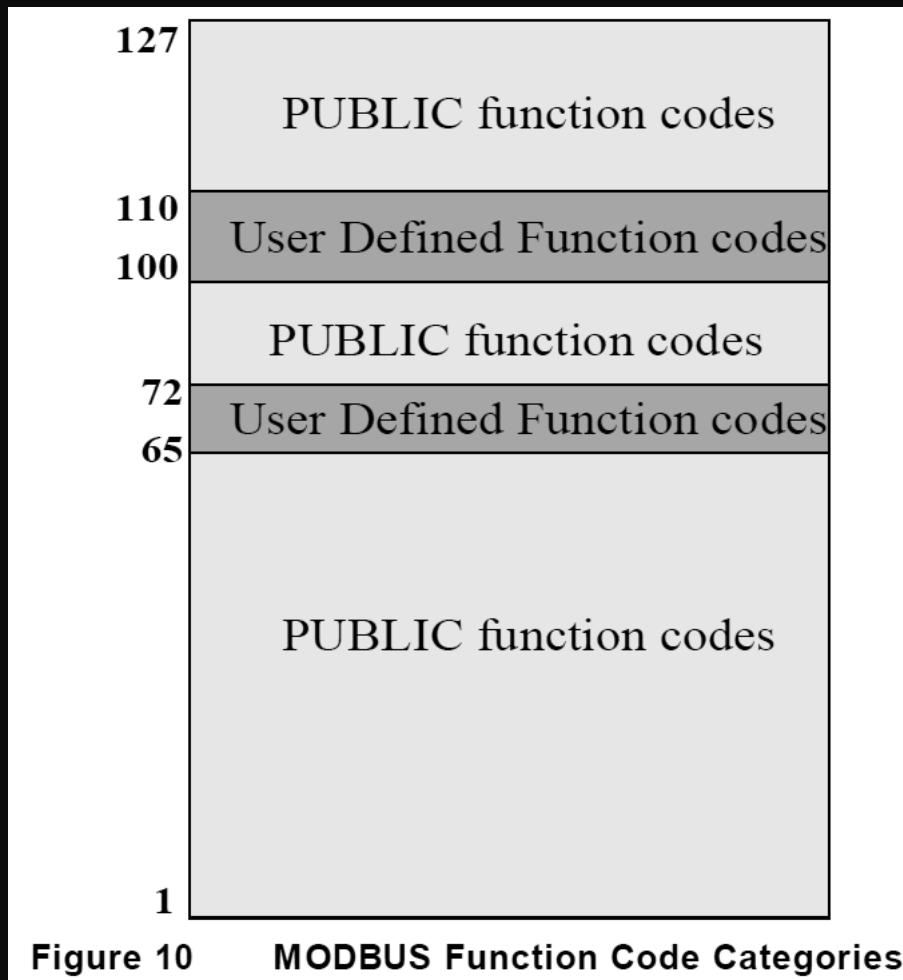
# Modbus – Cấu trúc bức điện

- Mô hình địa chỉ



# Modbus – Cấu trúc bức điện

- Mã hàm



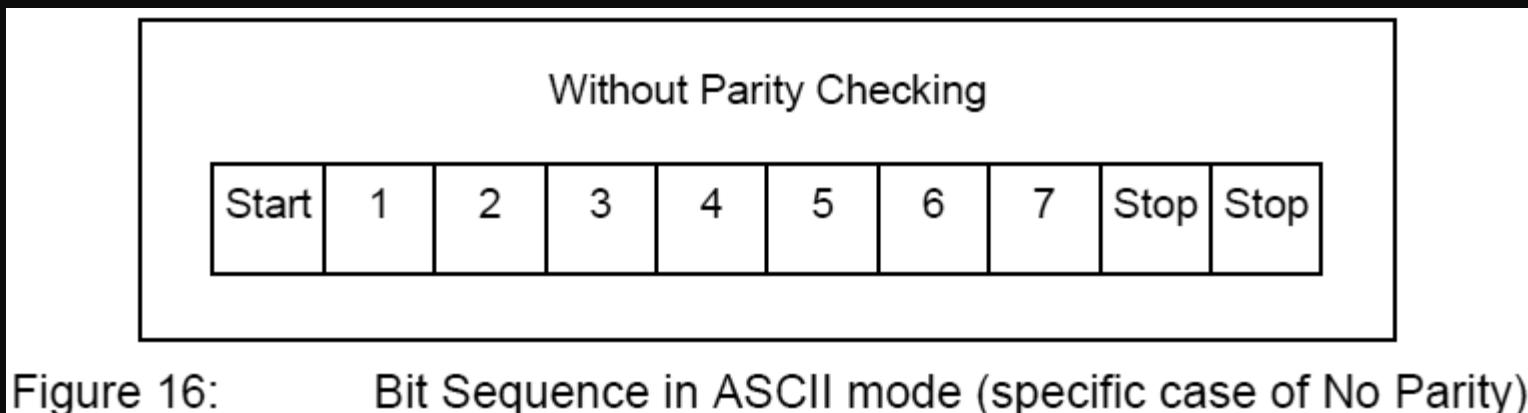
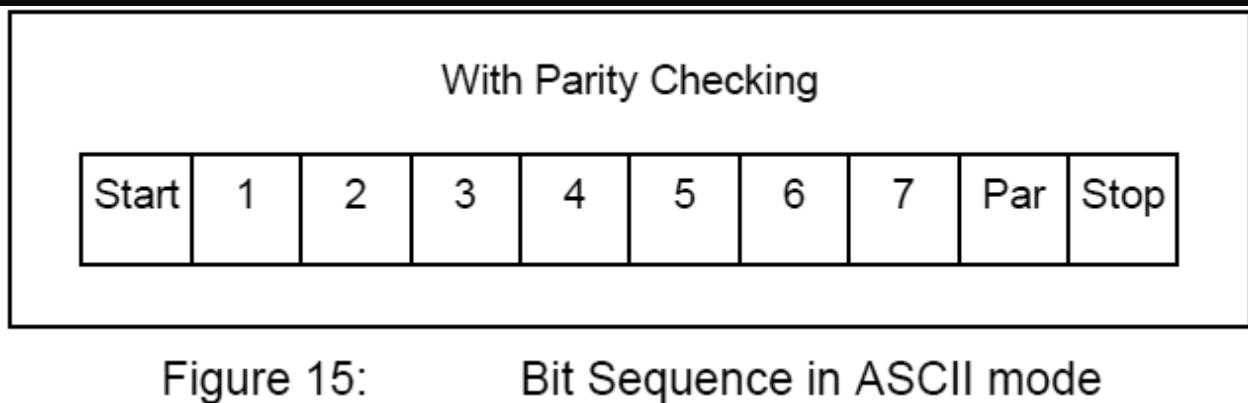
# Modbus – Cấu trúc bức điện

- Mã hàm

				Function Codes				
				code	Sub code	(hex)	Section	
Data Access	Bit access	Physical Discrete Inputs	Read Discrete Inputs	02		02	6.2	
		Internal Bits Or Physical coils	Read Coils	01		01	6.1	
			Write Single Coil	05		05	6.5	
			Write Multiple Coils	15		0F	6.11	
	16 bits access	Physical Input Registers	Read Input Register	04		04	6.4	
		Internal Registers Or Physical Output Registers	Read Holding Registers	03		03	6.3	
			Write Single Register	06		06	6.6	
			Write Multiple Registers	16		10	6.12	
			Read/Write Multiple Registers	23		17	6.17	
			Mask Write Register	22		16	6.16	
	File record access		Read FIFO queue	24		18	6.18	
			Read File record	20		14	6.14	
	Diagnostics		Write File record	21		15	6.15	
			Read Exception status	07		07	6.7	
			Diagnostic	08	00-18,20	08	6.8	
			Get Com event counter	11		OB	6.9	
			Get Com Event Log	12		0C	6.10	
			Report Server ID	17		11	6.13	
Other	Read device Identification			43	14	2B	6.21	
	Encapsulated Interface Transport			43	13,14	2B	6.19	
CANopen General Reference				43	13	2B	6.20	

# Modbus – ASCII

- Chế độ ASCII
  - Mỗi byte được truyền đi bằng hai ký tự ASCII 7 bit, mỗi ký tự biểu diễn một chữ số hex.



