

**TRƯỜNG ĐẠI HỌC KỸ THUẬT – CÔNG NGHỆ CẦN THƠ**  
**KHOA ĐIỆN – ĐIỆN TỬ – VIỄN THÔNG**



**ĐỒ ÁN HỌC PHẦN KỸ THUẬT ĐIỆN**  
**ỨNG DỤNG PLC S7 – 1200 ĐIỀU KHIỂN TRÌNH TỰ**  
**LÀM VIỆC CỦA MỘT NHÓM ĐỘNG CƠ**

**SVTH 1: NGUYỄN VĂN ĐÌNH – MSSV: 1350353**

**SVTH 2: THI MINH NHỰT – MSSV: 1350366**

**GVHD: THẦY VÕ MINH THIỆN**

Cần Thơ, ngày 24 tháng 04 năm 2016

## This image shows a full page of a handwriting practice worksheet. It consists of numerous horizontal rows, each defined by two parallel dotted lines. The rows are evenly spaced and extend across the entire width of the page, providing a guide for letter height and placement. There is no text or other markings on the page.

*Cần Thơ, ngày ..... tháng ..... năm 2016*

[illegible]

*Cần Thơ, ngày ..... tháng ..... năm 2016*

## This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the entire width of the page, providing a guide for handwriting or typing. There are no margins, text, or other markings on the page.

*Cần Thơ, ngày ..... tháng ..... năm 2016*

## LỜI CẢM ƠN

Chúng em chân thành cảm ơn cha mẹ, thầy cô và bạn bè đã tạo điều kiện giúp đỡ chúng em hoàn thành tốt đề tài.

Chúng em rất biết ơn sự giúp đỡ và hướng dẫn tận tình của thầy Võ Minh Thiện trong quá trình hướng dẫn chúng em thực hiện hoàn thành đồ án môn học Kỹ thuật điện. Cảm ơn thầy đã tạo điều kiện cung cấp thiết bị thực hành, tài liệu học tập để chúng em có thể hiểu được những kiến thức về lý thuyết để từ đó có thể áp dụng vào thực tế. Trong quá trình thực hiện, chúng em học được nhiều kiến thức mới và kỹ năng từ thầy. Sự hướng dẫn tận tình của thầy làm cho chúng em có thêm động lực để hoàn thành đề tài, không phụ công sức của thầy đã hướng dẫn chúng em. Chúng em xin chân thành cảm ơn thầy!

Cảm ơn anh chị, bạn bè đã chia sẻ tài liệu giúp cho nhóm, để nhóm có những kiến thức cơ bản để tiếp cận với vấn đề mới.

Xin chân thành cảm ơn!

*Nhóm sinh viên thực hiện*

Nguyễn Văn Đình

Thi Minh Nhựt

## **ĐẶT VẤN ĐỀ**

Với sự phát triển của công nghệ điện tử, các chip vi xử lý, vi điều khiển, bộ logic có thể lập trình, các máy tính đã lần lượt ra đời, tạo cơ sở thúc đẩy cho việc phát triển và ứng dụng tự động hóa vào các dây truyền sản xuất trong công nghiệp.

Trong nhiều ngành công nghiệp hiện nay, nhất là ngành công luyện kim, chế biến thực phẩm. . . Bộ logic có thể lập trình (Programmable Logic Controller – PLC) là một thiết bị không thể thiếu trong dây các truyền sản xuất. Từ đó thấy được tầm quan trọng của bộ điều khiển PLC và những được ứng dụng rộng rãi trong công nghiệp.

Trong phạm vi đồ án, nhóm em nghiên cứu về cách sử dụng và điều khiển bộ điều khiển PLC S7 – 1200 và ứng dụng nó vào điều khiển động cơ. Trên những cơ sở khi tìm hiểu về bộ điều khiển PLC S7-1200 để sau này có kiến thức cơ sở để sử dụng những bộ điều khiển PLC khác phục vụ cho nhu cầu công việc.

Để ứng dụng kiến thức học được nhóm chúng em thực hiện đề tài “Ứng dụng PLC S7 – 1200 điều khiển trình tự làm việc của một nhóm động cơ”.

Dù đã cố gắng thực hiện nhưng không thể tránh khỏi sai sót, nhờ thầy và các bạn cho ý kiến để chúng em có thể hoàn thành tốt bài báo cáo đồ án môn học và học được những kiến thức mới.

## Mục lục

<b>Chương 1 CƠ SỞ LÝ THUYẾT VỀ PLC S7 – 1200 .....</b>	<b>1</b>
1.1 Tổng quan về bộ điều khiển S7 – 1200 .....	1
1.2 Phân loại .....	1
1.3 Hình dạng bên ngoài.....	2
1.3.1 Hình dạng bên ngoài .....	2
1.3.2 Cấu trúc bên trong.....	3
1.4 Sơ đồ đấu dây .....	4
1.5 Module mở rộng .....	4
<b>Chương 2 NGÔN NGỮ LẬP TRÌNH CHO BỘ ĐIỀU KHIỂN S7 – 1200 .....</b>	<b>5</b>
2.1 Phương pháp lập trình điều khiển .....	5
2.2 Các ngôn ngữ lập trình.....	5
2.2.1 Ngôn ngữ lập trình LAD .....	5
2.2.2 Ngôn ngữ lập trình FBD .....	6
2.3 Phần mềm lập trình cho PLC S7 – 1200.....	6
2.3.1 Cách cài đặt phần mềm TIA Portal V13 .....	6
2.3.2 Sử dụng phần mềm TIA Portal V13 .....	7
2.4 Tập lệnh của PLC S7 – 1200 .....	13
2.4.1 Các lệnh về bit.....	13
2.4.1.1 Tiếp điểm .....	13
2.4.1.2 Lệnh đảo bit .....	13
2.4.1.3 Lệnh AND, OR, XOR .....	14
2.4.1.4 Lệnh ngõ ra .....	15
2.4.1.5 Lệnh SET và lệnh RESET .....	15
2.4.1.6 Lệnh cạnh lên và lệnh cạnh xuống .....	17
2.4.2 Nhóm lệnh định thời – Timer.....	20
2.4.3 Nhóm lệnh đếm – Counter .....	21
2.4.4 Nhóm lệnh so sánh .....	22
2.4.4.1 Lệnh so sánh byte.....	22
2.4.4.2 Lệnh In – range và Out – of – range .....	22
2.4.4.3 Lệnh OK và NOT OK.....	22

2.4.5 Nhóm lệnh toán học .....	22
2.4.6 Nhóm lệnh di chuyển .....	24
2.4.7 Nhóm lệnh chuyển đổi .....	25
2.4.8 Nhóm lệnh điều khiển chương trình .....	26
2.4.9 Nhóm lệnh dịch và quay .....	26
2.4.10 Lệnh vòng lặp PID .....	27
2.4.11 Lệnh đếm xung tốc độ cao .....	27
<b>Chương 3 ỨNG DỤNG PLC ĐIỀU KHIỂN TRÌNH TỰ CỦA MỘT NHÓM ĐỘNG CƠ.....</b>	<b>28</b>
3.1 Yêu cầu.....	28
3.2 Mạch điều khiển.....	29
3.3 Mạch động lực.....	29
3.4 Chương trình viết bằng ngôn ngữ LAD.....	30
<b>Chương 4 KẾT QUẢ VÀ KIẾN NGHỊ.....</b>	<b>32</b>
4.1 Kết quả .....	32
4.2 Kiến nghị.....	32
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>33</b>



# **Chương 1 CƠ SỞ LÝ THUYẾT VỀ PLC S7 – 1200**

## **1.1 Tổng quan về bộ điều khiển S7 – 1200**

PLC S7-1200 (Programmable Logic Controller) là những kết hợp I/O và các lựa chọn cấp nguồn, gồm 9 module các bộ cấp nguồn cả VAC – hoặc VDC – các bộ nguồn với sự kết hợp I/O DC hoặc Relay. Các module tín hiệu để mở rộng I/O và các module giao tiếp dễ dàng kết nối với các mặt của bộ điều khiển.

Các module tín hiệu có trong các model đầu vào, đầu ra và kết hợp loại 8, 16 và 32 điểm hỗ trợ các tín hiệu I/O DC, relay và analog. Bên cạnh đó, bảng điều khiển tín hiệu tiên tiến: kênh I/O số có 4 kênh và kênh I/O analog có 1 kênh được gắn trước bộ điều khiển, cho phép nâng cấp số chân I/O mà không cần thêm không gian, với điều chỉnh này cho phép điều chỉnh các ứng dụng từ 10 I/O đến 284 I/O.

Các đặc điểm khác: bộ nhớ 50kB (ghi dữ liệu người sử dụng và dữ liệu chương trình), đồng hồ thời gian thực, 16 vòng lặp PID với khả năng điều chỉnh tự động, cho phép bộ điều khiển xác định thông số vòng lặp gần tối ưu cho hầu hết các ứng dụng điều khiển ứng dụng thông dụng; một cổng giao tiếp Ethernet 10/100MBit tích hợp với hỗ trợ giao thức Profinet cho lập trình; kết nối HMI/SCADA; kết nối PLC với PLC.

Chức năng chính của PLC là kiểm tra trạng thái của các thiết bị đầu vào và điều khiển các quá trình hoặc các hệ thống máy móc thông qua các tín hiệu trên chính đầu ra của PLC.

## **1.2 Phân loại**

Dựa vào CPU mà PLC trang bị, thông dụng: CPU 1211C, CPU 1212C, CPU 1214C. PLC được chia thành 2 loại chính:

Loại cấp điện áp 220V AC:

- Ngõ vào: Kích hoạt mức 1 ở cấp điện áp +24V DC từ (15V DC – 30V DC).
- Ngõ ra: relay.
- Ưu điểm: dùng ngõ ra relay, dùng ở nhiều cấp điện áp khác nhau: 0V, 24V, 220V, ...
- Nhược điểm: do dùng ngõ ra relay, nên thời gian đáp ứng không nhanh cho điều chế độ rộng xung (PWM), ứng dụng có ngõ ra tốc độ cao, ...

Loại cấp điện áp 24V DC:

- Ngõ vào: Kích hoạt mức 1 ở cấp điện áp +24V DC từ (15V DC – 30V DC).

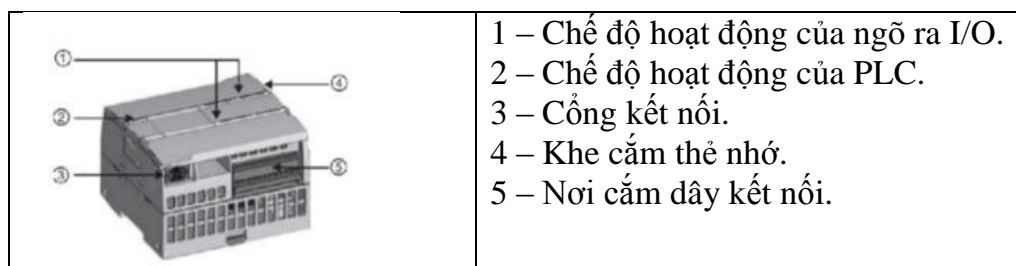
- Ngõ ra: transistor.
- Ưu điểm: dùng ngõ ra transistor, dùng để điều chế độ rộng xung, ứng dụng tốc độ cao,...
- Nhược điểm: do dùng ngõ ra transistor, nên chỉ dùng được một cấp điện áp +24VDC, gây khó khăn với ứng dụng có nhiều cấp điện áp khác nhau, phải thông qua bộ relay đệm 24VDC.

\* Tổng hợp các đặc điểm cơ bản của PLC S7 – 1200:

Đặc trưng	CPU 1211C	CPU 1212C	CPU 1214C
Kích thước (mm)	90 × 100 × 75		110 × 100 × 75
Bộ nhớ làm việc	25 Kbytes		2 Mbytes
Bộ nhớ sự kiện	2 Kbytes		2 Kbytes
Digital I/O	6 Input/ 4 Output	8 Input/ 6 Output	14 Input/ 10 Output
Analog I	2 Input		
Tốc độ xử lý ảnh	Input – Output : 1024bytes		
Mạch mở rộng	Không có	2	8
Mạch tính hiệu	1		
Mạch giao tiếp	3		
Bộ đếm tốc độ cao	3	4	6
Trạng thái đơn	3 – 100kHz	3 – 100kHz 1 – 30kHz	3 – 100kHz 3 – 30kHz
Trạng thái đôi	3 – 80kHz	3 – 80kHz 1 – 20kHz	3 – 80kHz 3 – 20kHz
Mạch ngõ ra	2		
Thẻ nhớ	Thẻ nhớ Simatic (tùy chọn)		
Thời gian lưu trữ khi mất điện	240h		
Profinet	Một cổng giao tiếp Ethernet		
Tốc độ thực thi phép toán số thực	18 $\mu$ s		
Tốc độ thực thi phép toán số thực	0.1 $\mu$ s		

### 1.3 Hình dạng bên ngoài

#### 1.3.1 Hình dạng bên ngoài



Bộ CPU 1212C gồm 10 ngõ ra và 6 ngõ vào, mở rộng: 2 module tín hiệu (SM), 1 mạch tín hiệu (SB), 3 module giao tiếp (CM).

