BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN

TRƯỜNG CAO ĐẲNG CƠ ĐIỆN XÂY DỰNG VIỆT XÔ

KHOA: ĐIỆN – ĐIỆN TỰ ĐỘNG HÓA

GIÁO TRÌNH

MÔ ĐUN: PLC CƠ BẢN

NGHỀ: ĐIỆN CÔNG NGHIỆP

TRÌNH ĐỘ TRUNG CẤP

Ban hành kèm theo Quyết định số: ..../QĐ ngày .....tháng ....năm 201..... .....của .....

Ninh Bình, 2019

TUYÊN BỐ BẢN QUYỀN:

LỜI GIỚI THIỆU

PLC cơ bản là một trong những mô đun chuyên môn mang tính đặc trưng cao thuộc nghề Điện công nghiệp. Sau khi học mô đun này, học viên có đủ kiến thức và kỹ năng để học tập tiếp các mô đun PLC nâng cao, Chuyên đề điều khiển lập trình cỡ nhỏ ...

Giáo trình PLC cơ bản được thiết kế theo mô đun thuộc hệ thống mô đun/môn học của chương trình đào tạo nghề Điện công nghiệp cho cấp trình độ Cao đẳng. Ngoài ra, giáo trình cũng có thể được sử dụng cho đào tạo ngắn hạn hoặc làm tài liệu tham khảo cho các công nhân kỹ thuật, các nhà quản lý và người sử dụng nhân lực tham khảo. Giáo trình mô đun này được triển khai sau các mô đun Kỹ thuật số; Lập trình vi điều khiển; Kỹ thuật cảm biến. Mô đun cung cấp những kiến thức cơ bản về ngôn ngữ lập trình trong PLC cũng như trang bị những kỹ năng về lắp đặt các bộ điều khiển lập trình và kỹ năng lập trình giải quyết các bài toán điều khiển cỡ nhỏ.

Mô đun được thiết kế gồm 6 bài:

Bài mở đầu: Giới thiệu chung về PLC và bài toán điều khiển

Bài 1: Đại cương về điều khiển lập trình

Bài 2: Các phép toán nhị phân của PLC

Bài 3: Các phép toán số của PLC

Bài 4: Xử lý tín hiệu analog

Bài 5: PLC của các hãng khác

Bài 6: Lắp đặt mô hình điều khiển bằng PLC

Trong quá trình biên soạn, do thời gian, kinh nghiệm và trình độ có hạn nên khó tránh thiếu sót, mong các thầy cô cũng như các độc giả nhận xét, đánh giá, bổ xung để tài liệu ngày một hoàn chỉnh hơn

Xin chân thành cảm ơn!

Ninh Bình, ngày tháng năm 20

Tham gia biên soạn

Trần Thị Thảo – Chủ biên

Mục lục

MÔ ĐUN: PLC CƠ BẢN
Mã môn học: MĐ21

Vị trí, tính chất, ý nghĩa và vai trò của mô đun:

- Vị trí: Mô đun được bố trí dạy cuối chương trình sau khi học xong các môn chuyên môn như điện tử công suất, trang bị điện....

- Tính chất: Là mô đun bắt buộc

Mục tiêu của môn học:

- Về kiến thức:

+ Trình bày được nguyên lý hệ điều khiển lập trình PLC; So sánh các ưu nhược điểm với bộ điều khiển có tiếp điểm và các bộ lập trình cỡ nhỏ khác.

+ Phân tích được cấu tạo phần cứng và nguyên tắc hoạt động của phần mềm trong hệ điều khiển lập trình PLC.

+ Trình bày được phương pháp kết nối dây giữa PC - CPU và thiết bị ngoại vi.

- Về kỹ năng:

+ Thực hiện được một số bài toán ứng dụng đơn giản trong công nghiệp.

+ Kết nối thành thạo phần cứng của PLC - PC với thiết bị ngoại vi.

+ Viết được chương trình một số bài toán ứng dụng đơn giản trong công nghiệp.

+ Phân tích được một số chương trình đơn giản, phát hiện lỗi và sửa chữa khắc phục.

- Về năng lực tự chủ và chịu trách nhiệm:

Phát huy tính tích cực, chủ động, sáng tạo, tác phong công nghiệp.

Nội dung của mô đun:

Số TT	Tên các bài trong mô đun	Thời gian (giờ)			
		Tổng số	Trong đó		
			Lý thuyết	Thực hành/ thực tập/thí nghiệm/ bài tập/thảo luận	Kiểm tra
	Bài mở đầu	**2**	**2**		
	Giới thiệu chung về PLC và bài toán điều khiển				
	1. Giới thiệu chung về PLC	1	1		
	2. Bài toán điều khiển	1	1		
	Bài 1: Đại cương về điều khiển lập trình	**4**	**3**	**1**	
	1.Tổng quát về một PLC	1	1		
	2. Kết nối dây giữa PLC và các thiết bị ngoại vi	2	1		

	3. Cài đặt và sử dụng phần mềm Step7-MicroWin	1	1	1	
	Bài 2: Các phép toán nhị phân của PLC	**22**	**10**	**10**	**2**
	1. Các liên kết logic	1	1		
	2. Các lệnh ghi/xóa giá trị cho tiếp điểm	1	1		
	3. Timer	4	2	2	
	4. Couter	4	1	3	
	5. Bài tập ứng dụng	10	4	6	
	Kiểm tra	2			2
	Bài 3: Các phép toán số của PLC	**22**	**9**	**11**	**2**
	1. Chức năng truyền dẫn	3	1	2	
	2. Chức năng so sánh	5	2	3	
	3. Chức năng chuyển đổi (Converter)	4	2	2	
	4. Chức năng toán học	4	2	2	
	5. Đồng hồ thời gian thực	4	2	2	
	Kiểm tra định kỳ	2			2
	Bài 4: Bộ xử lý tín hiệu Analog	**4**	**3**	**1**	
	1. Tín hiệu Analog	1	1		
	2. Biểu diễn các giá trị Analog	0.5	0.5		
	3. Kết nối ngõ vào-ra Analog	1	0.5	0.5	
	4. Hiệu chỉnh tín hiệu Analog	0.5	0.5		
	5. Giới thiệu về module analog PLC S7-200	1	0.5	0.5	
	Bài 5: PLC của các hãng khác	**12**	**4**	**8**	
	1. PLC của hãng Omron	8	3	5	
	2. PLC của hãng Siemens(PLC S7-300)	4	1	3	
	Bài 6: Lập trình điều khiển bằng PLC	**54**	**9**	**41**	**4**
	1. Lập trình điều khiển động cơ có đảo chiều quay	4	1	3	
	2. Lập trình điều khiển hệ thống cân và cấp liệu	6	1	5	
	3. Lập trình điều khiển đếm sản phẩm	6	1	5	
	4. Lập trình điều khiển đèn giao thông	8	1	7	
	5. Lập trình điều khiển xe chuyển nhiên liệu	6	1	5	
	Kiểm tra định kỳ	2			2

	6. Lập trình điều khiển trộn liệu	6	1	5	
	7. Lập trình điều khiển cầu trục	10	2	8	
	8. Lập trình điều khiển hệ thống nâng hàng	4	1	3	
	Kiểm tra định kỳ	**2**			**2**
Cộng		**120**	**40**	**72**	**8**

BÀI MỞ ĐẦU: GIỚI THIỆU CHUNG VỀ PLC VÀ BÀI TOÁN ĐIỀU KHIỂN

Giới thiệu:

Ngày nay khoa học kỹ thuật ngày càng phát triển. Trong các xí nghiệp hiện nay có nhiều hệ thống sản xuất sử dụng các bộ điều khiển lập trình. Trên thế giới có nhiều hãng sản xuất các bộ điều khiển lập trình khác nhau như: Siemens, Omron, Telemecanique, Allen Bredlay,… Về cơ bản, chúng đều có các tính năng tương tự, do đó, trong tài liệu này chỉ đề cập đến một loại PLC khá thông dụng và được dùng nhiều ở Việt Nam. Modul kỹ thuật điều khiển lập trình cơ bản (PLC cơ bản) là một modul chuyên môn của học viên ngành sửa chữa thiết bị điện công nghiệp. Modul này nhằm trang bị cho học viên các trường công nhân kỹ thuật, trung cấp và cao đẳng, các trung tâm dạy nghề những kiến thức về lĩnh vực điều khiển lập trình, với kiến thức này, học viên có thể áp dụng trực tiếp vào lĩnh vực sản xuất cũng như đời sống. Modul này cũng có thể làm tài liệu tham khảo cho các cán bộ kỹ thuật, các học viên của các ngành khác quan tâm đến lĩnh vực này.

Mục tiêu:

- Trình bày được khái niệm và đặc điểm của PLC.
- Phân tích được các dạng bài toán điều khiển và giải bài toán điều khiển.
- Rèn luyện đức tính tích cực, chủ động và sáng tạo.

Nội dung chính:

1. Giới thiệu chung về PLC:

Trong mấy năm trở lại đây ngành tự động hóa (TĐH) đã góp phần chứng tỏ được vai trò, vị thế của mình và bắt đầu đi vào cuộc sống, đặc biệt là trong các lĩnh vực sản xuất công nghiệp như: điều khiển các nhà máy thủy điện, nhiệt điện, các nhà máy chế biến lọc dầu, các nhà máy hóa chất.

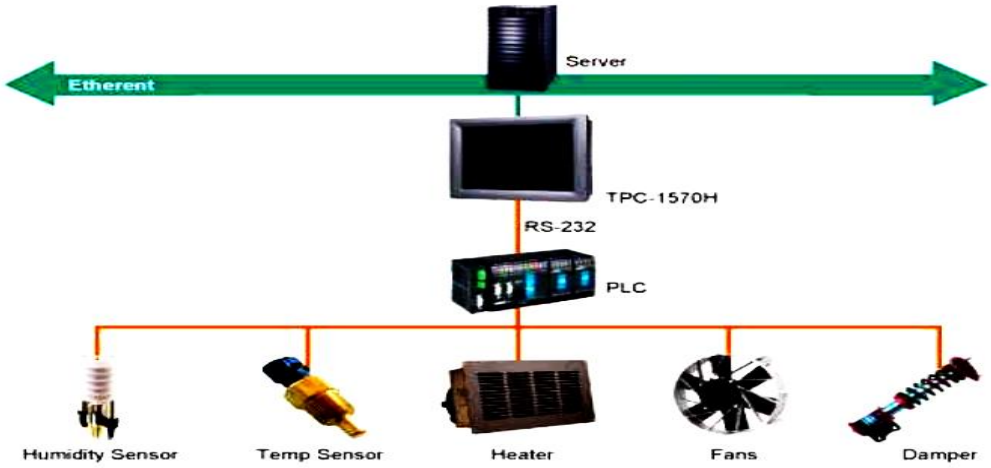
Ngoài ra, TĐH còn được áp dụng trong hầu hết các dây chuyền sản xuất tự động, cụ thể là trong sản xuất công nghiệp nhẹ; công nghiệp tàu thủy; công nghiệp chế tạo lắp ráp ô tô, xe máy; khai thác khoáng sản và luyện kim; chế tạo máy; lĩnh vực y tế và chăm sóc sức khỏe cộng đồng…

Cùng với sự phát triển của ngành điện - điện tử - tin học, "Tự động hóa trong công nghiệp" ngày nay đã đóng góp một phần khá quan trọng trong nền kinh tế Việt Nam. Với sự xuất hiện của nhiều tập đoàn tên tuổi trong lĩnh vực điện, điện tử, tự động đã làm cho thị trường thiết bị tự động ngày càng trở nên đa dạng.

Bộ điều khiển lập trình PLC (Programmable Logic Controller) được sáng tạo từ những ý tưởng ban đầu của một nhóm kỹ sư thuộc hãng General Motor vào năm 1968. Trong những năm gần đây, bộ điều khiển lập trình được sử dụng ngày càng rộng rãi trong công nghiệp của nước ta như là 1 giải pháp lý tưởng cho việc

tự động hóa các quá trình sản xuất. Cùng với sự phát triển công nghệ máy tính đến hiện nay, bộ điều khiển lập trình đạt được những ưu thế cơ bản trong ứng dụng điều khiển công nghiệp.

Như vậy, PLC là 1 máy tính thu nhỏ nhưng với các tiêu chuẩn công nghiệp cao và khả năng lập trình logic mạnh. PLC là đầu não quan trọng và linh hoạt trong điều khiển tự động hóa.



2. Bài toán điều khiển:

Bài toán điều khiển dùng rơle điện:

Sự bắt đầu về cuộc cách mạng khoa học kỹ thuật đặc biệt vào những năm 60 và 70, những máy móc tự động được điều khiển bằng những rơle điện từ như các bộ định thời, tiếp điểm, bộ đếm, relay điện từ. Những thiết bị này được liên kết với nhau để trở thành một hệ thống hoàn chỉnh bằng vô số các dây điện bố trí chằng chịt bên trong panel điện (tủ điều khiển).

Như vậy, với 1 hệ thống có nhiều trạm làm việc và nhiều tín hiệu vào/ra thì tủ điều khiển rất lớn. Điều đó dẫn đến hệ thống cồng kềnh, sửa chữa khi hư hỏng rất phức tạp và khó khăn. Hơn nữa, các rơle tiếp điểm nếu có sự thay đổi yêu cầu điều khiển thì bắt buộc thiết kế lại từ đầu.

Trong hệ điều khiển bằng rơ le các thiết bị trong hệ thống được chia thành 3 khối cơ bản sau:

- Khối các phần tử đầu vào bao gồm các công tắc, công tắc hành trình, nút ấn, cảm biến…
- Khối điều khiển bao gồm rơ le, cuộn hút, công tắc tơ, rơ le thời gian, bộ đếm….
- Khối đầu ra bao gồm động cơ điện, van điện từ, bộ gia nhiệt, bộ hiển thị….

Cả ba khối trên cũng được kết nối với nhau theo các sơ đồ điều khiển nhất định nhằm điều khiển các thiết bị của khối đầu ra hoạt động theo một yêu cầu nào đó.

- **Bài toán điều khiển dùng PLC:**

Trong hệ điều khiển bằng PLC các thiết bị trong hệ thống cũng được chia thành 3 khối cơ bản sau:

- Khối các phần tử đầu vào bao gồm các công tắc, công tắc hành trình, nút ấn, cảm biến…
- Khối điều khiển là một bộ điều khiển bằng PLC.
- Khối đầu ra bao gồm động cơ điện, van điện từ, bộ gia nhiệt, bộ hiển thị….

Cả ba khối trên cũng được kết nối với nhau theo các sơ đồ điều khiển nào đó căn cứ vào chương trình điều khiển được lập trình bằng PLC nhằm điều khiển các thiết bị của khối đầu ra hoạt động theo một yêu cầu nào đó.

BÀI 1: ĐẠI CƯƠNG VỀ ĐIỀU KHIỂN LẬP TRÌNH

Mã bài: MĐ21.01

Giới thiệu: Thiết bị điều khiển logic lập trình được PLC là dạng thiết bị điều khiển đặc biệt dựa trên bộ vi xử lý, sử dụng bộ nhớ lập trình được để lưu trữ các lệnh và thực hiện các chức năng, chẳng hạn tính logic, lập chuỗi, định giờ, đếm, và các thuật toán để điều khiển máy và các quá trình công nghệ. PLC được thiết kế cho các kỹ sư, không yêu cầu cao về kiến thức máy tính và ngôn ngữ máy tính, có thể vận hành. Chúng được thiết kế cho các nhà kỹ thuật có thể cài đặt hoặc thay đổi chương trình. Vì vậy, các nhà thiết kế PLC phải lập trình sẵn sao cho chương trình điều khiển có thể nhập bằng cách sử dụng ngôn ngữ đơn giản (ngôn ngữ điều khiển).

Mục tiêu:

- Trình bày được các ưu điểm của điều khiển lập trình so với các loại điều khiển khác và các ứng dụng của chúng trong thực tế.
- Trình bày được cấu trúc và nhiệm vụ các khối chức năng của PLC.
- Thực hiện được sự kết nối giữa PLC và các thiết bị ngoại vi.
- Lắp đặt được các thiết bị bảo vệ cho PLC theo yêu cầu kỹ thuật.
- Rèn luyện tính tỉ mỉ, cẩn thận trong công việc

Nội dung chính:

1. Tổng quát về một PLC

PLC là loại thiết bị cho phép thực hện linh hoạt các thuật toán điều khiển số thông qua các ngôn ngữ lập trình, thay cho việc phải thực hiện thuật toán đó bằng mạch số. Như vậy, với chương trình này, PLC trở thành một bộ điều khiển số nhỏ gọn, dễ thay đổi thuật toán, và đặc biệt, dễ trao đổi thông tin với môi trường xung quanh (với các PLC, với máy tính, hoặc các thiết bị ngoại vi khác...)

Toàn bộ chương trình điều khiển được lưu nhớ trong bộ nhớ của PLC dưới dạng các khối chương trình (khối OB, FC, hoặc FB) và được thực hiện lặp theo chu kỳ của vòng quét (Scan).

Để có thể thực hiện được một chương trình điều khiển, tất nhiên PLC phải có chức năng như một máy tính, nghĩa là phải có bộ xử lý (CPU), một bộ điều hành, bộ nhớ để lưu chương trình điều khiển, dữ liệu,... Ngoài ra, PLC còn phải có các cổng vào/ra để giao tiếp được các đối tượng điều khiển và để trao đổi thông tin với môi trường xung quanh.

Bên cạnh đó, nhằm phục vụ bài toán điều khiển số, PLC còn cần phải có thêm các khối chức năng đặc biệt khác như: bộ đếm (counter), bộ định thời (timer)... và những khối hàm chuyên dụng khác.

PLC được thiết kế sẵn thành bộ và chưa được cố định với một nhiệm vụ nào. Tất cả các cổng logic cơ bản, chức năng nhớ, timer, counter,... được nhà sản xuất tích hợp trong bộ PLC và kết nối với nhau bằng chương trình cho mỗi một nhiệm vụ điều khiển cụ thể nào đó. Có nhiều thiết bị điều khiển và được phân biệt với nhau qua các chức năng sau:

- Các ngõ vào/ra
- Dung lượng bộ nhớ
- Bộ đếm (counter)
- Bộ định thời (timer)
- Bít nhớ
- Các khối chức năng đặc biệt
- Tốc độ xử lý
- Loại xử lý chương trình.

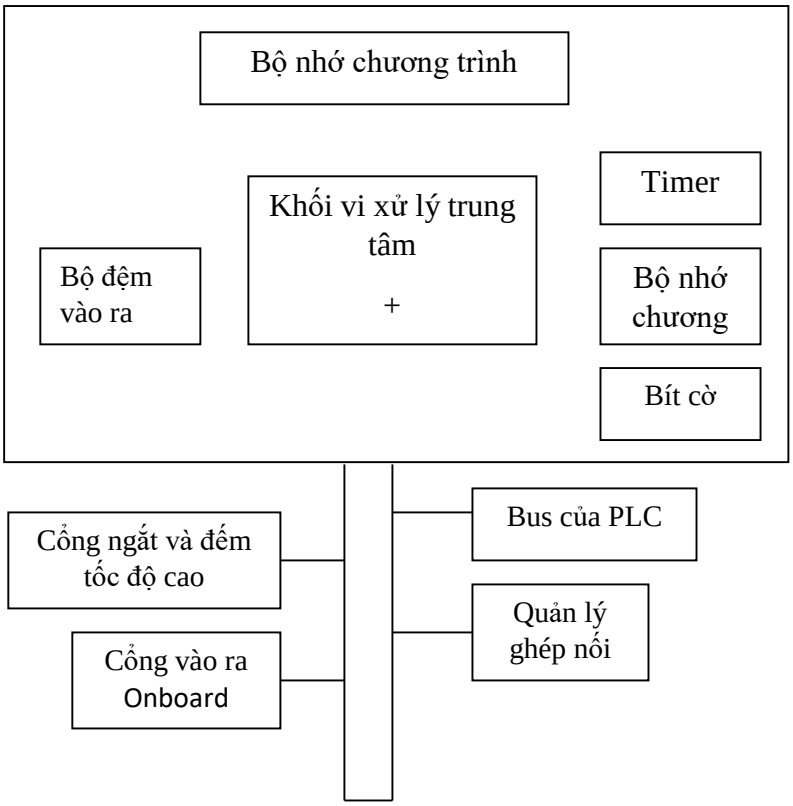
Các thiết bị điều khiển lớn thì được lắp thành các module riêng. Đối với các thiết bị điều khiển nhỏ, chúng được lắp đặt chung trong một bộ. Các bộ điều khiển này có số lượng ngõ vào/ra cho trước cố định.

Thiết bị điều khiển được cung cấp tín hiệu bởi các tín hiệu từ các cảm biến ở bộ phận ngõ vào của thiết bị tự động. Tín hiệu này được xử lý tiếp tục thông qua chương trình điều khiển đặt trong bộ nhớ chương trình. Kết quả xử lý được đưa ra bộ phận ngõ ra của thiết bị tự động để đến đối tượng điều khiển hay khâu điều khiển ở dạng tín hiệu.

1.1. Cấu trúc của một PLC:

Cấu trúc của một PLC có thể được mô tả như hình vẽ sau:

Thông tin xử lý trong PLC được lưu trữ trong bộ nhớ của nó. Mỗi phần tử vi mạch nhớ có thể chứa một bit dữ liệu. Bít dữ liệu (data binary digital) là một chữ số nhị phân, chỉ có thể là một trong hai giá trị 0 hoặc 1. Tuy nhiên các vi mạch nhớ thường được tổ chức thành nhóm để có thể chứa 8 bít dữ liệu. Mỗi chuỗi 8 bít dữ liệu được gọi là một byte. Mỗi mạch nhớ là 1 byte (byte nhớ), được xác nhận bởi một con số gọi là địa chỉ (address). Byte nhớ đầu tiên có địa chỉ 0. Dữ liệu chứa trong byte nhớ gọi là nội dung.



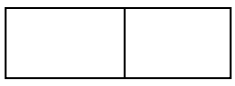
Hình 1.1: Cấu trúc của một PLC.

Địa chỉ của một byte nhớ là cố định và mỗi byte nhớ trong PLC có một địa chỉ riêng của nó. Địa chỉ của byte nhớ khác nhau sẽ khác nhau, nội dung chứa trong một bute nhớ là đại lượng có thể thay đổi được. Nội dung byte nhớ chính là dữ liệu được lưu trữ tức thời trong bộ nhớ.

Để lưu giữ một dữ liệu mà một byte nhớ không thể chứa hết được, thì PLC cho phép một cặp 2byte nhớ cạnh nhau được xem xét như một đơn vị nhớ và được gọi là một từ đơn (word). Địa chỉ thấp hơn 2 byte nhớ được dùng làm địa chỉ của từ đơn.

Ví dụ 1: Từ đơn có địa chỉ là 2 thì các byte nhớ có địa chỉ là 2 và 3 với 2 là địa chỉ byte cao và 3 là địa chỉ của byte thấp.a

IB2 IB3



IW2

IW2 là từ đơn có địa chỉ 2:

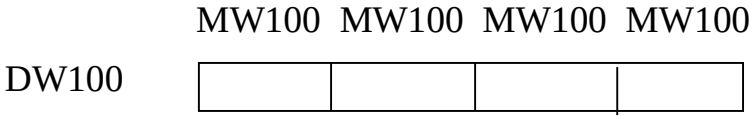
IB2 là byte có địa chỉ 2

IB3 là byte có địa chỉ 3

Trong trường hợp dữ liệu cần được lưu trữ mà một từ đơn không thể chứa hết được, PLC cho phép ghép 4byte liền nhau được xem xét là một đơn vị nhớ và

được gọi là từ kép (double word). Địa chỉ thấp nhất trong 4 byte nhớ này là địa chỉ của từ kép.

Ví dụ 2: từ kép có địa chỉ là 100 thì các byte nhớ trong từ kép này có địa chỉ là 100,101,102,103, trong đó 103 là địa chỉ byte thấp, 100 là địa chỉ byte cao.



Trong một PLC, bộ xử lý trung tâm có thể thực hiện một số thao tác như:

- Đọc nội dung các vùng nhớ (bit, byte, word, double word).

- Ghi dữ liệu vào vùng nhớ (bit, byte, word, double word).

Trong thao tác đọc, nội dung ban đầu của vùng nhớ không thay đổi mà chỉ lấy bản sao của dữ liệu để xử lý.

Trong thao tác ghi, dữ liệu được ghi vào trở thành nội dung của vùng nhớ và dữ liệu ban đầu bị mất đi.

Có hai loại bộ nhớ trong CPU của PLC:

- RAM (Random Access Memory): Bộ nhớ có thể đọc và ghi.

- ROM (Read Only Memory): Bộ nhớ chỉ đọc.

* Bộ nhớ RAM:

Có số lượng các ô nhớ xác định. Mỗi ô nhớ có một dung lượng nhớ cố định và nó chỉ tiếp nhận một lượng thông tin nhất định. Các ô nhớ được ký hiệu bằng các địa chỉ riêng của nó. Bộ nhớ này chứa các chương trình được sửa đổi hoặc caccs dữ liệu, kết quả tạm thời trong quá trình tính toán, lập trình.

Đặc điểm của bộ nhớ RAM là nội dung chứa trong các ô nhớ của nó bị mất đi khi mất nguồn điện.

* Bộ nhớ ROM:

Chứa các thông tin không có khả năng xóa được hoặc không thể thay đổi được, được nhà sản xuất sử dụng chứa các chương trình hệ thống. Chương trình trong bộ nhớ ROM có nhiệm vụ:

- Điều khiển và kiểm tra các chức năng hoạt động của CPU (hệ điều hành).

- Dịch ngôn ngữ lập trình thành ngôn ngữ máy.

- Khi bị mất nguồn điện, bộ nhớ ROM vẫn giữ nguyên nội dung của nó và không bao giờ bị mất.

* Bộ xử lý trung tâm:

Bộ xử lý trung tâm (CPU – Central Processing Unit) điều khiển và quản lý tất cả các hoạt động bên trong PLC. Việc trao đổi thông tin giữa CPU, bộ nhớ và khối vào/ra được thực hiện thông qua hệ thống BUS dưới sự điều khiển của CPU.

Một mạch dao động thạch anh cung cấp xung clock tần số chuẩn cho CPU, thường là 1 hay 8MHz, tùy thuộc vào bộ xử lý sử dụng. Tần số xung Clock xác định tốc độ hoạt động của PLC và được dùng để thực hiện sự đồng bộ cho tất cả các phần tử trong hệ thống.

* Hệ điều hành:

Sau khi bật nguồn, hệ điều hành sẽ đặt các counter, timer và bít nhớ với thuộc tính non_retentive (không được nhớ bởi pin dự phòng) cũng như accu về 0.

Để xử lý chương trình, hệ điều hành đọc từng dòng chương trình từ đầu đến cuối. Tương ứng hệ điều hành thực hiện chương trình theo các câu lệnh.

* Bít nhớ: (memory bit):

Các memory bit là các phần tử nhớ mà hệ điều hành ghi nhớ trang thái tín hiệu.

* Bộ đệm:

Bộ đệm là một vùng nhớ, mà hệ điều hành ghi nhớ trạng thái tín hiệu ở các ngõ vào/ra nhị phân.

* Accumulator:

Accumulator là một bộ nhớ trung gian mà qua nó, timer hay counter được nạp vào hay thực hiện các phép toán số học.

* Counter, timer:

Timer và counter cũng là các vùng nhớ, hệ điều hành ghi nhớ các giá trị đếm trong nó.

* Hệ thống bus:

Bộ nhớ chương trình, hệ điều hành và các module ngoại vi (các ngõ vào/ra) được kết nối với PLC thông qua BUS nối. Một BUS bao gồm các dây dẫn nà các dữ liệu được trao đổi. Hệ điều hành tổ chức việc truyền dữ liệu trên các dây dẫn này.

1.2. Thiết bị điều khiển lập trình S7-200:

S7-200 là thiết bị điều khiển lập trình loại nhỏ của hãng Siemens (CHLB Đức) có cấu trúc theo kiểu module và có các module mở rộng.

	CPU 221	CPU222	CPU224	CPU226	CPU226XM
Bộ nhớ chương trình	2048W	2048W	4096W	4096W	8192W
Bộ nhớ dữ liệu	1024W	1024W	2560W	2560W	5120W

Khả năng dự phòng bộ nhớ khi mất điện	50 giờ	50 giờ	190 giờ	190 giờ	190 giờ
I/O địa chỉ	6In/4Out	8In/6Out	14In/10Out	24In/16Out	24In/16Out
Đồng hồ thời gian thực	Cartrige	Cartrige	Tích hợp	Tích hợp	Tích hợp
Kích thước bộ đếm	256 (128 vào, 128 ra)				
Tốc độ thực hiện lệnh logic	0,37 us/lệnh				

Bảng 1.1: Cấu trúc của một số CPU của PLC S7-200

Địa chỉ các ngõ vào/ra

Địa chỉ ô nhớ trong s7 gồm hai phần: phần chữ và phần số

Ví dụ 3:

PIW304 hoặc I0.0

Phần chữ phần số Phần chữ Phần số

Phần chữ chỉ vị trí và kích thước ô nhớ

M: Chỉ ô nhớ trong miền các biến cờ có kích thước là 1 bít

MB: Chỉ ô nhớ trong miền các biến cờ có kích thước là 1 byte (8bít)

MW: Chỉ ô nhớ trong miền các biến cờ có kích thước là 2 byte (16 bít)

MD: Chỉ ô nhớ trong miền các biến cờ có kích thước là 4 byte (32 bít)

I: Chỉ ô nhớ có kích thước là 1 bít trong miền bộ đệm ngõ vào số

IB: Chỉ ô nhớ có kích thước là 1 byte trong miền bộ đệm ngõ vào số

IW: Chỉ ô nhớ có kích thước là 2 byte (1 từ) trong miền bộ đệm ngõ vào số

ID: Chỉ ô nhớ có kích thước là 4 byte (2 từ) trong miền bộ đệm ngõ vào số

Q: Chỉ ô nhớ có kích thước là 1 bít trong miền bộ đêm ngõ ra số

QB: Chỉ ô nhớ có kích thước là 1 byte trong miền bộ đêm ngõ ra số

QW: Chỉ ô nhớ có kích thước là 2 byte trong miền bộ đêm ngõ ra số

QD: Chỉ ô nhớ có kích thước là 4 byte trong miền bộ đêm ngõ ra số

T: Chỉ ô nhớ trong miền nhớ của bộ thời gian (Timer)

C: Chỉ ô nhớ trong miền nhớ của bộ đếm (Counter)

PIB: Chỉ ô nhớ có kích thước là 1 byte thuộc vùng Peripheral Input, thường là cổng vào của các modul tương tự

PIW: Chỉ ô nhớ có kích thước là 2 byte thuộc vùng Peripheral Input, thường là cổng vào của các modul tương tự

PID: Chỉ ô nhớ có kích thước là 4 byte thuộc vùng Peripheral Input, thường là cổng vào của các modul tương tự

PQB: Chỉ ô nhớ có kích thước là 1 byte thuộc vùng Peripheral output, thường là cổng ra của các modul tương tự

PQW: Chỉ ô nhớ có kích thước là 2 byte thuộc vùng Peripheral output, thường là cổng ra của các modul tương tự

Phần số chỉ địa chỉ của byte hoặc bit trong miền nhớ đã xác định

Nếu ô nhớ đã được xác định thông qua phần chữ có kích thước 1 bit thì phần số sẽ là địa chỉ của byte và số thứ tự của bit trong byte đó, được tách với nhau bằng dấu chấm.

Ví dụ 4:

I 0.0: Chỉ bit 0 của byte 0 trong miền nhớ bộ đệm ngõ vào số PII

Q 4.1: Chỉ bit 1 của byte 4 của miền nhớ bộ đệm ngõ ra số PIQ

M105: Chỉ bit 5 của byte 10 trong miền các biến cờ M

Trong trường hợp ô nhớ dã được xác định là byte, từ hoặc từ kép thì phần số sẽ là địa chỉ của byte đầu tiên trong mảng byte của ô nhớ đó.

Ví dụ 5:

DIB 15: Chỉ ô nhớ có kích thước 1byte (byte 15) trong khối DB đã được mở bằng lệnh OPN DI

DIW 18: Chỉ ô nhớ có kích thước 2 byte (byte 18,19) trong khối DB đã được mở bằng lệnh OPN DI

DB2.DBW15: Chỉ ô nhớ có kích thước 2 byte 15,16 trong khối dữ liệu DB2.

M 105: Chỉ ô nhớ có kích thước 2 từ gồm 4 byte 105,106,107,108 trong miền nhớ các biến cờ M.

Cấu trúc của bộ nhớ S7-200

Bộ nhớ của S7-200 được chia làm 3 vùng: vùng nhớ chương trình, vùng nhớ dữ liệu và vùng nhớ thông số. Vùng nhớ chương trình, vùng nhớ thông số và một phần vùng nhớ dữ liệu được chứa trong ROM điện EFPROM. Đối với CPU cho phép cắm thêm khối nhớ mở rộng để chứa chương trình mà không cần đến thiết bị lập trình. Phần sau đây mô tả chi tiết về các vùng nhớ.

* Vùng nhớ chương trình:

Vùng nhớ chương trình chứa các chỉ thị điều khiển vi xử lý để thực hiện yêu cầu điều khiển, chương trình ứng dụng sau khi soạn thảo được nạp vào ROM và vẫn tồn tại khi mất điện.

* Vùng nhớ thông số:

Gồm các ô nhớ chứa các thông số cài đặt, mật khẩu, địa chỉ thiết bị điều khiển và các thông tin về các vùng trống có thể sử dụng. Nội dung của vùng nhớ này được chứa trong ROM giống như vùng chương trình.

* Vùng nhớ dữ liệu:

Vùng nhớ dữ liệu là nơi làm việc, vùng này gồm các địa chỉ để lưu trữ các phép tính, lưu trữ tạm thời các kết quả trung gian, và chứa các hằng số được sử dụng trong các chỉ dẫn hoặc các thông số điều chỉnh khác. Ngoài ra trong vùng này còn có các phần tử và đối tượng như: Bộ định thời, bộ đếm, các bộ đếm tốc độ cao và các ngõ vào/ra analog. Một phần của vùng nhớ dữ liệu được chứa trong ROM, vì vậy các hằng số, cũng như các thông tin khác vẫn được duy trì khi mất điện giống như trong vùng nhớ chương trình. Một phần khác được chứa trong RAM, nội dung trong RAM cũng được duy trì trong khoảng thời gian nhất định khi mất điện bằng một điện dung có độ rỉ thấp.