# Modbus – ASCII

- Khung ASCII

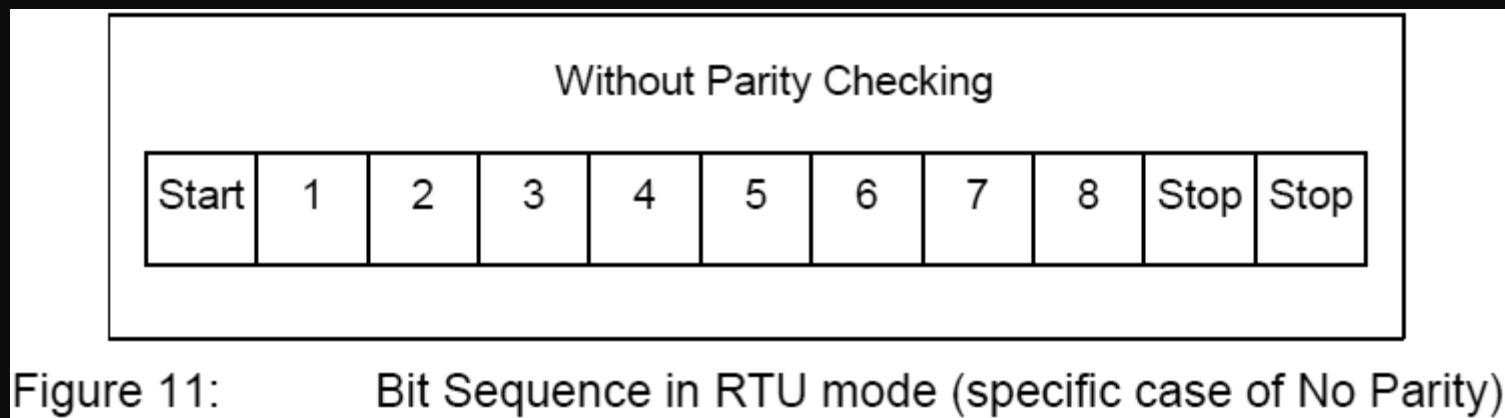
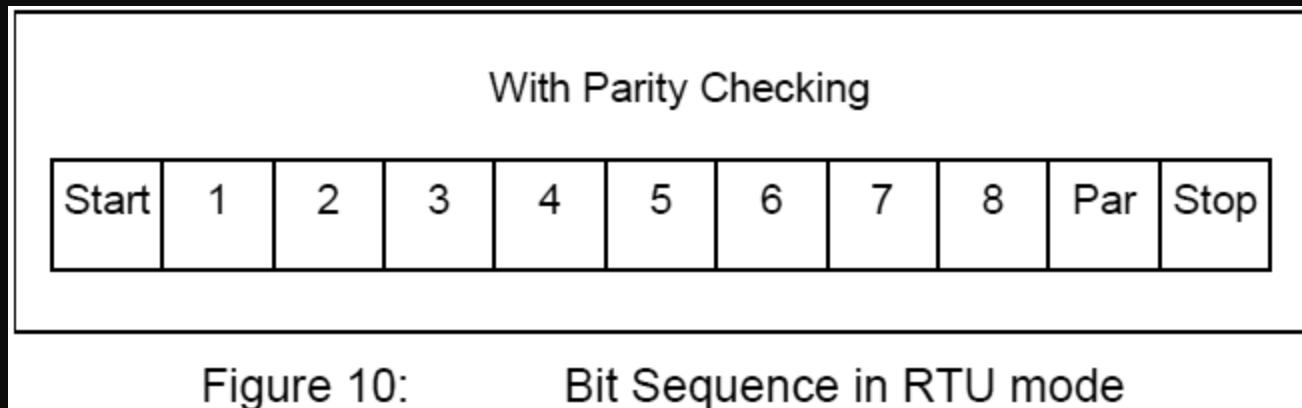
Start	Address	Function	Data	LRC	End
1 char :	2 chars	2 chars	0 up to 2x252 char(s)	2 chars	2 chars CR,LF

Figure 17:      ASCII Message Frame

- Khởi đầu khung bằng dấu hai chấm ( : )
- Kết thúc khung bằng hai ký tự quay lại (CR) và xuống dòng (LF)
- Thời gian tối đa giữa hai ký tự trong một thông báo là 1 giây.
- Thông tin kiểm tra lỗi LRC (Longitudinal Redundancy Check): cộng đại số toàn bộ các byte của dãy bit nguồn, bỏ qua phần tràn, sau đó lấy bù hai của kết quả.