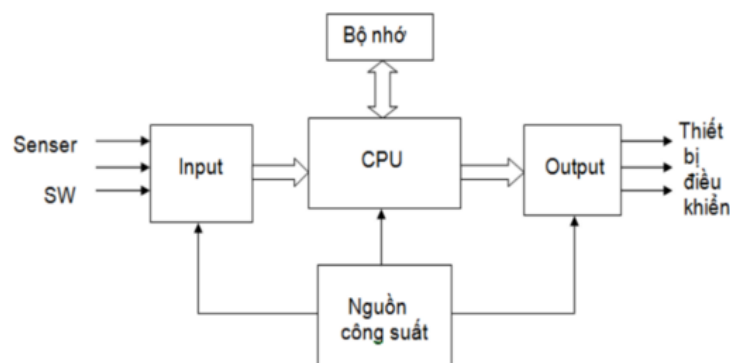
Các đèn báo trên CPU 1212C:

- STOP/RUN (màu cam/xanh): CPU ngừng/đang thực hiện chương trình đã nạp vào bộ nhớ.
- ERROR (màu đỏ): báo hiệu thực hiện chương trình đã xảy ra lỗi.
- MAINT: báo hiệu có chèn thẻ nhớ vào hay không.
- LINK (màu xanh): báo hiệu việc kết nối với máy tính thành công.'
- TX/RX (màu vàng): nhấp nháy, báo hiệu tín hiệu được truyền.
- Đèn cổng ra vào:
- Ix.x (màu xanh): báo hiệu trạng thái tức thời của cổng theo giá trị công tắc.
- Qx.x (màu xanh): báo trạng thái tức thời của cổng theo giá trị logic.

### 1.3.2 Cấu trúc bên trong

Gồm 4 thành phần cơ bản: bộ xử lý, bộ nhớ, bộ nguồn và bộ xuất – nhập I/O.

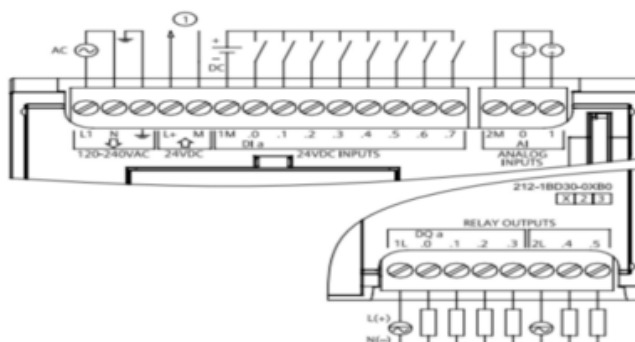
- Bộ xử lý: chứa vi xử lý, biên dịch các tín hiệu nhập, thực hiện các hoạt động điều khiển theo chương trình được lưu trong PLC. Truyền tín hiệu dưới dạng các tín hiệu hoạt động đến các thiết bị xuất.
- Bộ nhớ: lưu trữ chương trình điều khiển.
- Bộ nguồn: chuyển đổi điện áp AC thành DC (24VDC), cung cấp cho bộ xử lý và các module giao tiếp nhận và xuất nhập.
- Bộ xuất nhập: nhận thông tin từ thiết bị ngoại vi và truyền thông tin đến bộ điều khiển. Tín hiệu nhập là: công tắc, cảm biến,. . . Tín hiệu xuất là: các cuộn dây của bộ khởi động động cơ, các van solenoid,. . .
- Chương trình điều khiển được nạp vào bộ nhớ nhờ trợ giúp của bộ lập trình hay máy vi tính.



## 1.4 Sơ đồ đấu dây

Ví dụ sơ đồ đấu dây cho CPU 1212C:

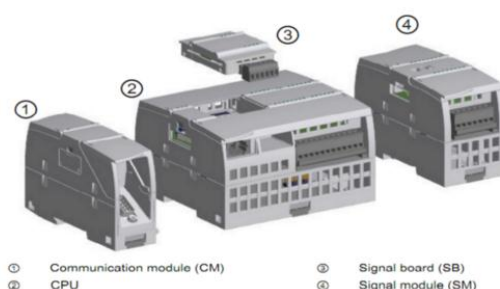
- Cấp nguồn: cấp nguồn 120 – 240VAC ( $f = 47 - 63\text{Hz}$ ), điện áp thay đổi 85 – 264V,  $V = 264\text{V}$  thì  $I = 2\text{A}$ ) hoặc 24VDC (điện áp thay đổi từ 20.4 – 28.8V,  $V = 28.8\text{V}$  thì  $I = 12\text{A}$ ). Tùy theo ứng dụng mà chọn cách nối nguồn thích hợp.



- Ngõ nhập – xuất: Các ngõ vào được tác động ở mức 24V DC, ngõ ra mức 0 khi công tắc hở hay điện áp  $U \leq 5\text{V DC}$ . Ngõ vào mức 1 khi công tắc đóng điện hay điện áp  $U \geq 15\text{V DC}$ . Thời gian thay đổi trạng thái là  $0.1\mu\text{s}$ . Ngõ ra có thể là 5 – 30V DC hay 5 – 250V AC.

## 1.5 Module mở rộng

Các họ PLC S7 – 1200 cung cấp nhiều nhất là 8 module tín hiệu đa dạng và 1 mạch tín hiệu cho bộ xử lý có khả năng mở rộng. Ngoài ra có thể thêm 3 module giao tiếp nhờ vào giao thức truyền thông.



Module		Ngõ vào	Ngõ ra	Ngõ vào – ra
Module tín hiệu (SM)	Digital	8 × DC	8 × DC 8 × Relay	8 × DC/8 × DC 16 × DC/8 × Relay
		16 × DC	16 × DC 16 × Relay	16 × DC/16 × DC 16 × Relay/16 × Relay
	Analog	2 × Analog	4 × Analog	4 × Analog/2 × Analog
Board tín hiệu (SB)	Digital	–	–	2 × DC/2 × DC
	Analog	–	1 × Analog	–
Module giao tiếp (CM)	RS485			
	RS232			

## Chương 2 NGÔN NGỮ LẬP TRÌNH CHO BỘ ĐIỀU KHIỂN S7 – 1200

### 2.1 Phương pháp lập trình điều khiển

Khác với phương pháp điều khiển phần cứng, trong hệ thống điều khiển có lập trình, *cấu trúc của bộ điều khiển và cách đấu dây độc lập với chương trình.*

*Để thay đổi tiến trình điều khiển, ta chỉ cần thay đổi nội dung bộ nhớ điều khiển, không cần phải thay đổi cách đấu dây bên ngoài.*

*Phương pháp này thực hiện theo 5 bước sau:*

- Xác định yêu cầu.
- Thiết kế giải thuật.
- Soạn thảo chương trình.
- Nạp chương trình vào bộ nhớ.
- Chạy thử và kiểm tra.

### 2.2 Các ngôn ngữ lập trình

Người ta thường dùng ngôn ngữ lập trình LAD (Ladder Logic) và ngôn ngữ lập trình FBD (Function Block Diagram). Nhóm chúng em chọn ngôn ngữ lập trình LAD để viết chương, em xin trình bày một số đặc điểm chính của hai ngôn ngữ này.

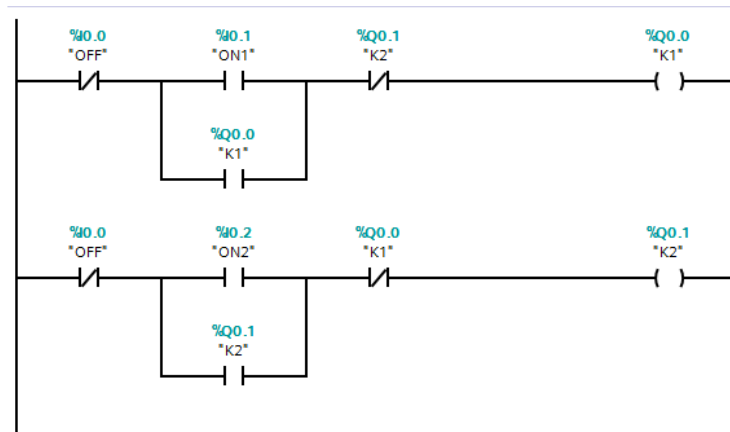
#### 2.2.1 Ngôn ngữ lập trình LAD

Chương trình được viết bằng ngôn ngữ LAD gồm: cột dọc là nguồn điện cùng với các ký hiệu công tắc logic tạo thành một mạch điện logic nằm ngang.

*Các quy ước của ngôn ngữ lập trình LAD:*

- Các đường dọc: đường công suất, các mạch được nối với đường này.
- Mỗi nấc thang: xác định một hoạt động trong quá trình điều khiển.
- Sơ đồ thang: được đọc từ trái sang phải và từ trên xuống dưới. Nấc ở đỉnh thang: đọc từ trái sang phải, nấc thứ 2 (tính từ đỉnh xuống): đọc từ trên xuống rồi từ trái sang phải, cách đọc tương tự cho các nấc còn lại.
- Nấc thang: bắt đầu với một hoặc nhiều ngõ vào và kết thúc với ít nhất một ngõ ra.
- Các thiết bị điện: được trình bày ở điều kiện chuẩn của chúng. Công tắc thường hở được trình bày ở trạng thái hở, công tắc thường đóng được trình bày ở trạng thái đóng.
- Các thiết bị có thể lập lại trên nhiều nấc thang.
- Các ngõ vào và ra được nhận biết theo địa chỉ của chúng.

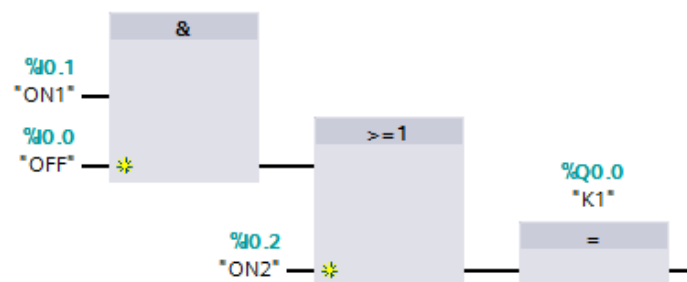
Ví dụ: chương trình sau được viết bằng ngôn ngữ LAD.



## 2.2.2 Ngôn ngữ lập trình FBD

Phương pháp này dùng các tiếp điểm thay cho các cổng logic. Ví dụ: các tiếp điểm nối tiếp được thay bằng cổng AND, các tiếp điểm song song được thay thế bằng cổng OR, các tiếp điểm thường đóng được thay thế bằng cổng NOT.

Ví dụ: chương trình sau được viết bằng ngôn ngữ FBD.



## 2.3 Phần mềm lập trình cho PLC S7 – 1200

Để viết chương trình nạp cho PLC S7 – 1200, chúng ta sử dụng phần mềm TIA Portal, khi làm đồ án, em sử dụng phiên bản TIA Portal V13 để viết chương trình nạp cho PLC.

### 2.3.1 Cách cài đặt phần mềm TIA Portal V13

a. Yêu cầu cấu hình máy tính:

Máy tính có bộ xử lý Core i5-3320M , 3.3 GHz, RAM 8GB, độ phân giải màn hình 1920 x 1080 pixel.

b. Địa chỉ tải phần mềm:

Do phần mềm TIA Portal có tính phí, nên chúng ta tải ở các trang do Siemens do cung cấp (có bản dùng thử).

*c. Cách cài đặt phần mềm:*

Việc cài đặt phần mềm tương đối dễ, chỉ cần làm theo hướng dẫn trong quá trình cài đặt. Khi cài đặt ta chọn ngôn ngữ tiếng Anh (English) để cài đặt.

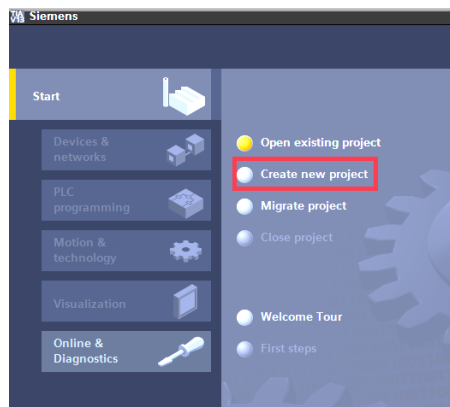
### 2.3.2 Sử dụng phần mềm TIA Portal V13

*a. Tạo một Project*

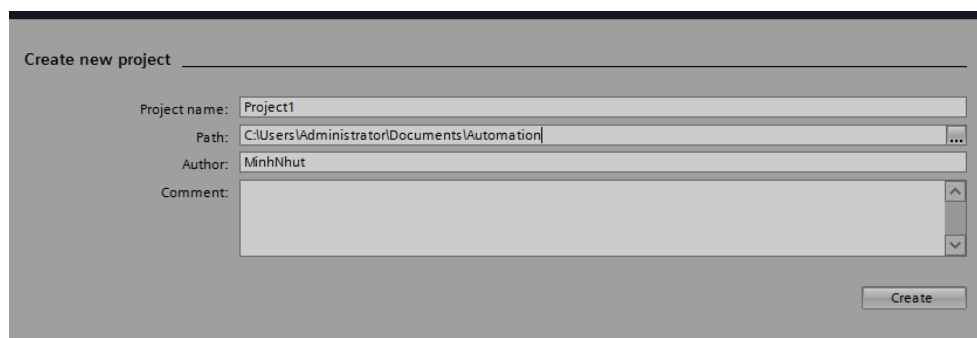
- Click đúp vào biểu tượng TIA Portal V13 trên màn hình Desktop (do phần mềm dung lượng nặng, nên cần thời gian để mở lên).