Vùng dữ liệu gồm các ô biến, vùng đệm của các ngõ vào/ra, vùng nhớ trong và vùng nhớ đặc biệt. Phạm vi của vùng nhớ rất linh hoạt và cho phép đọc cũng như ghi trên toàn bộ vùng nhớ, ngoại trừ một vài ô nhớ đặc biệt chỉ cho phép đọc, các dạng dữ liệu cho phép trong vùng này là: Bit, Byte, Word hoặc Double Word.

Ngôn ngữ lập trình trong PLC:

Trong S7-200 cho phép lựa chọn 3 ngôn ngữ lập trình:

Ngôn ngữ LADDER (LAD)

Ngôn ngữ STL

Ngôn ngữ FBD

Ngôn ngữ LAD: Là ngôn ngữ lập trình đồ họa dựa trên cơ sở sơ đồ trang bị điện, việc kết nối lập trình đồ họa giống với việc thiết lập các sơ đồ relay-contactor. Một chương trình nguồn viết bằng LAD được tổ chức thành các network, mỗi network thực hiện một công việc nhỏ.

S7-200 đọc chương trình từ trên xuống dưới, từ trái qua phải, sau đó lặp lại ở vòng quét tiếp theo.

Ngôn ngữ STL: là ngôn ngữ lập trình dưới dạng Text gần giống với lập trình hợp ngữ trong vi điều khiển và vi xử lý, là một ngôn ngữ mạch cho phép tạo ra một chương trình mà LAD hoặc FBD rất khó tạo ra. Một chương trình nguồn viết bằng STL được tổ chức thành các network, mỗi network thực hiện một công việc nhỏ.

Ngôn ngữ FBD: Là ngôn ngữ lập trình đồ họa dựa trên cơ sở kết nối các khối hàm, sử dụng các ký hiệu logic giống với đại số BOOLEAN. Các hàm toán học phức tạp cũng được thể hiện dưới dạng khối với các đầu vào đầu ra thích hợp.

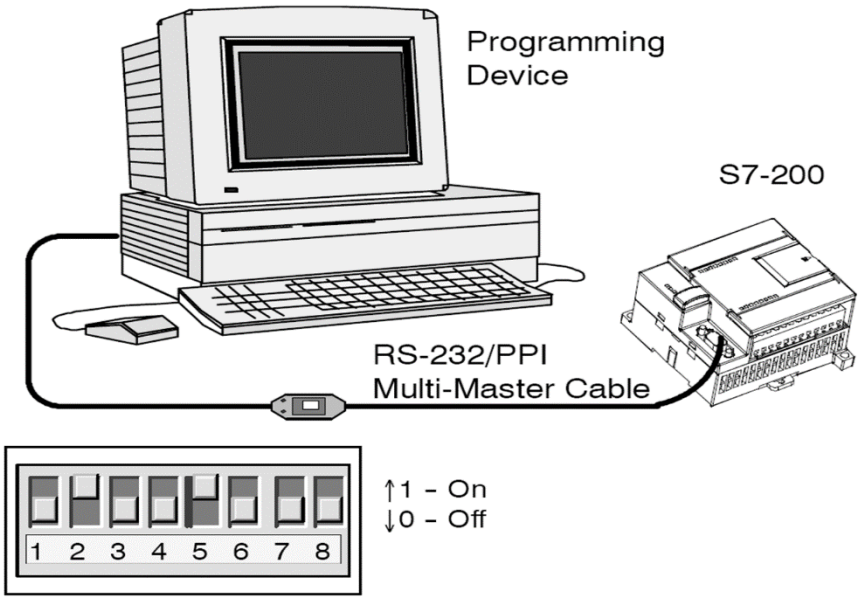
2. Kết nối dây giữa PLC và các thiết bị ngoại vi

Việc kết nối dây giữa PLC và thiết bị ngoại vi rất quan trọng. Nó quyết định đến việc PLC có thể giao tiếp được với thiết bị lập trình (máy tính) cũng như hệ

thống điều khiển có thể hoạt động theo đúng yêu cầu được thiết kế hay không. Ngoài ra việc nối dây còn liên quan đến an toàn cho PLC cũng như hệ thống điều khiển.

a, Kết nối với máy tính

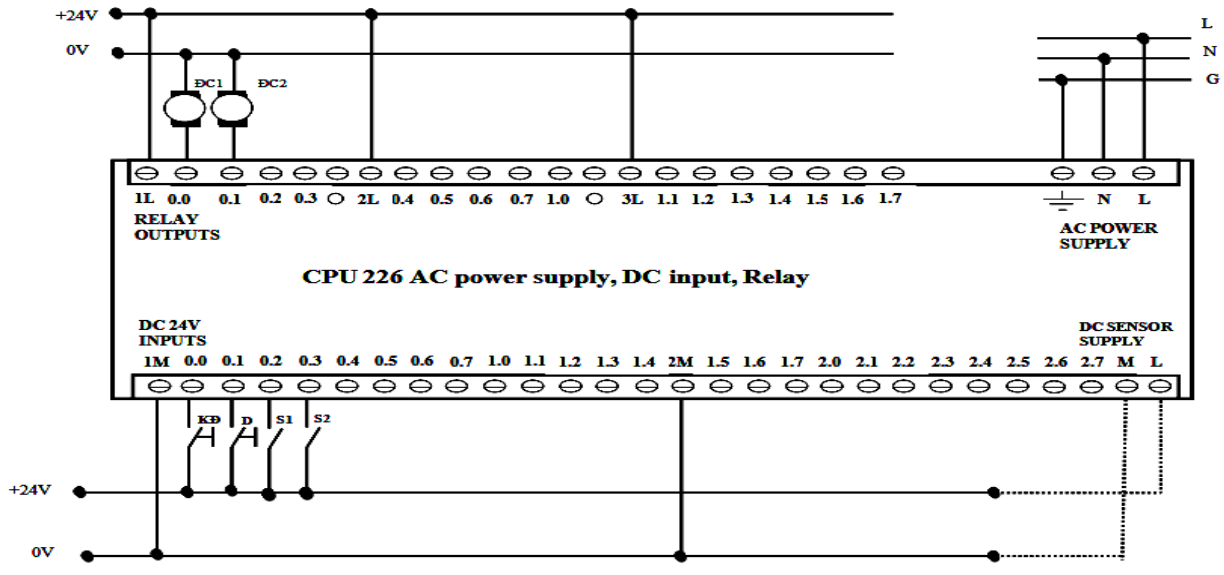
Đối với các thiết bị lập trình của hãng Siemens, có các cổng giao tiếp PPI thì có thể kết nối trực tiếp với PLC thông qua một sợi cáp. Tuy nhiên đối với máy tính cá nhân, cần thiết phải có cáp chuyển đổi PC/PPI. Sơ đồ nối máy tính với CPU thuộc họ S7-200



Hình 1.2: Kết nối máy tính với CPU qua cổng truyền thông PPI
Sử dụng cáp PC/PPI.

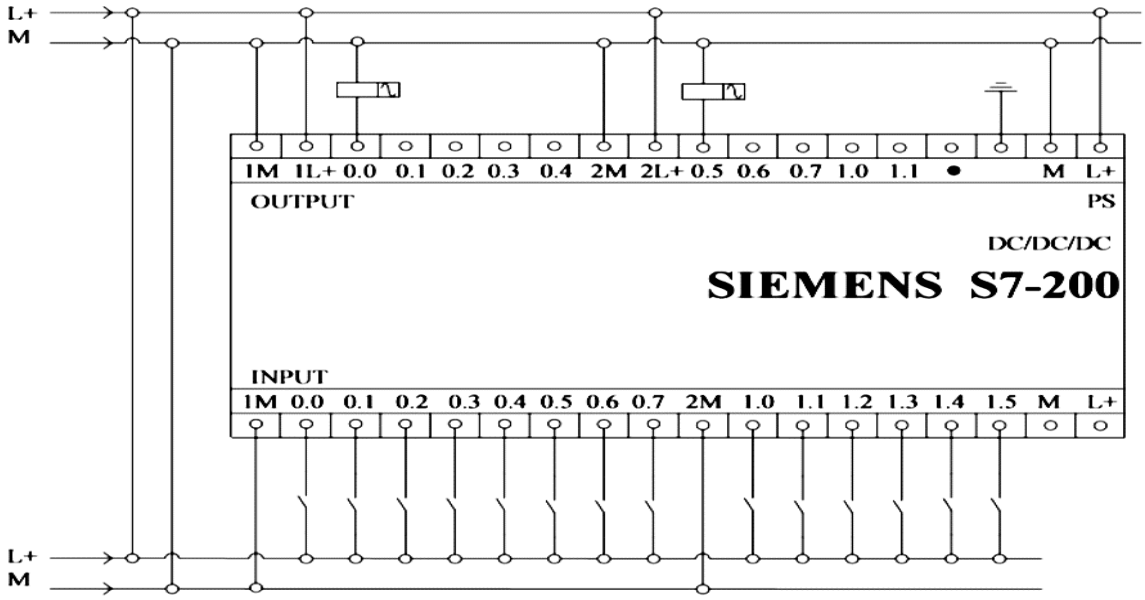
b. Kết nối vào/ra với ngoại vi

Kết nối PLC với nguồn cấp AC/DC/Relay:



Hình 1.3: Sơ đồ kết nối PLC với nguồn cấp AC/DC/Relay

Kết nối PLC với nguồn cấp DC/DC/DC:



Hình 1.4: Sơ đồ kết nối PLC với nguồn cấp DC/DC/DC

3. Cài đặt và sử dụng phần mềm Step7-MicroWin:

a. Giới thiệu về Step7- MicroWin

Step7- MicroWin là phần mềm lập trình cho PLC Siemens. Step7- MicroWin có các phiên bản sau: Step7- MicroWin 3.1, Step7- MicroWin 4.0 …

b. Những yêu cầu với máy tính PC:

- Window 2000, Window XP.

- ổ cứng trống > 350MB.

c. Cài đặt phần mềm lập trình Step7- MicroWin

Khởi động từ ổ đĩa CD click chuột chọn Setup.exe màn hình cài đặt hiện ra chon ngôn ngữ English sau đó quá trình cài đặt tiếp tục cho đến kết thúc.

Sau khi kiểm tra bộ nhớ, ổ cứng hoàn toàn có đủ khả năng để cài phần mềm STEP 7 –Micro/win vào ổ cứng, thì lần lượt tiến hành các bước:

1/ Chèn đĩa CD vào ổ CD máy tính.

2/ Kích chuột vào nút **start** để mở menu Window

3/ Kích chuột vào mục **Run** của menu
4/ Nếu cài đặt từ:
+ Disk A: Trong hộp thoại **Run**, gõ a:\setup và **enter**
+ CD: Trong hộp thoại **Run**, gõ e:\setup và **enter**
5/ Sau đó sẽ nhận được các chỉ dẫn thao tác tiếp theo trên màn hình

6/ Khi kết thúc việc cài đặt, hộp thoại setup **PG/PC Interface** tự động xuất hiện. Kích **Cancle** để trở về cửa sổ chính của step 7 Micro/win.

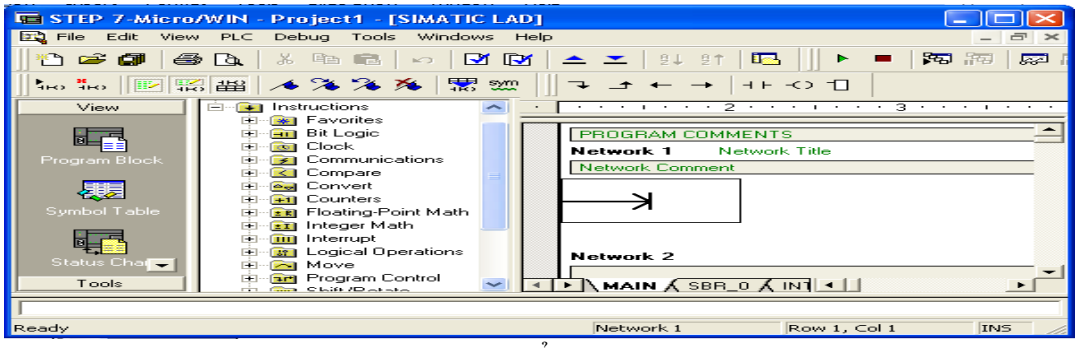
Sau khi cài đặt xong có thể bắt đầu soạn thảo chương trình bằng cách nhấp đúp vào biểu tượng của phần mềm để làm việc với giao diện trên màn hình.

d. Cách sử dụng phần mềm:

*1. Khởi động phần mềm:

*2. Tạo file mới: vào menu File chọn New

*3. Giới thiệu về cửa sổ lập trình



Hình 1.5: Cửa sổ lập trình

*4. Đặt symbol

Vào hộp Symbol Table để đặt:

			Symbol	Address	Comment
1					
2					
3					
4					
5					

Hình 1.6: Bảng đặt địa chỉ

*5. Cách lấy lệnh:

Nhấp đúp chuột trái vào biểu tượng lệnh trên cây lệnh.

*6. Nạp chương trình:

- Từ menu file chọn Download hoặc chọn biểu tượng 

*7. Chạy chương trình

- Từ menu PLC chọn RUN hoặc chọn biểu tượng 

*8. Lưu bài: Từ menu file chọn Save as. Sau đó chọn đường dẫn và viết tên file cần lưu.

BÀI 2: CÁC PHÉP TOÁN NHỊ PHÂN CỦA PLC

Mã bài: MĐ21.02

Giới thiệu: S7-200 biểu diễn một cách logic cứng bằng một dãy các lập trình. Chương trình bao gồm một dãy tập lệnh. S7-200 thực hiện chương trình bắt đầu từ lệnh đầu tiên và kết thúc ở lệnh cuối cùng trong một vòng quét.

Mục tiêu:

- Trình bày được các chức năng của RS, Timer, counter (bộ định thời, bộ đếm).

- Ứng dụng linh hoạt các chức năng của RS, Timer, counter trong các bài toán thực tế: Lập trình, kết nối, chạy thử...

- Rèn luyện đức tính tích cực, chủ động và sáng tạo

Nội dung chính:

1. Các liên kết logic:

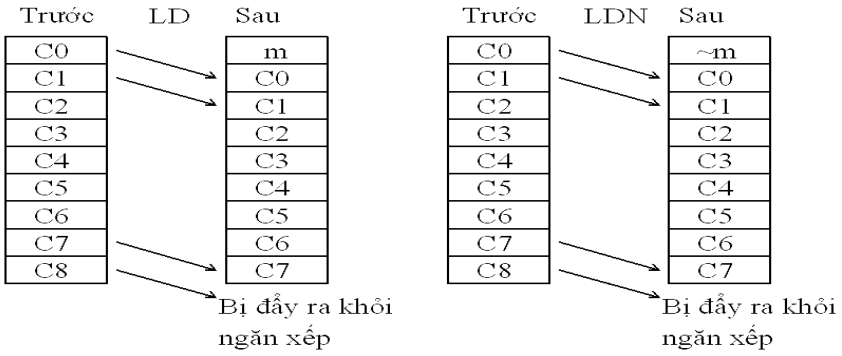
Một vòng quét trong PLC S7-200 bắt đầu từ việc đọc trạng thái của đầu vảo và sau đó thực hiện chương trình. Vòng quét kết thúc bằng việc thay đổi trạng thái đầu ra. Trước khi bắt đầu một vòng quét tiếp theo S7-200 thực hiện các nhiệm vụ bên trong và nhiệm vụ truyền thông. Chu trình thực hiện chương trình là một chu trình lặp.

Cách lập trình cho S7-200 nói riêng và cho các PLC của Siemens nói chung dựa trên hai phương pháp cơ bản: phương pháp hình thang (Ladder logic) và phương pháp liệt kê (Statement List). Nếu chương trình được viết theo kiểu LAD, thiết bị lập trình sẽ tự tạo ra một chương trình tương ứng theo kiểu STL. Ngược lại không phải mọi chương trình viết theo kiểu STL đều có thể chuyển sang LAD.

1.1 Các lệnh vào/ra và các lệnh tiếp điểm đặc biệt:

Load (LD): lệnh LD nạp giá trị của một tiếp điểm vào trong bit đầu tiên của ngăn xếp, các giá trị cũ còn lại trong ngăn xếp bị đẩy xuống 1 bit.

Load Not (LDN): lệnh LDN nạp giá trị nghịch đảo của 1 tiếp điểm vào trong bit đầu tiên của ngăn xếp, các giá trị cũ còn lại bị đẩy xuống 1 bít.



Hình 2.1: Trạng thái ngăn xếp trước và sau lệnh LD; LDN.

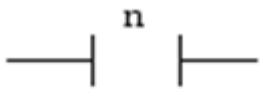
Cú pháp: LD n; LDN n

n: I, Q, M, SM, T, C, V (bit)

OUTPUT (=): Lệnh sao chép nội dung bit đầu tiên trong ngăn xếp vào bit được chỉ định trong lệnh. Nội dung ngăn xếp không bị thay đổi.

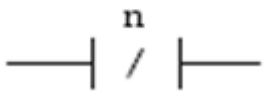
Lệnh tiếp điểm

+ Cú pháp:

Dạng LAD: 

Dạng STL:

```
LD   n
A    n
O    n
```

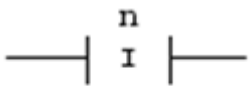



```
LDN  n
AN   n
ON   n
```

• Toán hạng cho phép (Operands) n: I, Q, M, SM, T, C, V, S.

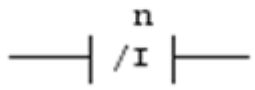
b. Lệnh các tiếp điểm tức thời

Cú pháp:

Dạng LAD 

Dạng STL

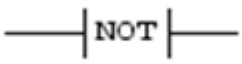
```
LDI  n
AI   n
OI   n
```

• Toán hạng 

M, SM, T, C, V, S.

```
LDNI  n
ANI   n
ONI   n
```

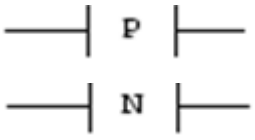
c. Tiếp điểm đảo trạng thái:

Dạng LAD 

Dạng STL NOT

Tiếp điểm đảo trạng thái của dòng cung cấp.

d. Tiếp điểm chuyển đổi theo sườn âm/ dương:

Dạng LAD 

Dạng STL EU

ED

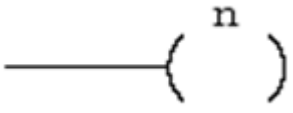
- Tiếp điểm chuyển đổi sườn dương (EU) cho phép dòng cung cấp thông mạch khi sườn xung chuyển từ 0 lên 1.

- Tiếp điểm chuyển đổi sườn âm (ED) cho phép dòng cung cấp thông mạch khi sườn xung chuyển từ 1 xuống 0.

e. Các lệnh đầu ra:

*. Lệnh đầu ra output:

+ Cú phép:

Dạng LAD 

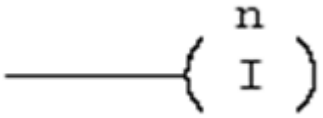
Dạng STL = n

Toán hạng cho phép: n: I, Q, M, SM, T, C, V, S

+ Ý nghĩa: Cuộn dây đầu ra ở trạng thái kích thích khi dòng điều khiển chạy qua.

*.Lệnh đầu ra tức thời:

+ Cú pháp:

Dạng LAD: 

Dạng STL =I n

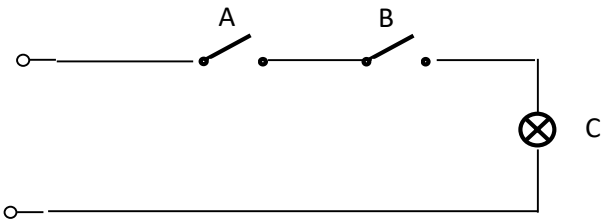
Toán hạng cho phép: n: I, Q, M, SM, T, C, V, S

+ Ý nghĩa: Cuộn dây đầu ra được kích thích tức thời khi dòng điều khiển chạy qua.

1.2. Các lệnh liên kết logic cơ bản:

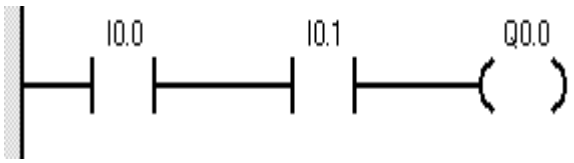
Lệnh AND

Lệnh AND sẽ tạo ra một logíc giống như hình dưới đây.



Như vậy lệnh AND thực hiện ghép nối tiếp nhiều tiếp điểm thường mở

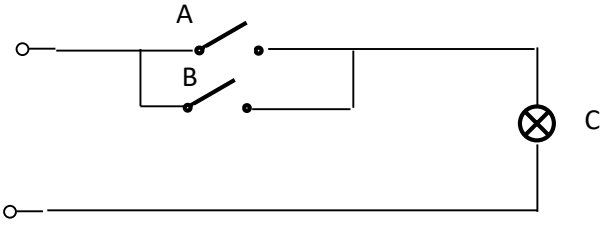
Ví dụ:




```
LD   I0.0
  A   I0.1
=   Q0.0
```

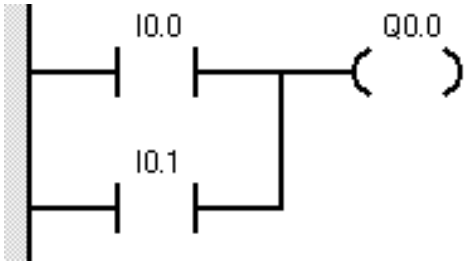
Lệnh OR

Lệnh OR sẽ tạo ra 1 logic giống như hình dưới đây:



Như vậy lệnh OR thực hiện mắc song song nhiều tiếp điểm thường mở.

Ví dụ:




```
LD I0.0
   O  I0.1
=  Q0.0
```

1.3. Liên kết các cổng logic cơ bản:

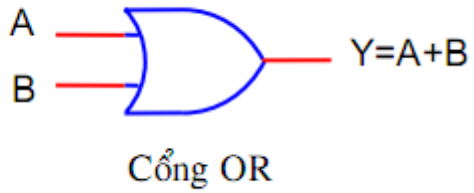
a, Phép toán OR và cổng OR

Gọi A và B là 2 biến logic độc lập. Khi A và B kết hợp qua phép toán OR, kết quả x được mô tả như sau:

$X = A + B$

Trong biểu thức này, dấu "+" không có nghĩa là phép cộng thuần túy. Nó là phép toán OR, kết quả của phép toán OR được cho trong bảng sự thật sau:

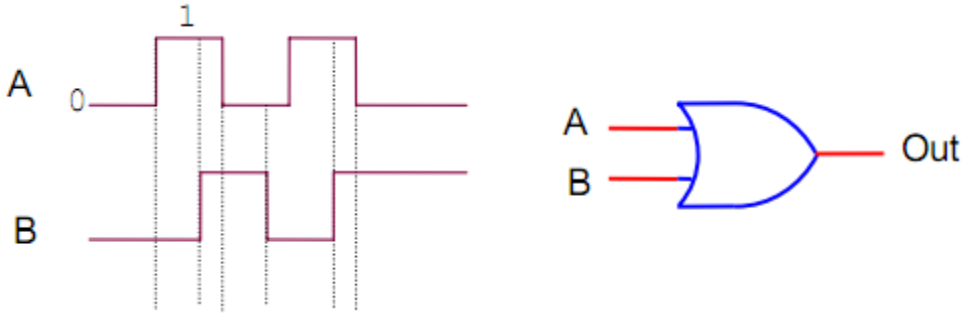
A	B	X=A+B
0	0	0
0	1	1
1	0	1
1	1	1



Cổng OR

Hình 2.2: Bảng sự thật của phép toán OR.

Ví dụ 2: Xác định dạng sóng ngo ra cổng OR khi ngo vào A, B thay đổi theo



giản đồ sau:

Hình 2.3: Giản đồ xung của phép toán .

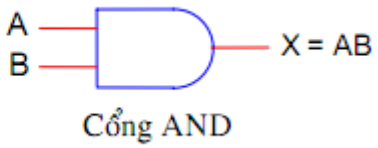
b, Phép toán AND và cổng AND

Nếu hai biến logic A và B được kết hợp qua phép AND, kết quả là:

$X = A.B$

Bảng sự thật của phép nhân 2 biến A và B như sau:

A	B	X=A.B
0	0	0
0	1	0
1	0	0
1	1	1



Hình 2.4: Bảng sự thật của phép toán AND.

Ví dụ 3: Xác định dạng sóng ngõ ra của cổng AND ứng với các ngõ vào như sau:

Hình 2.5: Giản đồ xung của phép toán AND.

Trong ví dụ này thấy rằng, ngõ ra sẽ bằng với ngõ vào A khi B ở mức logic 1. Vì vậy ta có thể xem ngõ vào B như ngõ vào điều khiển, nó cho phép dạng sóng ở ngõ vào A xuất hiện ở ngõ ra hay không.

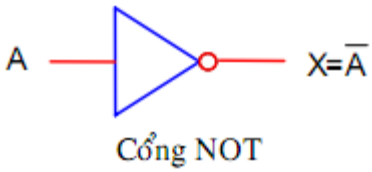
c, Phép toán NOT và cổng NOT

Nếu biến A được đưa qua phép toán NOT, kết quả x sẽ là:

$X = \overline{A}$

Cổng NOT chỉ có một ngõ vào và một ngõ ra. Trên hình 9 là bảng sự thật và kí hiệu của phần tử NOT

A	$X=\overline{A}$
0	1
1	0

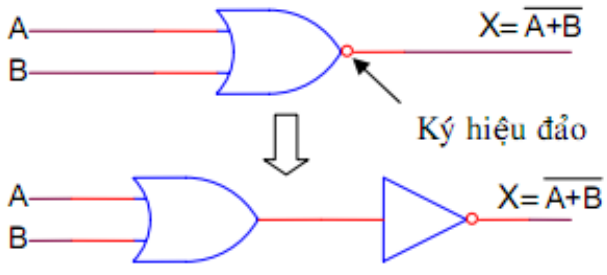


Hình 2.6: Bảng sự thật của phép toán NOT.

d, Phần thử NOR và cổng NOR

Cổng NOR hoạt động giống như hai cổng OR và NOT mắc nối tiếp như sau:

		OR	NOR
A	B	A+B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



Hình 2.7: Bảng sự thật của phép toán NOR

Trên sơ đồ mạch điện cổng NOR có kí hiệu giống như cổng OR nhưng có thêm vòng tròn ở phái đầu ra đại diện cho tín hiệu ra đảo so với cổng OR. Phần

tử OR có thể có hai hoặc nhiều đầu vào. Nếu các đầu vào của cổng OR được nối chung thì cổng OR có chưc năng như phần tử NOT.

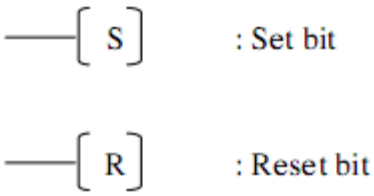
2. Các lệnh ghi/xóa giá trị cho tiếp điểm:

Mạch nhớ R_S:

Mạch nhớ là mạch có hai trạng thái ổn định và thông qua tín hiệu ngõ vào mà trạng thái của nó thay đổi. Đối với mạch điều khiển dùng relay và contactor ta có mạch tự duy trì. Còn trong PLC có khâu R-S (viết tắt của Reset và Set). Mạch nhớ R-S là rất cần thiết trong kỹ thuật điều khiển. Nó được xem là một chức năng cơ bản trong hầu hết các loại PLC và được chia thành hai loại là: Ưu tiên SET và ưu tiên RESET.

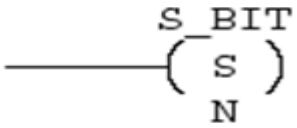
2.1. Lệnh Set (S) và Reset (R) trong PLC S7-200

Mạch nhớ R-S được thể hiện qua hai lệnh set và reset với các ví dụ ứng dụng dùng bộ nhớ với cú pháp như sau:

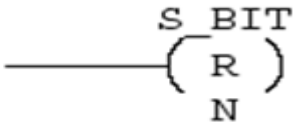


+ Cú pháp:

Dạng LAD Dạng STL



S S_BIT, N



R S_BIT, N

Toán hạng cho phép:

S_bit: I, Q, M, SM, T, C, V, S

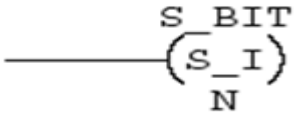
N: IB, QB, MB, SMB, VB, AC, Constant, *VD, *AC, SB.

+ Ý nghĩa: Thiết lập hoặc xoá một mảng gồm n tiếp điểm kể từ địa chỉ S-bit.

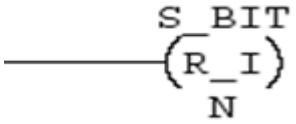
*. Lệnh Set, Reset tức thời:

+ Cú pháp:

Dạng LAD Dạng STL



SI S_BIT, N



RI S_BIT, N

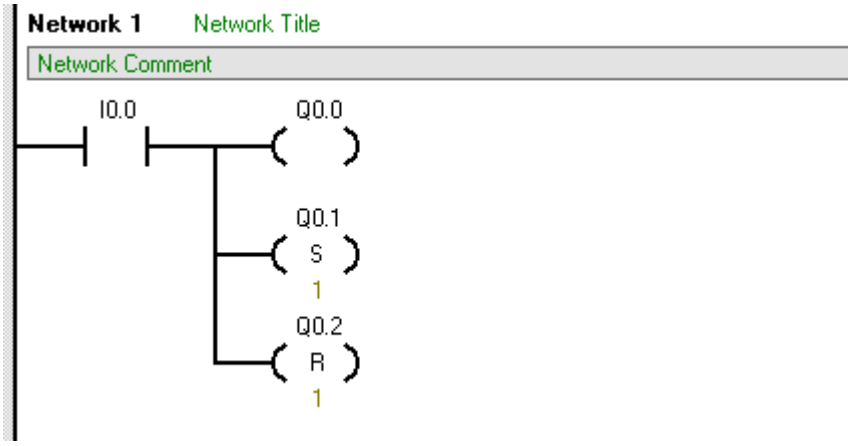
Toán hạng cho phép:

S_bit: Q

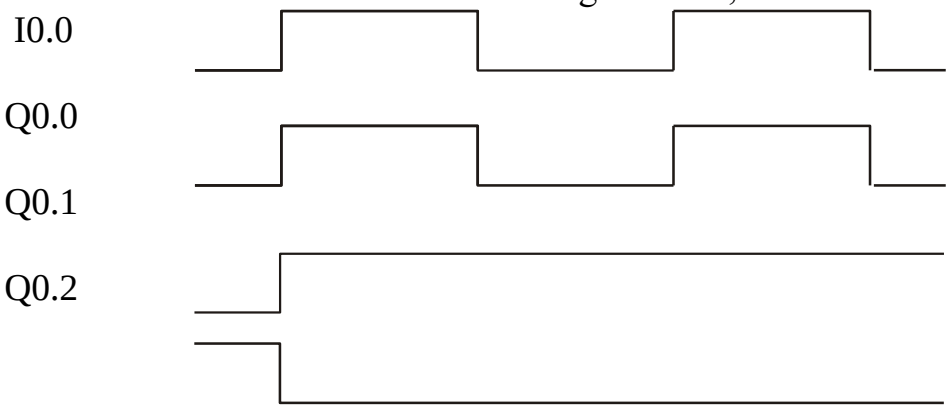
N: IB, QB, MB, SMB, VB, AC, Constant, *VD, *AC, SB.

+ Ý nghĩa: Thiết lập hoặc xoá tức thời một mảng gồm n tiếp điểm kể từ địa chỉ S-bit.

2.2. Các ví dụ ứng dụng dùng bộ nhớ:



Hình 2.8: Chương trình set, reset



Hình 2.9: Giản đồ thời gian sử dụng set, reset

3. Timer:

Timer là bộ tạo thời gian trễ giữa tín hiệu vào và tín hiệu ra. Nếu ký hiệu tín hiệu (logic) và là *x(t)* và thời gian trễ được tạo ra bằng timer là τ thì tín hiệu đầu ra của timer đó là $x(t-\tau)$. S7-200 có 64 timer (với CPU 212) và 128 timer (với CPU 214) được chia làm 2 loại khác nhau, đó là:

- Timer tạo thời gian trễ không có nhớ (On-delay timer), ký hiệu TON

- Timer tạo thời gian trễ có nhớ (Retentive On-delay timer), ký hiệu là TONR.

Hai kiểu timer của S7-200 (TON và TONR) cùng bắt đầu tạo thời gian trễ tín hiệu kể từ thời điểm có sườn lên ở tín hiệu đầu vào, tức là khi tín hiệu đầu vào chuyển trạng thái logic từ 0 lên 1, được gọi là *thời điểm timer được kích*, và không tính khoảng thời gian khi đầu vào có giá trị logic 0 vào thời gian trễ tín hiệu được đặt trước.

Khi đầu vào có giá trị logic bằng 0, TON sẽ tự động reset còn TONR thì không tự động Reset. Timer TON được dùng để tạo thì gian trễ trong một khoảng thời gian (*miền liên thông*), còn với TONR thời gian trễ sẽ được tạo ra trong nhiều khoảng thời gian khác nhau.

Timer TON và TONR bao gồm 3 loại với độ phân giải khác nhau, độ phân giải 1ms, 10ms và 100ms. Thời gian trễ τ được tạo ra chính là tích của độ phân giải của bộ timer và giá trị đặt trước cho timer.

Ví dụ như một bộ timer có độ phân giải bằng 10ms và giá trị đặt là 50 thì thời gian trễ τ=10x50=500ms.

Timer của S7-200 có những tính chất cơ bản sau:

- Các bộ timer được điều khiển bởi một cổng vào và giá trị đếm tức thời. Giá trị đếm tức thời của timer được nhớ trong thanh ghi 2 byte (T-word) của timer, xác định khoảng thời gian trễ kể từ khi timer được kích. Giá trị đặt trước của các bộ timer được ký hiệu rong LAD và STL là PT. Giá trị đếm tức thời của thanh ghi T-word thường xuyên được so sánh với giá trị đặt trước của timer.

- Mỗi timer, ngoài thanh thi 2 byte T-word lưu giá trị đếm tức thời còn có một bit, ký hiệu bằng T-bit, chỉ trạng thái logic đầu ra. Giá trị logic này phu thuộc vào kết quả so sánh giữa giá trị đếm tức thời với giá trị đặt trước.

- Trong khoảng thời gian tín hiệu x(t) có giá trị logic 1, giá trị đếm tức thời trong T-word luôn được cập nhật và thay đổi tăng dần cho đến khi đạt giá trị cực đại. Khi giá trị đếm tức thời lớn hơn hay bằng giá trị đặt trước, T-bit có giá trị logic 1.

- Bảng 2.1: Mô tả các kiểu timer và độ phân giải ứng với CPU 214 và CPU 226.