# Modbus – RTU

- Chế độ RTU

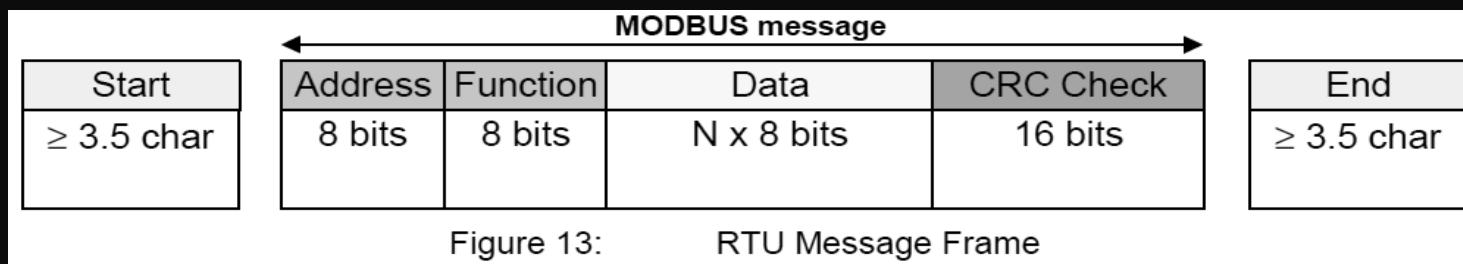


# Modbus – RTU

- Khung RTU

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low, CRC Hi

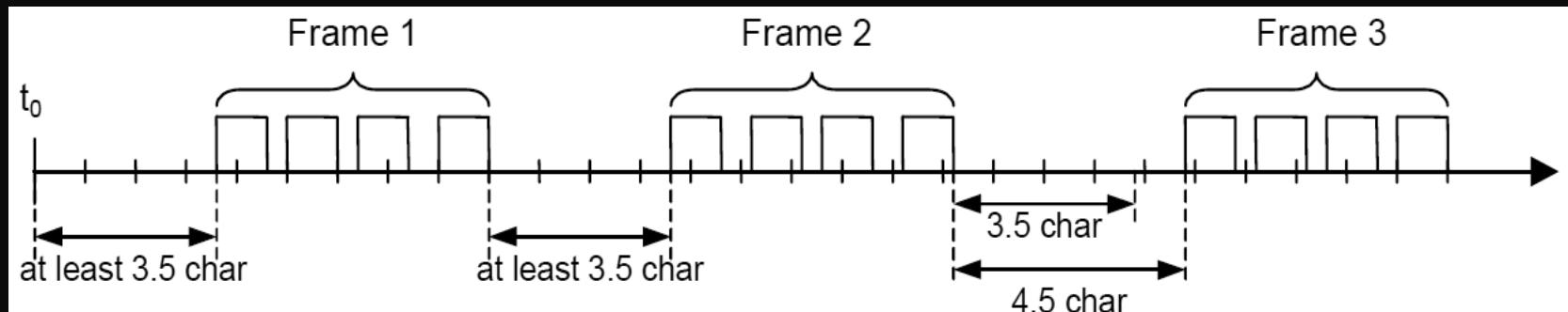
Figure 12: RTU Message Frame



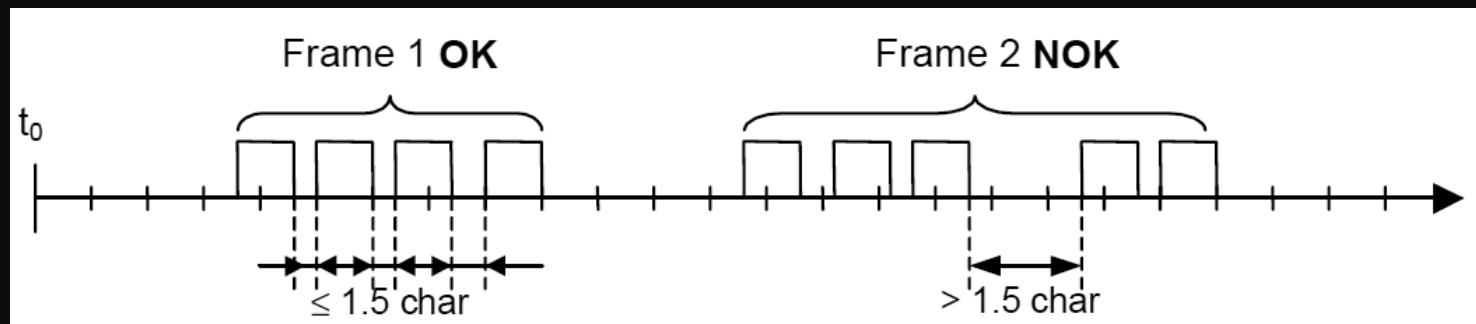
- Khởi đầu khung bằng một khoảng trống tối thiểu 3,5 thời gian ký tự được truyền
- Kết thúc khung bằng một khoảng trống tối thiểu 3,5 thời gian ký tự
- CRC dài 16 bit, đa thức phát G=1010 0000 0000 0001.

# Modbus – RTU

- Khung RTU
  - ❑ Khoảng trống kết thúc khung trước cũng có thể là khoảng khởi đầu khung tiếp theo.



- ❑ Khoảng trống 1,5 thời gian ký tự giữa các ký tự trong một khung truyền sẽ gây lỗi

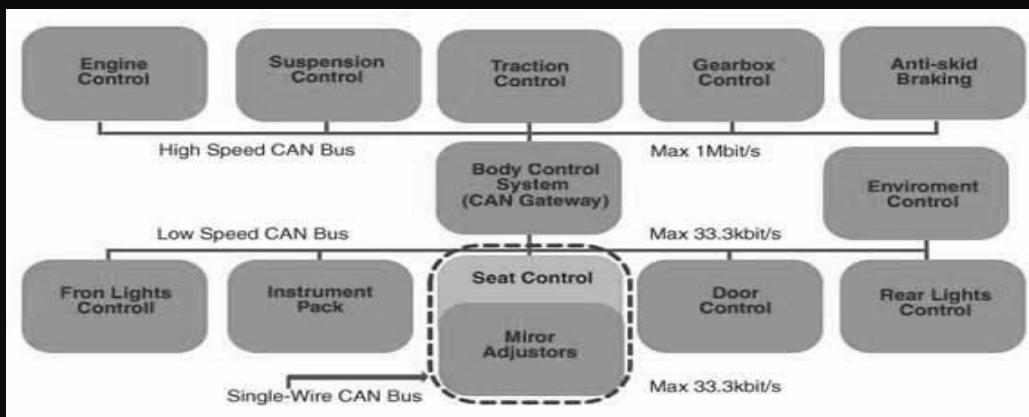
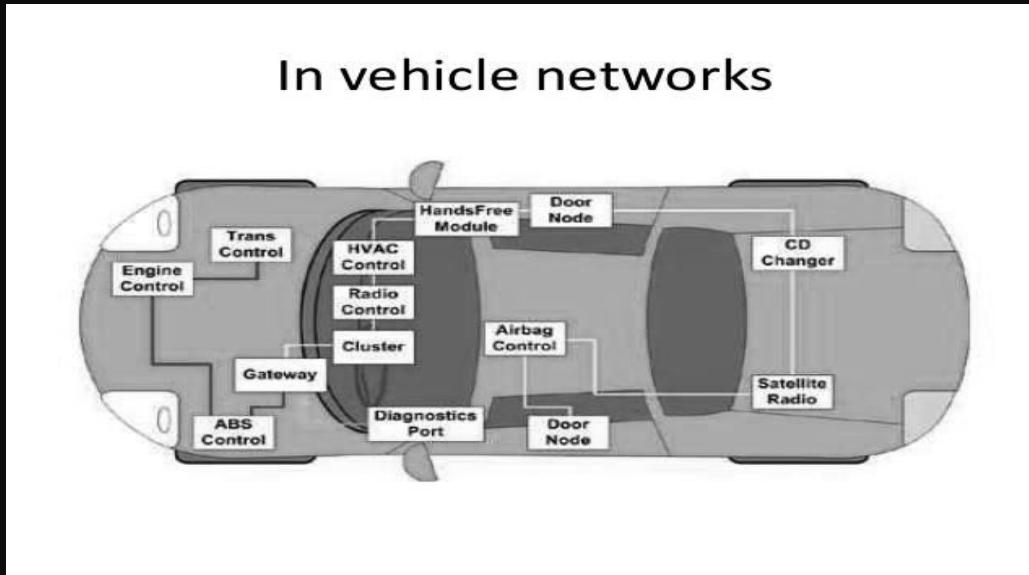


# CAN

- Lịch sử
  - Phát triển từ 1983 bởi BOSCH, phục vụ việc nối mạng các thiết bị trên các phương tiện giao thông
  - Chính thức được giới thiệu bởi SAE(Society of Automotive Engineers) 1986
  - Chuẩn hóa 1993 trong ISO 11898
- Ứng dụng
  - Được sử dụng chủ yếu trong công nghiệp oto, kết nối hầu hết các thiết bị điện tử trên một chiếc xe
  - Đang được ứng dụng ngày càng rộng rãi trong tự động hóa do giá thành rẻ và tốc độ cao
  - Ứng dụng trong các hệ thống điều khiển truyền động như robot, tay máy...