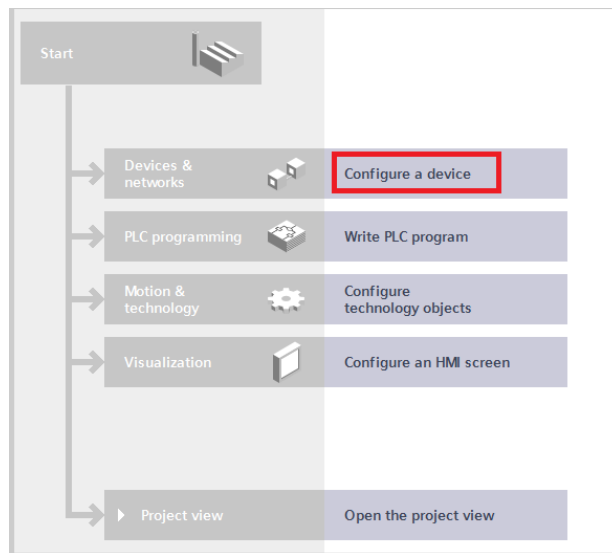
- Giao diện mới hiện lên: chọn Create new project



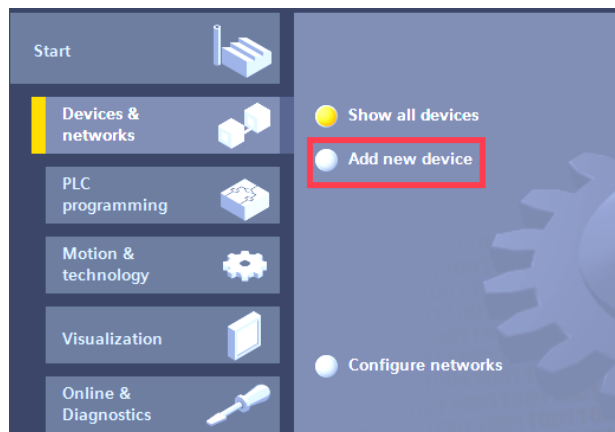
- Xuất hiện của sổ mới: chúng ta điền vào tên dự án (Project name), chọn nơi lưu dự án (Path). Rồi chọn Create.



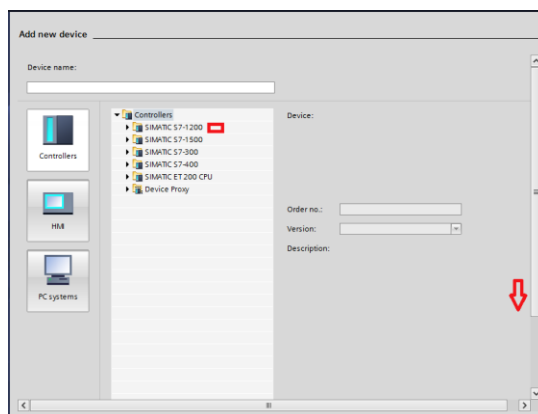
- Khi xuất hiện giao diện như bên dưới, ta chọn: Configure a device



- Tiếp tục giao diện mới ta chọn Add new device.

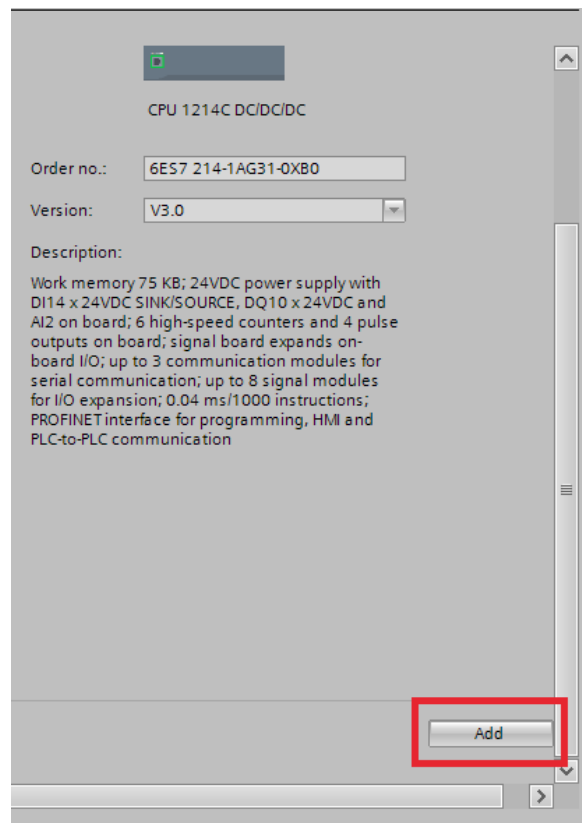


- Xuất hiện một danh sách các loại PLC. Tùy vào loại PLC mà ta có thể chọn loại CPU thích hợp.



- Sau khi chọn được loại CPU, ta kéo thanh cuộn xuống chọn Add

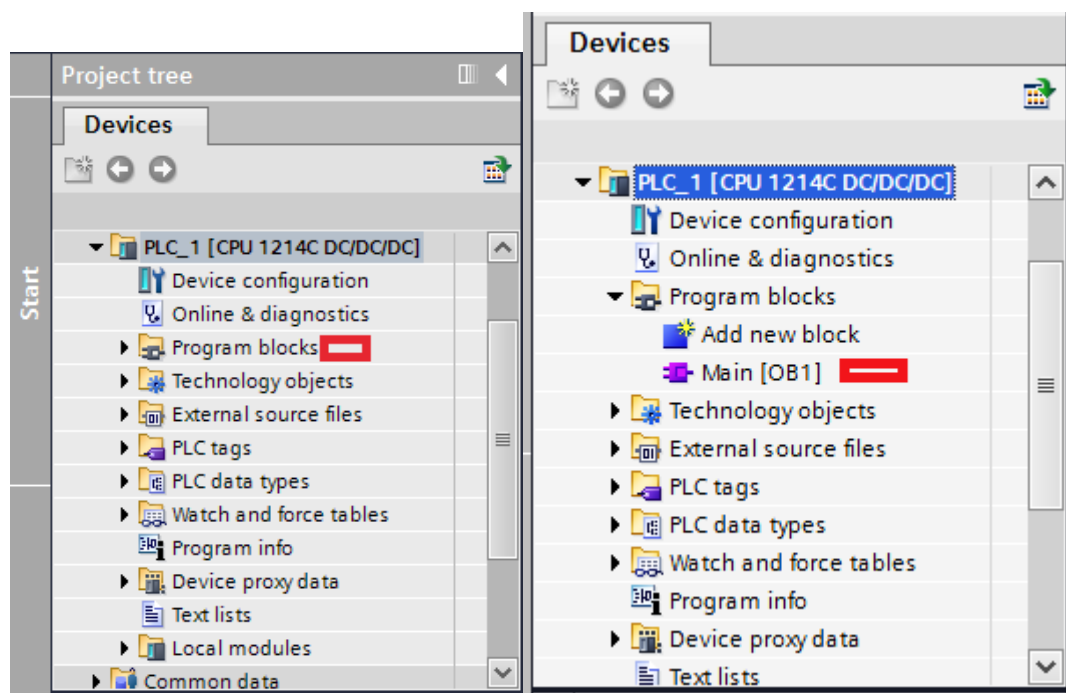




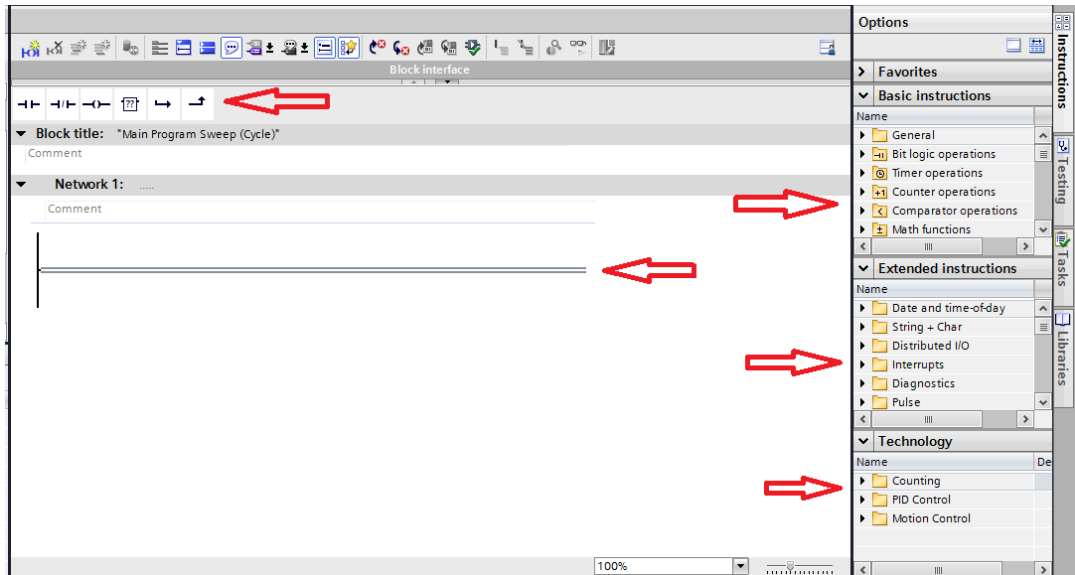
#### b. Viết chương trình

Sau khi đã tạo xong dự án, ta tiến hành viết chương trình.

- Tìm đến cửa sổ sau và mở tag *Program blocks*, chọn *Main [OB1]*



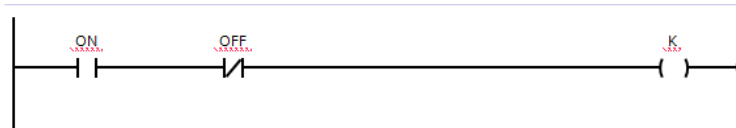
- Xuất hiện giao diện để ta viết chương trình như bên dưới.



- Để viết chương trình ta chọn kí hiệu, kéo thả ký hiệu vào vị trí cần đặt ký hiệu.
- Trong quá trình viết, ta nên đặt tên cho ký hiệu, lưu lại chương trình (nhấn Ctrl + S).
- Các vị trí đánh dấu mũi tên là những vị trí mà ta cần thao tác khi chọn các ký hiệu để viết chương trình.

*c. Gán địa chỉ của ký hiệu (thiết bị điện) với các cổng xuất nhập I/O trên CPU*

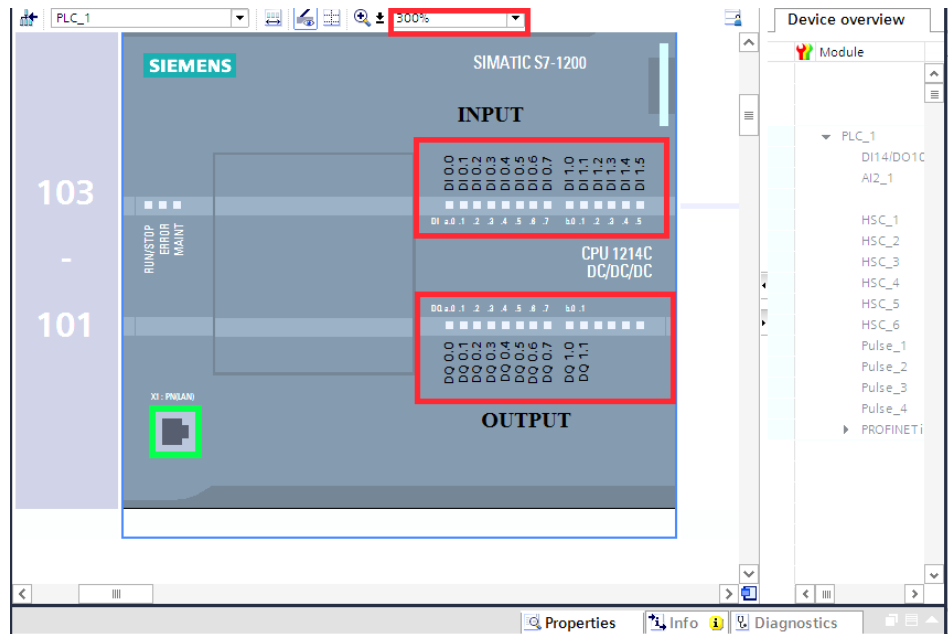
Ví dụ, ta đã viết xong chương trình, tiếp theo thực hiện gán địa chỉ cho các ký hiệu (thiết bị điện) trên sơ đồ.