Lệnh	**Độ phân giải**	**Giá trị cực đại**	**CPU 214**	**CPU 226**

TON	1ms	32,767s	T32, T96	T32, T96
	10ms	327,67s	T33-T36, T97-	T33-T36, T97-T100
	100ms	3276,7s	T37-T63, T101-	T37-T63, T101-T255
TONR	1ms	32,767s	T0	T0, T64
	10ms	327,67s	T1-T4	T1-T4, 65-T68
	100ms	3276,7s	T5-T31	T5-T31, T69-T95

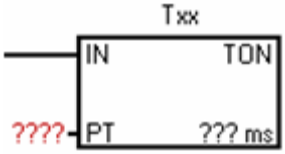
Bảng phân loại TON và TONR

Cú pháp khai báo sử dụng timer trong LAD, bao gồm 2 loại như sau:

3.1. On - delay Timer (TON)

- Cú pháp:

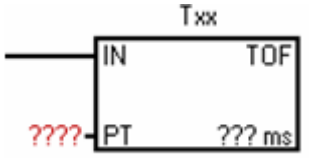
Hình 2.10: Khai báo sử dụng TON.



Khai báo timer số hiệu Txx kiểu TON để tạo thời gian trễ tính từ khi đầu vào IN được kích. Nếu như giá trị đếm tức thời lớn hơn hoặc bằng giá trị đặt trước PT thì T-bit có giá trị logic bằng 1. Có thể reset timer kiểu TON bằng lệnh R hoặc bằng giá trị logic 0 tại đầu vào IN.

3.2. Retentive On-Delay Timer (TONR)

- Cú pháp:



Hình 2.11: Khai báo sử dụng TONR.

Cú pháp khai báo sử dụng timer trong STL như sau: Khai báo timer của S7-200 là lệnh có điều kiện. Tại thời điểm khai báo tín hiệu đầu vào có giá trị logic bằng giá trị logic của bit đầu tiên trong ngăn xếp.

Khi sử dụng timer kiểu TONR, giá trị đếm tức thời được lưu lại và không bị thay đổi trong khoảng thời gian tín hiệu đầu vào logic 0. Giá trị của T-bit không được nhớ mà hoàn toàn phụ thuộc vào kết quả so sánh giữa giá trị đếm tức thời và giá trị đặt trước.

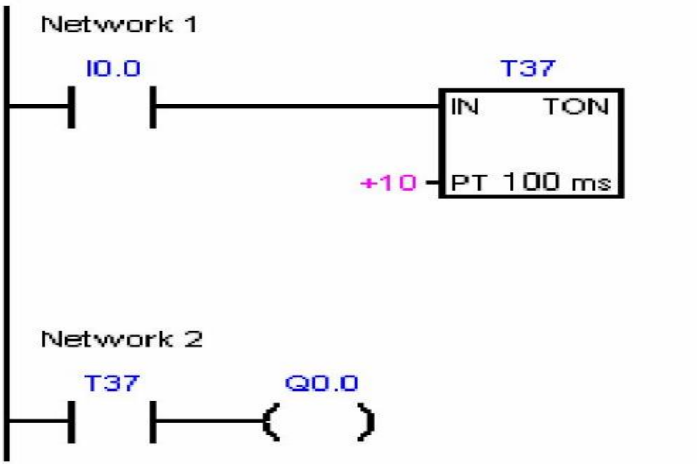
Các timer được đặt tên là Txx, trong đó xx là số hiệu của timer. Txx đồng thời cũng là đại chỉ hình thức của T-word và T-bit của timer đó. Tuy chúng có cùng địa chỉ hình thức, song T-word và T-bit vẫn được phân biệt với nhau nhờ

kiểu sử dụng lệnh với Txx. Khi dùng lệnh làm việc với từ, Txx được hiểu là T-word, ngược lại khi sử dụng lệnh làm việc với tiếp điểm, Txx được hiểu là T-bit.

Khi timer được reset, T-word và T-bit của nó đồng thời được xoá và có giá trị 0. Đối với timer TON có hai phương pháp reset là xoá tín hiệu đầu vào hoặc dùng lệnh R. Còn đối với timer TONR chỉ có một phương pháp reset duy nhất là dùng lệnh R (R Txx K1).

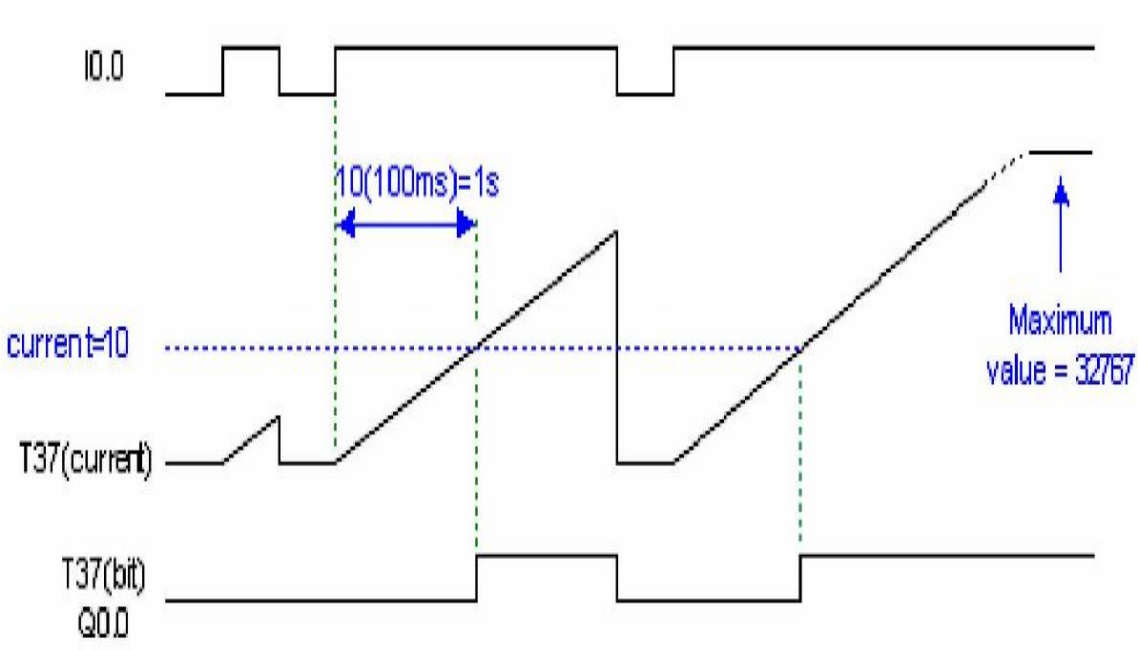
Ví dụ 7: Cách khai báo và sử dụng sử dụng timer kiểu TON.

Khi ngõ vào I0.0 =1 Timer



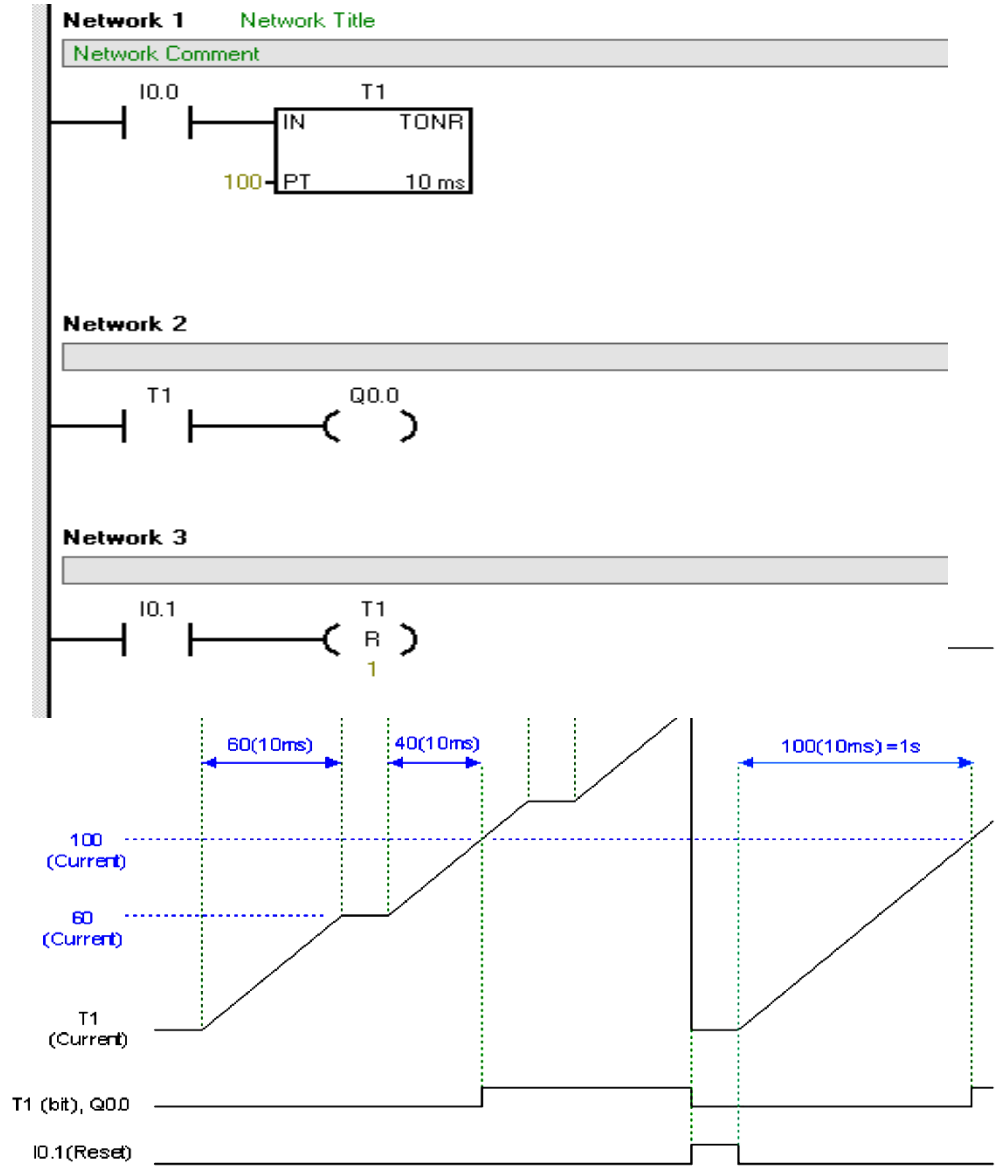
Hình 2.12: Chương trình sử dụng timer TON

T37 được kích, Nếu sau 10x100ms =1s I0.0 vẫn giữ trạng thái thì bit T37 sẽ lên 1 (Khi đó Q0.0 lên 1). Nếu I0.0 =1 không đủ thời gian 1S thì bit T37 sẽ không lên 1.



Hình 2.13: Giản đồ thời gian timer TON

Ví dụ 8: Cách khai báo và sử dụng sử dụng timer kiểu TONR



Hình 2.15: Giản đồ thời gian timer TONR

Ngõ vào I0.0 có tác dụng kích thời gian cho Timer,khi ngõ I0.0 =1 thời gian Timer được tính,khi I0.0=0 thời gian không bị Reset về 0.Khi đủ thời gian thì Bit T1 sẽ lên 1. Thời gian Timer chỉ bị Reset khi có tín hiệu Reset Timer (tín hiệu từ ngõ I0.1)

- **Cập nhật timer có độ phân giải 1ms**

CPU của S7-200 có các bộ timer có độ phân giải 1ms cho phép PLC cập nhật và thay thay đổi giá trị đếm tức thời trong T-word mỗi 1ms một lần.

Các bộ timer với độ phân giải thấp này có khả năng điều khiển chính xác các thao tác.

Ngay sau khi bộ timer với độ phân giả 1ms được kích, việc cập nhật để thay đổi giá trị đếm tức thời T-word *hoàn toàn tự động*. Chỉ nên đặt giá trị rất nhỏ cho PT của bộ timer có độ phân giải 1ms. Tần số cập nhật để thay đổi giá trị đếm tức thời và của T-bit của một bộ timer có độ phân giải 1ms không phụ thuộc vào vòng quét (scan) cảu bộ điều khiển và vòng quét của chương trình đang chạy. Giá trị đếm tức thời và T-bit của timer này có thể được cập nhật vào bất cứ thời điểm nào trong vòng quét và được cập nhật nhiều lần trong một một vòng quét nếu thời gian vòng quét đó lớn hơn 1ms.

Do việc cập nhật T-word của timer với độ phân giải 1ms hoàn toàn tự động nên thời gian trễ có thể bị trôi trong khoảng thời gian 1ms. Bởi vậy, ví dụ để có thể có được thời gian trễ không dưới 56ms ta nên đặt giá trị ban đầu cho PT là 57.

Thực hiện lệnh R (reset) đối với một timer có độ phân giải 1ms đang ở trạng thái làm việc có nghĩa là đưa timer đó về trạng thái ban đầu, giá trị đếm tức thời của timer được đưa về 0 và T-bit nhận giá trị logic 0.

- **Cập nhật timer có độ phân giải 10ms**

CPU của S7-200 có các bộ timer với độ phân giải 10ms. Sau khi được kích, việc cập nhật T-word và T-bit để thay đổi giá trị đếm tức thời và trạng thái logic đầu ra của các bộ timer này không phụ thuộc vào chương trình và được tiến hành *hoàn toàn tự động* mỗi vòng quét một lần vào thời điểm đầu vòng quét.

Thực hiện lệnh R đối với timer có độ phân giải 10ms đang ở trạng thái làm việc là đưa timer về trạng thái ban đầu và đưa T-word và T-bit giá trị 0.

Do việc cập nhật T-word của timer chỉ được thực hiện tự động mỗi vòng quét một lần nên thời điểm thay đổi giá trị đếm tức thời và giá trị logic của T-bit của timer có thể bị trôi trong khoảng 10ms. Bởi vậy, ví dụ để tạo được một khoảng thời gian trễ ít nhất là 140ms, nên chọn giá trị đặt trước cho timer có độ phần giải 10ms là PT=15.

- **Cập nhật timer có độ phân giải 100ms**

Hầu hết các bộ timer của S7-200 đều là timer có độ phân giải 100ms. Giá trị để lưu trữ trong bộ timer 100ms được tính tại đầu mỗi vòng quét và thời gian để tính sẽ là khoảng thời gian từ đầu vòng quét trước đó.

Việc cập nhật để thay đổi giá trị đếm tức thời của timer chỉ được tiến hành ngay tại thời điểm có lệnh khai báo cho timer trong chương trình. Bởi vậy quá trình cập nhật giá trị đếm tức thời không phải là một quá trình tự động và không nhất thiết phải được thực hiện một lần trong mỗi thời gian vòng quét ngay cả khi timer đã được kích. Đối với trường hợp một lệnh timer 100ms được khai báo nhiều lần trong một vòng quét thì có thể xảy ra trường hợp giá trị lưu trữ bị cộng nhiều lần với giá trị đếm tức thời, vì vậy nên sử dụng lệnh khai báo timer 100ms

chính xác một lần trong một vòng quét.

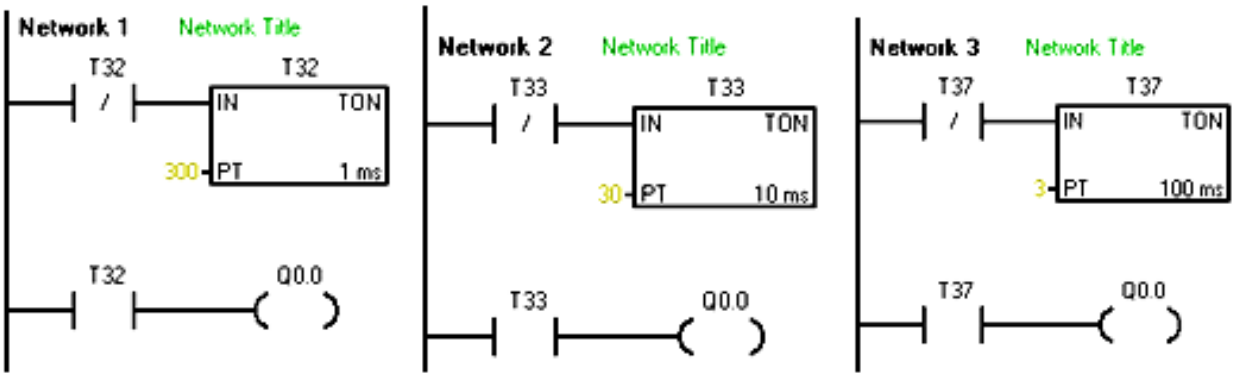
- **Hiệu quả của việc cập nhật giá trị đếm tức thời của timer 1ms, 10ms, 100ms.**

Việc cập nhật giá trị đếm tức thời của các timer với độ phân giải khác nhau được thực hiện tại các thời điểm khác nhau phụ thuộc vào các sử dụng timer đó. Ví dụ sau mô phỏng sự khác nhau đó với 3 loại timer cùng đặt thời gian 300ms.

- Trong trường hợp sử dụng timer 1ms, Q0.0 sẽ có giá trị logic bằng 1 trong khoảng thời gian 1 vòng quét nếu thời điểm cập nhật giá trị tức thời xảy ra trước khi tiếp điểm thường mở T32 và tiếp điểm thường đóng T32 chuyển trạng thái.

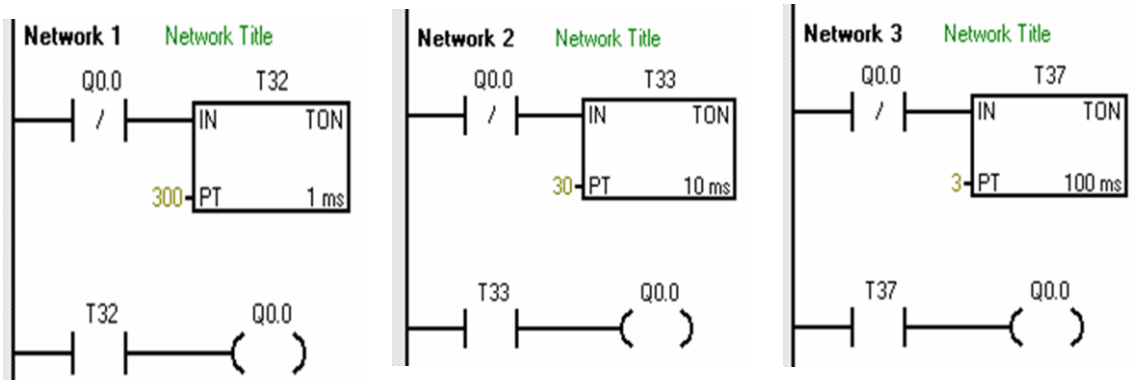
- Trong trường hợp sử dụng timer có độ phân giải 10ms, Q0.0 sẽ luôn có giá trị logic 0 vì khi bít T33 có giá trị logic 1 ở đầu vòng quét thì ngay sau đó sẽ bị chuyển về trạng thái 0.

- Trong trường hợp sử dụng timer 100ms, Q0.0 sẽ luôn có giá trị logic 1 trong khoảng thời gian 1 vòng quét mỗi khi giá trị đếm tức thời bằng giá trị đặt trước.



Hình 2.16: Ảnh hưởng của độ phân giải đến đầu ra của timer.

Việc sử dụng tiếp điểm thường đóng Q0.0 làm tín hiệu đầu vào cho timer đảm bảo Q0.0 sẽ có giá trị logic bằng 1 trong một vòng quét ở mỗi thời điểm mà giá trị đếm của bộ timer đạt được giá trị đặt trước PT:



Hình 2.17: Khắc phục ảnh hưởng của độ phân giải đến đầu ra của timer.

4. Counter

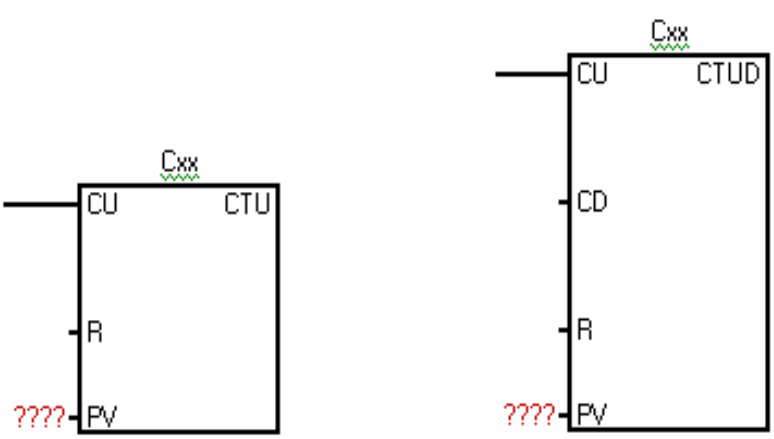
Counter là bộ đếm thực hiện chức năng đếm sườn xung trong S7-200. Các bộ đếm của S7-200 được chia ra làm hai loại: bộ đếm tiến (CTU) và bộ đếm tiến/lùi (CTUD).

Bộ đếm tiến CTU đếm số sườn lên của tín hiệu logic đầu vào, tức là đếm số lần thay đổi trạng thái từ 0 lên 1 của tín hiệu. Số sườn xung đếm được, được ghi vào thanh ghi 2 byte của bộ đếm gọi là C-word.

Nội dung của C-word gọi là giá trị đếm tức thời, luôn được so sánh với giá trị đặt trước của bộ đếm, được ký hiệu là PV. Khi giá trị đếm tức thời bằng hoặc lớn hơn giá trị đặt trước này thì bộ đếm báo ra ngoài bằng cách đặt giá trị logic 1 vào C-bit. Trường hợp giá trị đếm tức thời nhỏ hơn giá trị đặt trước thì C-bit có giá trị logic 0.

Khác với các bộ timer, các bộ đếm CTU đều có chân nối với tín hiệu điều khiển xoá để thực hiện việc đặt lại chế độ ban đầu (reset) cho bộ đếm, được ký hiệu R trong LAD hay được quy định là trạng thái logic của bit đầu tiên trong ngăn xếp STL. Bộ đếm được reset khi tín hiệu xoá này có mức logic là 1 hoặc khi lệnh R được thực hiện với C-bit. Khi bộ đếm được reset, C-word và C-bit đều có giá trị 0.

- Cú pháp hai bộ đếm CTU và CTUD của s7-200



Hình 2.18: Khai báo và sử dụng Counter.

Bộ đếm tiến/lùi CTUD đến tiến khi gặp sườn lên của xung vào cổng đếm tiến (ký hiệu CU trong LAD) hoặc bit thứ 3 của ngăn xếp trong STL và đếm lùi khi gặp sườn lên của xung vào cổng đếm lùi (ký hiệu CD trong LAD) hoặc bit thứ 2 của ngăn xếp trong STL.

- Giống như bộ đếm CTU, bộ đếm CTUD cũng được đưa về trạng thái khởi phát ban đầu bằng 2 cách:

+ Khi đầu vào của chân xoá, ký hiệu bằng R trong LAD hoặc bit thứ nhất của ngăn xếp trong STL có giá trị logic bằng 1.

+ Bằng lệnh R với C-bit.

CTUD có giá trj đếm tức thời đúng bằng giá trị đếm và được lưu trữ trong thanh ghi 2 byte C-word của bộ đếm. Giá trị đếm tức thời luôn được so sánh với giá trị đặt trước PV của bộ đếm. Nếu giá trị đếm tức thời lớn hơn hoặc bằng giá trị đặt trước thì C-bit có giá trị logic bằng 1. Còn các trường hợp khác C-bit có giá trị logic 0.

Bộ đếm tiến CTU có miền giá trị đếm tức thời từ 0 đến 32.767. Bộ đếm tiến/lùi CTUD có miền giá trị đếm tức thời là -32.768 đến 32.768.

Về nguyên lý hoạt động, có thể mô tả như sau:

4.1. Counter up (CTU)

Khai báo bộ đếm tiến theo sườn lên CU. Khi giá trị đếm tức thời C-word của Cxx lớn hơn hoặc bằng giá trị đặt trước PV, C-bit có giá trị logic bằng 1. Bộ đếm được reset khi đầu vào R có giá trị logic bằng 1. Bộ đếm ngừng khi C-word đạt giá trị cực đại bằng 32.767.

Các toán hạng

Cxx	CPU 212: 0-47
(word)	CPU214: 0-47, 80-127.
PV	VW, T, C, IW,
(word)	QW, MW, SMW, AC, AIW, hằng số, *VD, *AC

4.2. Counter up – down (CTUD)

Khai báo bộ đếm tiến/lùi, đếm tiến theo sườn lên của CU và đếm lùi theo sườn lên của CD. Khi giá trị đếm tức thời C-word lớn hơn hoặc bằng giá trị đặt trước PV, C-bit có giá trị logic bằng1.Bộ đếm ngừng đếm tiến khi C-word đạt giá trị 32,767 và ngừng đến lùi khi đạt giá trị cực tiểu -32,767, CTUD reset khi đầu vào R có giá trị logic bằng 1.

Các toán hạng:	Cxx	CPU 212: 48-63
	(word)	CPU 214: 48-79
	PV	VW, T, C, IW,
	(word)	QW, MW, SMW, AC, AIW, hằng số, *VD, *AC

Các bộ đếm được đánh số từ 0-63 (với CPU 212) hoặc từ 0-127 (với CPU 214) và ký hiệu bằng Cxx, trong đó xx là số thứ tự của bộ đếm. Ký hiệu Cxx đồng thời cũng là địa chỉ hình thức của C-word và C-bit.

Ví dụ 9:

```
LD    C48          //lệnh làm việc với C-bit của C48
LDW   C48   K2     //lệnh làm việc với C-word của C48
```

Ví dụ 10: Về bộ đếm CTUD trong LAD và STL

-Viết bằng STL

NETWORK 1 // I0.0 counts up - I0.1 counts down - I0.2 resets current value to 0

LD I0.0

LD I0.1

LD I0.2

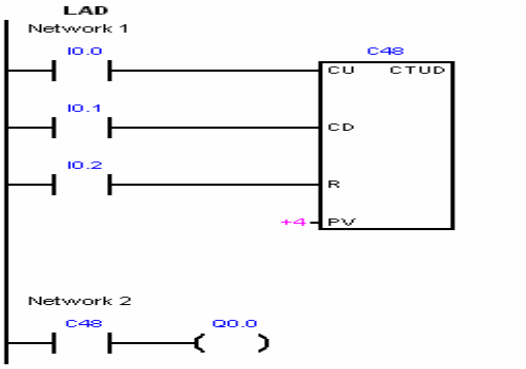
CTUD C48 +4

NETWORK 2 // Count Up/Down counter C48 turns on C48 bit when current value >= 4

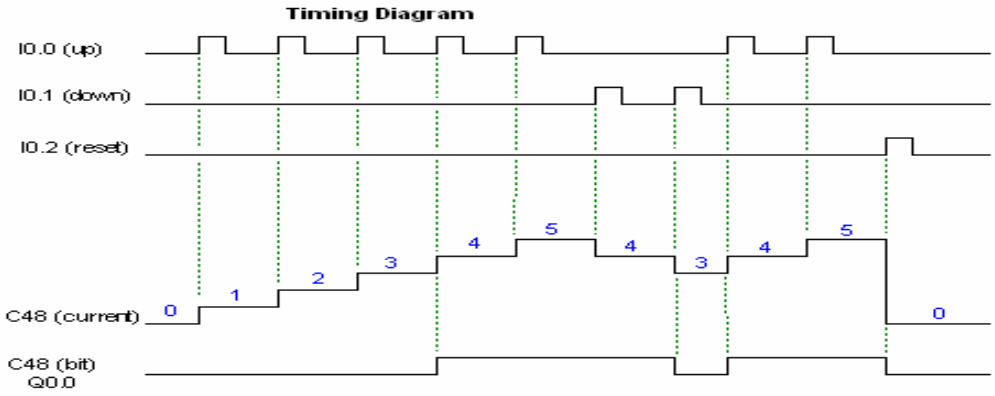
LD C48

= Q0.0

- Viết bằng LAD



- Giản đồ thời gian



Hình 2.19: Giản đồ thời gian bộ đếm CTUD

5. Bài tập ứng dụng:

1, Sử dụng phương pháp mạch tự giữ để khởi động động cơ theo phương pháp sao/tam giác.

2, Sử dụng các tập lệnh về bít để thực hiện khởi động tuần tự động cơ theo thứ tự sau:

- Khi ấn start1: động cơ 1 khởi động, ấn stop1, động cơ 1 tắt.

- Khi động cơ 1 không đủ tải, nhấn start2, động cơ 2 sẽ hoạt động, nhấn

stop2, động cơ 2 tắt (khi đã dư tải).

- Tương tự cho động cơ 3 và 4 (sẽ được khởi động khi tải tương ứng không đủ)

Trong quá trình hoạt động gặp sự cố ta có thể nhấn nút dừng khẩn cấp để dừng toàn bộ hệ thống.

3, Điều khiển cho tín hiệu đèn tại các ngã tư giao thông với 2 chế độ ngày và đêm.

BÀI 3: CÁC PHÉP TOÁN SỐ CỦA PLC

Mã bài: MĐ21.03

Giới thiệu: Trong PLC S7-200 có chứa các chức năng truyền dẫn, chức năng so sánh, chức năng toán học và có chứa chức năng đồng hồ thời gian thực

Mục tiêu của bài

- Trình bày được các phép toán so sánh, các phép toán số.
- Vận dụng các bài toán vào thực tế: Lập trình, kết nối, chạy thử...
- Rèn luyện đức tính tích cực, chủ động và sáng tạo

Nội dung chính:

1. Chức năng truyền dẫn

1.1. Truyền Byte, Word, Doubleword:

- Phép truyền Move Byte sẽ được thực hiện copy dữ liệu Byte tại ngõ vào IN và truyền tới Byte tại ngõ ra OUT.

- Phép truyền Move Word sẽ thực hiện copy dữ liệu Word tại ngõ vào In và truyền tới Word tại ngõ ra OUT

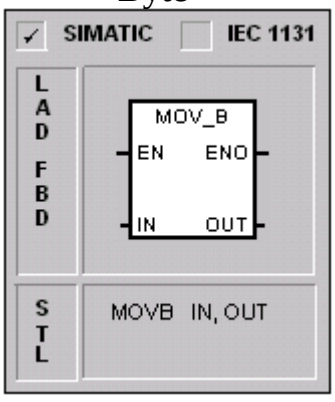
- Phép truyền Move DoubleWord sẽ thực hiện copy dữ liệu DoubleWord tại ngõ vào In và truyền tới DoubleWord tại ngõ ra OUT

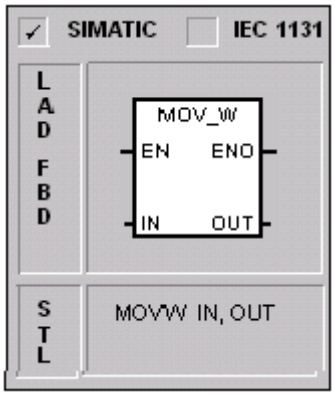
- Phép truyền Real sẽ thực hiện copy một số thực 32 bit tại DoubleWord ngõ vào IN và truyền tới DoubleWord tại ngõ ra OUT.

Khi xảy ra lỗi thì ngõ ENO bị SET = 0.

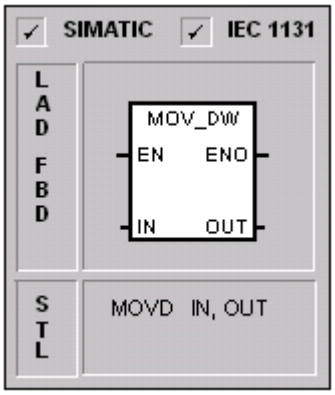
Bảng giới hạn vùng toán hạng và dạng dữ liệu hợp lệ

Block Move	Inputs/Output	Toán hạng	Dạng dữ liệu
Byte	In	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC	Byte
	Out	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC	Byte
Word	In	VW, IW, QW, MW, SW,	Word, Int

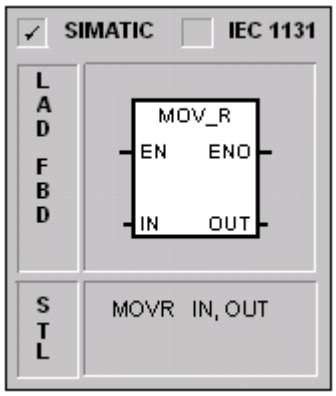




		SMW, LW, T, C, AIW, Constant, AC, *VD, *AC, *LD	
	Out	VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD	Word, Int



DoubleWord	In	VD, ID, QD, MD, SD, SMD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, &SMB, &AIW, &AQW AC, Constant, *VD, *LD, *AC	Dword, Dint
	Out	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	Dword, Dint

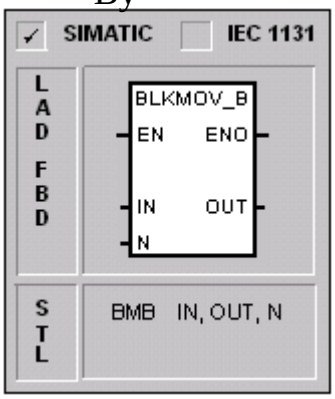
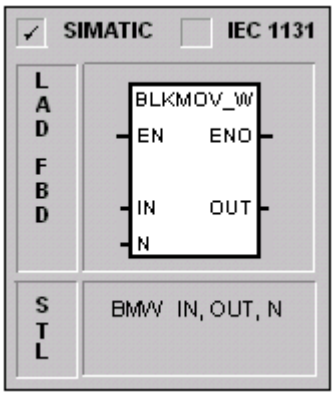
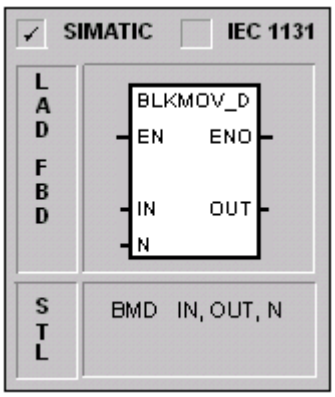


Real	In	VD, ID, QD, MD, SD, SMD, LD, AC, Constant, *VD, *LD, *AC	Real
	Out	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	Real

1.2. Truyền một vùng nhớ dữ liệu

Phép truyền Block Move Byte, Block Move Word, Block Move Doubleword sẽ thực hiện truyền một số lượng Byte (N) có địa chỉ Byte đầu tại ngõ vào IN sang vùng nhớ có địa chỉ tại ngõ ra OUT. N là số lượng Byte có giới hạn từ 1 đến 255.