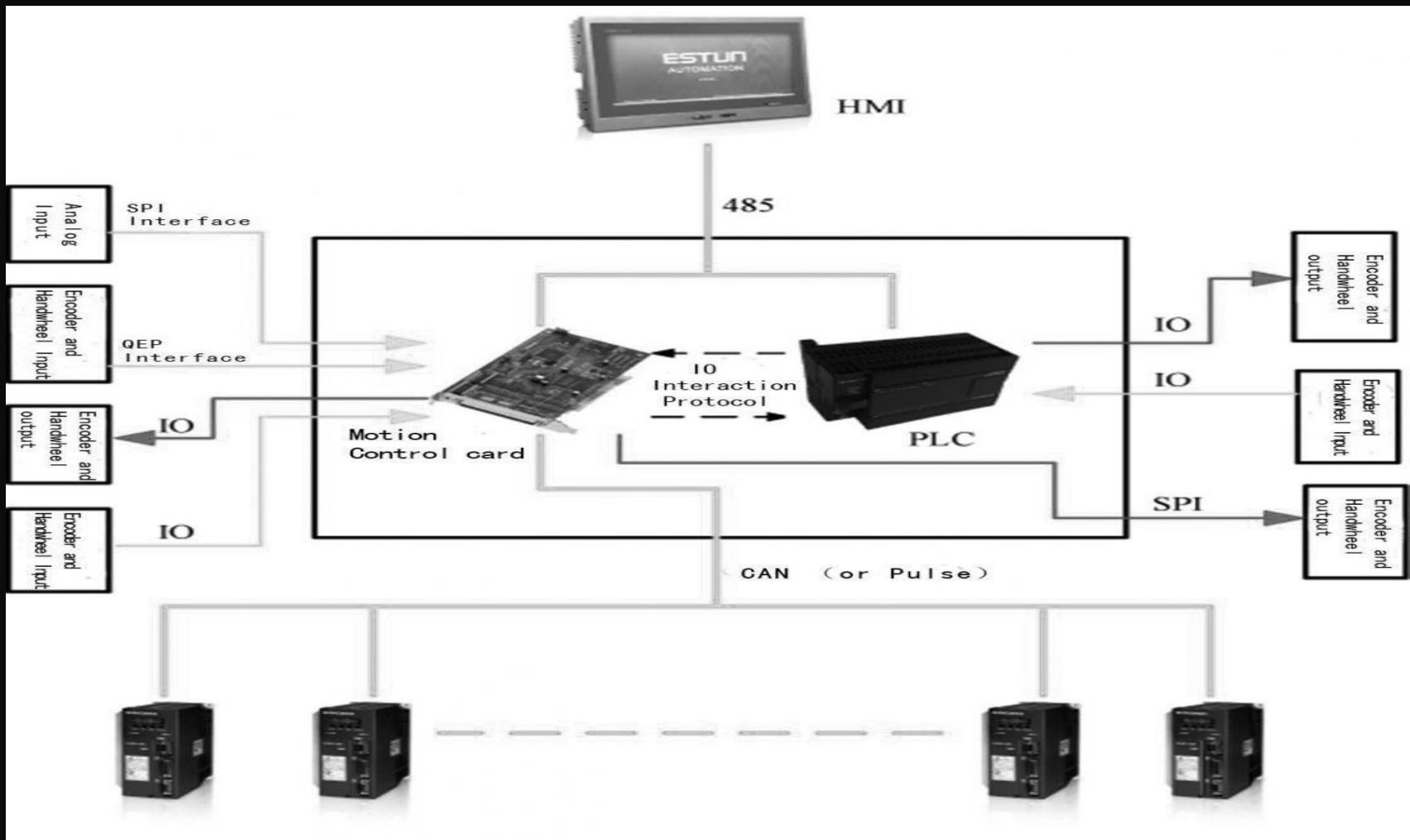
# CAN

- Ứng dụng



# CAN

- Ứng dụng



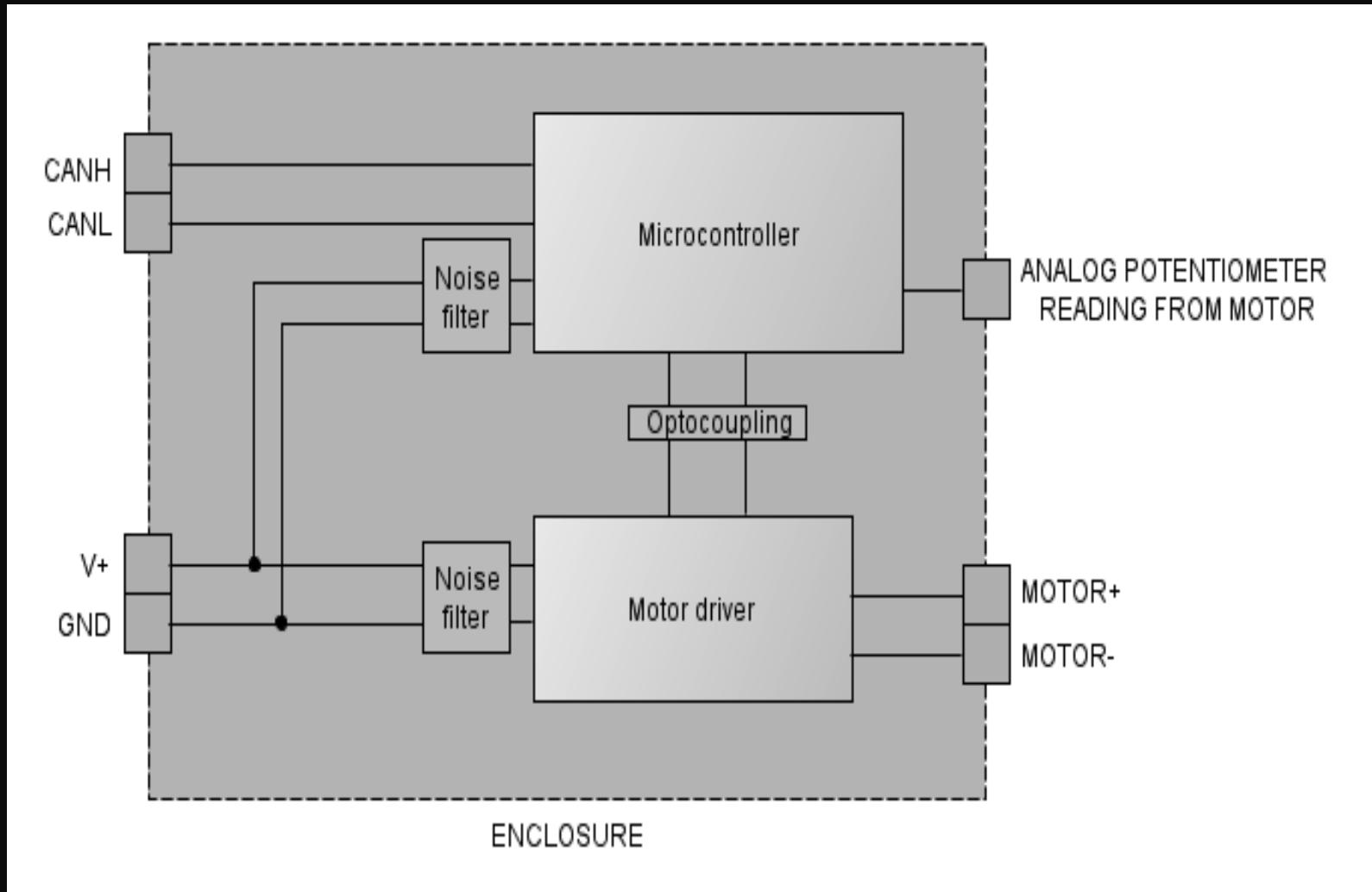
# CAN

- Ứng dụng



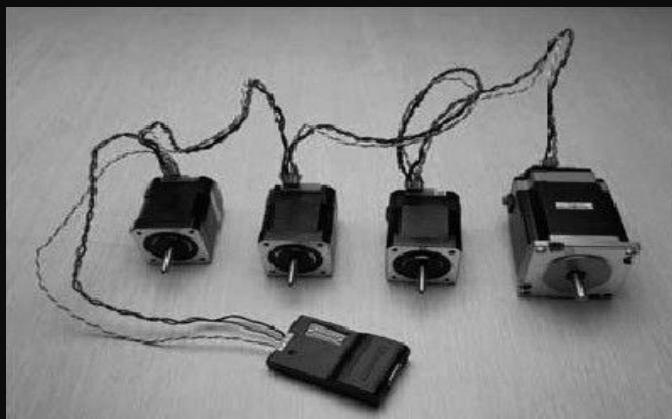
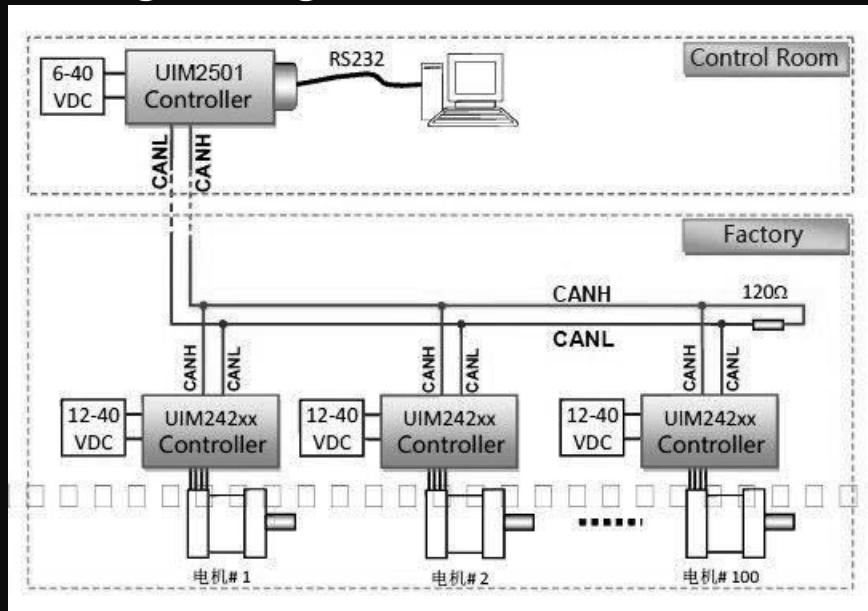
# CAN