- Khi chưa gán địa chỉ cho các ký hiệu sẽ có đường răng cưa màu đỏ dưới chân của các tên ký hiệu.
- Chọn tag PLC\_1 để mở cửa sổ CPU (thanh dưới cùng của cửa sổ).



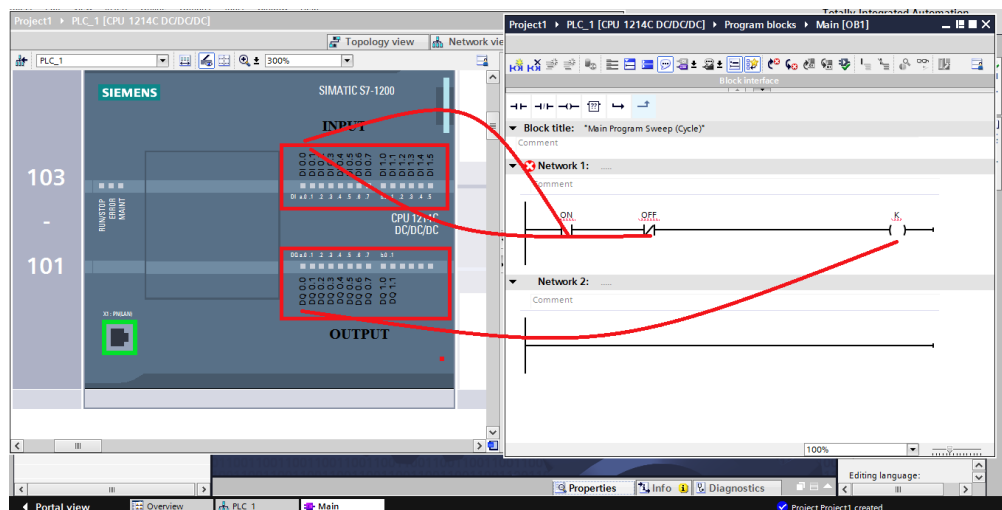
- Sau khi mở cửa sổ CPU, ta gặp giao diện sau (cần Zoom lên để CPU được lớn, tiện lợi cho quá trình gán địa chỉ).



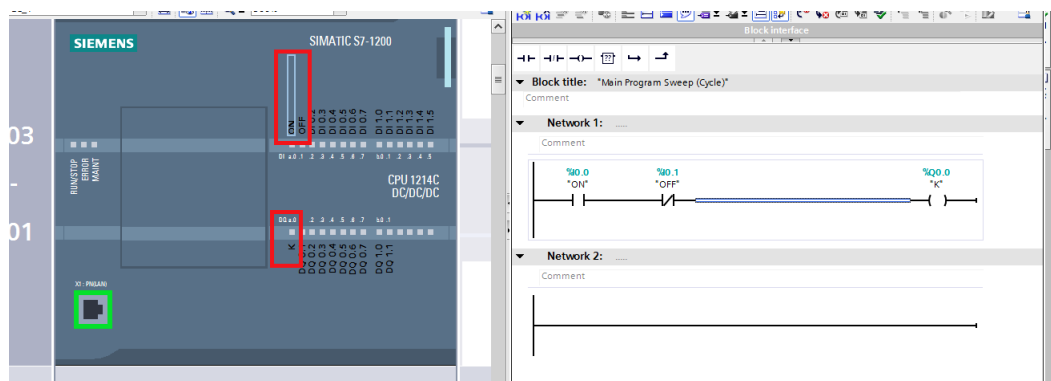
- Chọn ô vuông (như trong hình) để thu nhỏ cửa sổ lại thuận tiện cho việc kéo thả.



- Làm tương tự đối với thẻ Main để được như hình sau, khi đó ta chỉ cần kéo tên ký hiệu kết nối với các cổng INPUT, OUTPUT trên CPU là được.



- Sau khi kéo thả xong, ta được như hình dưới:



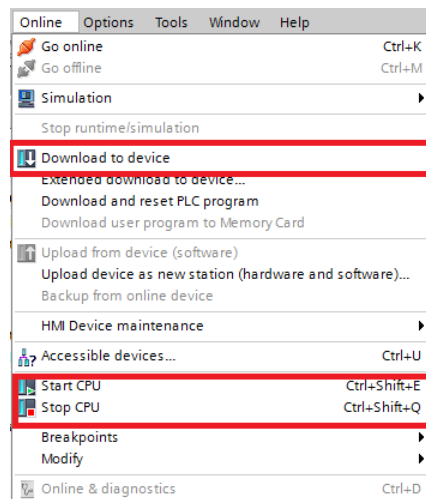
- Ta đã thực hiện xong quá trình gán địa chỉ. Tiếp theo nạp chương trình xuống PLC.

*d. Nạp chương trình xuống PLC*

- Kết nối cáp với máy tính.

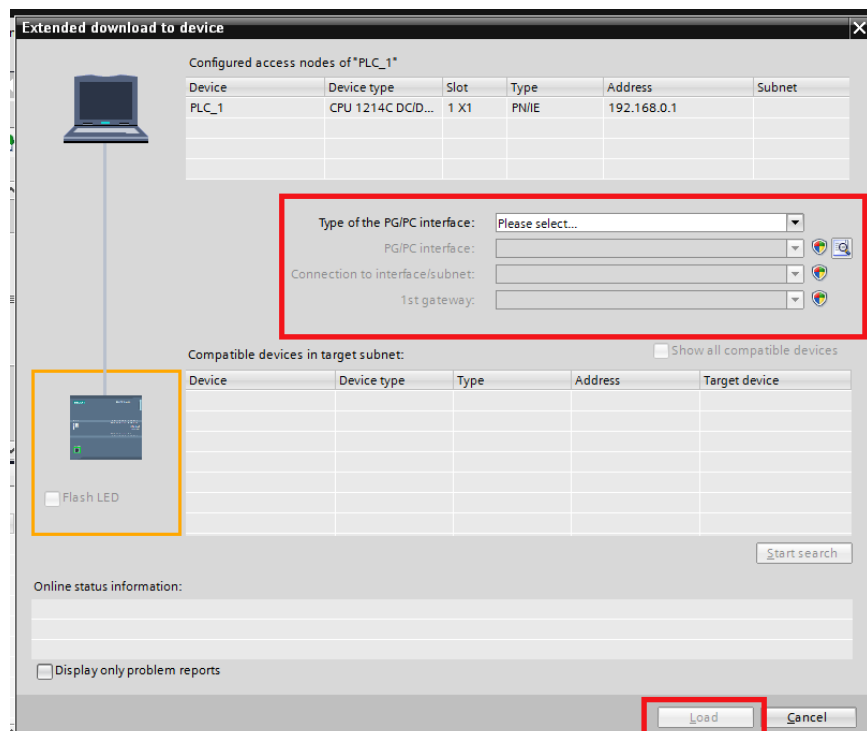


- Chọn *Online/Download* to device để nạp chương trình xuống PLC.



Nạp chương trình

- Chọn kết nối, mạng và nhấn *Load* để nạp xuống PLC.





- Nếu máy tính nhận được PLC thì ta sẽ nạp được, trường hợp không nhận thì kiểm tra lại nguồn, kết nối cáp.
- Trường hợp nhận PLC mà nạp không được thì do chương trình bị lỗi cần kiểm tra lại.

*Kết luận: 3 bước thực hiện cơ bản trên, chúng ta có thể tạo, viết và nạp chương trình xuống cho PLC.*

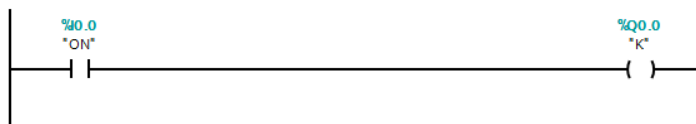
## 2.4 Tập lệnh của PLC S7 – 1200

### 2.4.1 Các lệnh về bit

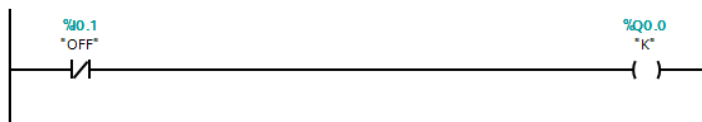
#### 2.4.1.1 Tiếp điểm

Loại	Thường hở - NO	Thường đóng - NC
Ký hiệu		
Trạng thái	Đóng: bit = 1	Đóng: bit = 0

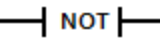
*Ví dụ 1:* Tiếp điểm NO – Khi chưa tác động vào nhấn ON (bit = 0) thì ngõ ra cuộn K ở mức thấp (hở mạch), khi tác động vào nút ON (bit = 1) thì ngõ ra cuộn K ở mức cao (kín mạch).



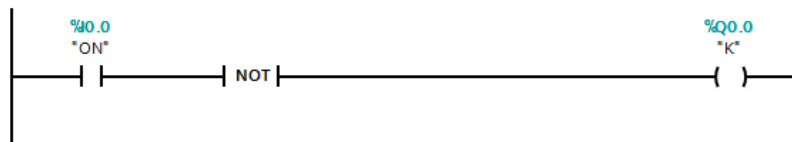
*Ví dụ 2:* Tiếp điểm NC – Khi chưa tác động vào nhấn OFF (bit = 0) thì ngõ ra cuộn K ở mức cao (kín mạch), khi tác động vào nút ON (bit = 1) thì ngõ ra cuộn K ở mức thấp (hở mạch).



#### 2.4.1.2 Lệnh đảo bit

Kí hiệu	
Trạng thái	$bit\ 1 \xrightarrow{NOT} bit\ 0$ $bit\ 0 \xrightarrow{NOT} bit\ 1$

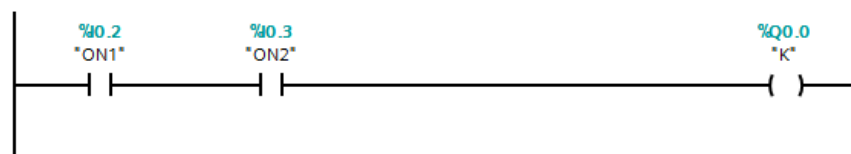
*Ví dụ 3:* Lệnh NOT – Khi chưa tác động vào nút ON (bit = 0) thì qua lệnh NOT trở thành bit = 1 thì ngõ ra cuộn K ở mức cao; khi tác động vào nút ON (bit = 1) thì qua cổng NOT trở thành bit = 0 thì ngõ ra cuộn K ở mức thấp.



### 2.4.1.3 Lệnh AND, OR, XOR

Lệnh	AND	OR	XOR
Mô tả	Gồm các tiếp điểm <i>nối tiếp</i> nhau	Gồm các tiếp điểm <i>song song</i> nhau	
Trạng thái	<ul style="list-style-type: none"> <li>- Ngõ ra = 1: khi và chỉ khi <i>tất cả</i> tiếp điểm có bit = 1</li> <li>- ngõ ra = 0: chỉ cần một tiếp điểm có bit = 0</li> </ul>	<ul style="list-style-type: none"> <li>- Ngõ ra = 1: khi và chỉ khi <i>có một</i> tiếp điểm có bit = 1</li> <li>- ngõ = 0: khi tất cả các tiếp điểm có bit = 0</li> </ul>	<ul style="list-style-type: none"> <li>- Ngõ ra = 1 khi và khi các tiếp điểm <i>cùng trạng thái bit</i> (cùng là 0 hoặc cùng là 1).</li> <li>- ngõ ra = 0: khi các bit có trạng thái khác nhau.</li> </ul>

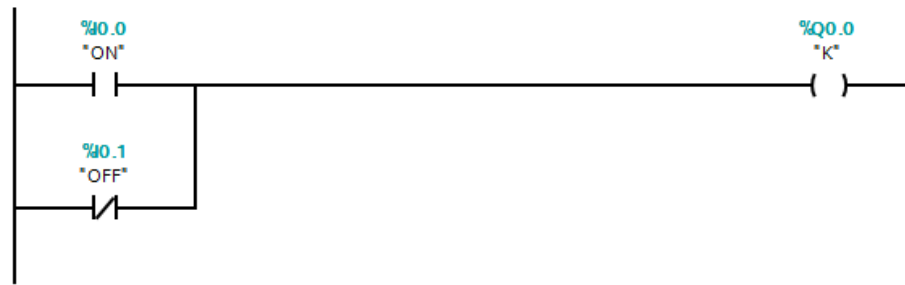
*Ví dụ 4:* Lệnh AND – ngõ ra cuộn K ở mức cao khi ON1 và ON2 cùng có bit = 1; ngõ ra cuộn K ở mức thấp khi ON1 có bit = 0 hoặc ON2 có bit = 0 (chỉ cần một tiếp điểm có bit = 0, các tiếp điểm còn lại không cần xét).