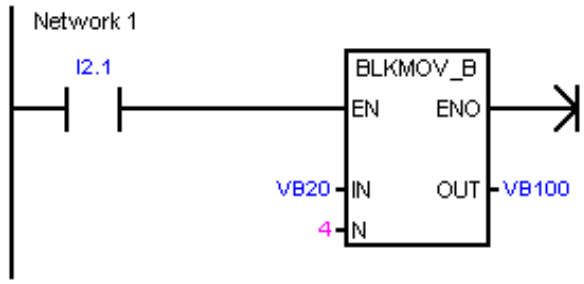
Bảng giới hạn vùng toán hạng và dạng dữ liệu hợp lệ

Block Move	Inputs/Output	Toán hạng	Dạng dữ liệu
By 	IN, OUT	VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD	Byte
	N	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD	Byte
Word 	IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, *VD, *LD, *AC	Word
	IN	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC	Byte
	OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, *VD, *LD, *AC	Word
DoubleWord 	IN, OUT	VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD	Doubleword
	N	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD	Byte

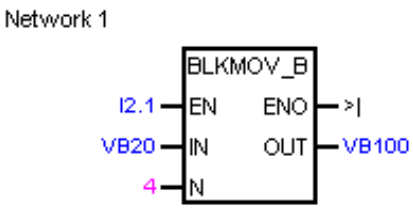
Ví dụ về truyền một mảng dữ liệu BLKMOVE:

Mảng dữ liệu thứ nhất gồm 4 byte (N=4) thuộc vùng nhớ V có địa chỉ đầu từ VB0 được truyền đến một vùng nhớ V có địa chỉ đầu từ VB100 (mảng 2). Dữ liệu tại mảng 1 vẫn không đổi.

LAD



FBD



STL

```
NETWORK 1      //Move array 1 (VB20 to VB23)
//to array 2 (VB100 to VB103)

 LD I2.1
 BMB VB20 VB100 4
```

Array 1 Data	30	31	32	33
Data Addresses	VB20	VB21	VB22	VB23
Block Move Execution				
Loads Array 2				
Array 2 Data	30	31	32	33
Data Addresses	VB100	VB101	VB102	VB103

2. Chức năng so sánh

Các phép so sánh có thể sử dụng là ==, <>, >, >=, <= và chỉ có thể áp dụng cho Byte, số nguyên I, số nguyên kép DI, và số thực R.

Dữ liệu tại ngõ vào IN1 được so sánh với dữ liệu tại ngõ vào IN2.

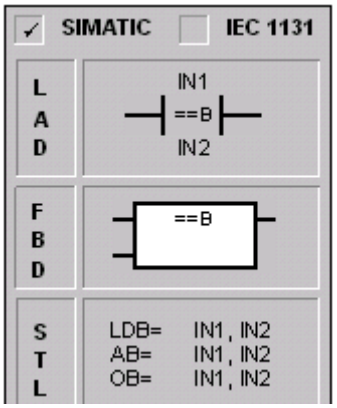
Trong lập trình LAD thì tiếp điểm lên mức logic 1 khi thỏa mãn điều kiện so sánh.

Trong lập trình STL, các lệnh LOAD, AND hoặc OR sẽ =1 khi phép so sánh là true.

2.1. So sánh Byte

Khối so sánh byte dùng để so sánh giá trị 2 byte IN1 và IN2

Lệnh so sánh bao gồm:



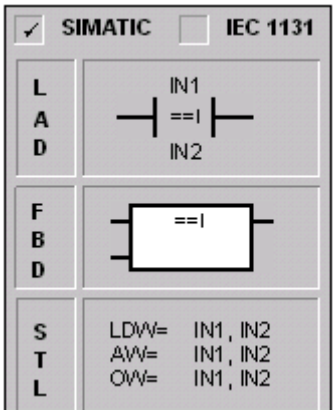
IN1 = IN2, IN1 > IN2
IN1>=IN2, IN1 < IN2
IN1<=IN2, IN1<>IN2

Bảng giới hạn toán hạng và vùng dữ liệu hợp lệ:

Inputs/Outputs	Operands	Data Types
Inputs	IB, QB, MB, SMB, VB, SB, LB, AC, Constant, *VD, *LD, *AC	BYTE
Output	I, Q, M, SM, T, C, V, S, L, Power Flow	BOOL

2.2. So sánh số nguyên Interger

Khối so sánh Interger dùng để so sánh giá trị 2 byte IN1 và IN2



Lệnh so sánh bao gồm: IN1 = IN2, IN1 > IN2
IN1>=IN2, IN1 < IN2

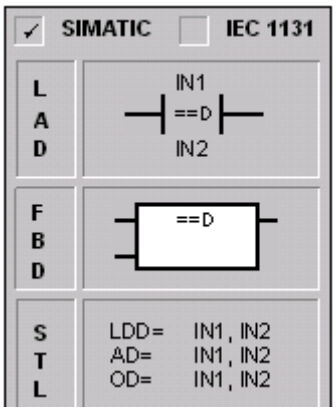
IN1<=IN2, IN1<>IN2

Bảng giới hạn toán hạng và vùng dữ liệu hợp lệ:

Inputs/Outputs	Operands	Data Types
Inputs	IW, QW, MW, SW, SMW, T, C, VW, LW, AIW, AC, Constant, *VD, *LD,*AC	INT
Output	I, Q, M, SM, T, C, V, S, L, Power Flow	BOOL

2.3. So sánh số nguyên kép Double Interger (DI)

Khối so sánh DI cũng dùng để so sánh giá trị 2 byte IN1 và IN2



Lệnh so sánh bao gồm: IN1 = IN2, IN1 > IN2
IN1>=IN2, IN1 < IN2

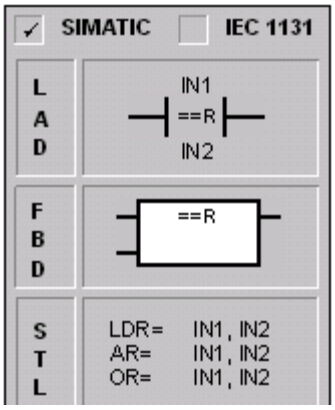
IN1<=IN2, IN1<>IN2

Bảng giới hạn toán hạng và vùng dữ liệu hợp lệ:

Inputs/Outputs	Operands	Data Types
Inputs	ID, QD, MD, SD, SMD, VD, LD, HC, AC, Constant, *VD, *LD, *AC	DINT
Output	I, Q, M, SM, T, C, V, S, L, Power Flow	BOOL

2.4. So sánh số thực Real (R)

Khối so sánh R cũng dùng để so sánh giá trị 2 byte IN1 và IN2



Lệnh so sánh bao gồm:
IN1 = IN2, IN1 > IN2
IN1>=IN2, IN1 < IN2
IN1<=IN2,IN1<>IN2

Bảng giới hạn toán hạng và vùng dữ liệu hợp lệ:

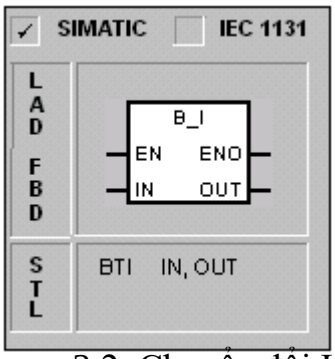
Inputs/Outputs	Operands	Data Types
Inputs	ID, QD, MD, SD, SMD, VD, LD, AC, Constant, *VD, *LD, *AC	REAL
Output	I, Q, M, SM, T, C, V, S, L, Power Flow	BOOL

3. Chức năng chuyển đổi (Converter)

3.1. Chuyển đổi Byte sang Integer

Lệnh chuyển đổi B_I chuyển đổi dữ liệu chứa trong Byte có địa chỉ tại ngõ IN sang giá trị số nguyên, kết quả chứa vào biến xác định tại ngõ ra OUT.

Bảng giới hạn vùng toán hạng và dạng dữ liệu hợp lệ:

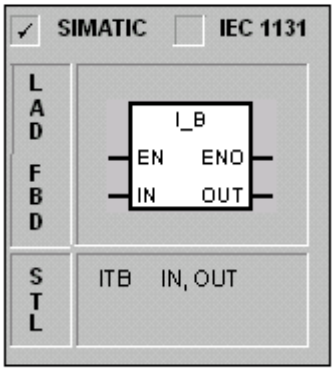


In/Out	Operands	Data Types
In	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *AC, *VD, *LD	Byte
Out	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	Int

3.2. Chuyển đổi Integer sang Byte

Lệnh chuyển đổi I_B chuyển đổi dữ liệu chứa trong Word có địa chỉ tại ngõ IN sang giá trị Byte, kết quả chứa vào biến xác định tại ngõ ra OUT. Các số nguyên có thể chuyển đổi là 0 đến 255.

Bảng giới hạn vùng toán hạng và dạng dữ liệu hợp lệ:

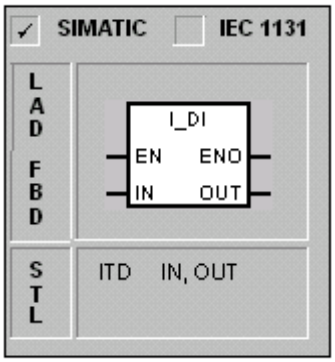


In/Out	Operands	Data Types
In	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	Int
Out	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *AC, *VD, *LD	Byte

3.3. Chuyển đổi Integer sang Double Integer

Lệnh chuyển đổi I_DI chuyển đổi giá trị số I tại ngõ IN sang một giá trị số nguyên kép DI, kết quả chứa vào biến xác định tại ngõ ra OUT.

Bảng giới hạn vùng toán hạng và dạng dữ liệu hợp lệ:



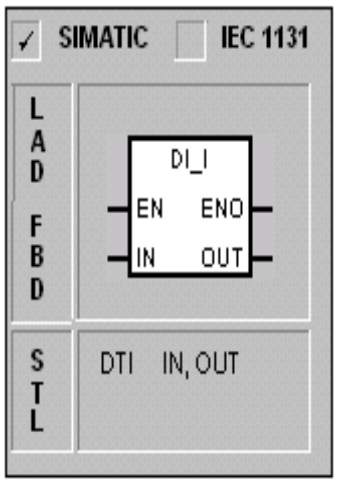
In/Out	Operands	Data Types
In	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, Constant, *VD, *LD, *AC	Int
Out	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	Dint

3.4. Chuyển đổi Double Integer sang Integer

Lệnh chuyển đổi DI_I chuyển đổi giá trị số nguyên kép DI tại ngõ IN sang một giá trị số nguyên I, kết quả chứa vào biến xác định tại ngõ ra OUT. Nếu phép

biến đổi bị tràn (kết quả lớn hơn khả năng chứa của ngõ OUT) thì ngõ ra không thay đổi và trạng thái EN)=0.

Bảng giới hạn vùng toán hạng và dạng dữ liệu hợp lệ:

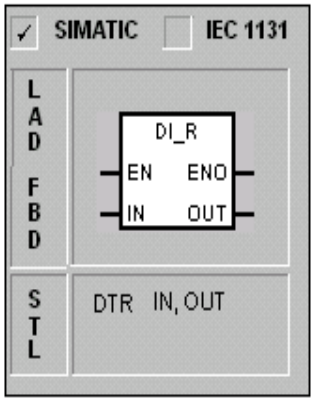


In/Out	Operands	Data Types
In	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	Dint
Out	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, Constant, *VD, *LD, *AC	Int

3.5. Chuyển đổi Double Integer sang Real

Lệnh chuyển đổi DI_R chuyển đổi chuyển đổi một số nguyên kép DI 32 bit sang một số thực R, đặt kết quả vào địa chỉ được xác định tại ngõ ra OUT.

Bảng giới hạn vùng toán hạng và dạng dữ liệu hợp lệ:



In/Out	Operands	Data Types
In	VD, ID, QD, MD, SD, SMD, LD, HC, AC, Constant, *VD, *LD, *AC	Dint
Out	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	REAL

3.6. Chuyển đổi số BCD_I và I_BCD

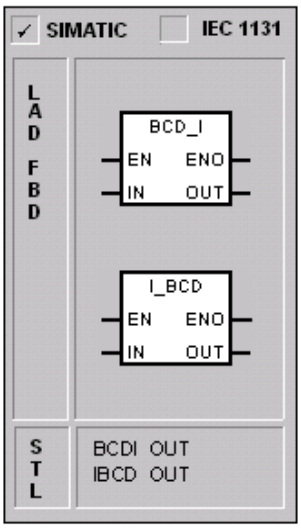
Lệnh chuyển đổi số BCD sang số Integer (BCD_I) sẽ thực hiện việc chuyển số BCD tại ngõ vào IN sang giá trị số nguyên I và chứa kết quả vào địa chỉ xác định tại ngõ ra OUT. Giá trị có thể nhập tại ngõ IN từ 0 đến 9999BCD

Khi xảy ra lỗi chuyển đổi thì trạng thái ENO=0

Lệnh chuyển đổi số I sang số BCD sẽ thực hiện việc chuyển số I tại ngõ vào IN sang giá trị số BCD và chứa kết quả vào địa chỉ xác định tại ngõ ra OUT. Giá trị có thể nhập tại ngõ IN từ 0 đến 9999 Integer.

Khi xảy ra lỗi chuyển đổi thì trạng thái ENO=0

Bảng giới hạn vùng toán hạng và dạng dữ liệu hợp lệ:



In/Out	Operands	Data Types
In	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, Constant, *VD, *AC, *LD	Word
Out	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	Word

4. Chức năng dịch chuyển

4.1. Dịch Byte

Chức năng này bao gồm dịch phải byte SHR_B và dịch trái byte SHL_B.

Các lệnh SHR_B và SHL_B sẽ dịch dữ liệu tại Byte ngõ vào IN sang phải hoặc sang trái với số vị trí dịch được nhập lại N, kết quả được chứa vào Byte ngõ ra OUT. Ở lệnh SHIFT thì tại vị trí các Bit bị dịch sẽ lấp đầy bằng số 0. Số vị trí Bit cần dịch được nhập tại ngõ N<=8.

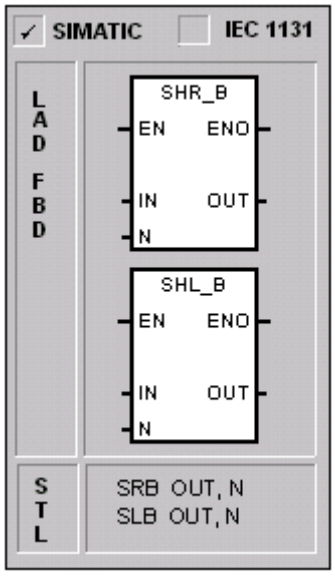
Trong trường hợp lỗi thì ENO=0

Bít đặc biệt:

SM1.0: Bít 0 được set nếu kết quả của lệnh shift là 0

SM1.1: Bit cao được set tới giá trị cuối cùng của bit được dịch

Bảng giới hạn toán hạng và vùng dữ liệu hợp lệ:



Inputs/Outputs	Operands	Data Types
In	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC	BYTE
N	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC	BYTE
Out	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC	BYTE

4.2. Dịch WORD

Chức năng này bao gồm dịch phải Word SHR_B và dịch trái Word SHL_B.

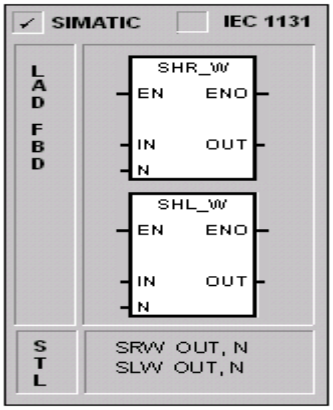
Các lệnh SHR_W và SHL_W sẽ dịch dữ liệu tại Byte ngõ vào IN sang phải hoặc sang trái với số vị trí dịch được nhập lại N, kết quả được chứa vào Word có địa chỉ tại ngõ ra OUT. Tại vị trí các Bit bị dịch sẽ lấp đầy bằng số 0. Số vị trí Bit cần dịch được nhập tại ngõ N<=16. Trong trường hợp lỗi thì ENO=0

Bít đặc biệt:

SM1.0: Bít 0 được set nếu kết quả của lệnh shift là 0

SM1.1: Bit cao được set tới giá trị cuối cùng của bit được dịch

Bảng giới hạn toán hạng và vùng dữ liệu hợp lệ:



Inputs/Outputs	Operands	Data Types
In	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, Constant, *VD, *LD, *AC	WORD
N	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC	BYTE
Out	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	WORD

4.3. Dịch Double Word

Chức năng này bao gồm dịch phải byte SHR_B và dịch trái byte SHL_B.

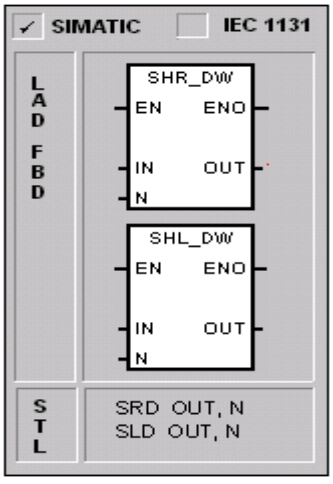
Các lệnh SHR_B và SHL_B sẽ dịch dữ liệu tại Byte ngõ vào IN sang phải hoặc sang trái với số vị trí dịch được nhập lại N, kết quả được chứa vào Byte ngõ ra OUT. Ở lệnh SHIFT thì tại vị trí các Bit bị dịch sẽ lấp đầy bằng số 0. Số vị trí Bit cần dịch được nhập tại ngõ N<=8.

Trong trường hợp lỗi thì ENO=0

Bít đặc biệt: SM1.0: Bít 0 được set nếu kết quả của lệnh shift là 0

SM1.1: Bit cao được set tới giá trị cuối cùng của bit được dịch

Bảng giới hạn toán hạng và vùng dữ liệu hợp lệ:



Inputs/Outputs	Operands	Data Types
In	VD, ID, QD, MD, SD, SMD, LD, AC, HC, Constant, *VD, *LD, *AC	DWORD
N	VB, IB, QB, MB, SB, SMB, LB, AC,	BYTE

	Constant, *VD, *LD, *AC	
Out	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	DWORD

5. Chức năng toán học

Các lệnh số học dùng để thực hiện các phép tính số học trong chương trình.

Trong LAD, bốn khối toán học (math box) thực hiện các phép tính cộng, trừ 16 bit và 32 bit. Khối nhân (multiply box) nhân hai số nguyên 16 bit và kết quả là một số nguyên 32 bít. Khối chia (divide box) chia hai số 16 bit, thương là 16 bit và dư cũng là một số 16 bit và được nạp vào từ ngay trước. Nếu lập trình bằn LAD, có thể tiết kiệm ô nhớ bằng cách sử dụng đầu vào IN1 đồng thời cũng là đầu ra OUT.

Trong STL, lệnh thực hiện bốn phép tính số học được quy định cho toán hạng 16 bit và 32 bit. Khối nhân thực hiện phép nhân hai số nguyên 16 bit và tích số là một số nguyên 32 bit. Lệnh chia thực hiện phép chia một số nguyên 16 bit với 16 bit cuối của một số nguyên 32 bit. Kết quả là một giá trị từ kép (32 bit) trong đó từ thấp (từ bit 0 đến bit 15) là thương số và từ cao (từ bit 16 đến 32 bit) là số dư của phép tính.

5.1. Phép cộng trừ (ADD và SUB).

a, Phép cộng số nguyên 16 bit

ADD_I (LAD)

+I (STL)

Lệnh thực hiện phép cộng các số nguyên 16_bit IN1 và IN2. Trong LAD kết quả là một số nguyên 16 bit được ghi vào OUT, tức là IN1 + IN2 = OUT.

Trong STL, kết quả cũng là một giá trị 16 bit nhưng được ghi lại vào IN2, tức là IN1 + IN2 = IN2.

b, Phép trừ số nguyên 16 bit

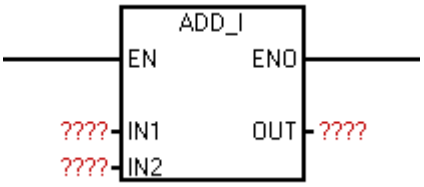
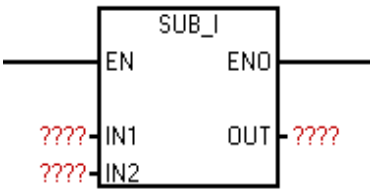
SUB_I (LAD)

-I (STL)

Lệnh thực hiện phép trừ các số nguyên 16_bit IN1 và IN2. Trong LAD kết quả là một số nguyên 16 bit được ghi vào OUT, tức là IN1 - IN2 = OUT.

Trong STL, kết quả cũng là một giá trị 16 bit nhưng được ghi lại vào IN2, tức là IN1 - IN2 = IN2.

Cú pháp dùng lệnh cộng trừ hai số nguyên 16 bit trong LAD và STL như sau:

LAD	STL	Toán hạng
	+I IN1 IN2	IN1, IN2 VW, IW, QW, MW, (INT) W, SMW, T, C, AC, LW, AIW, Constant, *VD, *LD, *AC OUT VW, IW, QW, MW, (INT) SW, SMW, T, C, LW, AC, *VD, *LD, *AC
	-I IN1 IN2	

c, Phép cộng số nguyên kép 32 bit

ADD_DI (LAD)

+D (STL)

Lệnh thực hiện phép cộng các số nguyên 32 bit IN1 và IN2. Trong LAD kết quả là một số nguyên 32 bit được ghi vào OUT, tức là IN1+IN2=OUT

Trong STL kết quả cũng là một số nguyên 32 bit nhưng được ghi lại vào IN2, tức là IN1+IN2=IN2.

d, Phép trừ số nguyên kép 32 bit

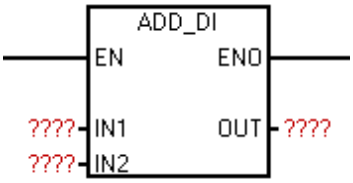
SUB_DI (LAD)

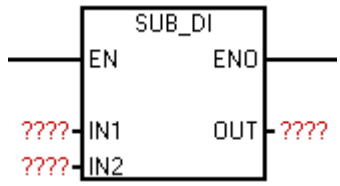
-D (STL)

Lệnh thực hiện phép trừ các số nguyên 32 bit IN1 và IN2. Trong LAD kết quả là một số nguyên 32 bit được ghi vào OUT, tức là IN1-IN2=OUT

Trong STL kết quả cũng là một số nguyên 32 bit nhưng được ghi lại vào IN2, tức là IN1- IN2=IN2.

Cú pháp dùng lệnh cộng trừ hai số nguyên 32 bit trong LAD và STL như sau:

LAD	STL	Toán hạng
	+D IN1 IN2	IN1, IN2 VD, ID, QD, MD, SMD, (DINT) SD, LD, AC, HC, Constant, *VD, *LD, *AC

	-D IN1 IN2	OUT VD, ID, QD, MD, SMD, (DINT) SD, LD, AC, *VD, *LD,*AC

e, Phép cộng số thực kép 32 bit

ADD_R (LAD)

+R (STL)

Lệnh thực hiện phép cộng các số thực 32 bit IN1 và IN2. Trong LAD kết quả là một số nguyên 32 bit được ghi vào OUT, tức là IN1+IN2=OUT

Trong STL kết quả cũng là một số nguyên 32 bit nhưng được ghi lại vào IN2, tức là IN1+IN2=IN2.

f, Phép trừ số thực kép 32 bit

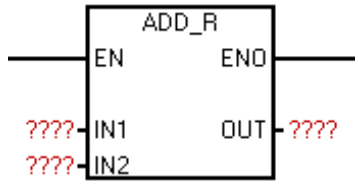
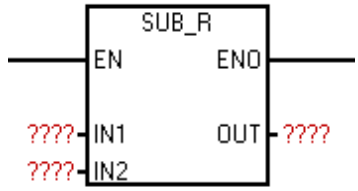
SUB_R(LAD)

-R (STL)

Lệnh thực hiện phép cộng các số thực 32 bit IN1 và IN2. Trong LAD kết quả là một số nguyên 32 bit được ghi vào OUT, tức là IN1- IN2=OUT

Trong STL kết quả cũng là một số thực 32 bit nhưng được ghi lại vào IN2, tức là IN1- IN2=IN2.

Cú pháp dùng lệnh cộng trừ hai số thực 32 bit trong LAD và STL như sau:

LAD	STL	Toán hạng
	+D IN1 IN2	IN1, IN2 VD, ID, QD, MD, SD, SMD, (REAL) LD, AC, Constant, *VD, *LD, *AC OUT VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC
	-D IN1 IN2	

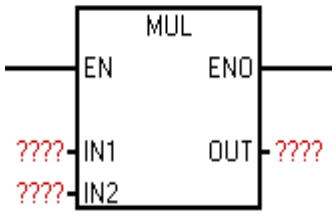
5.2. Phép nhân chia (MUL và DIV).

a, Phép nhân

Trong LAD: lệnh thực hiện phép nhân hai số nguyên 16 bit IN1 và IN2 và cho ra kết quả 32 bit chứa trong từ kép OUT (4 bytes).

Trong STL: lệnh thực hiện phép nhân hai số nguyên 16 bit n1 và số nguyên chứa trong từ thấp (từ bit 0 đến bit 15) của toán hạng 32 bit n2 (4 bytes). Kết quả 32 bit được ghi lại vào n2.

Cú pháp dùng lệnh nhân hai số nguyên trong LAD và STL như sau:

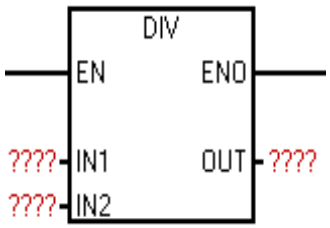
LAD	STL	Toán hạng
	MUL n1 n2	n1, IN1, IN2:VW, IW, QW, MW, SW, (INT) SMW, T, C, LW, AC,AIW, Constant, *VD, *LD, *AC n2, OUT: VD, ID, QD, MD, SMD, (DINT) SD, LD, AC, *VD, *LD,*AC

b, Phép chia

Trong LAD: lệnh thực hiện phép chia hai số nguyên 16 bit IN1 và IN2, cho ra kết quả 32 bit chứa trong từ kép OUT (*4 bytes*) gồm thương số ghi trong mảng 16 bit từ 0 đến 15 (*từ thấp*) và phần dư cũng gồm 16 bit trong mảng từ bit 16 đến 31 (từ cao).

Trong STL: lệnh thực hiện phép nhân hai số nguyên 16 bit n1 và số nguyên 16 bit nằm trong từ thấp (*từ bit 0 đến bit 15*) của toán hạng 32 bit n2 (*4 bytes*). Kết quả 32 bit được ghi lại vào n2 bao gồm thương số ghi trong bảng 16 bit từ 0 đến bit 15 (*từ thấp*) và phần dư ghi trong mảng 16 bit từ bit 16 đến bit 31 (*từ cao*).

Cú pháp dùng lệnh chia hai số nguyên trong LAD và STL như sau:

LAD	STL	Toán hạng
	DIV n1 n2	n1, IN1, IN2:VW, IW, QW, MW, SW, (INT) SMW, T, C, LW, AC, AIW, Constant, *VD, *LD, *AC n2, OUT: VD, ID, QD, MD, SMD, (DINT) SD, LD, AC, *VD, *LD, *AC

Tương tự, ta có các lệnh nhân chia sau

MUL_I: nhân hai số nguyên 16 bit

DIV_I: chia hai số nguyên 16 bit

MUL_DI: Nhân hai số nguyên 32 bit.

DIV_DI: Chia hai số nguyên 32 bit.

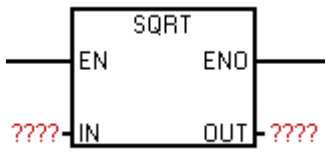
MUL_R: Nhân hai số thực.

DIV_R: Chia hai số thực.

5.3. Phép lấy căn bậc hai (SQRT)

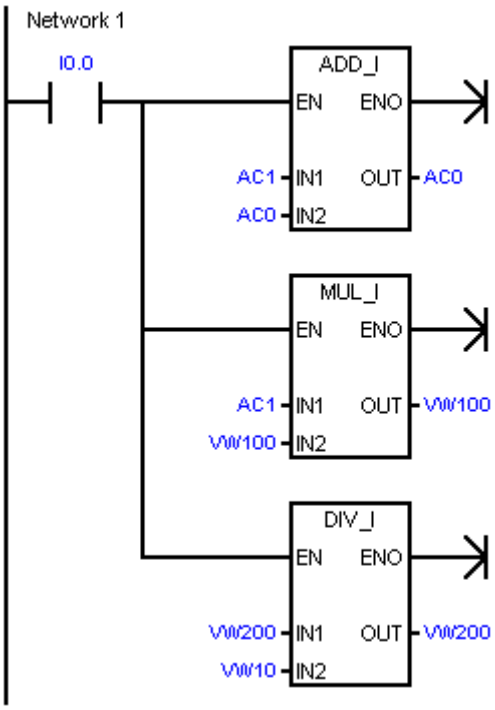
Lệnh thực hiện phép lấy căn bậc hai của số thực 32 bit IN. Kết quả cũng là một số 32 bit được ghi vào từ kép OUT (4 bytes).

Cú pháp dùng lệnh lấy căn bậc hai hai số nguyên trong LAD và STL như sau:

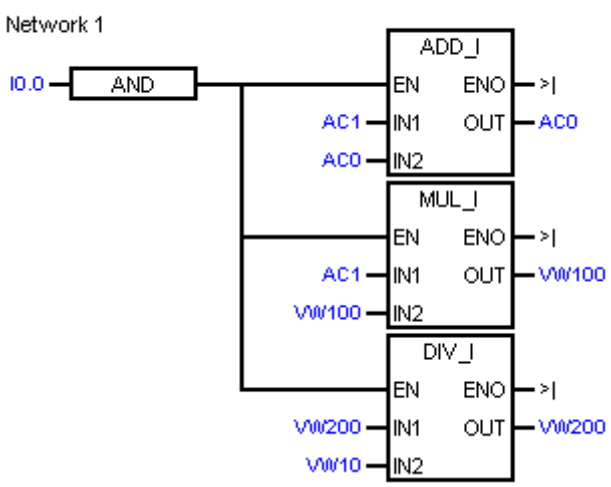
LAD	STL	Toán hạng
	SQRT IN OUT	IN: VD, ID, QD, MD, SMD, SD, LD, (REAL) AC, Constant, *VD, *LD, *AC OUT: VD, ID, QD, MD, SMD, SD (REAL) LD, AC, *VD, *LD, *AC

Ví dụ minh họa cách sử dụng một số lệnh số học:

LAD



FBD



STL

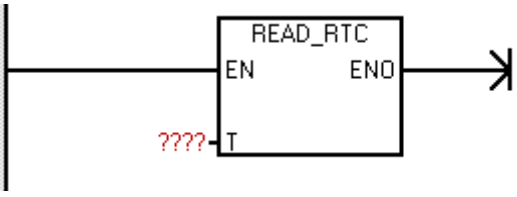
```
NETWORK 1
LD I0.0
+I AC1 AC0
*I AC1 VW100
/I VW10 VW200
```

Khi I0.0 ON, chương trình thực thi:

	IN1		IN2	OUT
Add Data	40	+	60	100
Data Address	AC1		AC0	AC0
Multiply Data	40	*	20	800
Data Address	AC1		VW102	VW100
Divide Data	4000	/	40	100
Data Address	VW200		VW10	VW200

6. Đồng hồ thời gian thực

6.1. Lệnh đọc thời gian thực Read_RTC



Bit EN: Bit cho phép đọc thời gian thực

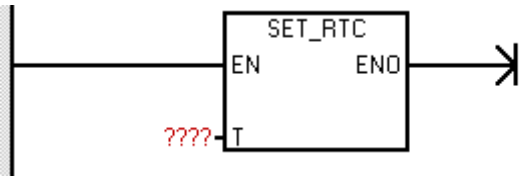
T (8 bit): VB,IB, QB, MB, SB, LB, *AC, *VD, *Ld

Được định dạng như sau:

T (byte)	Giá trị (định dạng BCD)
0	Năm (0-99)
1	Tháng (1 - 12)
2	Ngày (1 - 31)
3	Giờ (0 - 23)
4	Phút (0 - 59)
5	Giây (0 - 59
6	Trống 00
7	Ngày trong tuần (1- 7; 1- chủ nhật)

6.2. Lệnh set thời gian thực Set_R:

Khi có tín hiệu EN thì thời gian thực sẽ được set lại thông qua T. Cách định dạng Byte T hoàn toàn giống ở trên



Lệnh set thời gian thực Set_RTC: Khi có tín hiệu EN thì thời gian thực sẽ được set lại thông qua T. Cách định dạng Byte T hoàn toàn giống ở trên.

T (byte)	Giá trị (định dạng BCD)
0	Năm (0-99)
1	Tháng (1 - 12)
2	Ngày (1 - 31)
3	Giờ (0 - 23)
4	Phút (0 - 59)
5	Giây (0 - 59
6	Trống 00
7	Ngày trong tuần (1- 7; 1- chủ nhật)

BÀI 4: XỬ LÝ TÍN HIỆU ANALOG

Mã bài: MĐ21.04

Giới thiệu: Trong PLC S7-200 xử lý tín hiệu Analog thông qua bộ chuyển đổi ADC và DAC.

Mục tiêu:

- Trình bày được các bộ chuyển đổi đo.

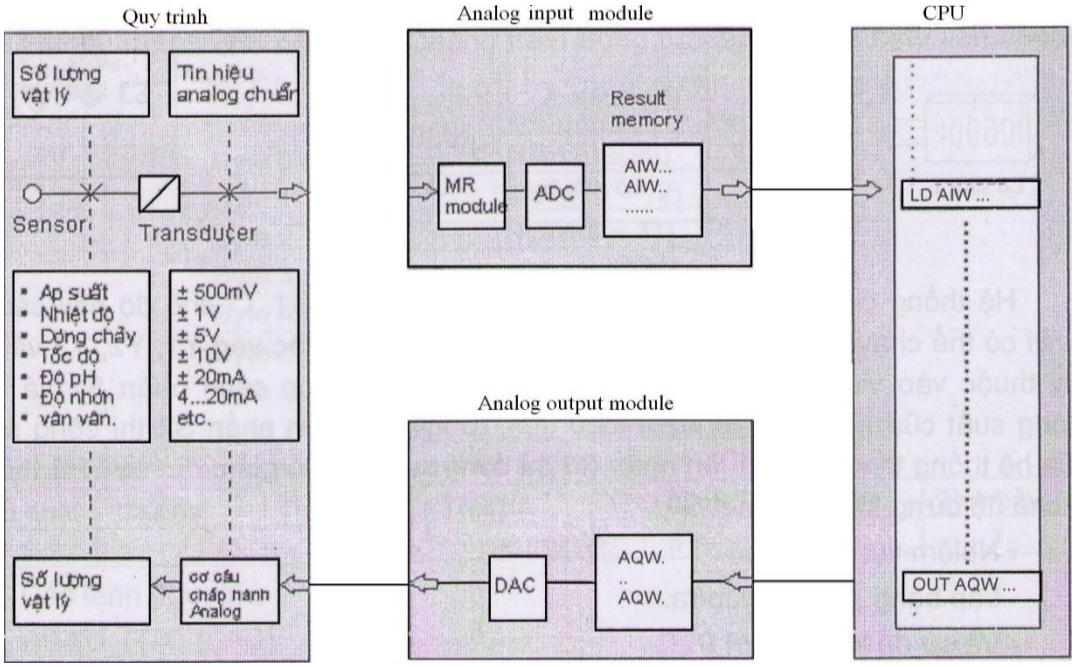
- Vận dụng các bài toán vào thực tế: Lập trình, kết nối, chạy thử...