- Ứng dụng



# CAN

- Ứng dụng

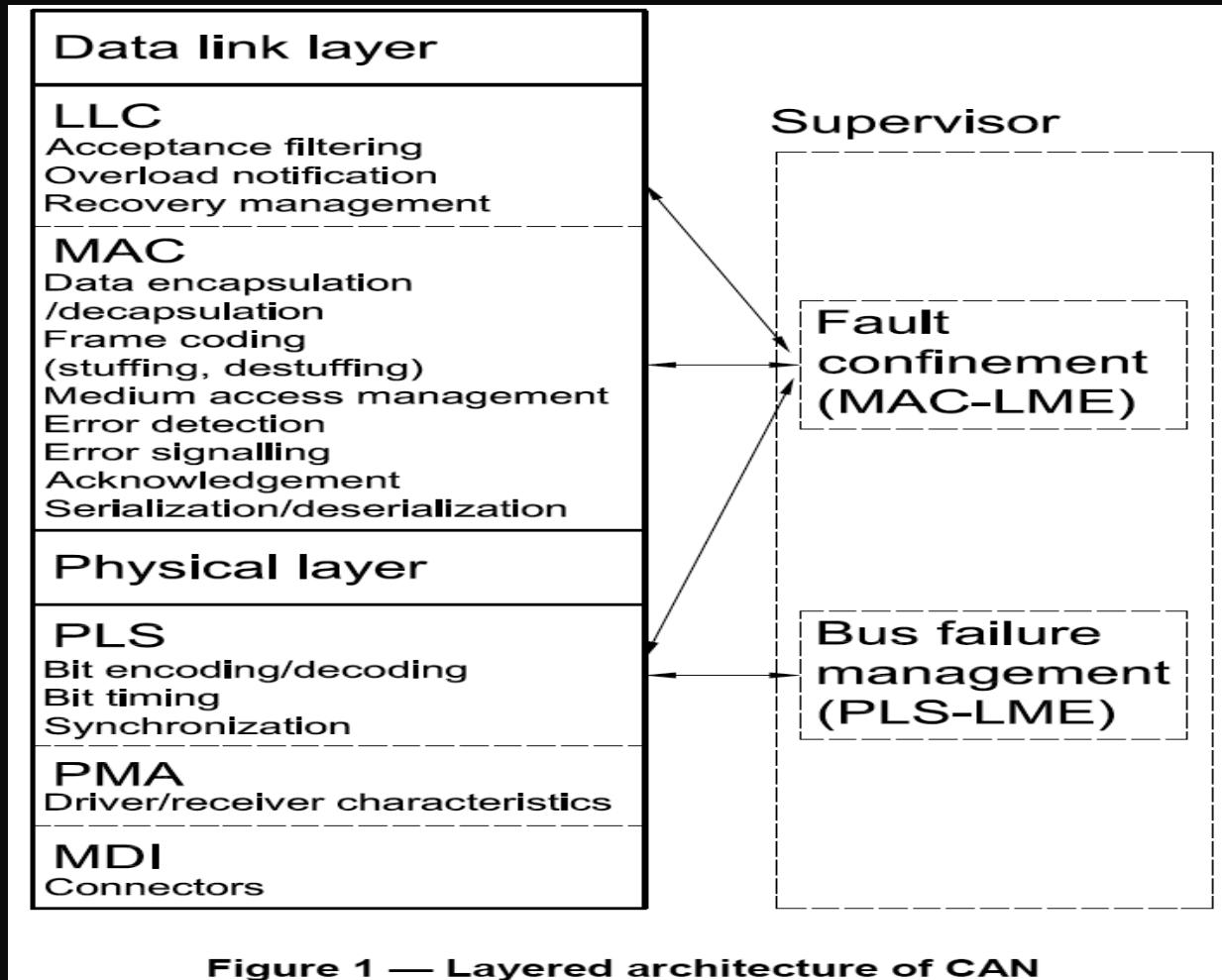


# CAN

- Kiến trúc giao thức
  - ❑ Đối chiếu mô hình 7 lớp OSI, CAN định nghĩa lớp liên kết dữ liệu, gồm 2 lớp con LLC(logical link control) và MAC(Medium access control)
    - ❑ Lớp điều khiển liên kết logic (LLC) đề cập các dịch vụ gửi dữ liệu và yêu cầu dữ liệu từ xa, lọc thông báo, báo cáo quá tải
    - ❑ Lớp điều khiển truy nhập môi trường (MAC) tạo khung thông báo, điều khiển truy nhập bus, xác nhận thông báo và kiểm soát lỗi.
    - ❑ Lớp vật lý đề cập tới việc truyền tín hiệu, định nghĩa phương thức định thời, mã hóa bit, đồng bộ hóa.
    - ❑ Hầu hết các hệ thống mạng công nghiệp dựa trên CAN thực hiện lớp vật lý theo chuẩn ISO 11898