*Ví dụ 5:* Lệnh OR – ngõ ra cuộn K ở mức cao khi ON1 có bit = 1 hoặc ON2 có bit = 1 (chỉ cần 1 tiếp điểm có bit = 1, các tiếp điểm còn lại không cần xét); ngõ ra cuộn K ở mức thấp khi ON1 và ON2 cùng có bit = 0.



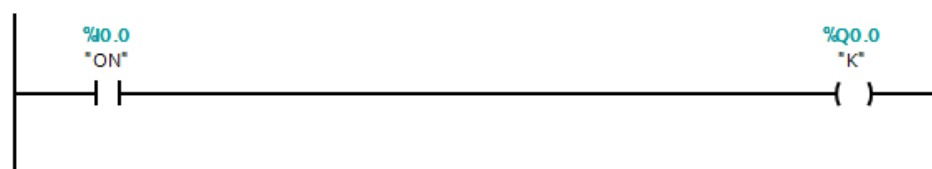
*Ví dụ 6:* Lệnh XOR – Khi chưa tác động vào I0.0 và I0.1: thì tiếp điểm NC I0.1 (bit = 0), tiếp điểm NO I0.1 chưa đóng (bit = 0): trạng thái bit của I0.0 = I0.1 = 0 nên ngõ ra cuộn K ở mức cao. Khi cùng tác động vào 2 tiếp điểm I0.0 và I0.1: thì tiếp điểm NO I0.0 có bit = 1; tiếp điểm NC I0.1 có bit = 1: trạng thái bit của I0.0 = I0.1 = 1 nên ngõ ra cuộn K ở mức cao. Ngõ ra cuộn K ở mức thấp khi ta chỉ tác động vào một tiếp điểm.



#### 2.4.1.4 Lệnh ngõ ra

Lệnh	Ngõ ra không đảo	Ngõ ra đảo
Ký hiệu		
Trạng thái	Ngõ ra ở mức cao khi ngõ vào ở mức cao (bit = 1).	Ngõ ra ở mức cao khi ngõ vào ở mức thấp (bit = 0).

Ví dụ 7: Ngõ ra không đảo – Khi chưa tác động I0.0 (bit = 0) thì ngõ ra cuộn K ở mức thấp; Khi tác động vào I0.0 (bit = 1) thì ngõ ra cuộn K ở mức cao.

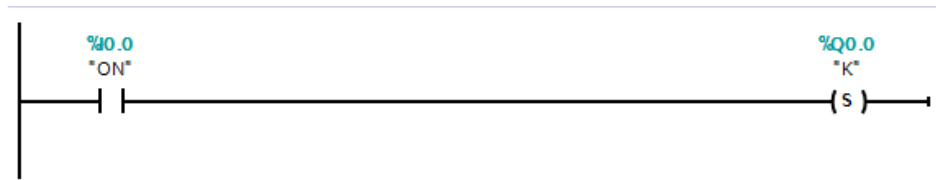


#### 2.4.1.5 Lệnh SET và lệnh RESET

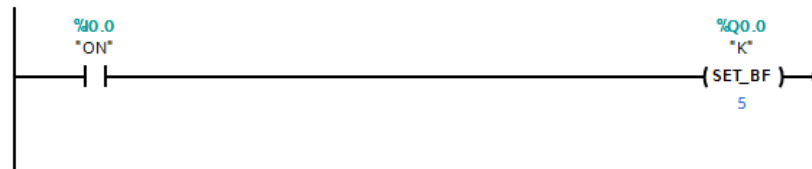
##### a. Lệnh SET

Ký hiệu SET cho 1 bit	
Trạng thái	Khi ngõ vào có bit = 1, thì ngõ ra luôn ở mức cao (cho dù hở mạch, do lệnh này có tính nhớ).
Ký hiệu SET cho một vùng bit Điền số bit cần SET vào <??>	
Trạng thái	Hoạt động giống như lệnh SET 1 vùng bit, nhưng sẽ có $n-1$ bit liên tiếp nữa được SET. ( $n$ : là số bit điền vào <??>)

Ví dụ 8: Lệnh SET cho 1 bit – Khi chưa tác động I0.0 thì ngõ ra cuộn K mức thấp (hở mạch). Khi tác động vào I0.0 thì ngõ ra ở cuộn K lên mức cao (liền mạch). Không còn tác động vào I0.0 nữa (hở mạch) nhưng ngõ ra cuộn K vẫn được giữ ở mức cao (do có tính nhớ dù hở mạch).



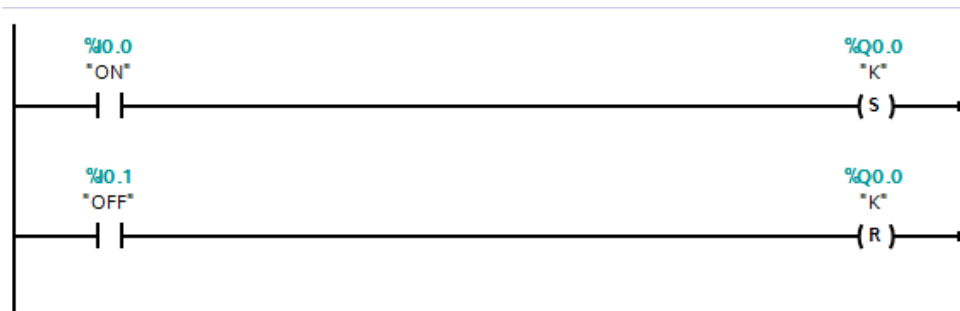
Ví dụ 9: Lệnh SET cho một vùng bit – Tương tự như ví dụ 8, nhưng sẽ có  $5-1=4$  bit liên tiếp với ngõ ra Q0.0 được SET lên mức 1: Q0.1 – Q0.4 (các ngõ ra này giống như Q0.0 có tính giữ).



#### b. Lệnh RESET

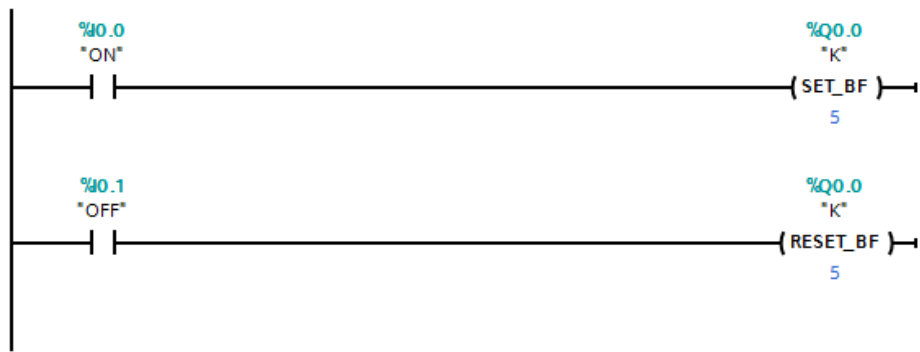
Ký hiệu RESET cho 1 bit	
Trạng thái	Khi ngõ vào có bit = 1, thì ngõ ra luôn ở mức thấp (cho dù kín mạch, do lệnh này có tính nhớ).
Ký hiệu RESET cho một vùng bit Điền số bit cần SET vào < ??? >	
Trạng thái	Hoạt động giống như lệnh RESET 1 bit, nhưng sẽ có $n-1$ bit liên tiếp nữa được RESET. ( $n$ : là số bit điền vào < ??? >)

Ví dụ 10: Lệnh RESET cho 1 bit – Ở nấc thang đầu tiên (từ trên đếm xuống) giải thích giống như ví dụ 8: ngõ ra cuộn K luôn ở mức cao (khi tác động I0.0 bit=1). Muốn cho ngõ ra cuộn K xuống mức thấp, ta sử dụng lệnh RESET. Cụ thể, khi tác động vào I0.1 bit=1 thì ngõ ra của cuộn K sẽ xuống mức thấp.



Ví dụ 11: Lệnh RESET cho một vùng bit – Tương tự khi 5bit: Q0.0 – Q0.4 được SET lên mức cao thì RESET một vùng bit sẽ đưa 5bit: Q0.0 – Q0.4 xuống mức thấp.





c. *Lệnh ưu tiên SET và RESET*

Lệnh	Ký hiệu	Mô tả bằng bảng trạng thái		
Ưu tiên SET		R	S1	Q
		0	0	Không đổi
		1	0	0
		0	1	1
		1	1	1
Mô tả	Dấu <???' là bit nhớ			

Khi cả 2 lệnh SET và RESET được gọi thì qua khối RS sẽ ưu tiên lệnh SET, ngõ ra lên mức 1.

Lệnh	Ký hiệu	Mô tả bằng bảng trạng thái		
Ưu tiên RESET		R	S1	Q
		0	0	Không đổi
		1	0	1
		0	1	0
		1	1	0
Mô tả	Dấu <???' là bit nhớ			

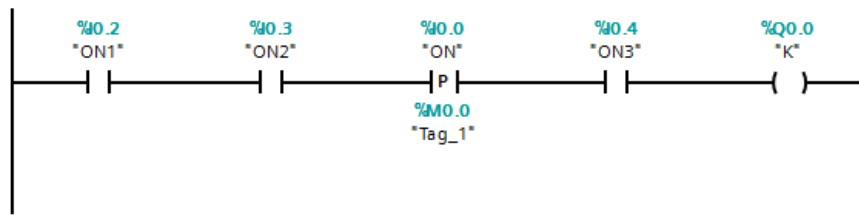
Khi cả 2 lệnh SET và RESET được gọi thì qua khối SR sẽ ưu tiên lệnh RESET, ngõ ra xuống mức 0.

#### 2.4.1.6 Lệnh cạnh lên và lệnh cạnh xuống

a. *Lệnh cạnh lên*

Ký hiệu			
Mô tả	Dấu <???' phía trên là tiếp điểm nhận xung. Dấu <???' phía dưới, là bit nhớ		
Trạng thái	Tiếp điểm của P sẽ đóng khi phát hiện có tín hiệu chuyển từ <i>mức thấp lên mức cao</i> (tại chân tiếp điểm nhận xung).		

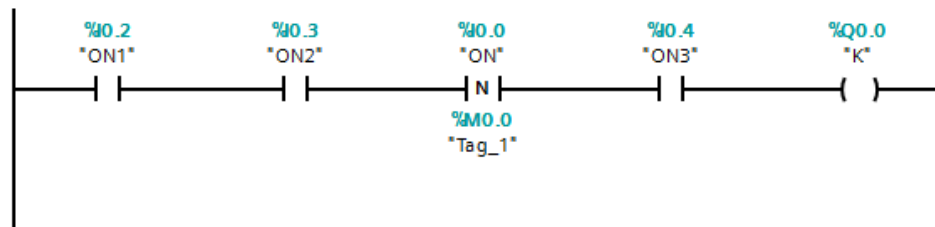
Ví dụ 12: Ngõ ra của K lên mức cao khi: I0.1, I0.2, I0.4 có  $bit = 1$  và có xung cạnh lên I0.0.



#### b. Lệnh cạnh xuống

<b>Ký hiệu</b>	
<b>Mô tả</b>	Dấu <??> phía trên là tiếp điểm nhận xung. Dấu <??> phía dưới, là bit nhớ
<b>Trạng thái</b>	Tiếp điểm của N sẽ đóng khi phát hiện có tín hiệu chuyển từ <i>mức cao xuống mức thấp</i> (tại chân tiếp điểm nhận xung).

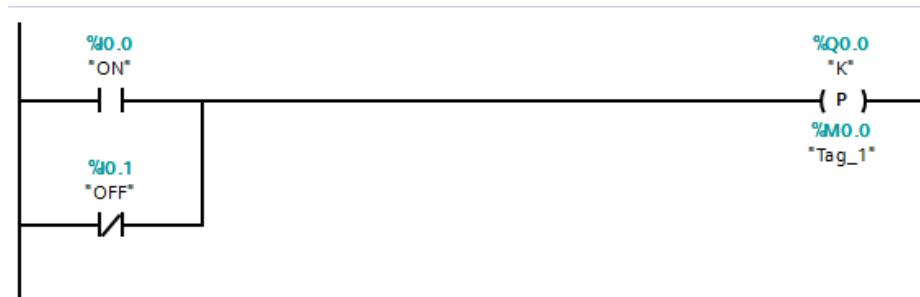
Ví dụ 13: Ngõ ra của K lên mức cao khi: I0.1, I0.2, I0.4 có  $bit = 1$  và có xung cạnh xuống I0.0.



#### c. Lệnh cạnh lên ngõ ra

<b>Ký hiệu</b>	
<b>Mô tả</b>	Dấu <??> phía trên là ngõ ra nhận xung. Dấu <??> phía dưới, là bit nhớ
<b>Trạng thái</b>	Ngõ ra P sẽ ON khi phát hiện có tín hiệu chuyển từ <i>mức thấp lên mức cao</i> (tại chân ngõ ra nhận xung).