- Rèn luyện đức tính tích cực, chủ động và sáng tạo

Nội dung chính:

1. Tín hiệu Analog

Trong quá trình điều khiển một hệ thống tự động hóa có thể có các yêu cầu điều khiển liên quan đến việc xử lý các tín hiệu analog. Các đại lượng vật lý như: nhiệt độ, áp suất, tốc độ, dòng chảy, độ PH… cần phải được các bộ Transducer chuẩn hóa tín hiệu trong phạm vi định mức cho phép trước khi nối tín hiệu vào ngõ vào analog. Ví dụ: chuẩn của tín hiệu điện áo là từ 0 đến 10VDC hoặc chuẩn của tín hiệu analog là dòng từ 4 đến 20mA. Các Module ngõ vào analog (Al) bên trong có các bộ chuyển đổi ADC (Analog Digital Converter) để chuyển đổi các tín hiệu analog nhận được thành số đưa về CPU qua Bus dữ liệu. Các module ngõ ra (AO) bên trong có bộ chuyển đổi DAC (Digital Analog Converter) chuyển các tín hiệu số nhận được từ CPU ra các giá trị analog có thể là áp hoặc dòng.



Hình 4.1: Sơ đồ khối hệ thống có sử dụng tín hiệu analog.

2. Biểu diễn các giá trị Analog

Mỗi một tín hiệu ngõ vào Analog sau khi qua bộ chuyển đổi ADC trong Module Al được chuyển thành các số nguyên Integer 16 bit có giá trị từ 0 đến ±27648. Do đó, địa chỉ vùng nhớ chứa giá trị này là 1 Word. Độ chính xác của phép chuyển đổi này phụ thuộc vào Module Analog hiện có, phạm vi độ phân giải là từ 8 đến 15 bit. Module Analog có độ phân giải càng cao thì giá trị chuyển đổi càng chính xác. Việc chuyển đổi từ tín hiệu Analog sang tín hiệu số là tỷ lệ thuận và có dạng đường thẳng. Các giá trị Analog sau khi được chuyển đổi thành giá trị số sẽ được chứa vào 1 word 16 Bit và lấp đầy các bit trong word này theo thứ tự từ bên trái sang. Các bit trống sẽ được lấp đầy bằng số 0. (Chú ý bit thứ 15 là bit dấu:=0 khi giá trị chuyển đổi là số nguyên dương và =1 khi giá trị chuyển đổi là số nguyên âm).

Bit số		Các đơn vị		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Giá trị bit		Dec.	Hex.	VZ	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Độ phân giải tính bằng bit có dấu	8	128	80	*	*	*	*	*	*	*	*	1	0	0	0	0	0	0	0
	9	64	40	*	*	*	*	*	*	*	*	*	1	0	0	0	0	0	0
	10	32	20	*	*	*	*	*	*	*	*	*	*	1	0	0	0	0	0
	11	16	10	*	*	*	*	*	*	*	*	*	*	*	1	0	0	0	0
	12	8	8	*	*	*	*	*	*	*	*	*	*	*	*	1	0	0	0
	13	4	4	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	0
	14	2	2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0
	15	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1

* = 0 or 1

Hình 4.2: Độ phân giải của module Analog.

3. Kết nối ngõ vào/ra Analog

Để đảm bảo tín hiệu Analog có được độ chính xác cao và ổn định cần tuân thủ các điều kiện sau:

+ Đảm bảo rằng điện áp 24VDC cấp nguồn cho Sensor không bị ảnh hưởng bởi nhiễu và ổn định.

+ Định tỷ lệ cho module (được mô tả bên dưới).

+ Dây nối cho Sensor chần để ngắn nhất tới mức có thể.

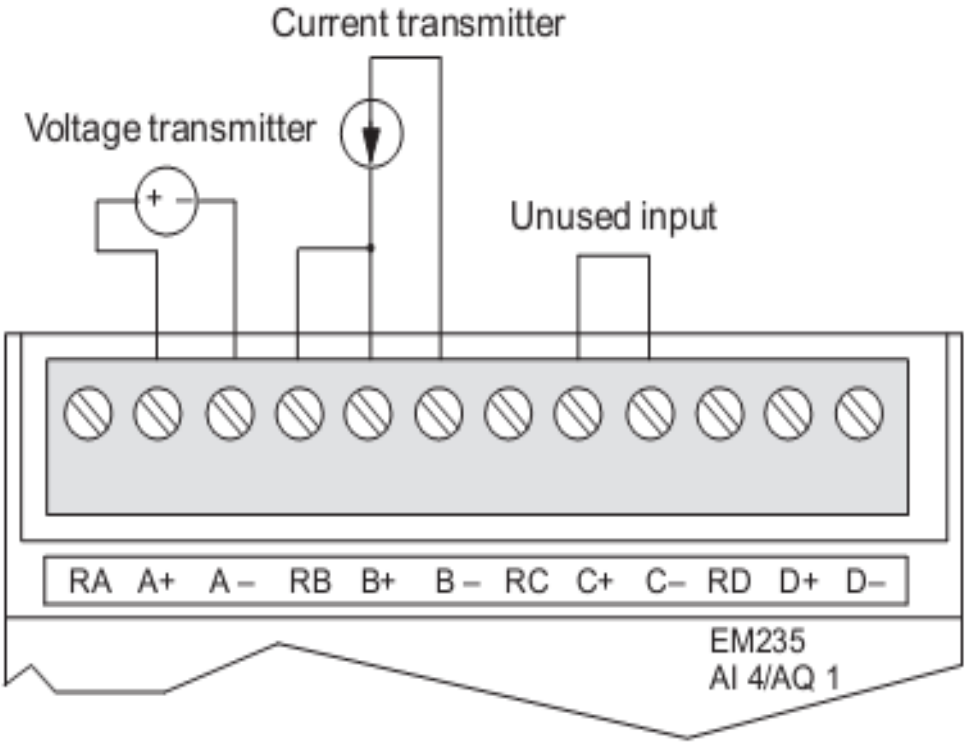
+ Sử dụng cáp đôi dây xoắn cho sensor.

+ Tất cả các ngõ vào không sử dụng phải nối tắt.

+ Tránh bẻ cong dây dẫn thành những góc nhọn.

+ Sử dụng máng đi dây hay các ống đi dây cho tuyến dây.

+ Tránh đặt các đường dây tín hiệu Analog gần với các đường dây có điện áp cao, nếu hai đường dây này cắt nhau phải đặt chúng vuông góc với nhau.



Hình 4.3: Sơ đồ kết nối module Analog.

- Phương pháp định tỷ lệ ngõ vào Analog:

Việc định tỷ lệ ngõ vào Analog có ảnh hưởng đến tất cả các ngõ vào của

modul EM và AI. Để định tỷ lệ ngõ vào một cách chính xác, cần sử dụng một chương trình thiết kế để tính trung bình các giá trị đọc được từ module. Có thể sử dụng Analog Input Filtering wizard trong STEP 7 Micro/win để tạo ra chương trình này. Nên sử dụng 64 giá trị lấy mẫu hoặc hơn để tính giá trị trung bình của tín hiệu Analog.

Để thực hiện việc định tỷ lệ cần theo các bước sau:

+ Tắt nguồn cung cấp cho module, chọn phạm vi ngõ vào mong muốn.

+ Cấp nguồn lại cho CPU và module có AI.

+ Sử dụng một Transmiter, một nguồn áp, hay một nguồn dòng và đặt giá trị 0 cho một trong các ngõ vào.

+ Đọc giá trị mà CPU nhận được tại ngõ vào tương ứng.

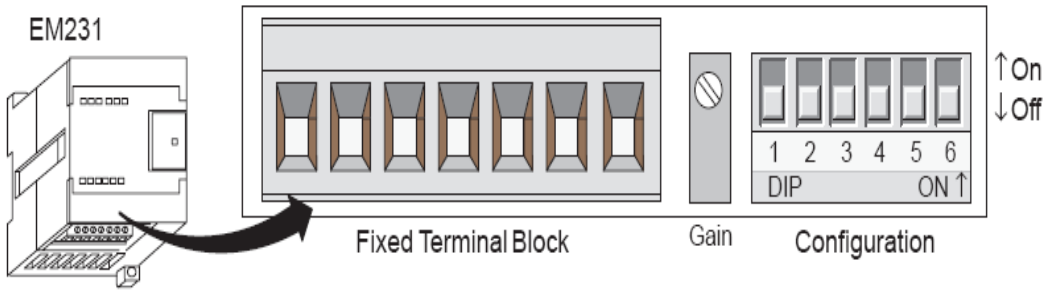
+ Điều chỉnh biến trở đặt lại giá trị offset cho tới khi giá trị đọc được là 0.

+ Điều chỉnh để tăng giá trị đặt vào tới mức và xem giá trị mà CPU nhận được.

+ Điều chỉnh biến trở GAIN cho tới khi giá trị nhận được là 32000 hoặc tới 1 giá trị số mong muốn.

+ Lặp lại các bước trên nếu cần.

Trên hình 4. là cách đặt hệ tỉ lệ cho modul Analog



Hình 4.4: Điều chỉnh tỉ lệ ngõ vào của module Analog .

- Điều chỉnh các Swich và biến trở điều chỉnh GAIN

Việc chính định các công tắc (Switch) trên module Analog EM sẽ thay đổi các phạm vi đo lường định mức và độ phân giải của module.

Sơ đồ công tắc, chỉnh định phạm vi đo định mức và độ phân giải phụ thuộc vào từng module Analog. Các thông tin này được lấy từ sổ tay phần cứng của module.

Bảng 4.1: Dải đầu vào và độ phân giải tương ứng với vị trí của switch.

Vị trí của Switch						Dải đầu vào	Độ phân giải
1[1]	3	5	7	9	11		

ON	ON	OFF	ON	OFF	OFF	0 đến 50mV	12.5µV
ON	ON	OFF	OFF	ON	OFF	0 đến 100mV	25µV
ON	OFF	ON	ON	OFF	OFF	0 đến 500mV	125µV
ON	OFF	ON	OFF	ON	OFF	0 đến 1V	250µV
ON	OFF	OFF	ON	OFF	OFF	0 đến 5V	12.5mV
ON	OFF	OFF	ON	OFF	OFF	0 đến 20mA	5µV
ON	OFF	OFF	OFF	ON	OFF	0 đến 10V	2.5mV
OFF	ON	OFF	ON	OFF	OFF	±25mV	12.5µV
OFF	ON	OFF	OFF	ON	OFF	±50mV	25µV
OFF	ON	OFF	OFF	OFF	ON	±100mV	50µV
OFF	OFF	OFF	ON	OFF	OFF	±250mV	125µV
OFF	OFF	ON	OFF	ON	OFF	±500mV	250µV
OFF	OFF	ON	OFF	OFF	ON	±1V	500µV
OFF	OFF	OFF	ON	OFF	OFF	±2.5V	1.25mV
OFF	OFF	OFF	OFF	ON	OFF	±5V	2.5mV

4. Hiệu chỉnh tín hiệu Analog

Module analog thường có nhiều tầm đo khác nhau, tín hiệu ngõ vào có thể là dòng điện hoặc điện áp. Việc chuyển đổi từ tầm đo này sang tầm đo khác thiết quả chuyển đổi thường có những sai số nhất định do cấu trúc của mạch chuyển đổi. Do vậy thông thường khi sử dụng module analog, người lập trình cần phải hiệu chỉnh trước khi sử dụng để kết quả chuyển đổi được chính xác hơn. Dưới đây trình bày việc hiệu chỉnh cho ngõ vào là điện áp, tầm đo 10V, ngõ vào chuyển đổi là AIW0.

- Cấp điện cho module analog hoạt động khoảng 10 phút.

- Chọn điện áp vào là 10V (độ phân giải 2,5mV)

- Chỉnh biến trở tại ngõ vào AIW0 để ngõ vào đạt giá trị 0V.

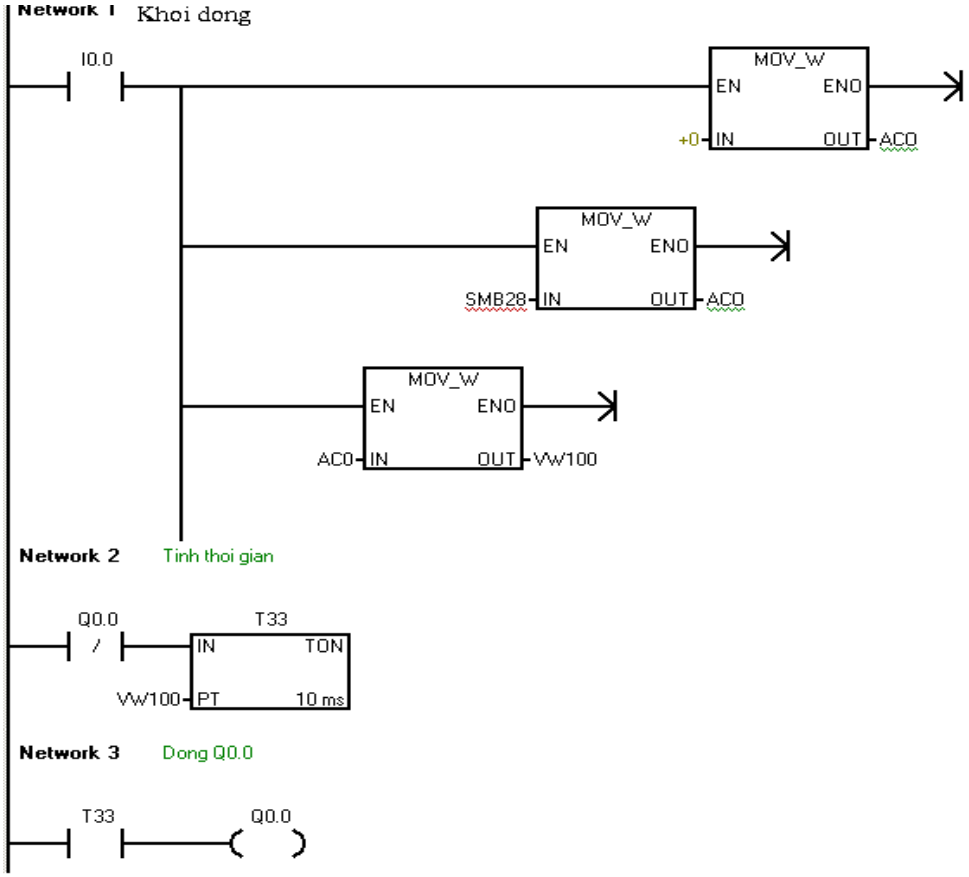
- Dùng chương trình đọc giá trị analog vào và quan sát giá trị. Nếu chưa bằng không thì hiệu chỉnh độ lợi (Gain) để đạt giá trị = 0.

- Chỉnh biến trở tại ngõ vào AIW0 để ngõ vào đạt giá trị 10V.

- Dùng chương trình đọc giá trị analog vào và quan sát giá trị. Nếu chưa bằng 32000 thì hiệu chỉnh độ lợi (Gain) để đạt giá trị = 32000.

Byte nhớ SMB 28 lưu trữ giá trị só biểu diễn vị trí chỉnh 0. SMB 29 lưu trữ giá trị số biểu diễn vị trí chỉnh 1. Sự điều chỉnh Analog có giá trị giới hạn từ 0 tới 255 và độ tin cậy tốt nhất trong phạm vi từ 10 đến 200.

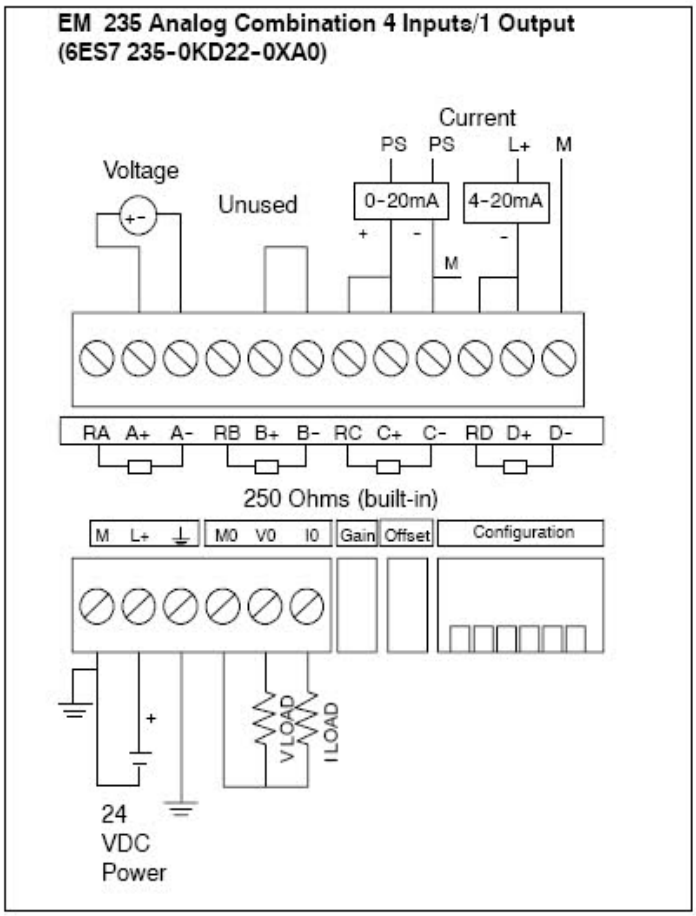
Để thực hiện việc điều chỉnh này, ta dùng 1 tuốc nơ vít nhỏ để xoay biến trở sang phải hoặc sang trái để tăng hoặc giảm giá trị.



Hình 4.5: Chương trình điều khiển module EM 235.

5. Giới thiệu về module Analog PLC S7-200

5.1. Module analog EM235



Hình 4.6: Module EM235 và cách đấu nối.

a, Đầu vào của module EM235

- Ngõ vào tương tự: 4 AI, DC +/- 10V

- Ngõ ra tương tự: 1 AO, DC +/- 10V 12 bit converter

Dải đầu vào/trở kháng đầu vào:

- 0 đến 50 mV; 0 đến 100 mV; 0 đến 500 mV; 0 đến 1V; 0 đến 5 V; 0 đến 10 V; 0 đến 20 mA; +/- 25 mV; +/- 50 mV; +/- 100 mV; +/- 200 mV; +/- 500 mV; +/- 1 V; +/- 2.5 V; +/-5 V; +/- 10V độ phân giải12 bit converter

Thời gian biến đổi tương tự sang số: <250us

b, Đầu ra của module EM235

- Số đầu ra: 1

Dải đầu ra:

- Dòng: 0 đến 20 mA
- Áp : -10 đến +10 V

Độ phân giải:

- Đầu ra áp: 12 bit
- Đầu ra dòng: 11 bit

Dải giá trị biến đổi:

- Tín hiệu đơn cực: 0 đến 32 000
- Tín hiệu hai cực: - 32000 đến + 32000

Công suất: 2W

5.2. Đọc tín hiệu Analog

Tín hiệu analog là tín hiệu tương tự (0 – 10VDC , hoặc 4 – 20mA …) hầu hết các ứng dụng của chương trình PLC siemens nói riêng hay các ứng dụng khác đều cần phải đọc các tín hiệu analog. Tín hiệu analog có thể là tín hiệu từ các cảm biến đo khoảng cách, cảm biến áp suất , cảm biến đo trọng lượng…Các bước đọc tín hiệu analog:

Đọc tín hiệu analog từ modul EM235

Các tín hiệu có thể đọc được thông qua modul EM 235 (tùy theo switch chọn trên modul):

+ Tín hiệu đơn cực: 0 – 50mV, 0 – 100mA, 0 – 500mV, 0 – 1V, 0 – 5 VDC, 0 – 20mA, 0 – 10 VDC.

+ Tín hiệu lưỡng cực: ±25mV, ±50mV, ±100µV, ±250µV, ±500µV, ±1VDC, ±2.5VDC, ±5VDC, ±10VDC.

BÀI 5: PLC CỦA CÁC HÃNG KHÁC

Mã bài: MĐ21.05

Giới thiệu: Ngoài PLC S7-200, giáo trình giới thiệu thêm một số các loại PLC: PLC của hãng OMRON, PLC của hãng SIEMENS loại S7-300

Mục tiêu:

- Trình bày nguyên lý, cấu tạo của các họ PLC Omron, ...
- Thực hiện lập trình của các họ PLC nói trên.
- Rèn luyện đức tính tích cực, chủ động và sáng tạo

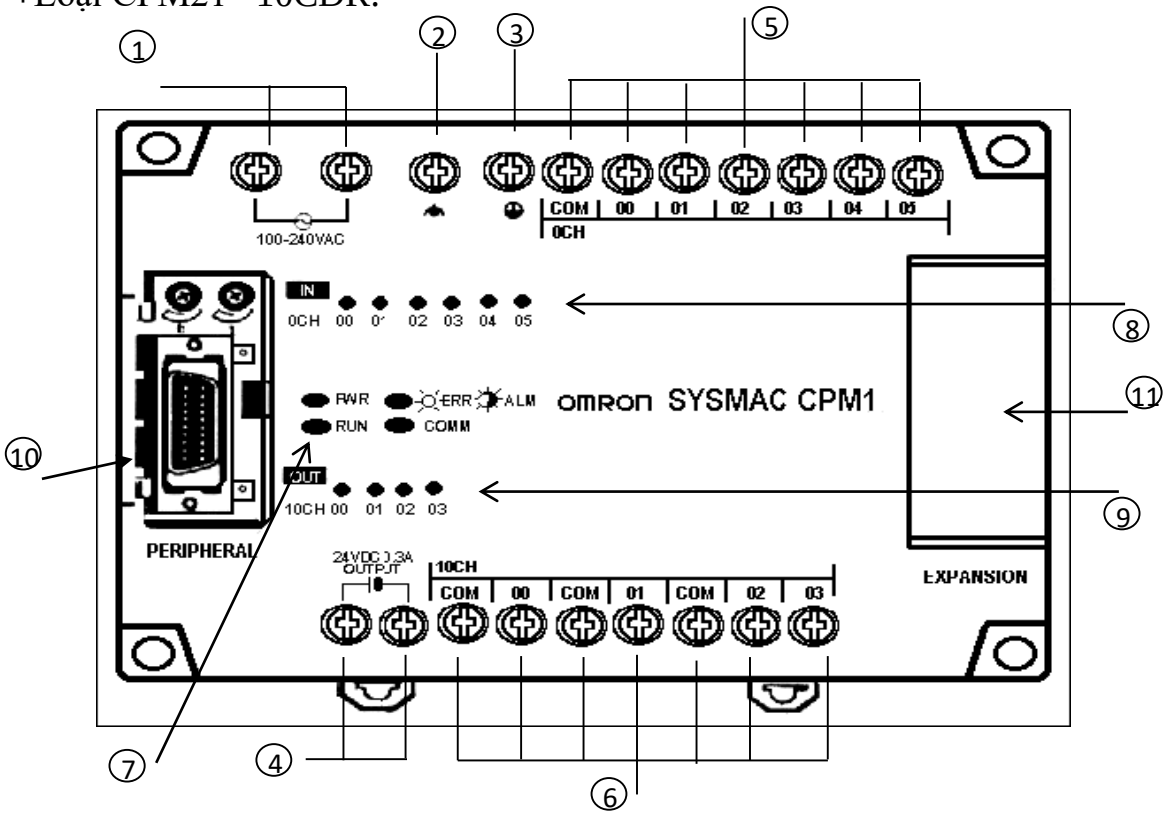
Nội dung chính:

1. PLC của hãng Omron:

1.1. Cấu trúc của một PLC Ômron

a. Các họ của PLC Ômron CPM1

+Loại CPM21– 10CDR:



Hình 5.1: Cấu hình cứng PLC CPM1- 10CDR

Trong đó:

1. Nguồn cấo cho PLC

2,3. Các đầu nối mát và nối đất

4. Nguồn ra 24V một chiều

5. Các đầu vào

6. Các đầu ra

7. Các đèn báo

8 Các đèn báo đầu vào

9. Các đèn báo đầu ra

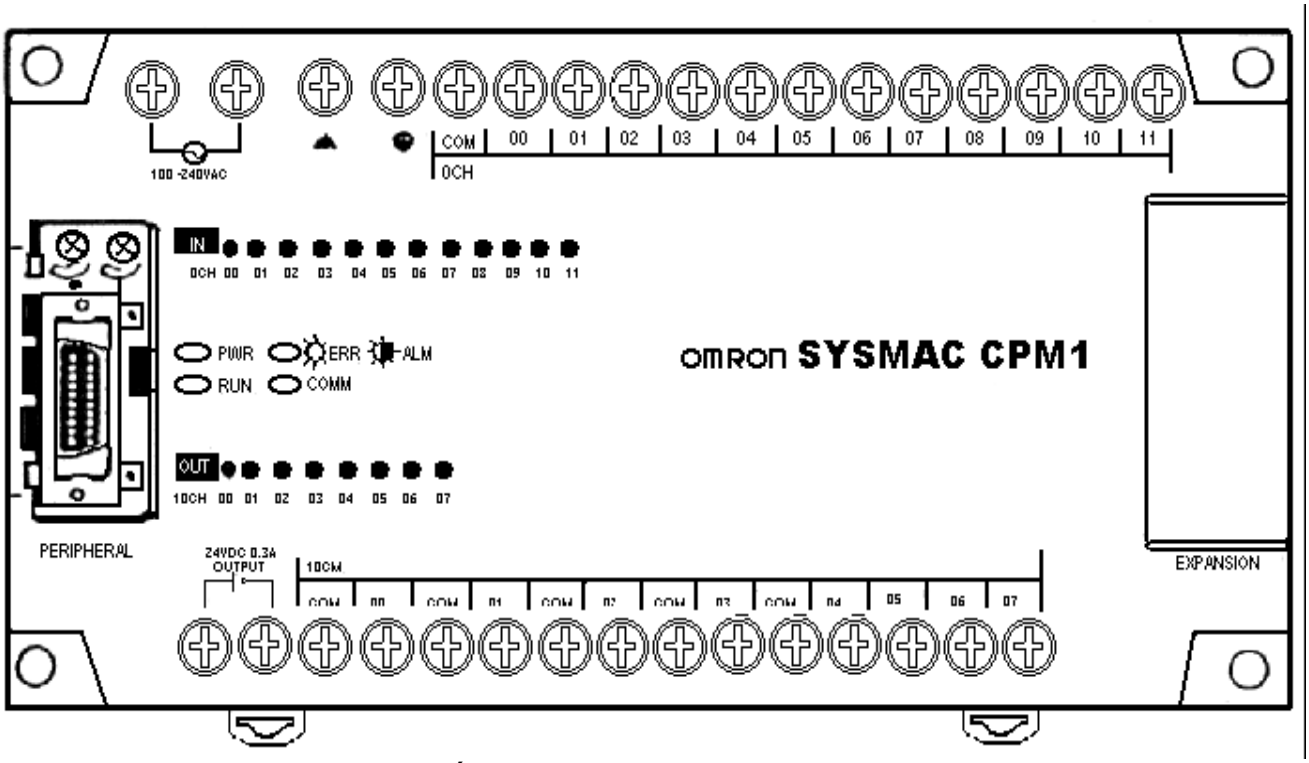
10. Cổng kết nối với máy vi tính

11. Cổng kết nối với mô dul mở rộng

Loại này có 6 đầu vào là: 00, 01, 02, 03, 04, 05.

Và 4 đầu ra là: 00, 01, 02, 03.

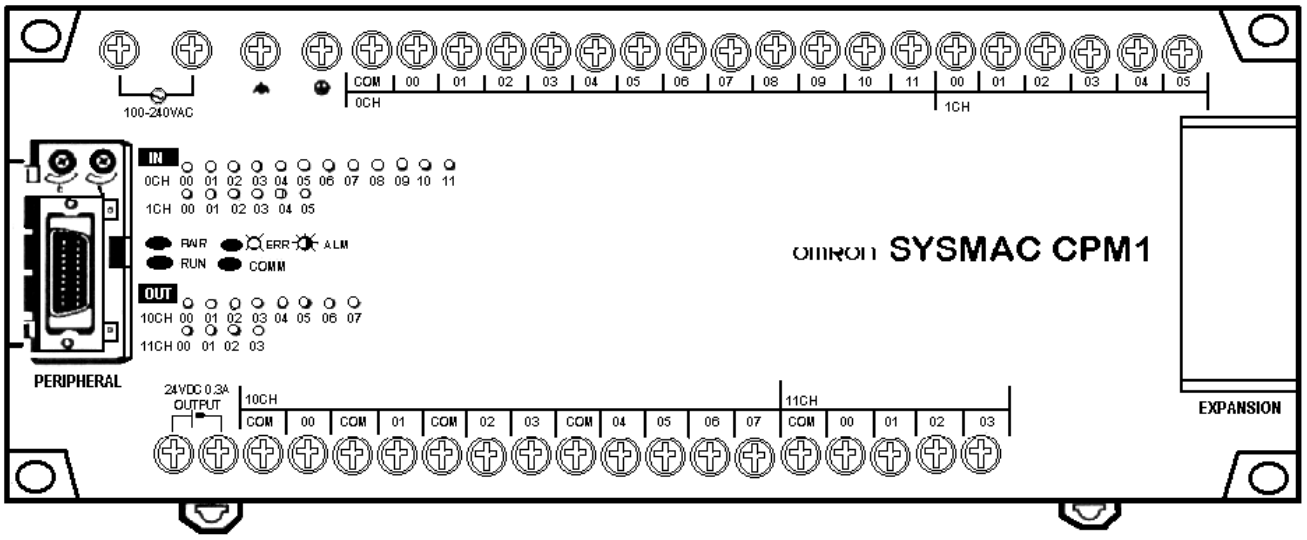
+ Loại CPM1 – 20CDR



Hình 5.2: Cấu hình cứng của PLC CPM1-20CDR

Loại này có kết cấu tương tự như loại trên, nhưng có 12 đầu vào là: 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11 và 8 đầu ra là: 00, 01, 02, 03, 04, 05, 06, 07.

+ Loại CPM1– 30CDR.



Hình 5.3: Cấu hình cứng của PLC CPM1-30CDR

Loại này có kết cấu tương tự như loại trên, nhưng có 16 đầu vào là: 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11 ở kênh 0 CH cộng với các đầu ở kênh 10 CH là: 00, 01, 02, 03, 04, 05 và 8 đầu ra là: 00, 01, 02, 03, 04, 05, 06, 07 ở kênh 010 CH cộng với các đầu 00, 01, 02, 03 ở kênh 011CH.

b. Ghép nối vào ra của PLC

* Mạch đầu vào (Input Unit)

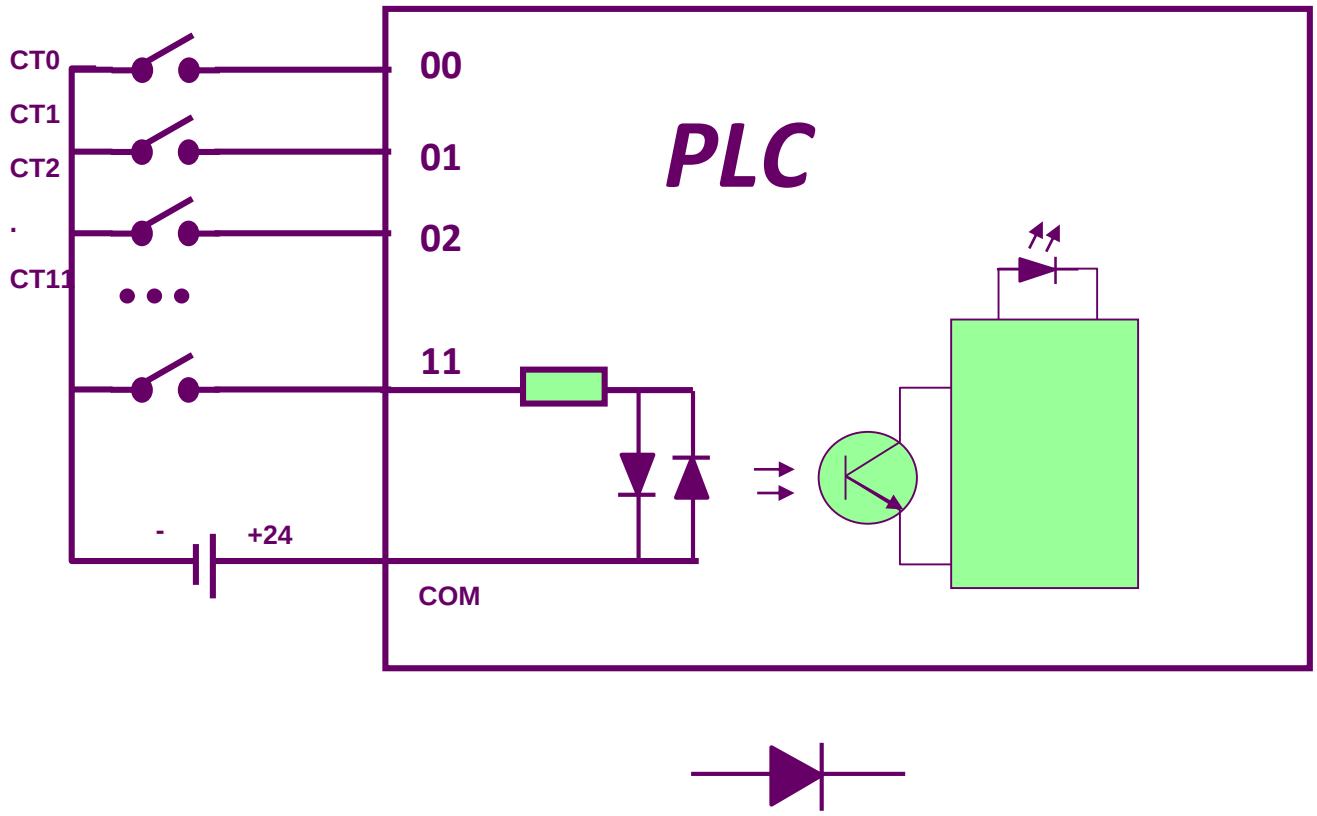
Là các mạch điện tử làm nhiệm vụ phối ghép chuyển đổi giữa tín hiệu điện đầu vào (Input) và tín hiệu số sử dụng bên trong PLC. Kết quả của việc xử lý sẽ được lưu ở vùng nhớ Input Area. Mạch đầu vào được cách ly về điện với các mạch trong của PLC nhờ các điốt quang. Bởi vậy, hư hỏng mạch đầu vào sẽ không ảnh hưởng đến hoạt động của CPU.