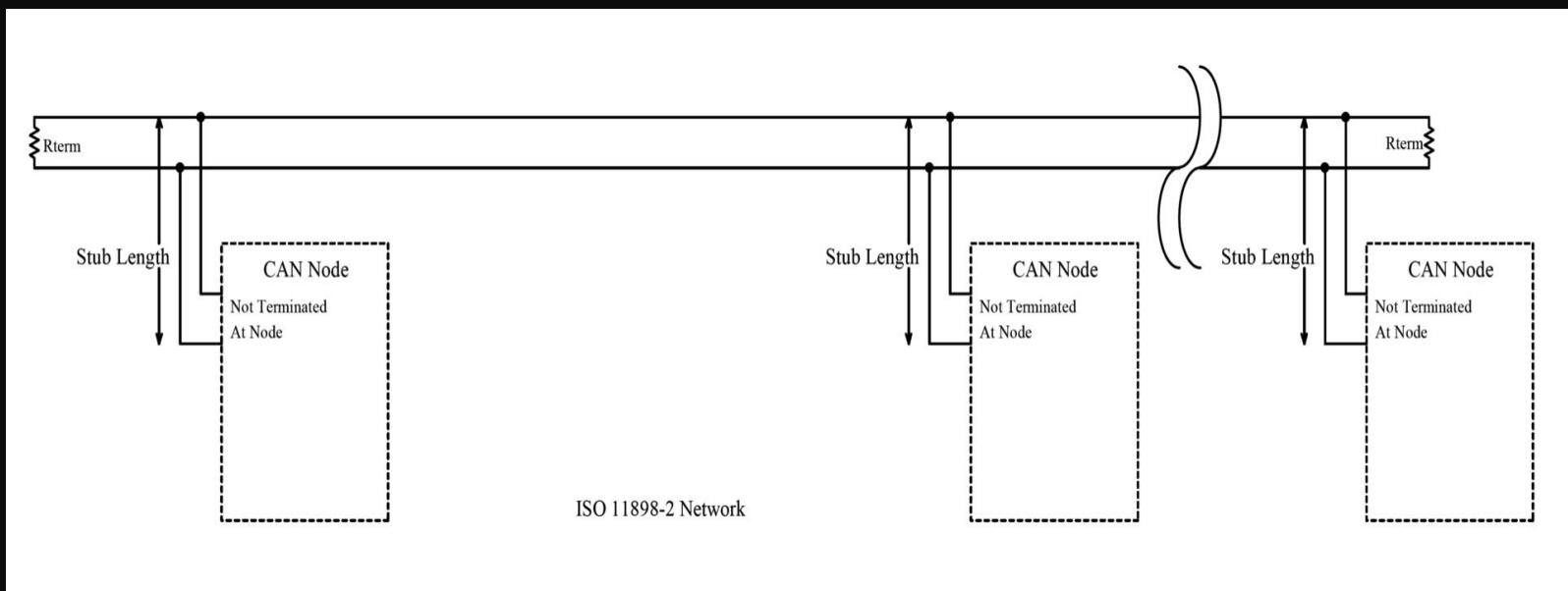
# CAN

- Chuẩn ISO 11898



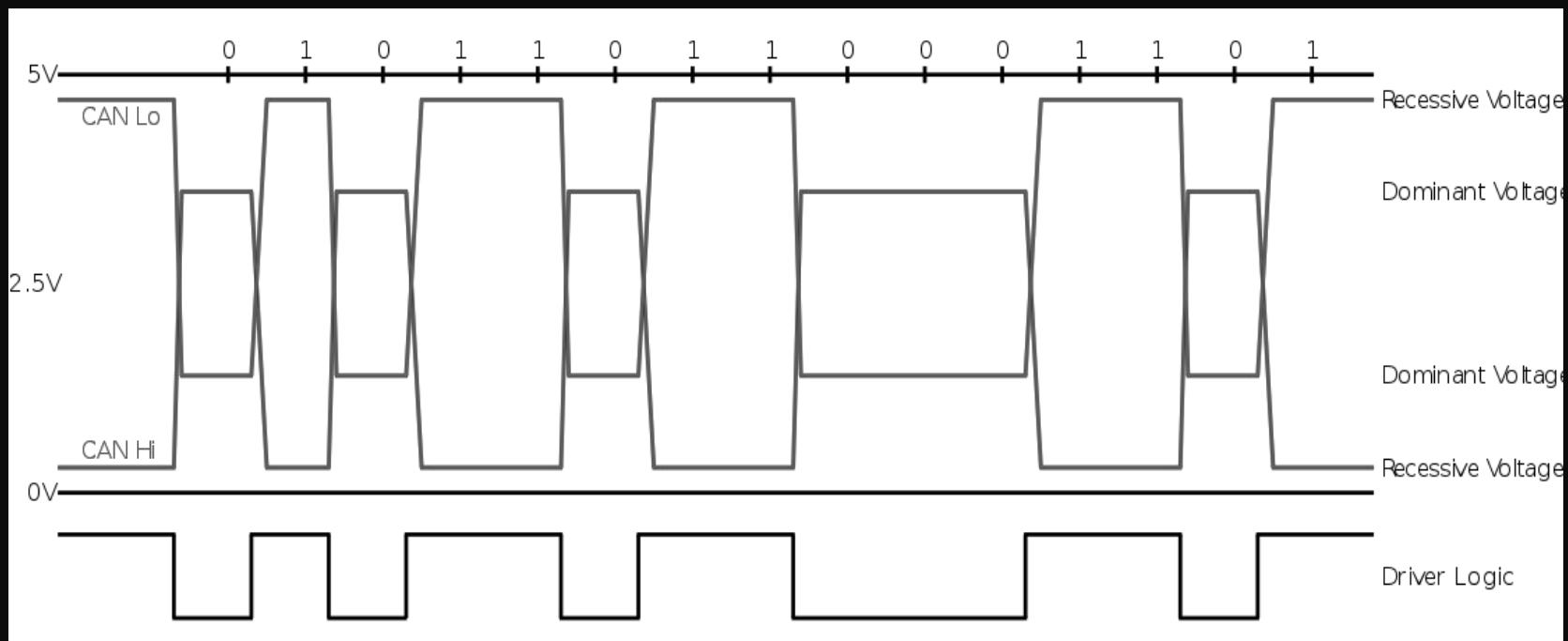
# CAN

- Cấu trúc mạng
  - Thực tế chủ yếu sử dụng cáp đôi dây xoắn và chuẩn điện RS-485, cũng có thể sử dụng cáp quang
  - Với cáp đôi dây xoắn, chủ yếu sử dụng cấu trúc đường thẳng, cũng có trường hợp dùng đường trực/đường nhánh (với chiều dài đường nhánh dưới 0,3m)



# CAN

- Kỹ thuật truyền dẫn
  - Sử dụng phương pháp truy nhập bus CSMA/CA
  - Tốc độ truyền tối đa 1Mbit/s ở khoảng cách 40m và 50kbit/s ở khoảng cách 1000m
  - Mức logic trội (dominant) và lặn (recessive)



# CAN – Cơ chế giao tiếp

- Đặc trưng giao tiếp hướng đối tượng
  - ❑ Mỗi thông tin được coi như một đối tượng, được gán một mã số căn cước
  - ❑ Mỗi trạm tự quyết định tiếp nhận, xử lý thông báo hoặc không, thông qua phương thức lọc thông báo (message filtering)
  - ❑ Mỗi trạm không cần biết cấu hình hệ thống, việc bổ sung hay bỏ một trạm trong mạng không yêu cầu thay đổi phần cứng và phần mềm ở các trạm khác
  - ❑ Mỗi bức điện có một phần chứa căn cước của đối tượng