Ví dụ 14: Chưa chưa tác động: I0.0 = 0; I0.1 = 1, suy ra ngõ vào của P là 1 (lệnh OR), ngõ ra của P vẫn là 0 (không có xung thay đổi từ 0 lên 1). Khi tác động xung vào I0.0, có sự thay đổi trạng thái từ 0 lên 1 ở ngõ vào P, nên ngõ ra của P lên mức cao.



d. *Lệnh cạnh xuống ngõ ra*

<b>Ký hiệu</b>	
<b>Mô tả</b>	Dấu <???' phía trên là ngõ ra nhận xung. Dấu >???' phía dưới, là bit nhớ
<b>Trạng thái</b>	Ngõ ra N sẽ ON khi phát hiện có tín hiệu chuyển từ <i>mức cao xuống mức thấp</i> (tại chân ngõ ra nhận xung).

Ví dụ 15: Chưa chưa tác động: I0.0 = 0; I0.1 = 1, suy ra ngõ vào của N là 1 (lệnh OR), ngõ ra của N vẫn là 0 (không có xung thay đổi từ 1 xuống mức 0). Khi tác động xung vào I0.1, có sự thay đổi trạng thái từ 1 xuống 0 ở ngõ vào N, nên ngõ ra của N lên mức cao.



d. *Lệnh cạnh lên xung CLK*

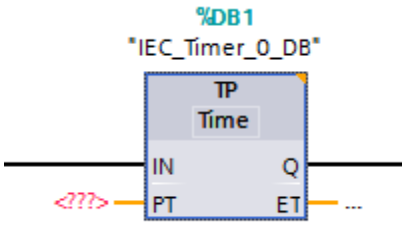
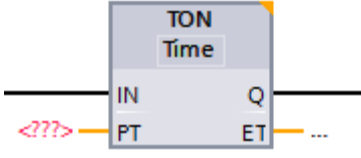
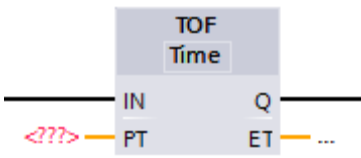
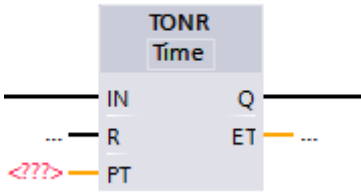
<b>Ký hiệu</b>	
<b>Mô tả</b>	Dấu <???' là bit nhớ
<b>Trạng thái</b>	Ngõ ra Q sẽ ON khi phát hiện có tín hiệu chuyển từ <i>mức thấp lên mức cao của xung CLK</i> .

e. *Lệnh cạnh xuống xung CLK*

<b>Ký hiệu</b>	
<b>Mô tả</b>	Dấu <???' là bit nhớ
<b>Trạng thái</b>	Ngõ ra Q sẽ ON khi phát hiện có tín hiệu chuyển từ <i>mức cao xuống mức thấp của xung CLK</i> .

### 2.4.2 Nhóm lệnh định thời – Timer

Được sử dụng trong chương trình để tạo thời gian trễ.

Lệnh	Ký hiệu	Mô tả
TP		<ul style="list-style-type: none"> <li>- Dấu &lt;??&gt; là thời gian trôi qua (đơn vị ms).</li> <li>- Ngõ vào IN = 1 thì ngõ ra Q là ON, đủ thời gian cài đặt PT: Q là OFF.</li> <li>- Khi IN xuống 0 mà PT chưa hết thời gian thì vẫn tiếp tục.</li> </ul>
TON		<ul style="list-style-type: none"> <li>- Dấu &lt;??&gt; là thời gian trôi qua (đơn vị ms).</li> <li>- Ngõ vào IN = 1 thì sau thời gian PT ngõ ra Q sẽ ON.</li> <li>- Khi IN xuống 0 mà chưa đủ thời gian PT thì ngõ ra Q vẫn là OFF.</li> <li>- Khi Q đang ON mà IN xuống 0 thì Q sẽ OFF.</li> </ul>
TOF		<ul style="list-style-type: none"> <li>- Dấu &lt;??&gt; là thời gian trôi qua (đơn vị ms).</li> <li>- Khi IN = 1 thì Q sẽ ON.</li> <li>- Khi IN = 0, sau thời gian PT thì Q sẽ OFF.</li> <li>- Khi IN = 0 mà chưa đủ thời gian TP thì ngõ ra Q vẫn ON.</li> </ul>
TONR		<ul style="list-style-type: none"> <li>- Trì hoãn ngõ ra ON sau một khoảng thời gian đặt trước, ngõ ra chỉ OFF khi có tín hiệu từ chân RESET.</li> <li>- Dấu &lt;??&gt; là thời gian trôi qua (đơn vị ms).</li> <li>- Ngõ vào IN kích thời gian cho Timer: IN = 1, Timer được tính.</li> <li>- Khi IN xuống 0: thời gian được tính giữ nguyên.</li> <li>- Khi IN lên 1: thời gian được tính tiếp.</li> <li>- Khi thời gian <math>ET \geq PT</math> thì ngõ ra Q sẽ ON.</li> <li>- Khi có tín hiệu RESET ngõ ra sẽ OFF và thời gian được tính lại là 0.</li> </ul>

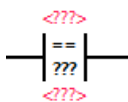
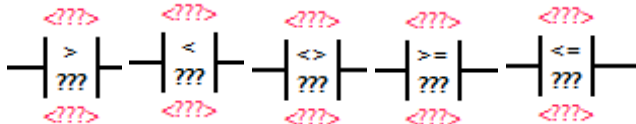
### 2.4.3 Nhóm lệnh đếm – Counter

Dùng để đếm các sự kiện bên trong chương trình hoặc bên ngoài chương trình.

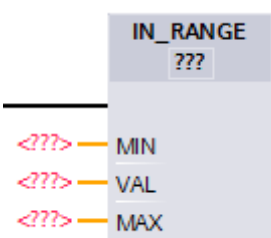
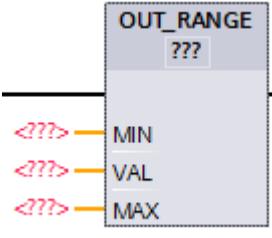
<p><b>Lệnh đếm lên</b></p>		<ul style="list-style-type: none"> <li>- Dấu &lt;??&gt; số cần đếm: bắt đầu là MW..., ví dụ MW20.</li> <li>- Khi có cạnh lên tại CU, bộ đếm tăng lên 1.</li> <li>- Khi giá trị <math>CV \geq PV</math>, ngõ ra sẽ bật lên ON.</li> <li>- Khi chân RESET được kích thì bộ đếm trở về hiện tại và Q sẽ OFF.</li> <li>- Giá trị tối đa của bộ đếm là 32767.</li> </ul>
<p><b>Lệnh đếm xuống</b></p>		<ul style="list-style-type: none"> <li>- Điền giá trị đếm vào chân PV.</li> <li>- Khi chân LD được kích cạnh lên thì PV nạp cho bộ đếm.</li> <li>- Mỗi lần có xung cạnh lên tại xung CD, bộ đếm giảm 1.</li> <li>- Khi <math>CV \leq 0</math> thì ngõ ra sẽ ON.</li> <li>- Giá trị tối đa của bộ đếm là -32767.</li> </ul>
<p><b>Lệnh đếm lên xuống</b></p>		<ul style="list-style-type: none"> <li>- Dấu &lt;??&gt; số cần đếm: bắt đầu là MW..., ví dụ MW20.</li> <li>- Mỗi lần có cạnh lên ở chân CU, giá trị tăng lên 1.</li> <li>- Mỗi lần có cạnh xuống ở chân CD, giá trị giảm xuống 1.</li> <li>- Khi <math>CV \geq PV</math>: QU sẽ ON.</li> <li>- Khi <math>CV \leq 0</math>: QD sẽ ON.</li> <li>- Khi kích vào chân R thì giá trị bộ đếm và ngõ ra sẽ là 0.</li> <li>- Khi kích vào chân LD: PV được đặt lại.</li> </ul>

## 2.4.4 Nhóm lệnh so sánh

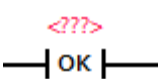
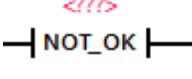
### 2.4.4.1 Lệnh so sánh byte

So sánh bằng	
Mô tả	<ul style="list-style-type: none"> <li>- Ngõ vào được điền vào 2 dấu &lt;??&gt;</li> <li>- Dấu ???: chọn kiểu dữ liệu so sánh.</li> <li>- Khi 2 ngõ vào giống nhau thì ngõ ra là 1, ngược lại là 0.</li> </ul>
Các khối so sánh khác hoạt động tương tự	

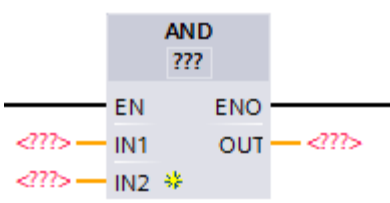
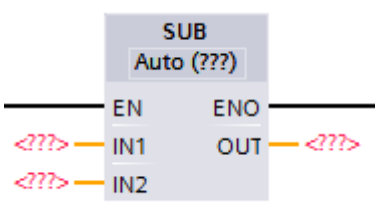
### 2.4.4.2 Lệnh In – range và Out – of – range

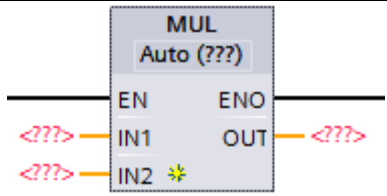
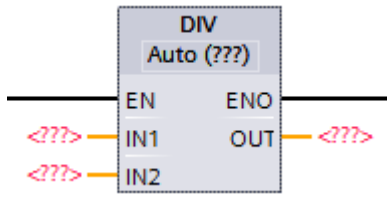
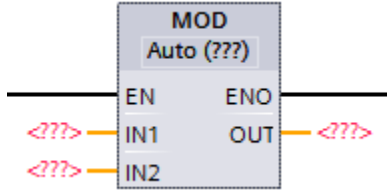
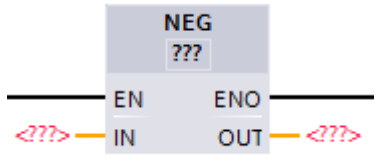
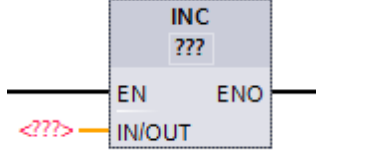
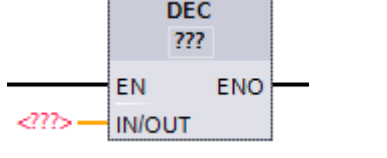
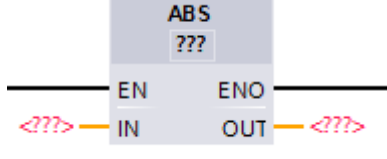
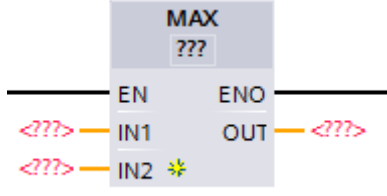
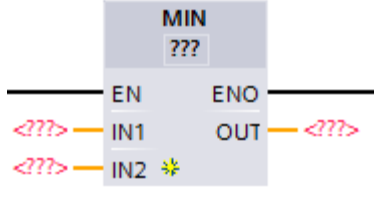
In – range	Mô tả	Out – of – range	Mô tả
	<p>Nếu giá trị <math>VAL: MIN \leq VAL \leq MAX</math></p> <p>Thì ngõ ra sẽ là ON, ngược lại là OFF.</p>		<p>Nếu giá trị <math>VAL: VAL &lt; MIN</math> hoặc <math>VAL &gt; MAX</math></p> <p>Thì ngõ ra sẽ là ON, ngược lại là OFF.</p>

### 2.4.4.3 Lệnh OK và NOT OK

Lệnh OK	Mô tả	Lệnh NOT OK	Mô tả
	<ul style="list-style-type: none"> <li>- Kiểm tra xem dữ liệu có phải là số thực hợp lệ hay không ( khác <math>\pm\infty, NaN</math>, giá trị không bình thường)</li> <li>- Nếu hợp lệ thì ngõ ra là 1, ngược lại là 0.</li> </ul>		<ul style="list-style-type: none"> <li>- Kiểm tra xem dữ liệu có phải là không số thực hợp lệ hay không ( <math>\pm\infty, NaN</math>, giá trị không bình thường)</li> <li>- Nếu không hợp lệ thì ngõ ra là 1, ngược lại là 0.</li> </ul>

## 2.4.5 Nhóm lệnh toán học

Lệnh cộng		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện cộng các giá trị ở chân IN1, IN2 (có thể mở rộng thêm nhiều chân).</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh trừ		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện trừ giá trị ở chân IN1 cho giá trị ở chân IN2.</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>