Bộ PLC Omron CPM1 có điện áp đầu vào là 24V một chiều.

Số lượng đầu vào phụ thuộc vào từng loại PLC.

Đặc điểm đầu vào :

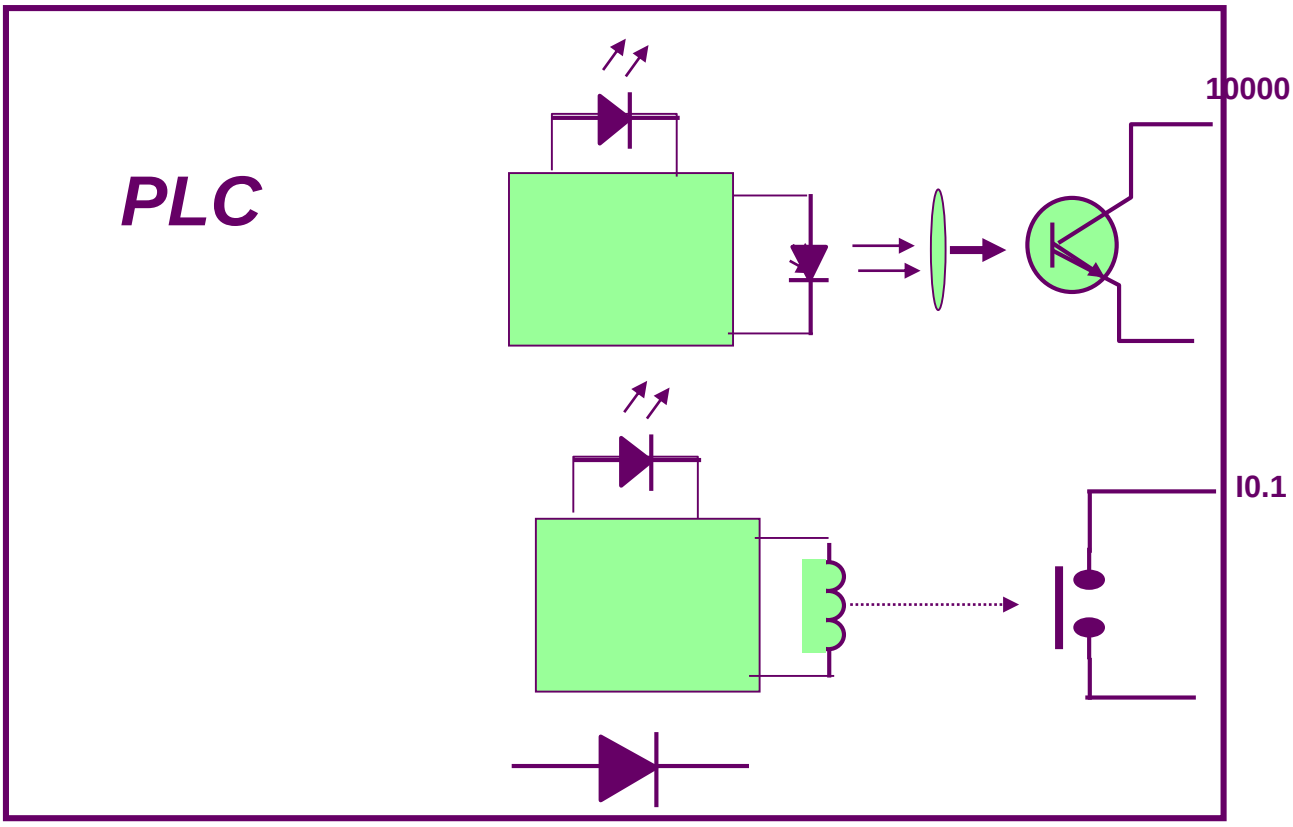
- Đầu vào được đánh số
- Đầu vào được tín hiệu hoá
- Đầu vào ghép quang cách li CPU với ngoại vi



Hình 5.3: Sơ đồ kết nối đầu vào

* Mạch đầu ra (Output Unit)

Mạch điện tử đầu ra sẽ biến đổi các lệnh mức logic bên trong PLC (trong vùng nhớ Output Area) thành các tín hiệu điều khiển như đóng mở rơle. Bộ CPM1 có mạch đầu ra bao gồm 8 tiếp điểm rơle, chịu được dòng tối đa 2 A Mạch đầu ra được cách ly về điện với các mạch trong của PLC nhờ các điốt quang. Bởi vậy, hư hỏng mạch đầu ra cũng sẽ không ảnh hưởng đến hoạt động của CPU.



Hình 5.4: Sơ đồ kết nối đầu ra

Số lượng đầu ra phụ thuộc vào từng loại PLC.

Đặc điểm đầu ra :

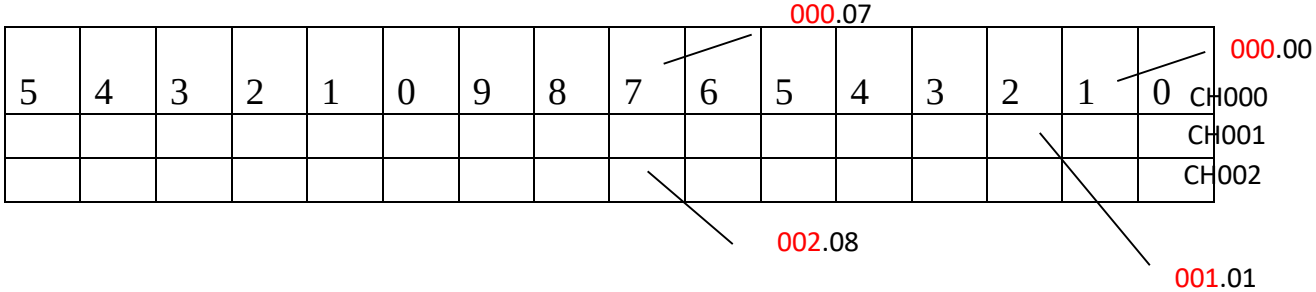
- Đầu vào được đánh số
- Đầu vào được tín hiệu hoá
- Đầu vào ghép quang cách li CPU với ngoại vi.

c. Địa chỉ các bộ nhớ trong PLC

Tất cả các đầu vào ra cũng như các bộ nhớ lưu trữ khác trên PLC khi sử dụng trong chương trình đều thông qua các địa chỉ bộ nhớ tương ứng. Các địa chỉ bộ nhớ được tổ chức thành các nhóm gồm 16 bit gọi là word hay Channel (CH). Mỗi bit có giá trị 0 hoặc 1. Các bit được đánh số từ 00 đến 15 từ phải qua trái.

Địa chỉ đầy đủ của mỗi bit sẽ được ký hiệu bằng 5 chữ số: 3 chữ số đầu từ trái qua là ký hiệu của channel, 2 chữ số tiếp theo là số thứ tự của bit.

Ví dụ: Địa chỉ 000.07 thì 000 là kênh, 07 là bít.



Địa chỉ các bộ nhớ thường dùng trong PLC CPM1

Bảng 5.1: Bảng địa chỉ của PLC CPM

CH000 ...CH009	Các đầu vào
CH010 ...CH019	Các đầu ra
CH200 ... CH239	Vùng nhớ hỗ trợ dùng tự do
SR240 ... SR255	Thanh ghi đặc biệt
TR0 ... TR7	Lệnh rẽ nhánh
HR00 ... HR19	
AR00 ... AR15	
LR00 ....LR15	
TIM/CNT000 ...TIM/CNT127	Địa chỉ của timer và counter

1.2. Các lệnh cơ bản PLC OMRON

1.2.1. Phần mềm lập trình Syswin

a. Giới thiệu về phần mềm Syswin

SYSWIN là 1 phần mềm lập trình cho PLC OMRON d−ới dạng

Ladder Diagram thực thụ chạy trong Windows. Syswin hiện có nhiều phiên bản như Syswin 3.1, Syswin 3.3, Syswin 3.4 … những phiên bản cao thường có nhiều tính năng ưu việt hơn.

b. Những yêu cầu với máy tính

Để cài đặt và chạy phần mềm này cần đảm bảo máy tính có cấu hình tối thiểu nh−

sau :

Windows 3.1, 3.11, Windows 95, Windows 98 trở lên

> 486 DX50 CPU

> 8 M Byte RAM

> 10 MB đĩa cứng trống.

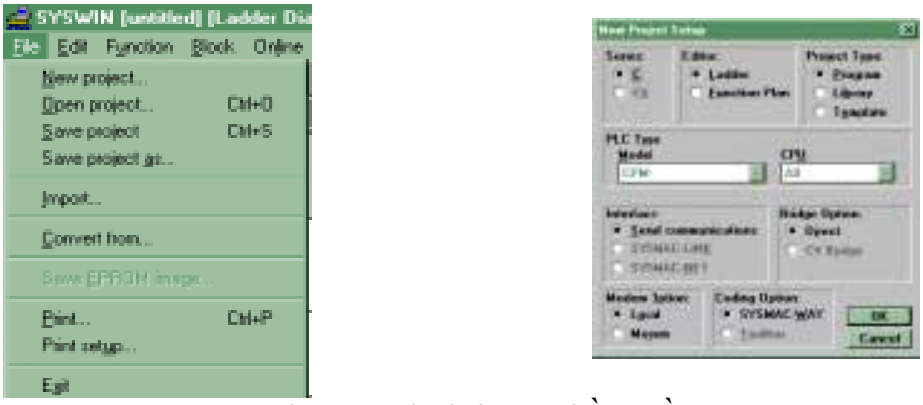
c. Cài đặt phần mềm Syswin.

Để cài đặt phần mềm syswin từ đĩa CD ta đưa đĩa chứa phần mềm cài đặt vào ổ đĩa CD, mở ổ đĩa CD → Omron → Softwarec → Syswin 3.4. Setup → disk 1 → Setup. Exe → chọn ngôn ngữ sử dụng là tiếng Anh U.S. English sau đó tiếp tục cài đặt như những phần mềm thông thường khác.

d. Sử dụng phần mềm Syswin

*1. Khởi động phần mềm lập trình Syswin bằng cách nháy đúp chuột trái vào biểu tượng của Syswin màn hình lập trình sẽ hiện ra

*2. Tạo file mới: Từ menu File chọn New project để tạo chương trình mới.



Hình 5.7: Khởi động phần mềm

PLC Type " CPM1

CPU " All

Series " C

Editor " Ladder

Project Type " Program

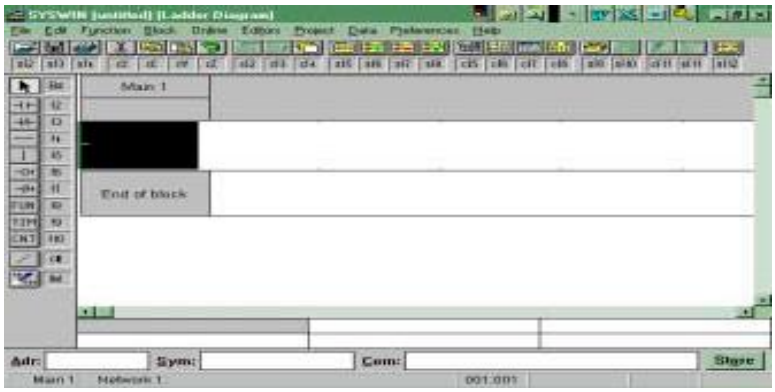
Interface " Serial Communications

Bridge Option " Direct

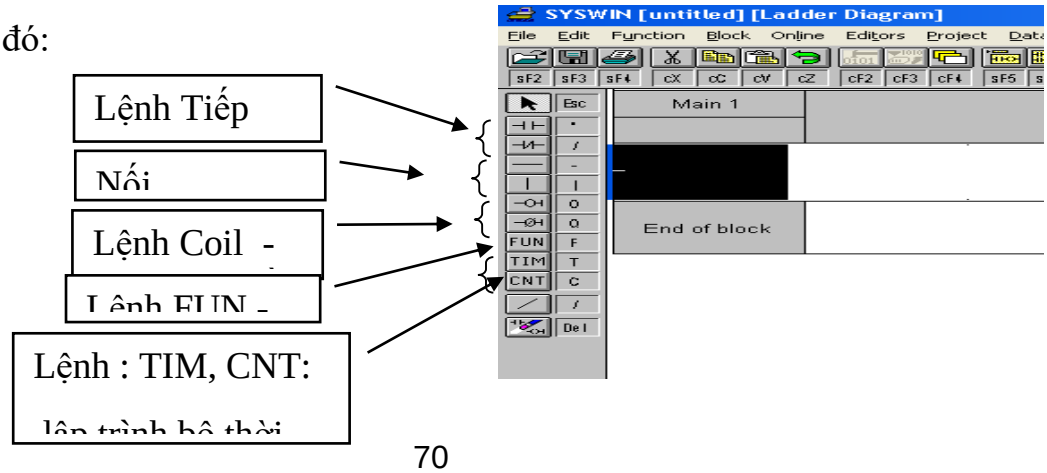
Chọn các mục trên ở hộp thoại New Project Setup xong rồi bấm OK

*3. Giới thiệu về cửa sổ lập trình:

Màn hình sẽ hiện ra 1 khung làm việc cho chương trình dạng Ladder Diagram



Trong đó:



Hình 5.8: Màn hình lập trình SYSWIN

*4. Cách lấy lệnh lập trình:

- Nhấp chuột chọn lệnh cần lấy.

- Di chuyển chuột ra ô cần đặt lệnh nhấp chuột lần nữa hộp thoại yêu cầu đặt địa chỉ hiện ra.

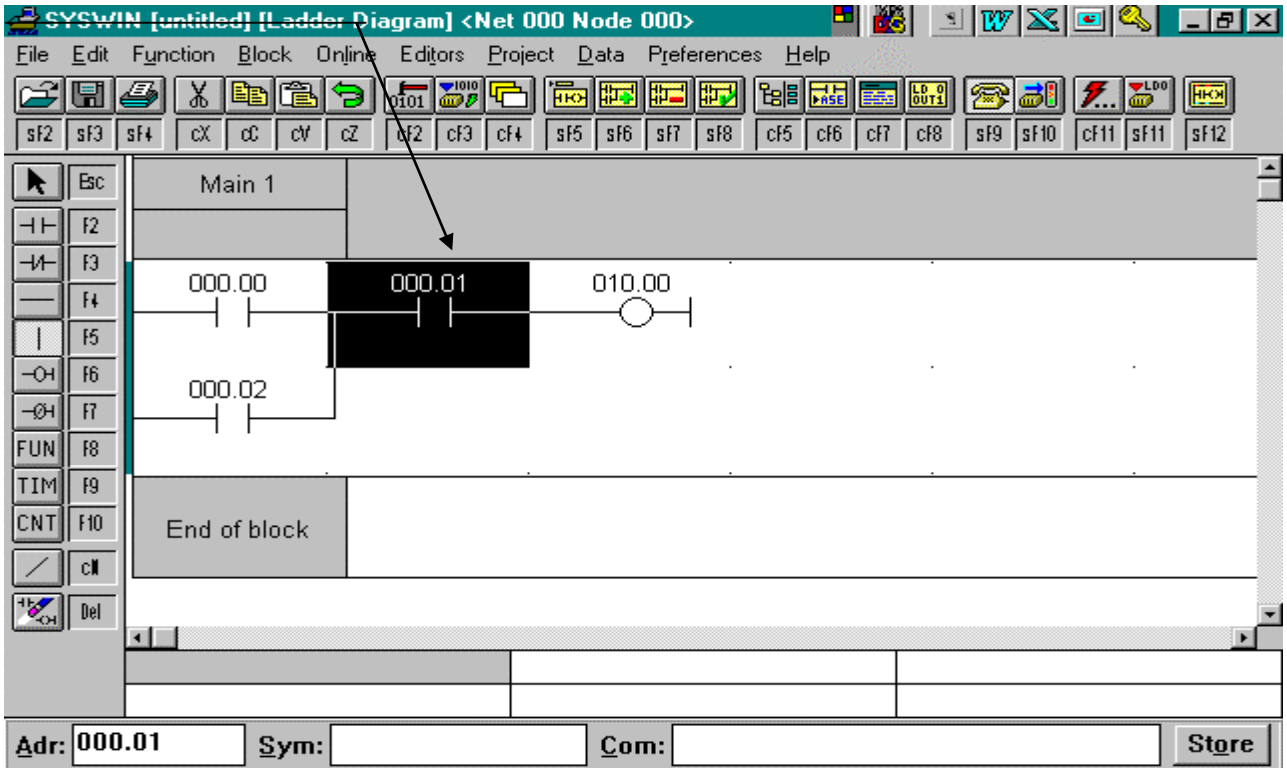
- Đặt địa chỉ cho lệnh vừa chọn và ấn OK.

*5. Chèn thêm Network mới:

- Chọn Insert Network

Insert Network

Hình 5.9: Chèn Network mới trong chương trình



- Từ hộp thoại hiện ra, chọn vị trí nơi sẽ chèn Network mới.ở đây ta sẽ chèn Network mới vào phí a dưới network hiện hành nên sẽ chọn BELOW Current Network và nhấn OK.

*6. Viết lệnh END(01).

- Chọn FUN ⇒ vào mã lệnh END là 01 ⇒ OK

*7. Đặt tên ký hiệu mô tả (SYMBOL) cho các địa chỉ

Để đặt tên ký hiệu mô tả cho các địa chỉ, trước tiên di ô chọn đến địa chỉ cần đặt tên, ô Adr ở cuối màn hình sẽ hiển thị địa chỉ hiện hành. Sau đó bấm vào

ô Sym và đánh vào 1 tên cho địa chỉ này. Phần mô tả địa chỉ có thể đánh vào ô Com. Lưu tên vừa đặt bằng cách bấm nút STORE.

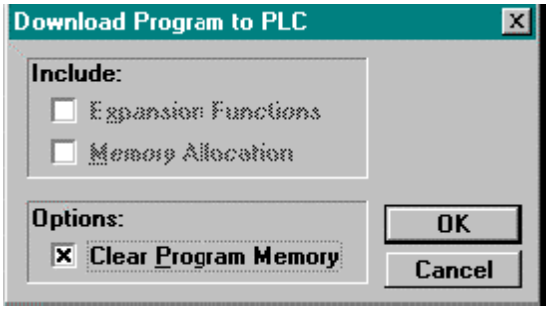
*8. Kết nối PLC với PC:

- Nối cáp từ PLC tới PC

- Từ Menu Online chọn connect để kết nối. Sau khi máy tính đã kết nối được với PLC, đèn COMM trên PLC sẽ nhấp nháy và các mục khác trên menu này trở thành màu đen (được phép lựa).

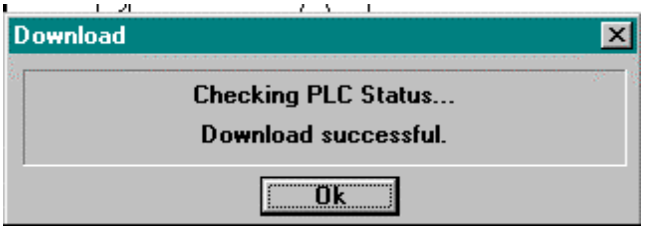
*9. Nạp chương trình vào PLC (Download program to PLC)

- Cũng từ menu Online chọn Download program to PLC. Một hộp thoại sau đây hiện ra hỏi ta có muốn xoá bộ nhớ chương trình trong PLC không (Clear Program Memory) trước khi nạp. Nên lựa tuỳ chọn này để tránh các vấn đề có thể xảy ra. Bấm OK để nạp chương trình vào PLC.



Hình 5.10: Xóa bộ nhớ chương trình

Khi việc nạp hoàn tất bấm nút OK ở hộp thoại sau để tiếp tục :

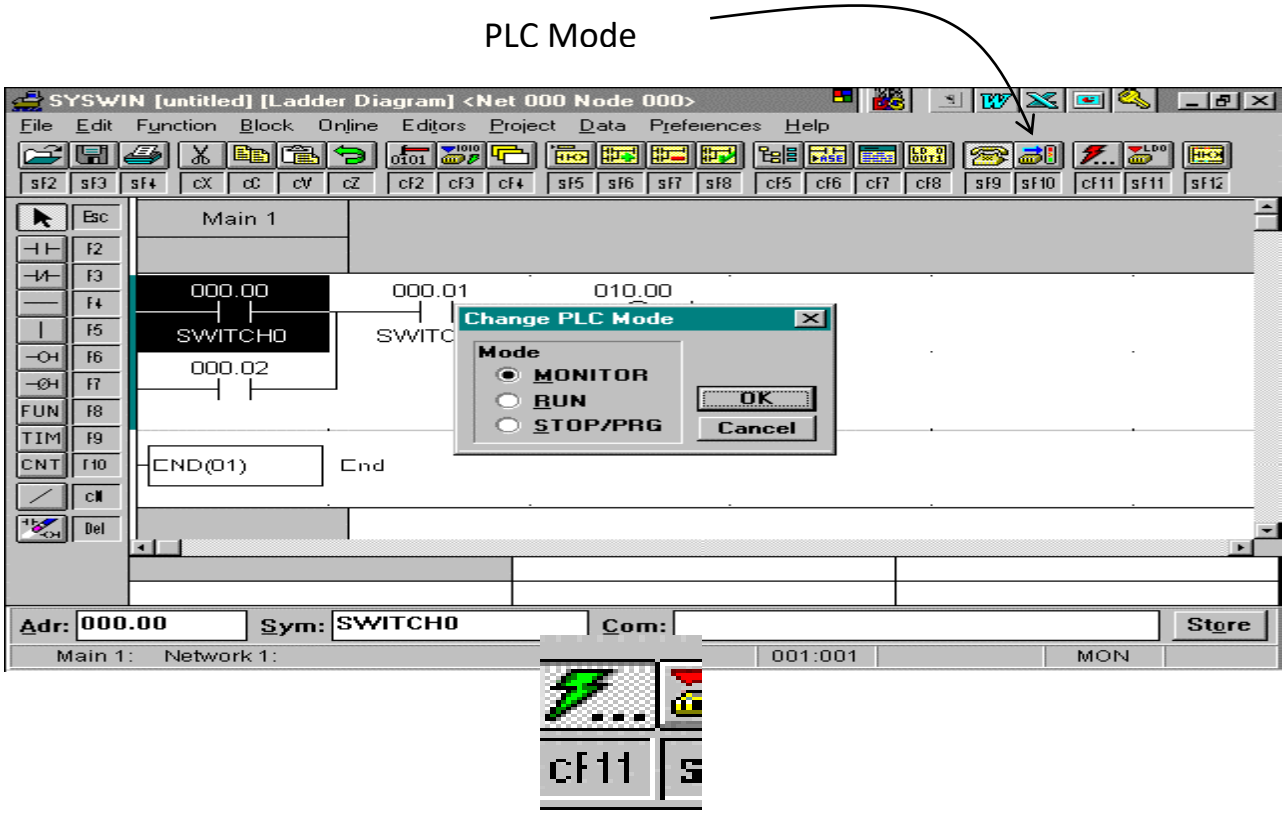


Hình 5.11: Nạp chương trình

Chú ý : Không thực hiện được việc Download vào PLC nếu PLC đang ở chế độ RUN.

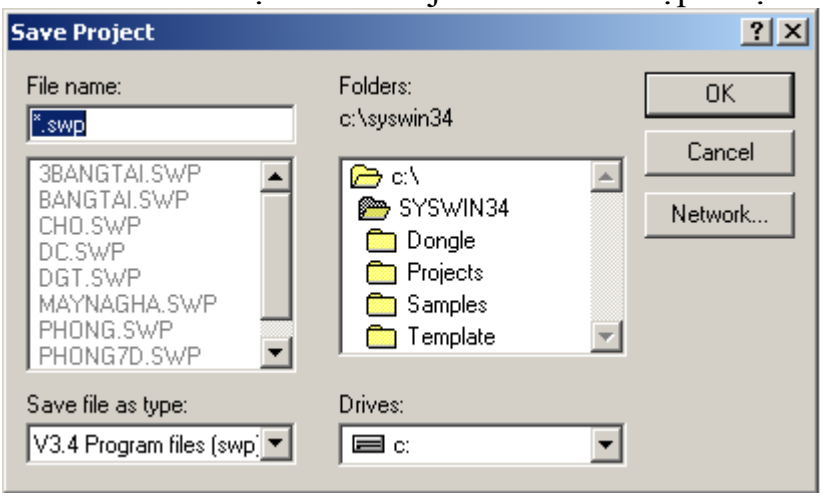
*10. Chạy chương trình (RUN)

Chuyển PLC sang chế độ RUN hoặc MONITOR bằng nút PLC Mode



*11. Lưu bài:

Từ menu File chọn Save Project as sau đó hộp thoại sau hiện ra:



Hình 5.13: Lưu bài

- Ô File name: Viết tên file.
- Ô Drives: Chọn ổ D
- Ô Folders: Chọn folder

Sau đó ấn OK.

1.2.2. Các lệnh cơ bản của PLC Omron

Các lệnh cơ bản.

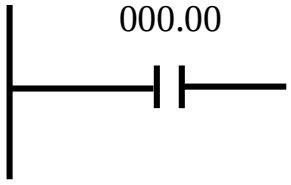
Các lệnh cơ bản của PLC Omron được lập trình dưới 2 dạng ngôn ngữ khác nhau là: lập trình bằng sơ đồ bậc thang và lập trình bằng bàn phím.. Dưới đây sẽ trình bầy các lệnh của 2 dạng ngôn ngữ trên.

Với sơ đồ bậc thang (LADDER DIAGRAM) thành phần luôn luôn phải có trong sơ đồ gọi là power bus, là nơi dẫn nguồn điện (tưởng tượng) đi vào và đi ra sơ đồ.

a. Lệnh LD.

Lệnh LD nối với power bus trái sẽ khởi đầu 1 network của sơ đồ Ladder Diagram. Số ghi phía trên ký hiệu lệnh là địa chỉ thông số của lệnh.

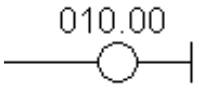
LD 000.00



b. OUT

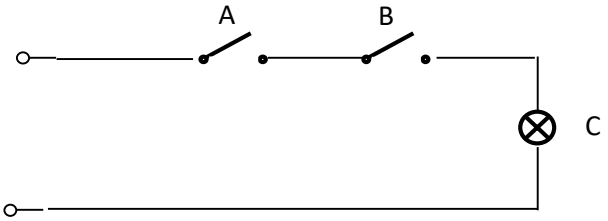
Lệnh OUT giống như 1 rơ le chấp hành đưa ra kết quả logic của các lệnh đi trước vào 1 tiếp điểm (bit) OUTPUT.

OUT 010.00



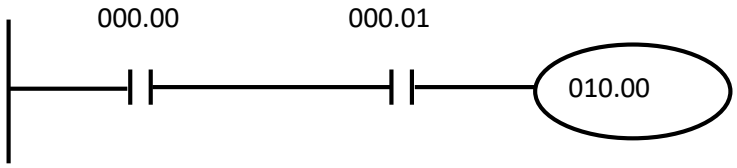
c. Lệnh AND

Lệnh AND sẽ tạo ra một logíc giống như hình dưới đây.



Như vậy lệnh AND thực hiện ghép nối tiếp nhiều tiếp điểm thường mở

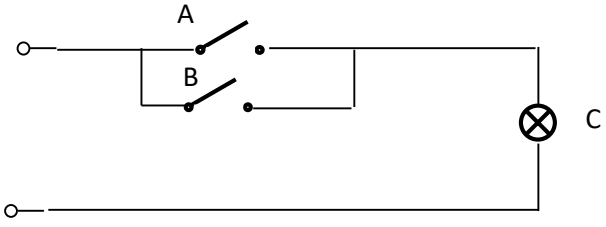
Ví dụ:



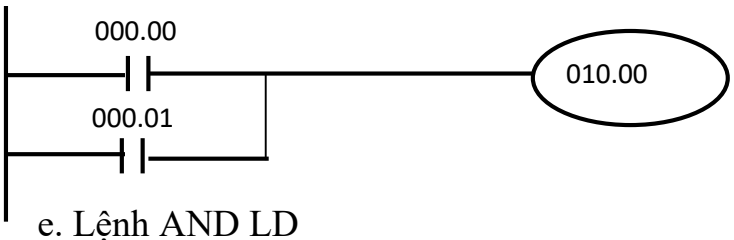
LD 000.00
AND 000.01
OUT 010.00

d. Lệnh OR

Lệnh OR sẽ tạo ra 1 logic giống như hình dưới đây:



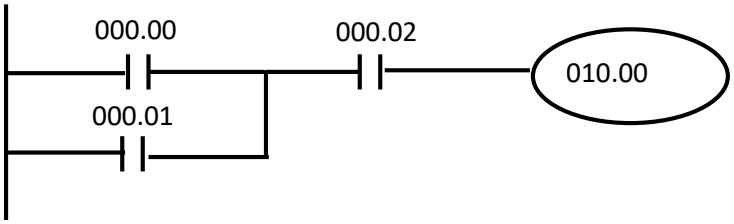
Như vậy lệnh OR thực hiện mắc song song nhiều tiếp điểm thường mở.
Ví dụ:




```
LD 000.00
OR 000.01
OUT 010.00
```

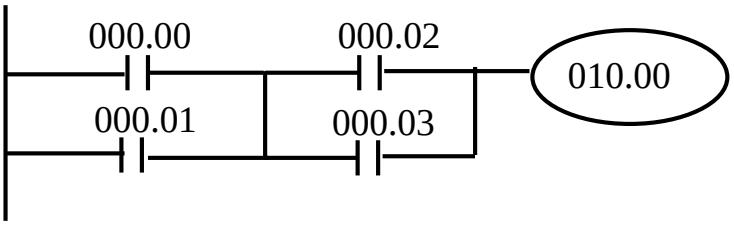
e. Lệnh AND LD

Lệnh AND LD được dùng để xây dựng các khối logic phức tạp hơn bằng cách ghép chúng nối tiếp với nhau. Giả sử ta có 1 đoạn chương trình như dưới đây, trong đó đầu ra 01000 sẽ bật khi đầu vào 00000 hoặc 00001 Và 00002 bật.

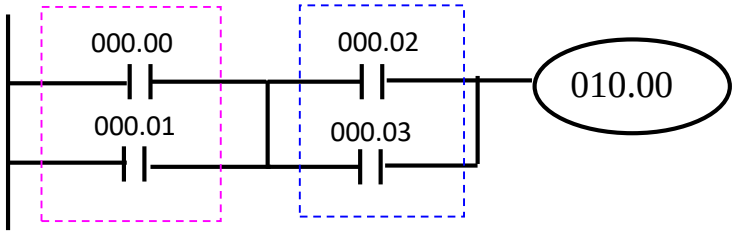



```
LD  000.00
OR  000.01
AND 000.02
OUT 010.00
```

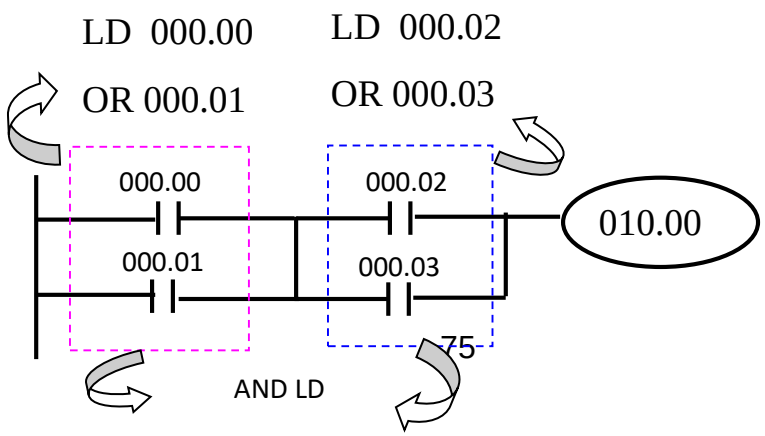
Nếu bây giờ điều kiện trên có thêm đầu vào 00003 như dưới đây :



Việc nhập vào đoạn chương trình này đòi hỏi phải chia nó ra làm 2 khối nối tiếp nhau



Sau đó nhập vào riêng rẽ các lệnh cho từng khối và nối 2 khối lại với nhau bằng lệnh AND LD.

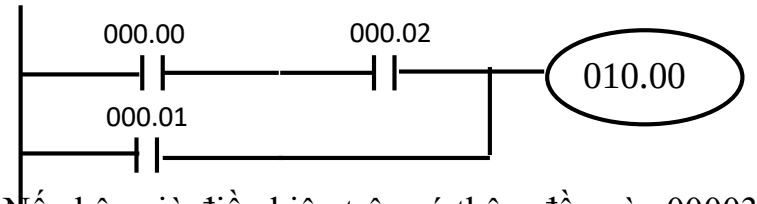



```
LD 000.00
OR 000.01
LD 000.02
OR 000.03
AND LD
```

OUT 010.00

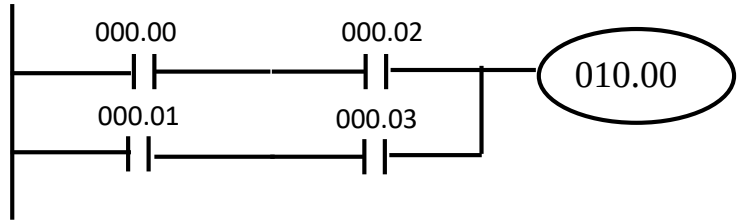
f. Lệnh OR LD.

Lệnh OR LD được dùng để xây dựng các khối logic bằng cách ghép chúng song song với nhau. Giả sử ta có 1 đoạn chương trình như dưới đây, trong đó đầu ra 01000 sẽ bật khi đầu vào 00000 và 00001 hoặc 00002 bật.