# CAN – Cấu trúc bức điện

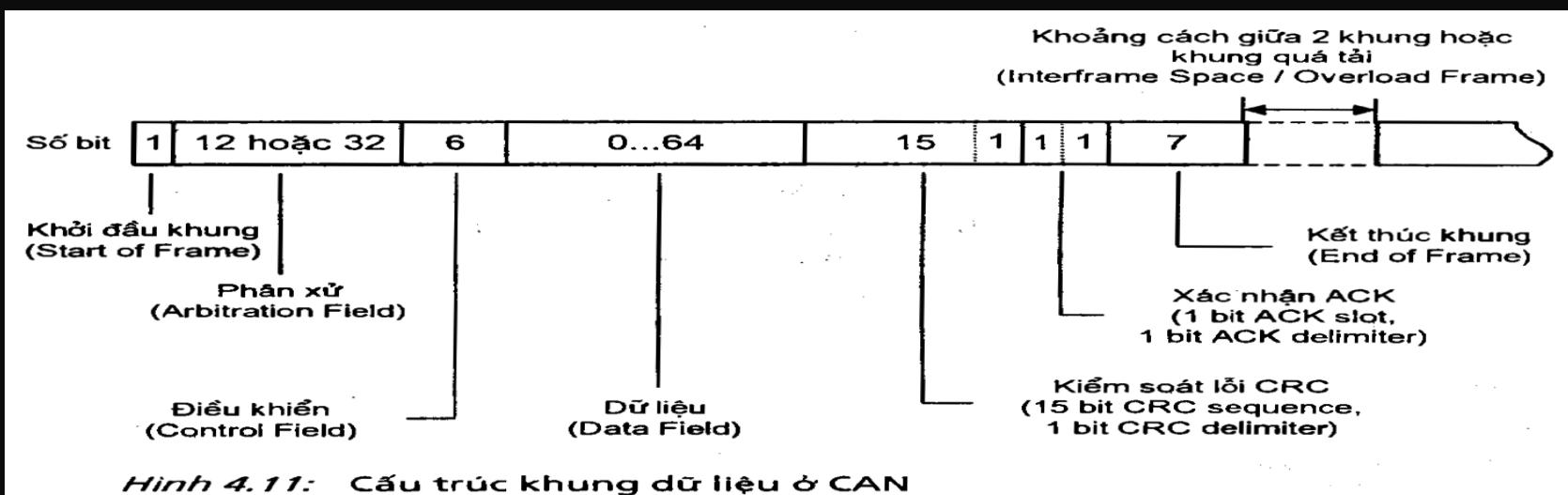
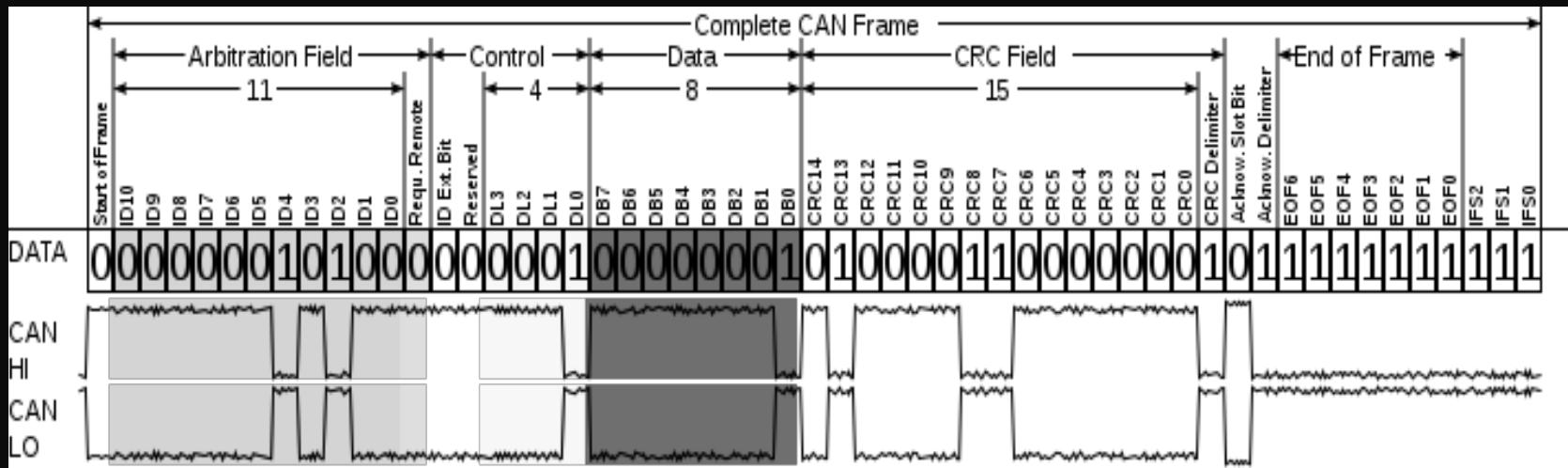
- Căn cước của đối tượng
  - Bức điện có 1 ô chứa căn cước, với chiều dài 11 bit (chuẩn 2.0A) hoặc 29 bit (không mở rộng, chuẩn 2.0B)
- CAN định nghĩa 4 kiểu bức điện
  - Khung dữ liệu (Data Frame)
  - Khung yêu cầu dữ liệu (Remote Frame)
  - Khung lỗi (Error Frame), được gửi từ bất kỳ trạm nào phát hiện lỗi bus
  - Khung quá tải (Overload Frame)

# CAN – Cấu trúc bức điện

- Khung dữ liệu/yêu cầu dữ liệu
  - Khởi đầu khung: 1bit
  - Trường phân xử (Arbitration Field): 12 hoặc 32 bit, với bit cuối RTR (Remote transmission request) dùng để phân biệt giữa khung dữ liệu (bit trội) và khung yêu cầu dữ liệu (bit lặn)
  - Ô điều khiển: 6bit, trong đó 4bit cuối mã hóa chiều dài dữ liệu
  - Trường dữ liệu (Data field): 0-8 byte
  - Ô kiểm soát lỗi CRC: 15 bit và 1bit lặn phân cách, đa thức phát  $G = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$
  - Ô xác nhận ACK: gồm 2 bit ACK Slot và ACK delimiter, được phát đi là các bit lặn. Trạm nhận sẽ kiểm tra bức điện, nếu đúng sẽ phát chòng một bit trội vào ACK Slot.
  - Kết thúc khung: 7bit lặn.