Lệnh nhân		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện nhân các giá trị ở chân IN1, IN2 (có thể mở rộng thêm nhiều chân).</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh chia		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện chia giá trị ở chân IN1 cho giá trị ở chân IN2.</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh chia dư		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện chia giá trị ở chân IN1 cho giá trị ở chân IN2.</li> <li>- Giá trị phần dư lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh phủ định		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện đảo dấu giá trị của tham số IN</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh tăng		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện tăng giá trị của một số nguyên.</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh giảm		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện giảm giá trị của một số nguyên.</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh trị tuyệt đối		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện lấy giá trị tuyệt đối của số nguyên hoặc số thực.</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh giá trị lớn nhất		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện lấy giá trị lớn nhất từ các dữ liệu chân IN1, IN2,...</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh giá trị nhỏ nhất		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện lấy giá trị nhỏ nhất từ các dữ liệu chân IN1, IN2,...</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>

Lệnh giới hạn		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện kiểm tra IN có: <math>MIN \leq IN \leq MAX</math></li> <li>- Nếu có giá trị IN được lưu vào cổng OUT.</li> <li>- Nếu IN ngoài phạm vi thì MIN hoặc MAX được lưu vào cổng OUT (giá trị gần với giá trị IN)</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh đảo		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện đảo từng bit nhị phân ở chân IN lưu vào chân OUT.</li> <li>- Nếu IN = 1 sau khi thực hiện thì ENO = 0.</li> <li>- Nếu IN = 0 sau khi thực hiện thì ENO = 1.</li> </ul>
Lệnh mã hóa		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện mã hóa mã nhị phân ở chân IN</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh giải mã		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện giải mã mã nhị phân ở chân IN</li> <li>- Giá trị lưu vào cổng OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh chọn		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện:</li> <li>+ Nếu G = 0: giá trị tại IN0 được sao chép.</li> <li>+ Nếu G = 1: giá trị tại IN1 được sao chép.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh đa hợp		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện sao chép giá trị ngõ vào đưa ra OUT.</li> <li>- Nếu vượt quá K thì giá trị ELSE sẽ đưa ra OUT</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>

#### 2.4.6 Nhóm lệnh di chuyển

Dùng để sao chép dữ liệu đến một địa chỉ bộ nhớ mới và chuyển đổi từ kiểu dữ liệu này sang kiểu dữ liệu khác.

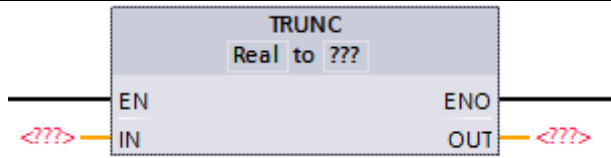
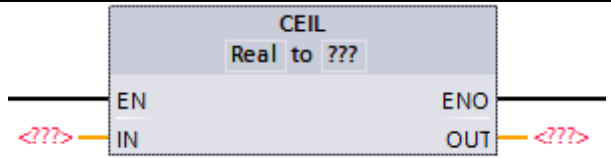
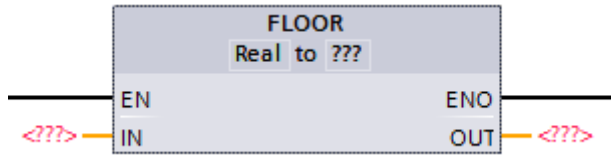
Lệnh Move		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện sao chép dữ liệu từ chân IN lưu vào chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
-----------	--	--



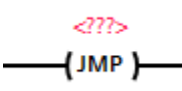
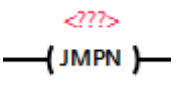
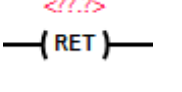
Lệnh Move Block		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện sao chép một khối dữ liệu lưu vào chân OUT.</li> <li>- COUNT: cho biết có bao nhiêu phần tử được sao chép.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Tóm tắt	Sau chép dữ liệu cơ bản, một cấu trúc dùng lệnh MOVE. Sau chép mảng hoặc cấu trúc dùng lệnh Move Block.	
Lệnh FILL_BLK/ UNFILL_BLK		
Tóm tắt	<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện sao chép dữ liệu chân IN đến dữ liệu chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>	
Lệnh SWAP		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện hoán đổi dữ liệu chân IN và lưu dữ liệu chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>

#### 2.4.7 Nhóm lệnh chuyển đổi

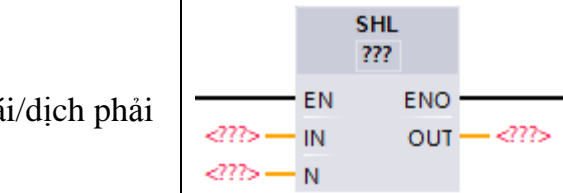
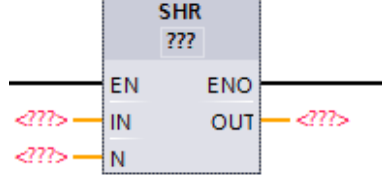
Lệnh CONV		<ul style="list-style-type: none"> <li>- Ký hiệu ??? to ??? là kiểu dữ liệu cần chuyển đổi.</li> <li>- Khi EN được kích, sẽ thực hiện chuyển đổi kiểu dữ liệu chân IN và lưu dữ liệu chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh SCALE_X		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện:  <math>OUT = VALUE(MAX - MIN) + MIN</math> lưu vào chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh NORM_X		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện:  <math>OUT = \frac{VALUE - MIN}{MAX - MIN}</math> lưu vào chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh ROUND		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện chuyển đổi kiểu dữ liệu số thực ở chân IN làm tròn và lưu dữ liệu chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>

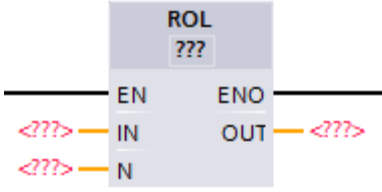
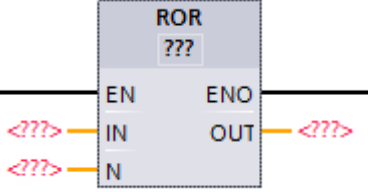
Lệnh TRUNC		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện chuyển đổi kiểu dữ liệu số thực ở chân IN làm tròn bỏ phần thập phân và lưu dữ liệu chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh CEIL		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện chuyển đổi kiểu dữ liệu số thực ở chân IN làm tròn lên và lưu dữ liệu chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>
Lệnh FLOOR		<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện chuyển đổi kiểu dữ liệu số thực ở chân IN làm tròn xuống và lưu dữ liệu chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>

#### 2.4.8 Nhóm lệnh điều khiển chương trình

Lệnh JMP		<ul style="list-style-type: none"> <li>- Dấu &lt;???: tên nhãn cần nhảy đến.</li> <li>- Ngõ vào mức 1: sẽ thực nhảy đến nhãn quy định</li> </ul>
Lệnh JMPN		<ul style="list-style-type: none"> <li>- Dấu &lt;???: tên nhãn cần nhảy đến.</li> <li>- Ngõ vào mức 0: sẽ thực nhảy đến nhãn quy định</li> </ul>
Lệnh RET		Lệnh RET được sử dụng để kết thúc một đoạn chương trình được gọi và quay trở lại ngay vị trí lệnh gọi chương trình con để thực hiện tiếp các lệnh kế tiếp.

#### 2.4.9 Nhóm lệnh dịch và quay

Lệnh dịch trái/dịch phải		
Mô tả	<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện dịch N bit từ chân IN.</li> <li>- Kết quả lưu vào chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>	

Lệnh quay trái/ quay phải	 
Mô tả	<ul style="list-style-type: none"> <li>- Khi EN được kích, sẽ thực hiện quay N bit từ chân IN.</li> <li>- Kết quả lưu vào chân OUT.</li> <li>- Sau khi thực hiện xong: ENO lên 1.</li> </ul>

#### 2.4.10 Lệnh vòng lặp PID

#### 2.4.11 Lệnh đếm xung tốc độ cao

Trong phạm vi đề tài, nhóm em không tìm hiểu về 2 nhóm lệnh này.

## **Chương 3 ỨNG DỤNG PLC ĐIỀU KHIỂN TRÌNH TỰ CỦA MỘT NHÓM ĐỘNG CƠ**

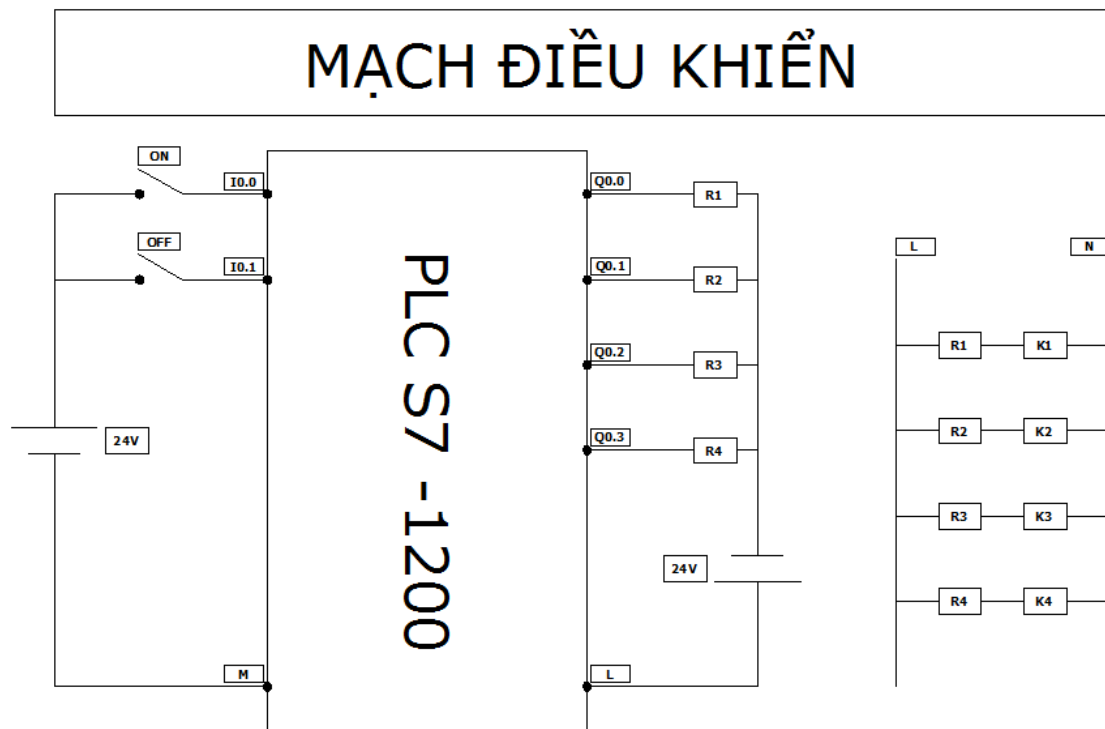
### **3.1 Yêu cầu**

Sử dụng bộ điều khiển PLC, điều khiển nhóm động cơ gồm 4 động cơ mở máy theo trình tự sau:

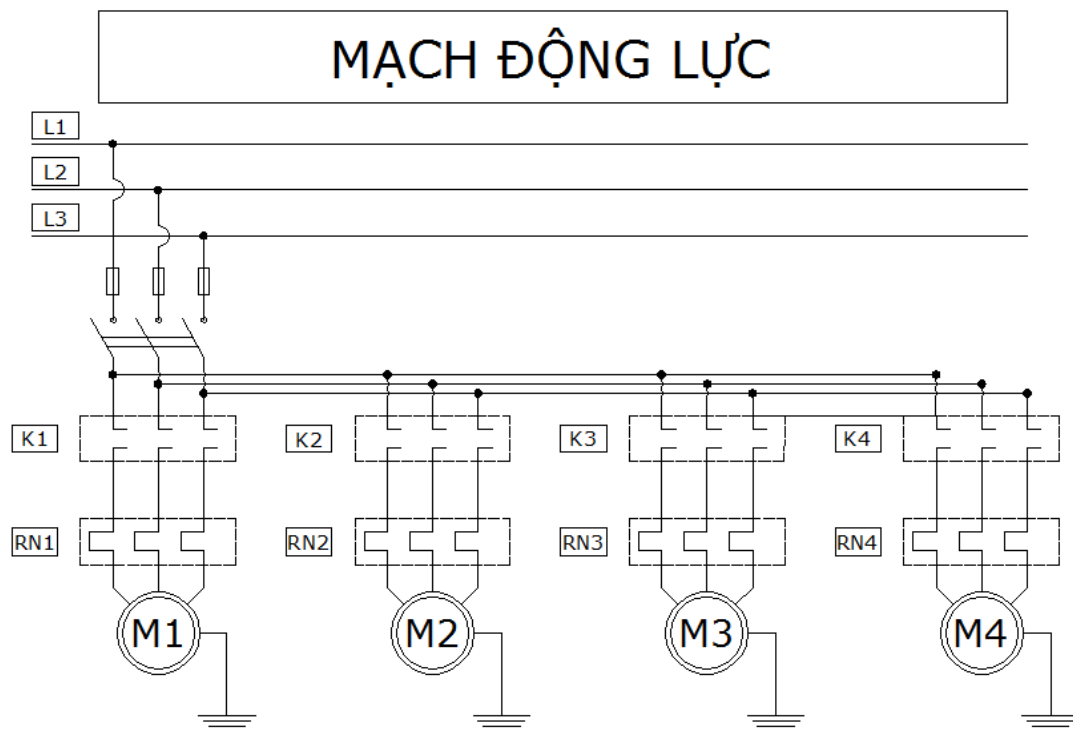
- Động cơ 1 chạy trong 10s.
- Động cơ 2 chạy trong 10s (Động cơ 1 và 2 cùng chạy).
- Động cơ 3 chạy trong 10s (Động cơ 1, 2 và 3 cùng chạy).
- Động cơ 4 chạy trong 10s (Động cơ 1, 2, 3 và 4 cùng chạy).
- Dừng lại trong 5s.
- Động cơ 1 chạy trong 10s.
- Động cơ 2 chạy trong 10s (Động cơ 1 dừng).
- Động cơ 3 chạy trong 10s (Động cơ 2 dừng).
- Động cơ 4 chạy trong 10s (Động cơ 3 dừng).
- Lặp lại quá trình trên.