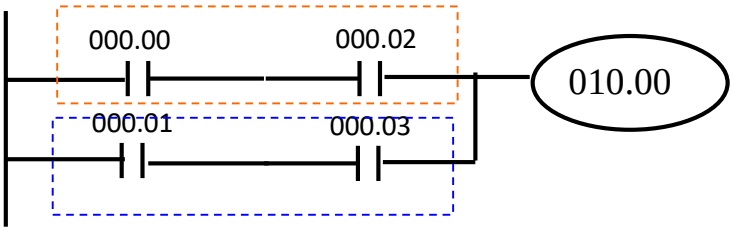


LD 000.00
AND 000.02
OR 000.01
OUT 010.00

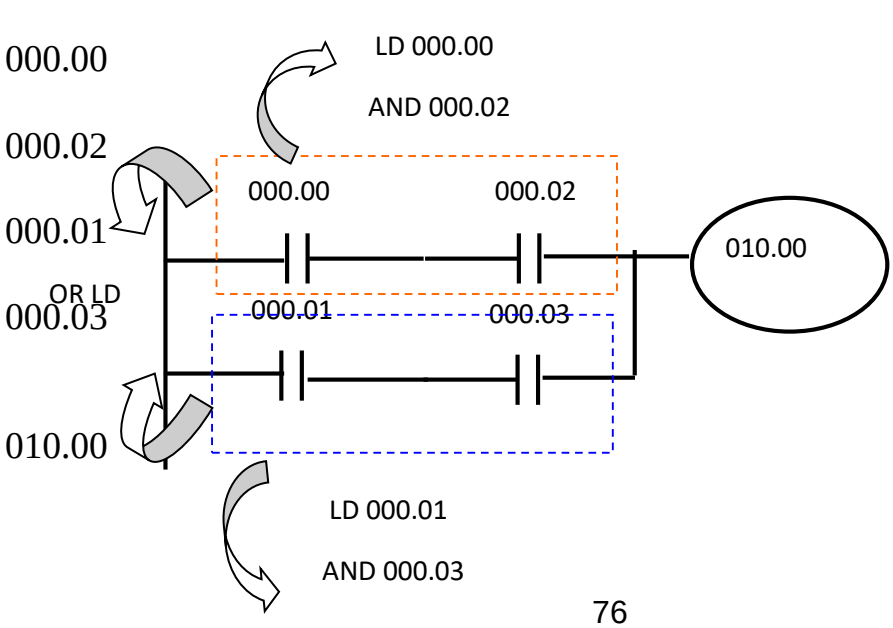
Nếu bây giờ điều kiện trên có thêm đầu vào 00003 như dưới đây :



Để nhập vào đoạn chương trình này ta phải chia nó ra làm 2 khối con nối song song với nhau như dưới đây :

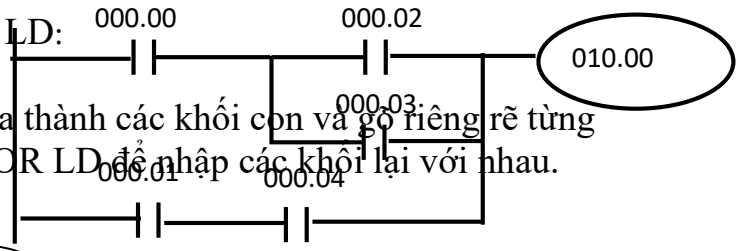


Sau đó nhập vào riêng rẽ các lệnh cho từng khối và nối 2 khối lại với nhau bằng lệnh OR LD.

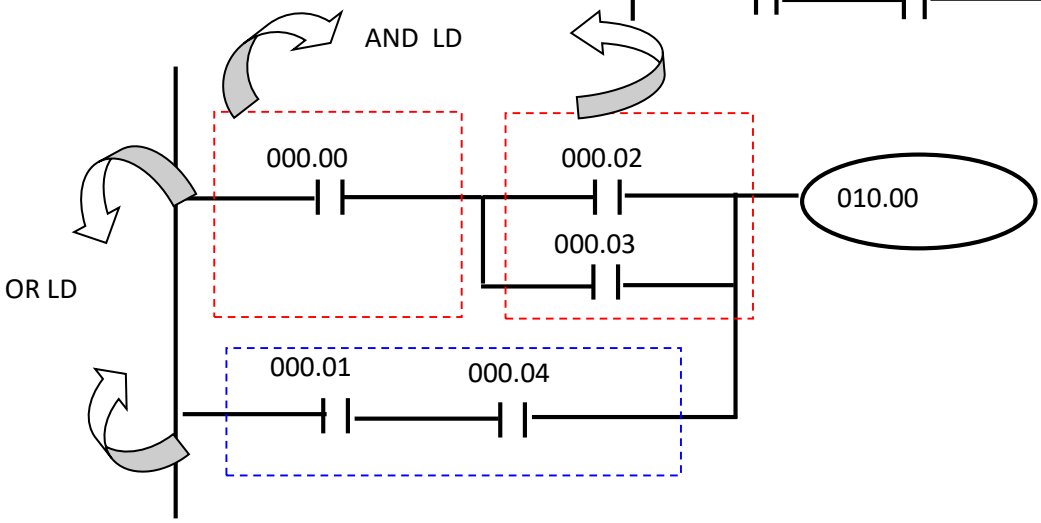


LD
AND
LD
AND
OR LD
OUT

Ví dụ kết hợp lệnh AND LD và OR LD:



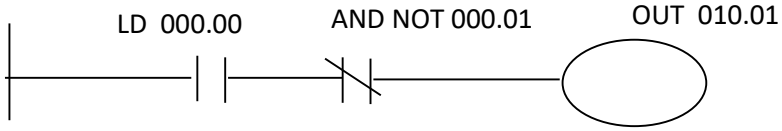
Trình tự thực hiện cũng phải chia thành các khối con và gõ riêng rẽ từng khối vào, sau đó dùng lệnh AND LD và OR LD để nhập các khối lại với nhau.



g. Các lệnh AND NOT và LD NOT

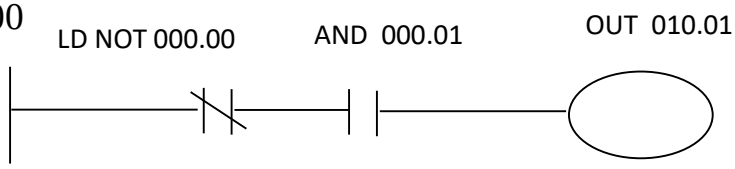
Đây là các lệnh phủ định lại lệnh AND và lệnh LD.

LD 000.00
AND NOT 000.01
OUT 010.01



Hoặc

LD NOT 000.00
AND 000.01
OUT 010.01



Ví dụ ứng dụng:

VD1. Lập trình điều khiển động cơ có đảo chiều quay:

a. Yêu cầu bài toán: Ấn nút quay thuận động cơ quay thuận, ấn nút quay ngược động cơ quay ngược, ấn nút dừng động cơ ngừng hoạt động. Động cơ được bảo vệ quá tải bằng rơ le nhiệt.

b. Liệt kê đầu vào /ra:

- Đầu vào: MT, MN, D, RN
- Đầu ra: KT, KN

c. Phân công địa chỉ:

- Đầu vào: 00000 – MT; 00001 – MN; 00002 – D; 00003 - RN

- Đầu ra: 01000 – KT; 01001 - KN

d. Soạn thảo chương trình

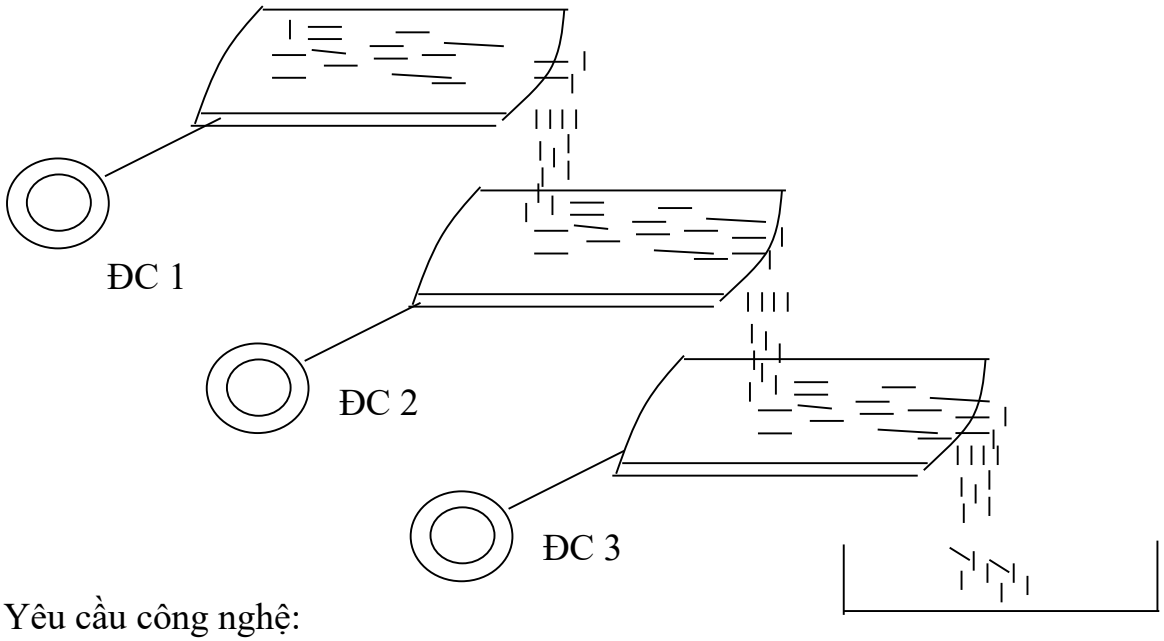
e. Nạp chương trình và chạy thử.

f. Lưu bài.

VD2. **Lập trình điều khiển hệ thống băng tải hoạt động theo tuần tự**

1. Yêu cầu của hệ thống:

Sơ đồ công nghệ của hệ thống:



Yêu cầu công nghệ:

Ấn nút M1 Băng tải 1 quay.

Ấn nút M2 Băng tải 2 quay.

Ấn nút M3 Băng tải 3 quay.

Nhấn nhầm Băng tải không quay.

Ấn nút D1 Băng tải 1 ngừng quay.

Ấn nút D2 Băng tải 2 ngừng quay.

Ấn nút D3 Băng tải 3 ngừng quay.

Nhấn nhầm Băng tải không ngừng quay.

2. Liệt kê đầu vào ra:

Đầu vào: M1, D1, M2, D2, M3, D3

Đầu ra: DC1, DC2, DC3

3. Phân công địa chỉ :

Đầu vào:				Đầu ra:	
M1	000.00	D1	000.03	DC1	010.00
M2	000.01	D2	000.04	DC2	010.01
M3	000.02	D3	000.05	DC3	010.02

4. Soạn thảo chương trình:

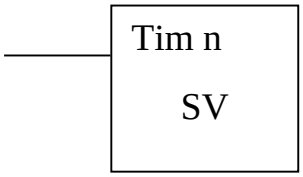
5. Download và chạy thử chương trình.

6. Lưu bài trên ổ

1.3. Lệnh Timer, Counter:

1.3.1 Lệnh Timer

a. Nguyên lý



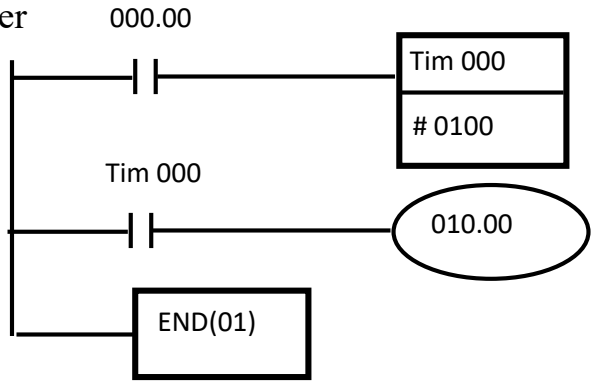
Trong đó: n là số thứ tự của timer

SV là dữ liệu đặt thời gian của timer

Đơn vị = 0.1 giây

SV = 0000 - 9999 - 999.9 giây

$t_{trễ}$ =SV x 0.1




```
LD 000.00
Tim 000 #0100
LD Tim 000
OUT 010.00
END (01)
```

Khi bật khoá CH000.00 lên, Timer số 000 sẽ bắt đầu đếm thời gian, khi 10 giây trôi qua, tiếp điểm của Timer là TIM 000 được bật lên ON và làm đầu ra CH010.00 cũng được bật lên ON. Timer cũng sẽ bị reset về giá trị đặt khi đầu vào 00000 tắt (OFF).

b. Ví dụ : Lập trình điều khiển động cơ khởi động đổi nối sao – tam giác

*1. Yêu cầu

Ấn nút mở thuận (hoặc mở ngược) thì động cơ khởi động quay thuận (hoặc ngược) ở hình sao sau 10s tự động chuyển sang làm việc ở chế độ hình tam giác.

Ấn nút Stop thì động cơ dừng.

*2. Liệt kê đầu vào ra:

- Đầu vào: MT, MN, D, RN

- Đầu ra: KT, KN, KY, KTG

*3. Phân công địa chỉ

Đầu vào:

D	000.00
RN	000.01
MT	000.02
MN	000.03

Đầu ra:

KT	010.00
KN	010.01
KY	010.02
KTG	010.03

*4. Soạn thảo chong trình.

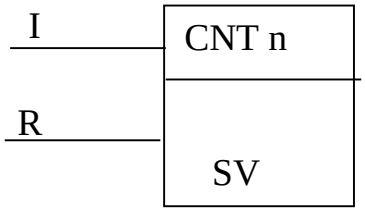
*5. Download và chạy thử chương trình.

*6. Lưu bài trên ổ

1.3.2. Bộ đếm COUNTER

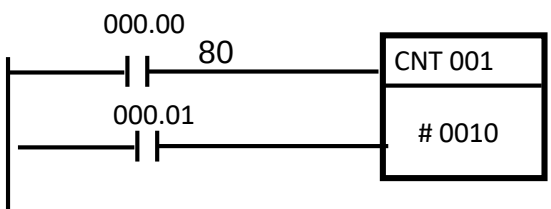
a. Nguyên lý

Trong đó: n là số tứ tự của bộ đếm



SV là dữ lệu đặt của bộ đếm

Lúc khởi đầu giá trị hiện hành của bộ đếm được bắt đầu tại SV. Bộ đếm sẽ giảm giá trị hiện hành của nó (CNT N) đi 1 đơn vị mỗi lần có sườn lên ở xung đầu vào I và cờ báo hoàn thành CNT N sẽ bật khi giá trị hiện hành của bộ đếm giảm về 0. Bộ đếm sẽ bị reset về giá trị đặt SV khi có sườn lên của đầu vào R.



LD 000.00

LD 000.01

CNT 001 # 0010

LD CNT 001

OUT 010.00

END(01)

Mỗi lần bật khoá CH 000.00, giá trị của Counter 000 giảm đi 1. Khi bật khoá CH000.00 đủ 10 lần thì cờ báo CNT 000 bật lên ON và đầu ra CH 010.00 cũng bật lên ON. Bộ đếm sẽ bị reset khi bật switch CH 000.01.

b. Ví dụ:

Yêu cầu : Ấn nút M 1 lần đèn sáng sau 10s đèn tắt.

Ấn nút M 2 lần đèn sáng thường trực

Ấn nút D đèn tắt.

Bài làm:

B1. Xác định yêu cầu bài toán.

B2. Xác định đầu vào/ ra.

Đầu vào: M, D

Đầu ra: Đèn

B3. Phân công địa chỉ.

M: 000.00

D : 000.01

Đèn : 010.00

B4. Soạn thảo chương trình.

B5. Download và chạy thử chương trình.

B6. Lưu bài trên ổ

2. PLC của hãng siemens S7-300

Giới thiệu:

PLC S7-300 là dòng sản phẩm cao cấp, nó được dùng cho các ứng dụng lớn với những yêu cầu I/O nhiều và thời gian đáp ứng nhanh, êu cầu kết nối mạng và có khả năng mở rộng cho sau này.

2.1. Các Module trong S7-300:

- Module nguồn (PS : Power Supply): Có 3 loại 2A, 5A, 10A

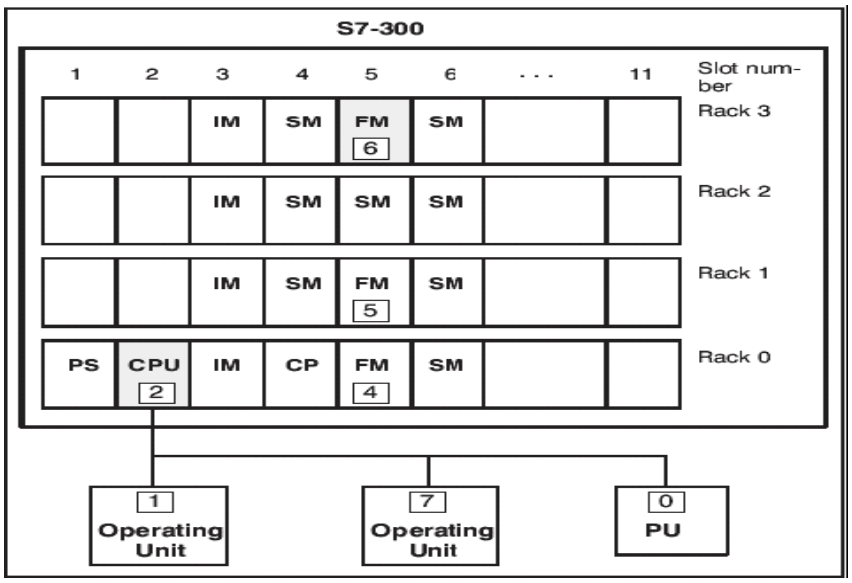
➢ Module CPU: Chứa bộ vi xử lý, hệ điều hành, bộ nhớ, bộ trễ thời gian, bộ đếm, cổng truyền thông và còn có thể có cổng vào/ ra số.

➢ Module kết nối (IM :Interface Module): là loại module dùng ghép nối các module lại với nhau

➢ Module tín hiệu vào/ ra (SM :Signal Module): Bao gồm tín hiệu số và tín hiệu tương tự

➢ Module chức năng (FM : Function Module): Module có chức năng riêng biệt như điều khiển nhiệt độ, điều khiển động cơ Servo....

➢ Module truyền thông (CP : Communication Module): Modul phục vụ truyền thoâng trong mạng giữa các PLC với nhau hoặc giữa PLC với máy tính.



Bộ nhớ PLC: gồm 3 vùng chính.

- Vùng chứa chương trình ứng dụng : Vùng chứa chương trình được chia thành 3 miền

+ OB (Organisation block) : miền chứa chương trình tổ chức.
+ FC (Function) : Miền chứa chương trình con ,được tổ chức thành hàm và có biến hình thức để trao đổi dữ liệu

+ FB (Function block) : Miền chứa chương trình con ,được tổ chức thành hàmvà có khả năng trao đổi dữ liệu với bất cứ 1 khối chương trình nào khác .Các dữ liệu này phải được xây dựng thành một khối dữ liệu riêng (Data Block khối DB)

- Vùng chứa tham số của hệ điều hành: Chia thành 7 miền khác nhau

+I (Process image input) : Miền dữ liệu các cổng vào số,trước khi bắt đầu thực hiện chương trình ,PLC sẽ đọc giá trị logic của tất cả các cổng đầu vào và cất giữ chúng trong vùng nhớ I.Thông thường chương trình ứng dụng không đọc

trực tiếp trạng thái logic của cổng vào số mà chỉ lấy dữ liệu của cổng vào từ bộ đệm I.

+Q (Process Image Output): Miền bộ đệm các dữ liệu cổng ra số .Kết thúc giai đoạn thực hiện chương trình,PLC sẽ chuyển giá trị logic của bộ đệm Q tới các cổng ra số.Thông thường chương trình không trực tiếp gán giá trị tới tận cổng ra mà chỉ chuyển chúng tới bộ đệm Q.

+M (Miền các biến cờ): CHương trình ứng dụng sử dụng những biến này để lưu giữ các tham số cần thiết và có thể truy nhập nó theo Bit (M) ,byte (MB),từ (MW) hay từ kép (MD).

+T (Timer): Miền nhớ phục vụ bộ thời gian (Timer) bao gồm việc lưu trữ giá trị thời gian đặt trước (PV-Preset Value),giá trị đếm thời gian tức thời (CV –Current Value) cũng như giá trị Logic đầu ra của bộ thời gian.

+ C (Counter): Miền nhớ phục vụ bộ đếm bao gồm việc lưu trữ giá trị đặt trước (PV- Preset Value),giá trị đếm tức thời (CV _ Current Value)và giá trị logic đầu ra của bộ đệm.

+ PI : Miền địa chỉ cổng vào của các Modul tương tự (I/O External input). Các giá trị tương tư tại cổng vào của modul tương tự sẽ được module đọc và chuyển tự động theo những địa chỉ.Chương trình ứng dụng có thể truy cập miền nhớ PI theo từng Byte (PIB), từng tư PIW hoặc từng từ kép PID .

+ PQ: Miền địa chỉ cổng ra cho các module tương tự (I/O External Output).Các giá trị theo những địa chỉ này sẽ được module tương tự chuyển tới các cổng ra tượng tự .Chương trình ứng dụng có thể truy nhập miền nhớ PQ theo từng Byte (PQB),từng từ (PQW) hoặc theo từng từ kép (PQD)

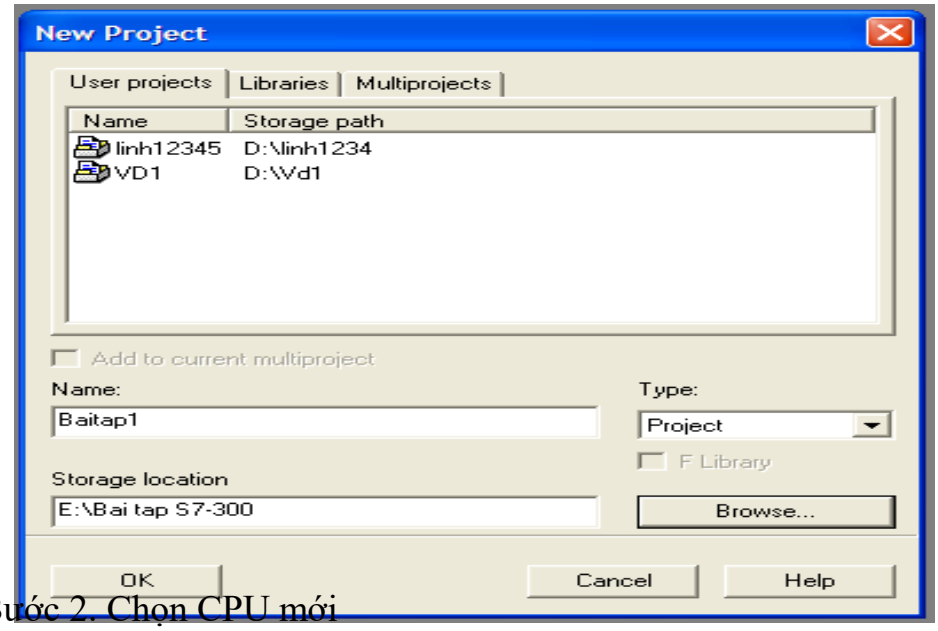
- Vùng chứa các khối dữ liệu: được chia làm 2 loại: DB(Data Block):Miền chứa dữ liệu được tổ chức thành khối .Kích thước cũng như số lượng khối do người sử dụng quy định ,phù hợp với từng bài toán điều khiển.Chương trình có thể truy nhập miền này theo từng bit (DBX),byte (DBB),từ (DBW) hoặc từ kép (DBD). L (Local data block) : Miền dữ liệu địa phương ,được các khối chương trình OB,FC,FB tổ chức và sử dụng cho các biến nháp tức thời và trao đổi dữ liệu của biến hình thức với những khối chương trình gọi nó .Nội dung của một khối dữ liệu trong miền nhớ này sẽ bị xoá khi kết thúc chương trình tương ứng trong OB ,FC,FB.Miền này có thể được truy nhập từ chương trình theo bit (L),byte(LB) từ (LW) hoặc từ kép (LD).

2.2. Phần mềm lập trình.

a. Khai báo cấu hình cứng:

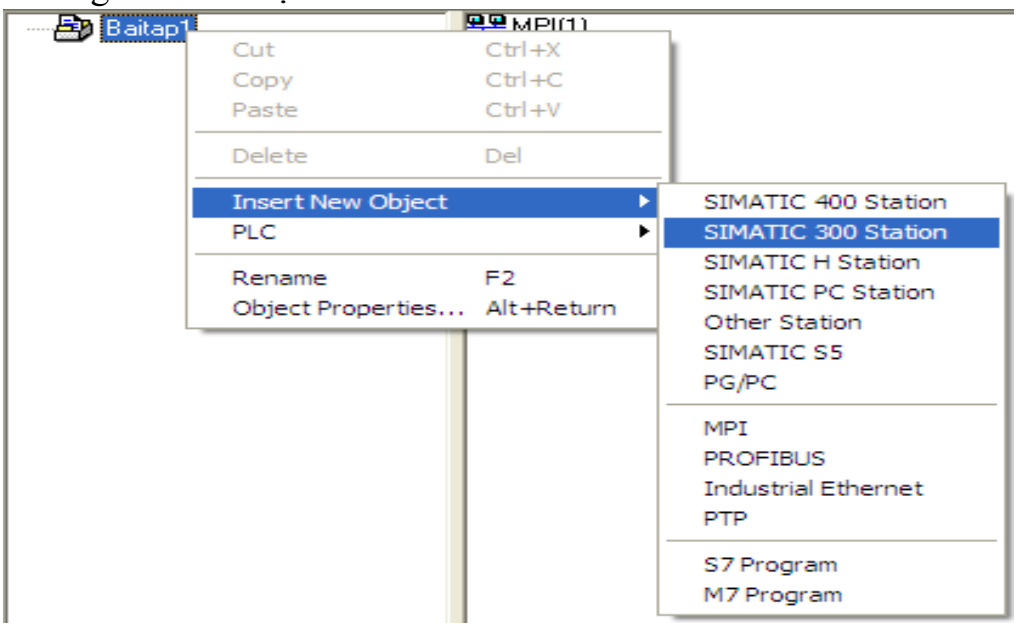
Bước 1. Tạo file mới chương trình

- Khởi động phần mềm
- Chọn File/ New để tạo mới chương trình
- Đặt tên cho chương trình, chọn thư mục cho chương trình.
- Chương trình SIMATIC sẽ nằm trong 1 thư mục có tên do người sử dụng đặt.



Bước 2. Chọn CPU mới

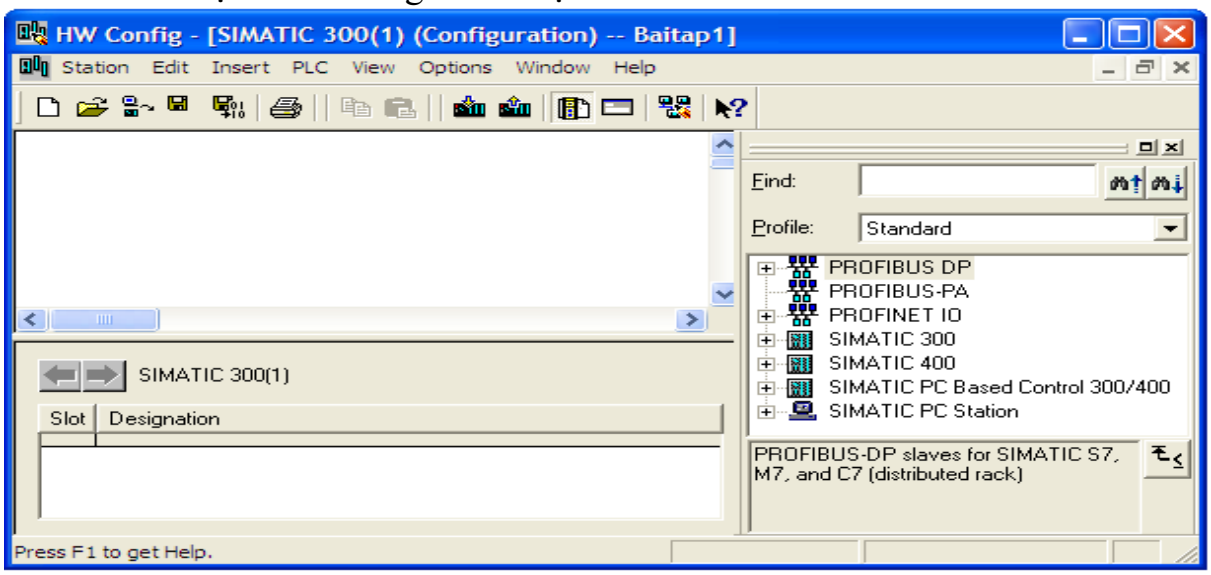
Chương trình sẽ chọn CPU mới như sau:



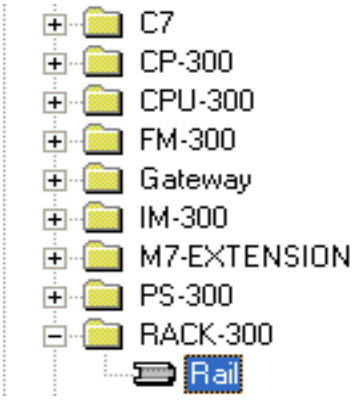
Bước 3. Khai báo cấu hình cứng:

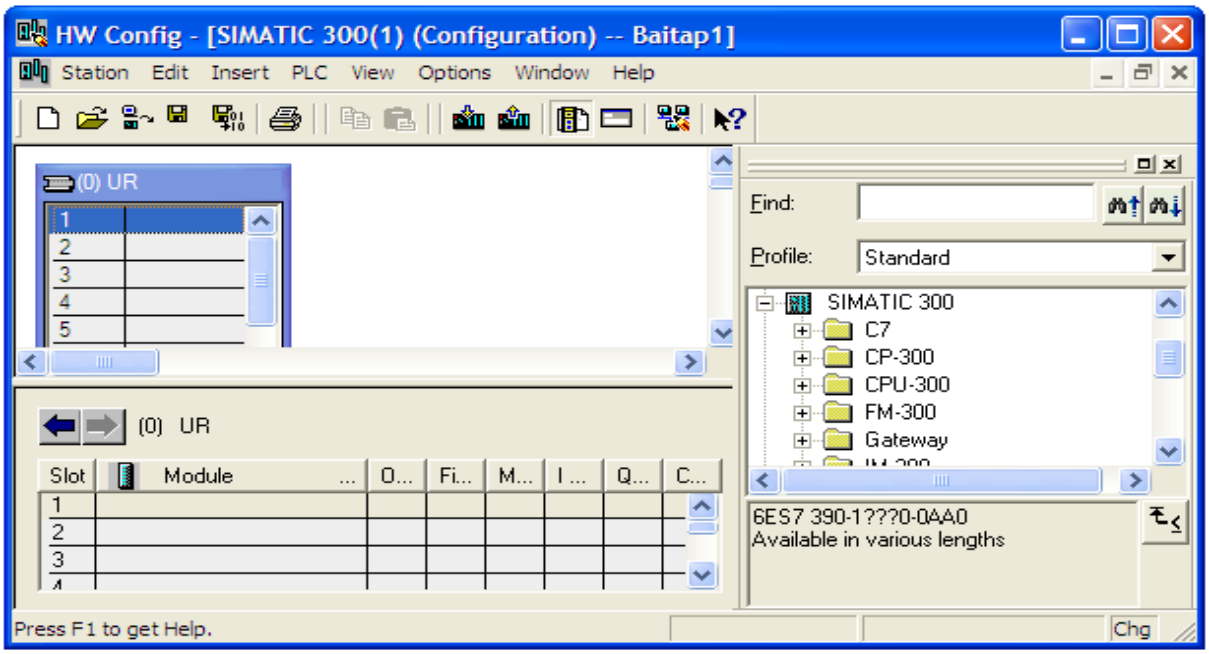
- Chọn SIMATIC 300(1) -> Hardware

- Giao diện HW configure sẽ hiện ra



- Tạo RACK: Chọn SIMATIC 300 -> RACK 300 -> Rail

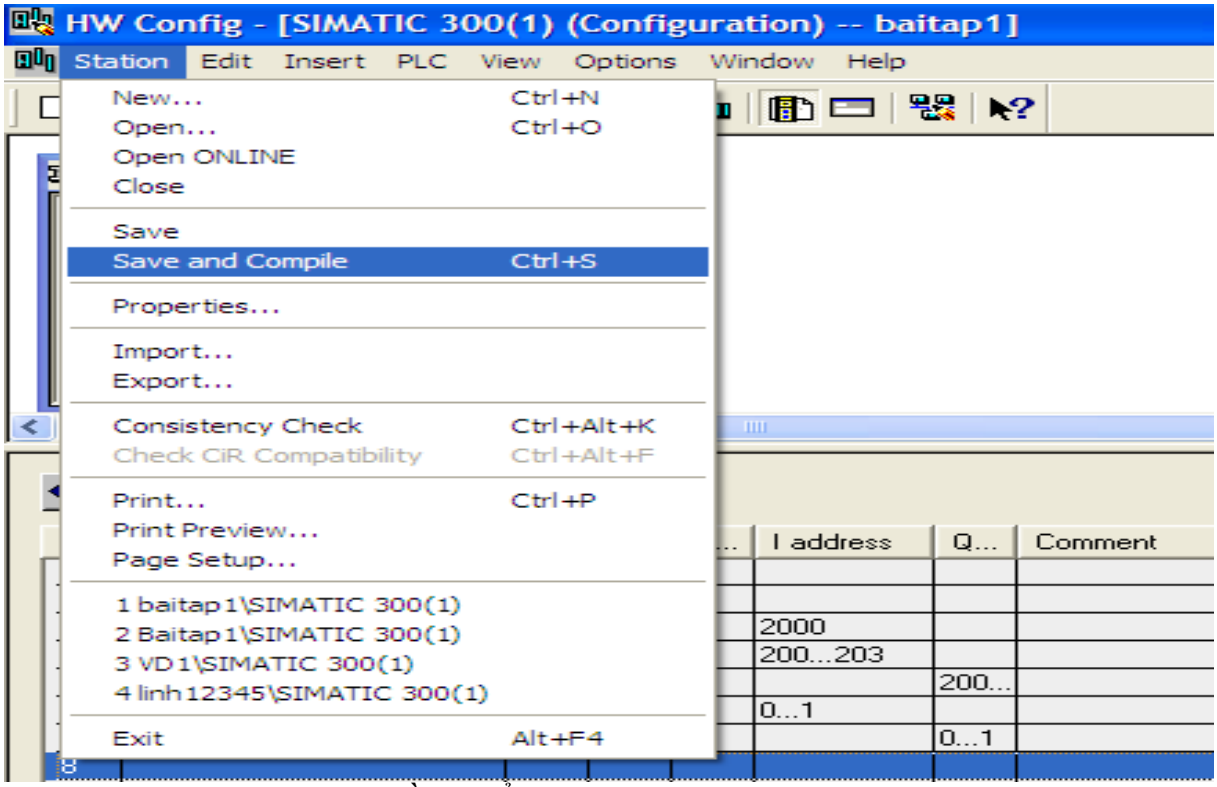




- Thêm các Module cho cấu hình phần cứng:
Slot1: Module nguồn
Slot 2: Module CPU
Slot kết tiếp: module IM, Module SM, Module FM, Module CP

Slot	Module	Order number	Firmware	MPI address	I addre...	Q addr...	Comment
1	PS 307 10A	6ES7 307-1KA00-0AA0					
2	CPU 314	6ES7 314-1AG13-0AB0	V2.6	2			
3	IM 360	6ES7 360-3AA00-0AA0			2000		
4	AI2x12Bit	6ES7 331-7KB02-0AB0			256...259		
5	AO4x12Bit	6ES7 332-5HD01-0AB0				272...279	
6	DI16xDC24V	6ES7 321-1BH02-0AA0			8...9		
7	DO16xDC24V/0.5A	6ES7 322-1BH01-0AA0				12...13	

Bước 4. Lưu cấu hình



b.Lập chương trình điều khiển

Bước 1. Tạo file lập trình trên OB1

Bước 2. Lập bảng phân công địa chỉ

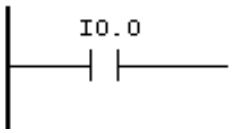
Bước 3. Viết chương trình

Bước 4. Nạp và chạy thử

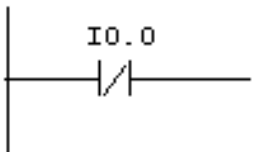
2.3. Tập lệnh trong PLC S7-300.

a. Lệnh về bit:

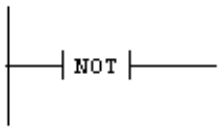
- Tiếp điểm thường hở: cho KQ=1 khi I0.0 = 1



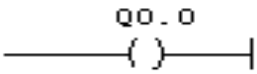
- Tiếp điểm thường đóng: cho KQ=1 khi I0.0 = 0



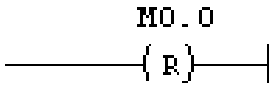
- Lệnh NOT: KQ thu được bằng đảo giá trị của đầu vào.