# CAN – Cấu trúc bức điện

- Khung dữ liệu

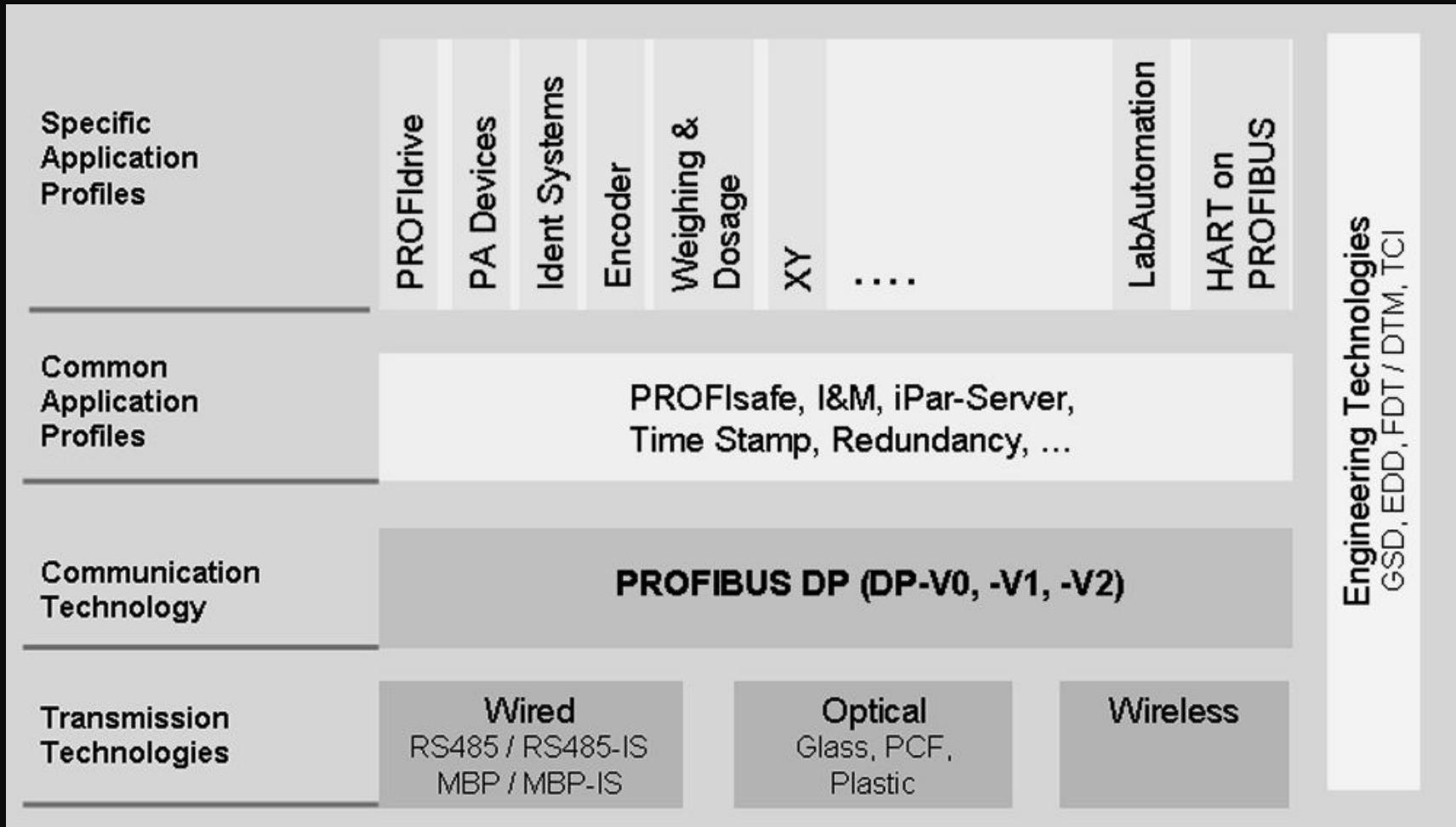


# Profibus

- Lịch sử
  - ❑ Phát triển từ 1986, do 21 công ty và cơ quan nghiên cứu của Đức hợp tác phát triển.
  - ❑ Trở thành chuẩn Châu Âu EN 50 170 trong năm 1996 và chuẩn quốc tế IEC 61158 vào năm 1999.
  - ❑ Hiện được hỗ trợ và quản lý bởi hiệp hội Profibus International (PI Organization)
- Ứng dụng
  - ❑ Profibus FMS được sử dụng chủ yếu cho giao tiếp giữa các máy tính điều khiển và điều khiển giám sát.
  - ❑ Profibus DP được sử dụng cho kết nối các thiết bị vào ra phân tán và các thiết bị cấp trường với máy tính/thiết bị điều khiển.
  - ❑ Profibus PA chủ yếu được sử dụng cho các thiết bị tự động trong môi trường dễ cháy nổ.

# Profibus

- Ứng dụng



# Profibus

- Ứng dụng

Market Segment	Process Automation Ex / non-Ex areas	Factory Automation	Motion Control	Safety Application
PROFIBUS Solution (Common term)	PROFIBUS PA	PROFIBUS DP	PROFIdrive	Safety
Application Profile	PA Devices (and others)	e.g. Ident Systems	PROFIdrive	PROFIsafe
Communication Technology	PROFIBUS DP	PROFIBUS DP	PROFIBUS DP	PROFIBUS DP
Transmission Technology	MBP / MBP-IS RS 485 / 485-IS	RS 485	RS 485	RS 485 MBP-IS

# Profibus

- Kiến trúc giao thức
  - Đối chiếu mô hình 7 lớp OSI, Profibus định nghĩa các lớp ứng dụng, liên kết dữ liệu, vật lý.

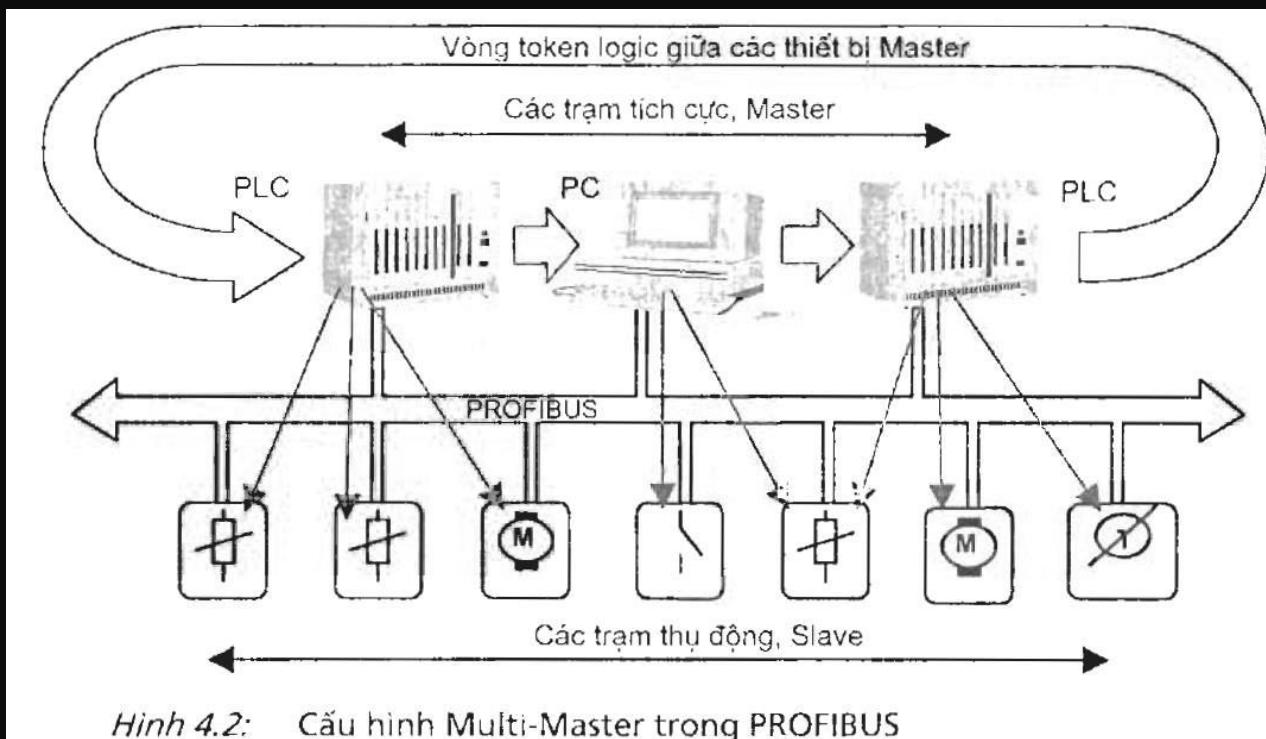
User program	Application profiles
7 Application Layer	PROFIBUS DP Protocol (DP-V0, DP-V1, DP-V2)
6 Presentation Layer.	
5 Session Layer	Not used
4 Transport Layer	
3 Network Layer	
2 Data link Layer	Fieldbus Data Link (FDL): Master Slave principle Token principle
1 Physical Layer	Transmission technology
OSI Layer Model	OSI implementation at PROFIBUS

# Profibus

- Cấu trúc mạng và kỹ thuật truyền dẫn
  - ❑ Thực tế chủ yếu sử dụng cáp đôi dây xoắn và chuẩn điện RS-485, cũng có thể sử dụng cáp quang
  - ❑ Chủ yếu sử dụng cấu trúc đường trực/đường nhánh hoặc daisy-chain, trong đó các tốc độ truyền 1,5Mbit/s trở lên yêu cầu cấu trúc daisy-chain.
  - ❑ Giao diện cơ học của đầu nối chủ yếu là loại D-Sub 9
  - ❑ Với cáp quang thì chỉ có thể đi theo cấu trúc mạng hình sao hoặc mạch vòng.

# Profibus

- Truy nhập bus
  - Master/Slave thích hợp với việc trao đổi giữa một thiết bị điều khiển với các thiết bị trường cấp dưới, sử dụng mạng DP, PA.
  - Token-Passing thích hợp với mạng FMS, cho các thiết bị điều khiển có quyền tương đương
  - Kết hợp hai kiểu trên với hệ thống phức tạp



# SERCOS

- Giới thiệu chung
  - SERCOS (serial real-time communication system) được phát triển bởi một số công ty chế tạo máy công cụ Đức từ 1987, được chuẩn hóa vào 1995 theo chuẩn IEC 61491.
  - Sercos được phát triển chủ yếu cho các hệ thống điều khiển chuyển động, và có thể được sử dụng cho nhiều hệ thống điều khiển khác với khả năng đáp ứng thời gian thực rất cao.
  - Sercos I, II hỗ trợ tốc độ 2, 4, 8 16 Mbit/s, chu kỳ cập nhật đạt tới 62,5 us.
  - Sử dụng đường truyền cáp quang, với cấu trúc mạch vòng (ring).
  - Phiên bản Sercos III được chuẩn hóa theo IEC 61800-7; IEC 61784-1, -2, -3 and IEC 61158.
  - Sercos III dựa trên đường truyền Ethernet, có thể hỗ trợ tới 60 bộ điều khiển chuyển động với chu kỳ cập nhật 250us

# Tổng kết

- Các mạng truyền thông công nghiệp có thể được sử dụng để kết nối máy tính điều khiển với các thiết bị điều khiển khác hoặc các thiết bị cấp trường