Chúng em trình bày sơ đồ đồ khiển, mạch động lực và chương trình LAD ở phần sau.

### 3.2 Mạch điều khiển

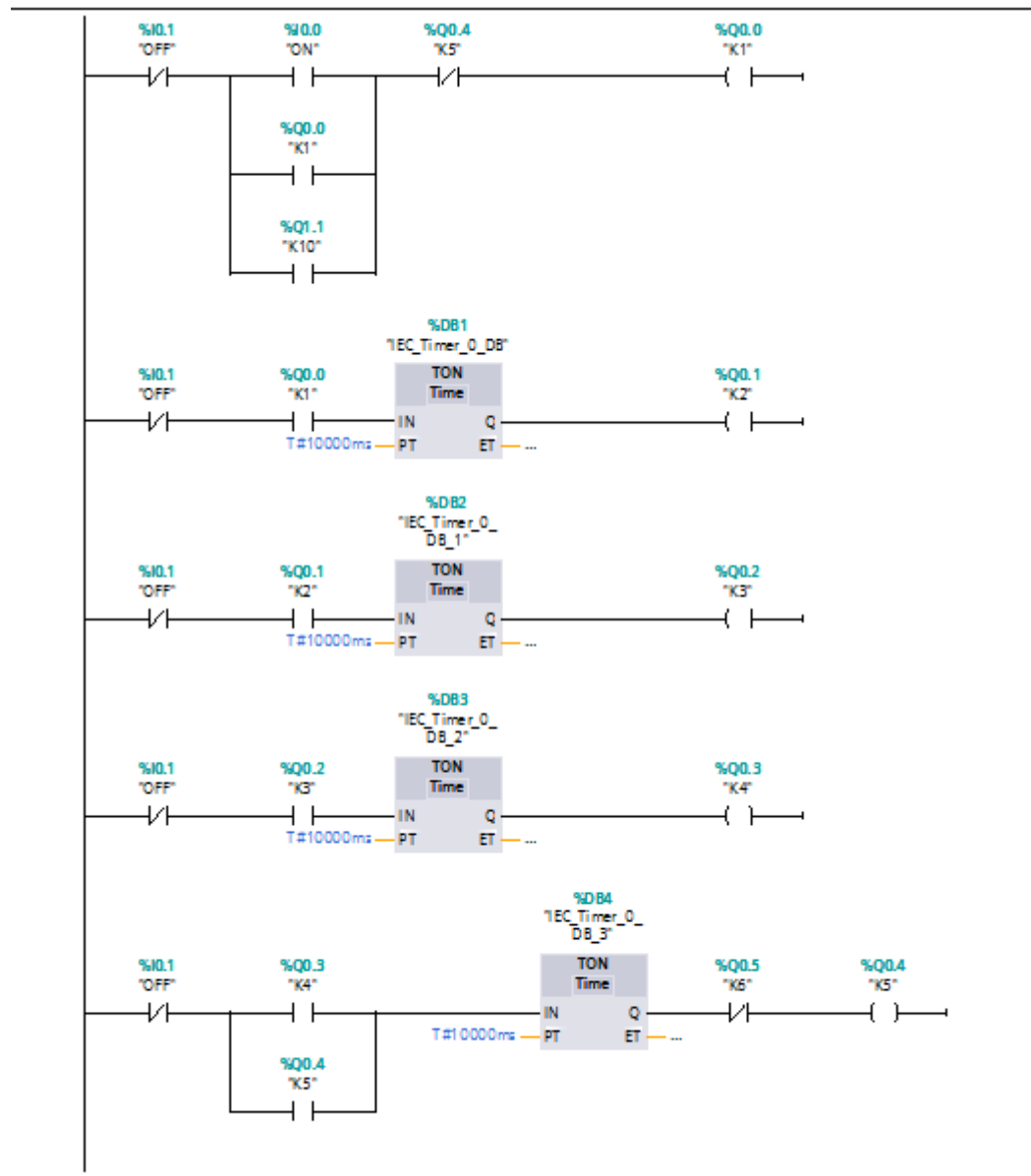


### 3.3 Mạch động lực



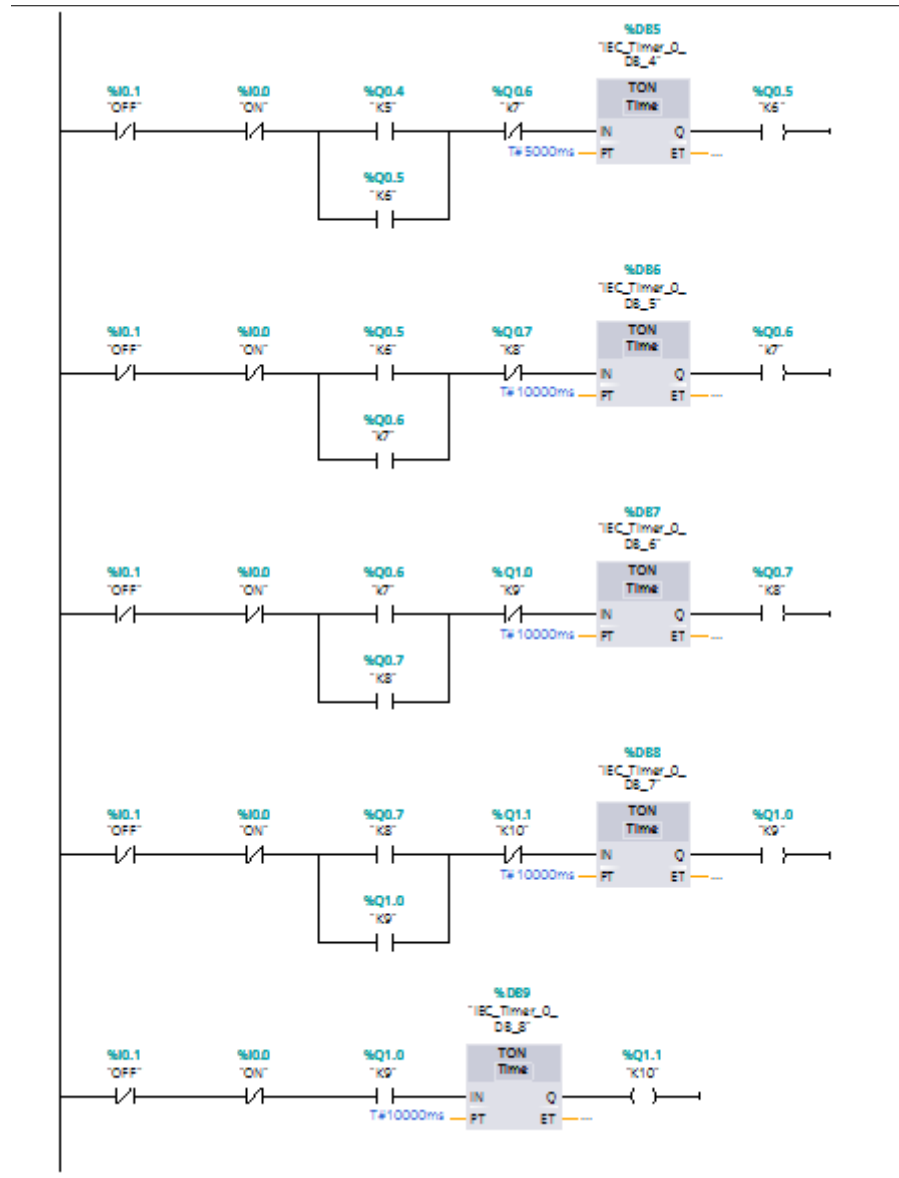
### 3.4 Chương trình viết bằng ngôn ngữ LAD

*Network 1: Khởi động tăng dần số động cơ*



Symbol	Address	Type
"K1"	%Q0.0	Bool
"K2"	%Q0.1	Bool
"K3"	%Q0.2	Bool
"K4"	%Q0.3	Bool
"K5"	%Q0.4	Bool
"K6"	%Q0.5	Bool
"K10"	%Q1.1	Bool
"OFF"	%I0.1	Bool
"ON"	%I0.0	Bool

## Network 2: Khởi động tuần tự động cơ



Symbol	Address	Type
"K5"	%Q0.4	Bool
"K6"	%Q0.5	Bool
"K7"	%Q0.6	Bool
"K8"	%Q0.7	Bool
"K9"	%Q1.0	Bool
"K10"	%Q1.1	Bool
"OFF"	%I0.1	Bool
"ON"	%I0.0	Bool

## **Chương 4 KẾT QUẢ VÀ KIẾN NGHỊ**

### **4.1 Kết quả**

Qua quá trình thực hiện đồ án, chúng em học được cách lập trình điều khiển cho bộ điều khiển PLC S7 – 1200, cách kết nối mạch điều khiển, mạch động lực để đưa vào chạy thực tế. Kết quả cụ thể là ứng dụng bộ điều khiển PLC vào điều khiển trình tự khởi động của động cơ.

Bộ điều khiển PLC là một chủ đề mới đối với chúng em, nên chương trình viết ra còn hạn chế chưa tối ưu: chỉ điều khiển 4 động cơ mà sử dụng hết 10 cổng Output của bộ điều khiển.

Chúng em khai thác sử dụng được nhiều chức năng của PLC, chỉ mới khai thác ở mức độ cơ bản, sử dụng được chức năng điều khiển qua các chân xuất nhập số, còn các lệnh nâng cao và một số tính năng như điều khiển với cảm biến có tính hiệu số hoặc tính hiệu liên tục (tính hiệu analog), kết hợp PLC với biến tần, màn hình HMI,... thì chúng em chưa khai thác được.

### **4.2 Kiến nghị**

Đề tài ứng dụng bộ điều PLC có thể nói là mới với sinh viên đại học năm 3 như chúng em, một số bạn có điều kiện thì có đi học ở bên ngoài, còn phần lớn là không được học. Nếu được em xin đưa ra đề xuất là trường mình có thể mở một học phần tự chọn về học lập trình PLC cho những bạn yêu thích về điều khiển trong công nghiệp có dịp được tiếp xúc với nó.

Riêng về chúng em, nhóm thực hiện đề tài, đã có được kiến thức cơ bản về lập trình PLC, mong thầy cho cho chúng em cơ hội được làm tiếp đồ án sau này về ứng dụng PLC để có thể tiếp tục nghiên cứu và khai thác bộ điều khiển PLC với tập lệnh và chức năng ở mức độ cao để có thể ứng dụng nó trong công việc sau này, để trở có thể trở thành người kỹ sư lành nghề, giúp ích cho đất nước.



## **TÀI LIỆU THAM KHẢO**

1. Tài liệu hướng dẫn PLC S7 – 1200 của Seimens  
[http://tudonghoatre.com/tai-lieu-plc-siemens-s7\\_1200-a47i46.html](http://tudonghoatre.com/tai-lieu-plc-siemens-s7_1200-a47i46.html)
2. Phần Help trong phần mềm TIAPortal V13.
3. Tác giả: Tạ Minh Liên – Phan Thành Năm – Luận văn tốt nghiệp Lập trình ứng dụng S7 – 1200 – Trường Đại Học Cần Thơ, Năm 2010.
4. Tác giả: Nguyễn Thị Hồng Cẩm – Phạm Tiến Huy – Luận văn tốt nghiệp – Lập trình ứng dụng S7 – 1200 - Trường Đại Học Cần Thơ, Năm 2011.
5. Các tài liệu hướng dẫn về PLC S7 – 1200 của [tailieudieukhientudonghoa.com](http://www.tailieudieukhientudonghoa.com)  
<http://www.tailieudieukhientudonghoa.com/2015/11/tai-lieu-lap-trinh-plc-siemens-s7-1200.html>
6. Kênh Youtube của tác giả Nguyễn Thanh Dân về lập trình PLC S7 – 200  
<https://www.youtube.com/playlist?list=PLhFjtzzUovr-catsm6X4IEgvzhR1hniQ1>