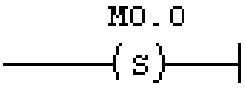
- Ngõ ra:



- Lệnh Reset Bit: Gán giá trị 0 cho đầu ra

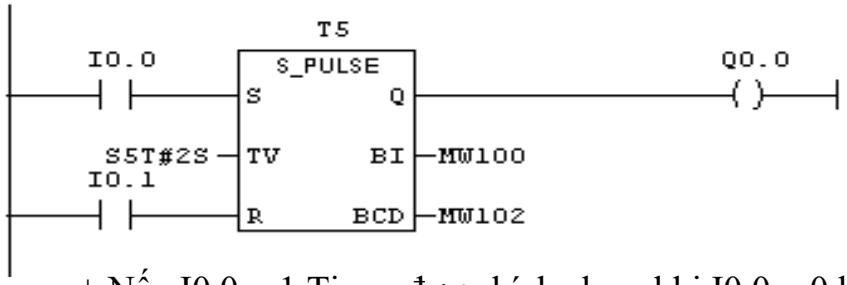


- Lệnh Set Bit: Gán giá trị 1 cho đầu ra



b.Lệnh Timer:

- Lệnh S_PULSE:



+ Nếu I0.0 = 1 Timer được kích chạy, khi I0.0 = 0 hoặc chạy đủ thời gian đặt 2s thì Timer dừng

+ Hoặc có tín hiệu I0.1 thì Timer cũng dừng

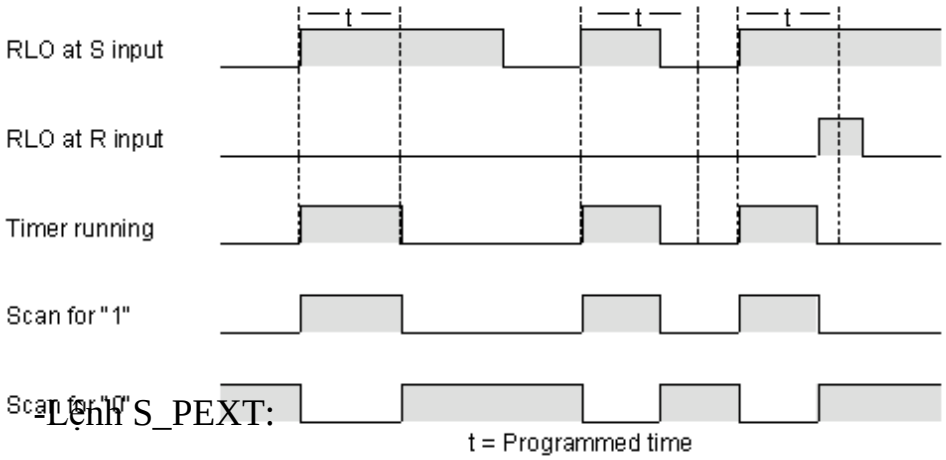
+ Tỉmer chỉ chạy lại khi có tín hiệu mới từ I0.0 (tức là I0.0 chuyển trạng thái từ 0 lên 1)

+ Q0.0= 1 khi Timer đang chạy

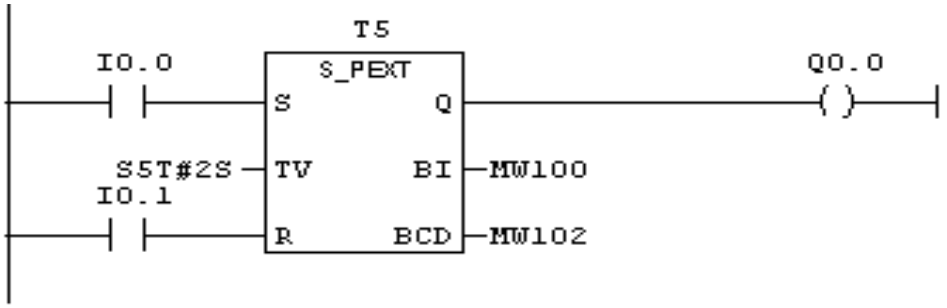
+ MW100 lưu giá trị đếm của Timer theo dạng Integer

+ MW102 lưu giá trị của Timer theo dạng BCD

+ Chức năng của Timer này là tạo xung có thời gian được đặt sẵn



+ Lệnh S_PEXT:



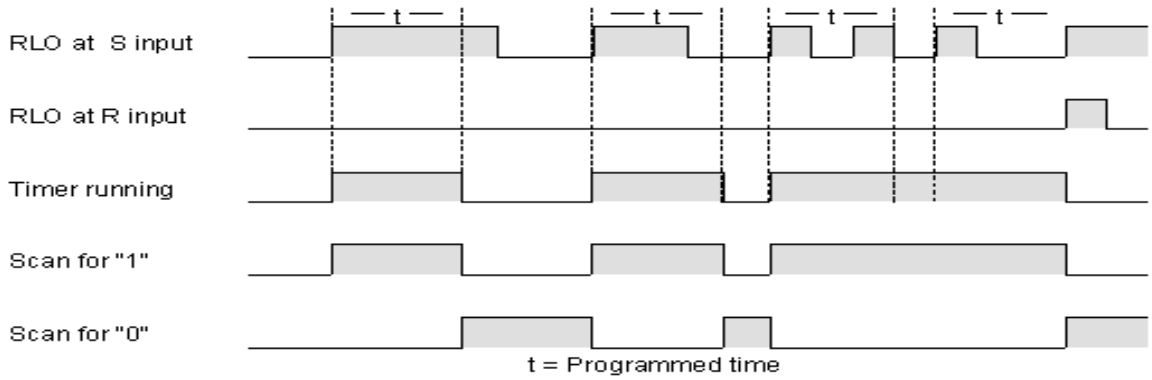
+ Timer kích có nhớ, khi có tín hiệu cạnh lên ở I0.0 Timer T5 chạy, nếu đủ thời gian đặt Timer dừng.

+ Trong quá trình chạy nếu có tín hiệu mới từ chân I0.0 thì thời gian Timer lại được tính lại từ đầu

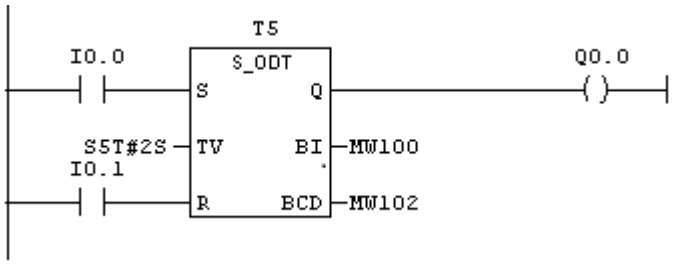
+ Trong quá trình chạy nếu có tín hiệu I0.1 thì Timer dừng

+ Q0.0 = 1 khi Timer đang chạy.

+ Các ô nhớ MW100 và MW102 lưu giá trị hiện thời của Timer theo dạng Integer và dạng BCD

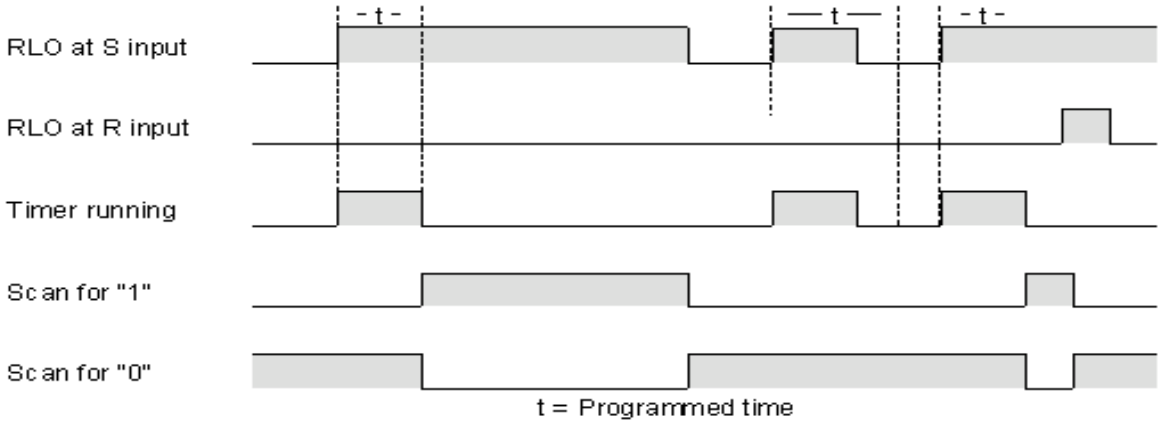


-Lệnh S_ODT:

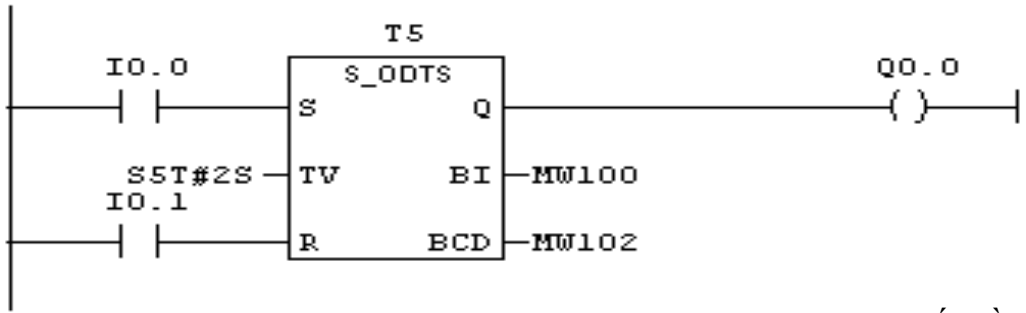


+ Nếu I0.0 =1 Timer bắt đầu chạy khi đủ thời gian thì ngưng khi đó ngõ Q0.0 sẽ lên 1 nếu I0.0 vẫn còn giữ trạng thái 1, khi có tín hiệu I0.1 thì tất cả phải được Reset về 0

+ Các ô nhớ MW100 và MW102 lưu giá trị hiện thời của Timer theo dạng Integer và dạng BCD

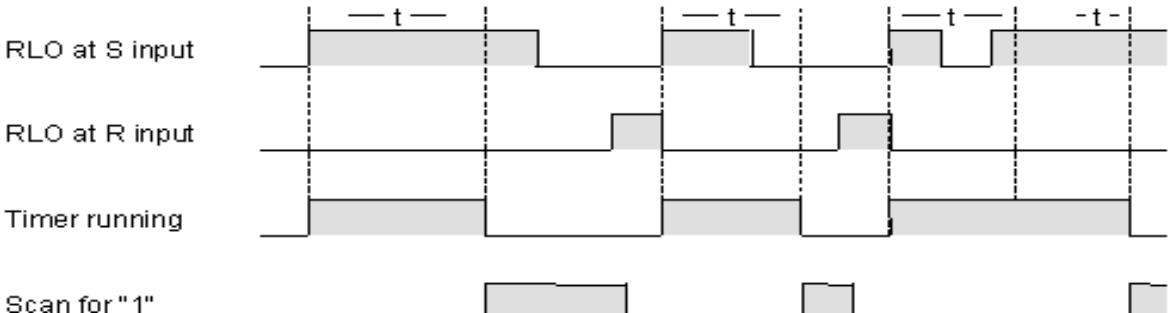


-Lệnh S_ODTS:

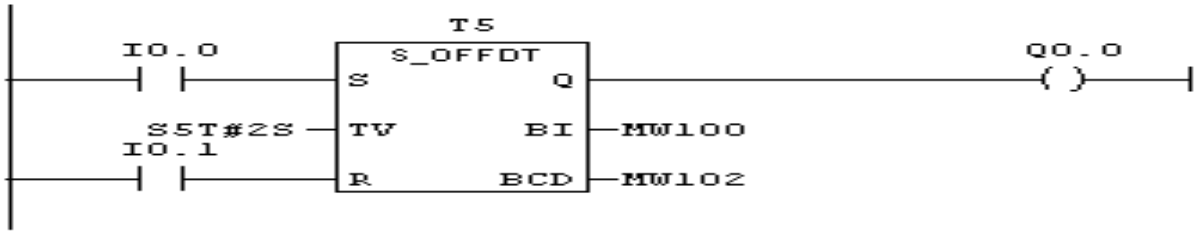


+ Timer kích có nhớ, khi có xung cạnh lên ở I0.0 Timer bắt đầu chạy, ngõ ra Q0.0 =1 khi Timer ngưng và chỉ tắt khi có tín hiệu Reset (tín hiệu I0.1)

+ Trong quá trình Timer chạy nếu có sự chuyển đổi tín hiệu từ chân I0.0 thêm 1 lần nữa thì Timer sẽ nhớ và tiếp tục chạy khi hết thời gian lần trước.

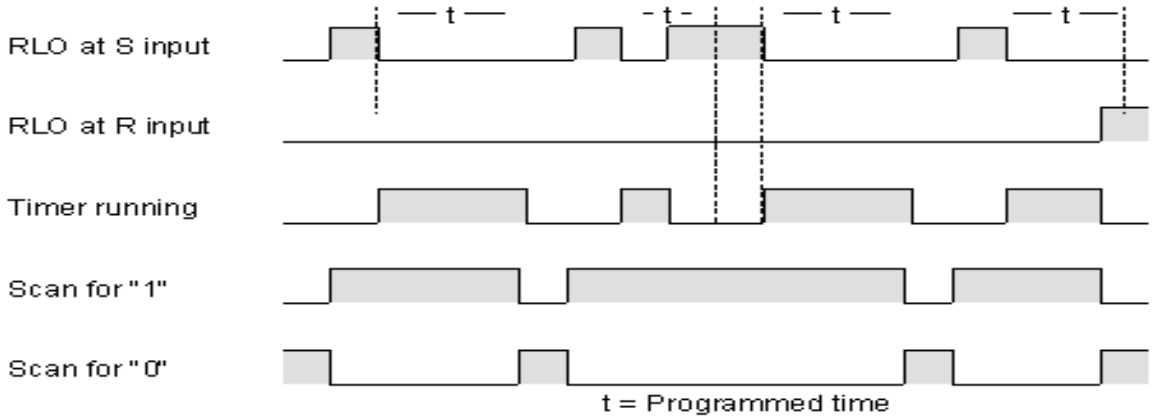


-Lệnh S_OFFDT:



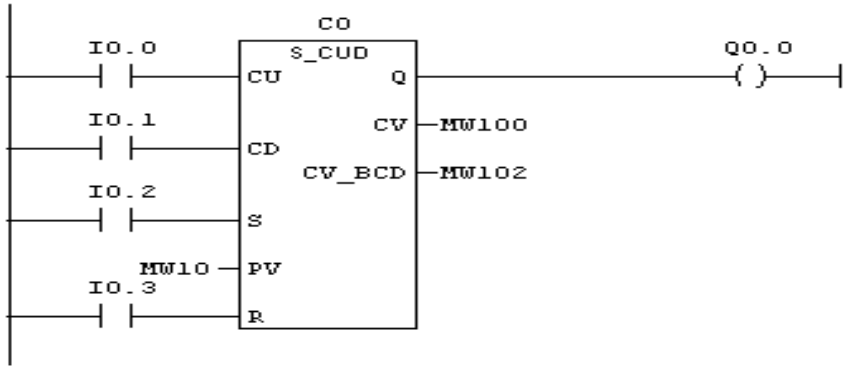
+ Khi I0.0 ON, Q0.0 = 1, khi I0.0 OFF Timer bắt đầu chạy và Q0.0 chỉ tắt khi đủ thời gian và I0.0 vẫn OFF

+ Khi có tín hiệu Reset I0.1 thì tất cả tín hiệu đều OFF.



c. Lệnh Counter

- Lệnh đếm lên xuống S_CUD:



+ Ngõ vào I0.2 = 1: đưa giá trị đếm vào PV

+ Khi I0.0 chuyển trạng thái từ 0 lên 1, C0 đếm tăng lên 1

+ Khi I0.1 chuyển trạng thái từ 0 lên 1, C0 đếm giảm xuống 1

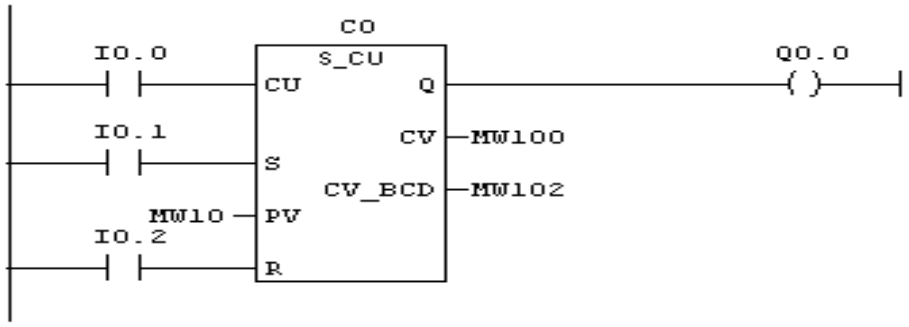
+ Khi I0.0 và I0.1 đều chuyển trạng thái thì C0 không thay đổi

+ Khi I0.3 =1 thì C0 bị Reset về 0

+ Giá trị bộ đếm hiện thời nằm trong 2 ô nhớ MW100 và MW102 dưới dạng Integer và dạng BCD, giá trị có tầm từ 0 – 999.

+ Ngõ ra Q0.0 =1 khi giá trị đếm lớn hơn 0

-Lệnh đếm lên S_CU:



+ Ngõ vào I0.1 = 1 đưa giá trị đếm vào PV

+ Khi I0.0 chuyển trạng thái từ 0 sang 1, C0 đếm tăng lên 1

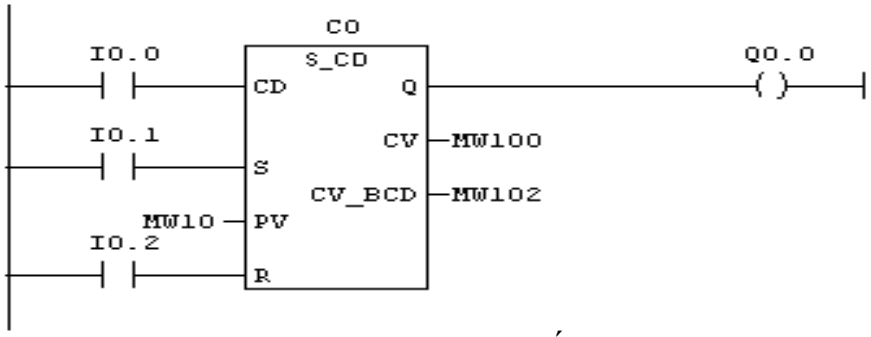
+ Khi I0.2 = 1 Counter bị Reset

+ Ngõ ra Q0.0 =1 khi giá trị đếm lớn hơn 0

+ Giá trị bộ đếm hiện thời nằm trong 2 ô nhớ MW100 và MW102 dưới dạng Integer và dạng BCD, giá trị này có tầm từ 0 – 999.

+ Ngõ ra Q0.0 = 1 khi giá trị đếm lớn hơn 0

-Lệnh đếm xuống S_CD:



+ Ngõ vào I0.1 = 1: đưa giá trị đếm vào PV

+ Khi I0.0 chuyển trạng thái từ 1 sang 0, C0 giảm đi 1

+ Khi I0.2 = 1 Counter bị Reset

+ Ngõ ra Q0.0 = 1 khi giá trị đếm lớn hơn 0

+ Giá trị bộ đếm hiện thời nằm trong 2 ô nhớ MW100 và MW102 dưới dạng Integer và dạng BCD, giá trị này có tầm từ 0-999.

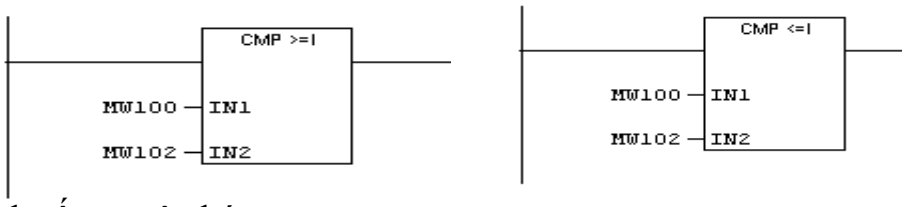
d. Lệnh so sánh

- Có 6 phép so sánh:

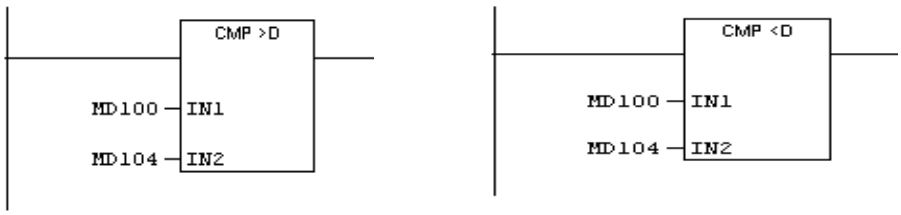
So sánh lớn hơn: >: GT

So sánh nhỏ hơn: <: LT

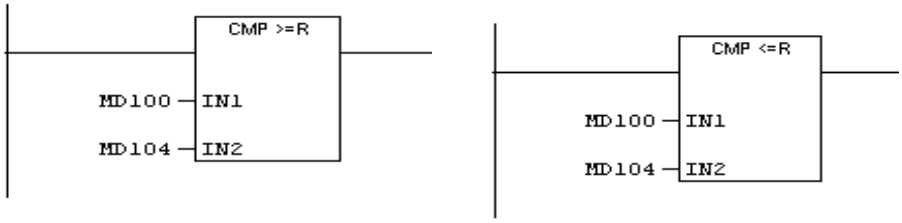
So sánh lớn hơn bằng: >=: GE
So sánh nhỏ hơn bằng: <=: LE
So sánh bằng: == : EQ
So sánh khác: <>: NE
- Khi 2 toán hạng thỏa mãn phép so sánh thì cho dòng điện chay qua
- Có 3 kiểu dữ liệu so sánh: số nguyên (I), Số nguyên kép (DI), số thực (R)
- So sánh số nguyên:



- So sánh số nguyên kép:



- So sánh số thực:



2.4. Bài tập ứng dụng.

Bài tập 1. Đặt cấu hình phần cứng cho PLC S7-300 theo cấu hình sau:

PS: 5A

CPU: CPU314-1AG13- 0AB0

IM: IM360

SM gồm: AI, AO, DI, DO

Bài tập 2: Lập trình điều khiển hệ thống băng tải hoat động tuần tự:

Ấn M1: ĐC1 quay	Ấn D1: ĐC1 dừng
Ấn M2: ĐC2 quay	Ấn D2: ĐC2 dừng
Ấn M3: ĐC3 quay	Ấn D3: ĐC3 dừng

Ấn nhầm ĐC không quay Ấn nhầm ĐC không dừng

Bài tập 3. Lập trình điều khiển hệ thống băng tải hoạt động trình tự

Ấn KĐ Động cơ 1 hoạt động sau 10s Động cơ 2 hoạt động sau 5s Động cơ 3 hoạt động

Ấn D Động cơ 3 dừng sau 10s Động cơ 2 dừng sau 5s Động cơ 1 dừng

Nếu trong quá trình hoạt động mà có sự cố xảy ra thì dừng các động cơ và đưa tín hiệu nháy đèn với chu lỳ T=5s.

BÀI 6: MỘT SỐ ỨNG DỤNG LẬP TRÌNH ĐIỀU KHIỂN BẰNG PLC

Mã bài: MĐ21.06

Giới thiệu: Nội dung bài này giới thiệu phương pháp và các kỹ năng lập trình trên phần mềm STEP 7- MicroWin.

Mục tiêu:

- Phân tích qui trình công nghệ của một số mạch máy sản xuất.
- Lập trình được một số mạch ứng dụng thường gặp trong thực tế.
- Nạp trình, vận hành và kiểm tra mạch hoạt động theo yêu cầu kỹ thuật.
- Rèn luyện đức tính tích cực, chủ động và sáng tạo

Nội dung chính:

1. Lập trình điều khiển động cơ có đảo chiều quay

1.1. Yêu cầu công nghệ

Ấn nút mở thuận MT thì động cơ quay thuận

Ấn nút mở ngược MN thì động cơ quay ngược

Ấn nút Stop thì động cơ dừng.

1.2. Trình tự thực hiện:

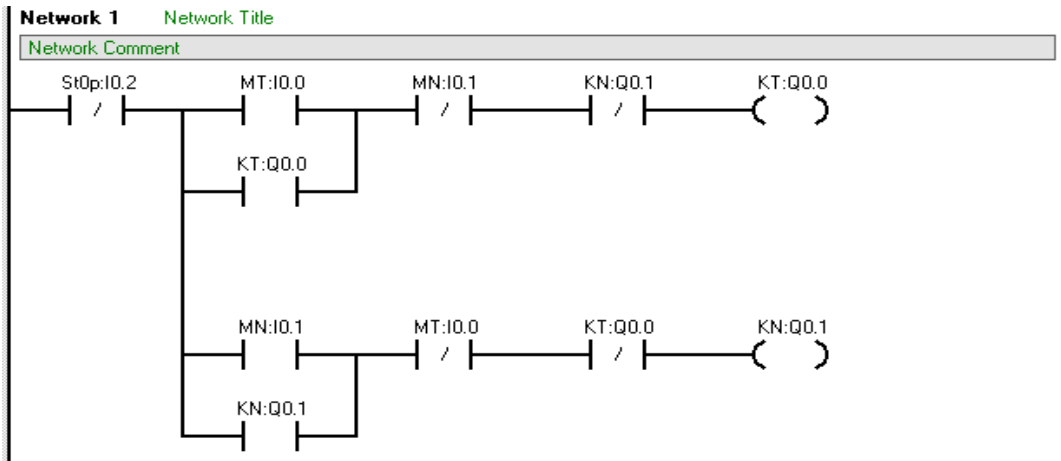
Bảng ký hiệu:

Ký hiệu	Địa chỉ	Chú thích
Mở thuận MT	I0.0	Khởi động động cơ quay thuận
Mở ngược MN	I0.1	Khởi động động cơ quay ngược
Nút dừng Stop	I0.2	Dừng động cơ
Quay thuận KT	Q0.0	Động cơ quay thuận
Quay ngược KN	Q0.1	Động cơ quay ngược

Chương trình điều khiển:

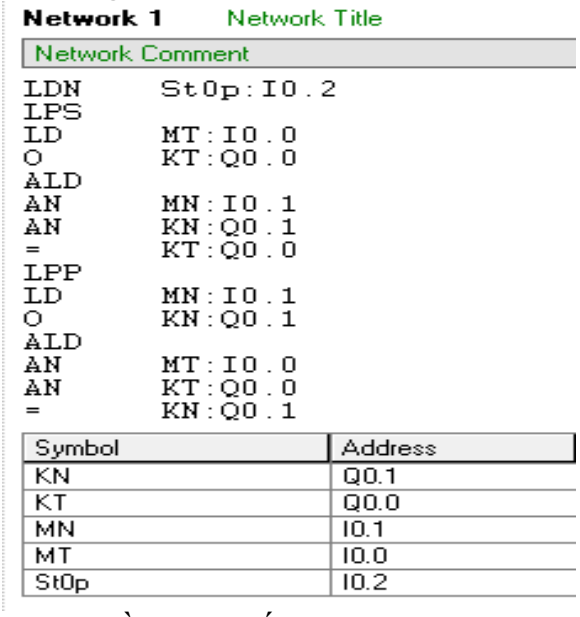
Đảo chiều trực tiếp:

Chương trình sơ đồ LAD:



Symbol	Address	Comment
KN	Q0.1	
KT	Q0.0	
MN	I0.1	
MT	I0.0	
St0p	I0.2	

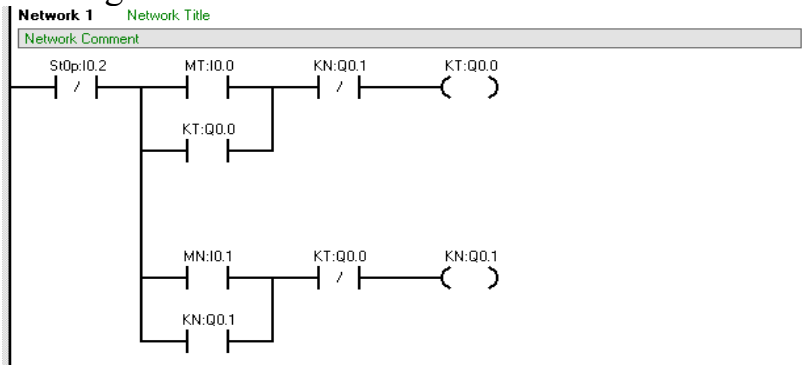
Chương trình STL:



Symbol	Address
KN	Q0.1
KT	Q0.0
MN	I0.1
MT	I0.0
St0p	I0.2

Đảo chiều gián tiếp:

Chương trình LAD:



Symbol	Address	Comment
KN	Q0.1	
KT	Q0.0	
MN	I0.1	
MT	I0.0	
St0p	I0.2	

Chương trình STL:

Network 1 Network Title

Network Comment

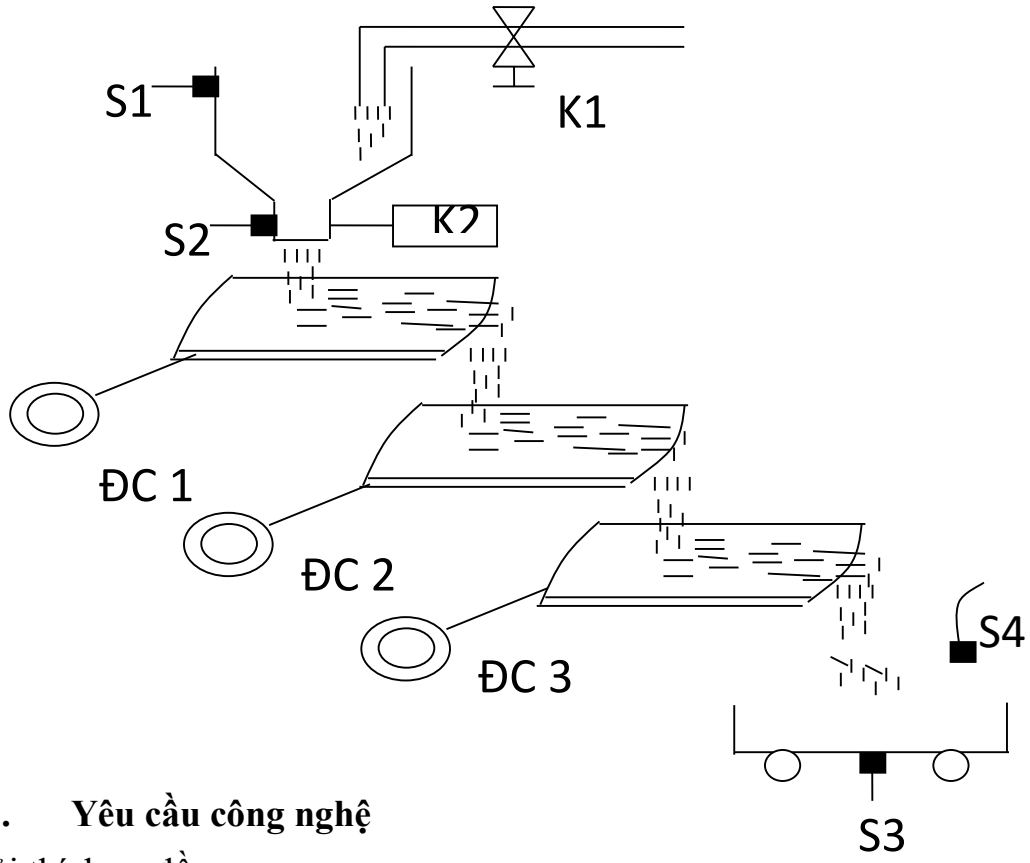
```
LDN     StOp:I0.2
LPS
LD      MT:I0.0
O       KT:Q0.0
ALD
AN      KN:Q0.1
=       KT:Q0.0
LPP
LD      MN:I0.1
O       KN:Q0.1
ALD
AN      KT:Q0.0
=       KN:Q0.1
```

Symbol	Address
KN	Q0.1
KT	Q0.0
MN	I0.1
MT	I0.0
StOp	I0.2

Network 2

2. Lập trình điều khiển hệ thống cân và cấp liệu

Sơ đồ công nghệ.



2.1. Yêu cầu công nghệ

Giải thích sơ đồ:

+ ĐC1, ĐC 2, ĐC3: Động cơ kéo băng tải cấp liệu.

+ K1: Van cấp liệu.

+ K2: Van xả liệu.

+ S1: Cảm biến báo đầy liệu trên thùng chứa.

+ S2: Cảm biến báo hết liệu trên thùng chứa.

+ S3: Cảm biến báo xe đầy liệu.

+ S4: Cảm biến báo có xe chứa liệu.

Yêu cầu:

Khi bấm nút **Start,** nếu thùng chứa hết liệu (S2 báo) K1 mở cấp liệu vào thùng chứa đến khi S1 báo đầy thì K1 đóng. Nếu S4 báo có xe đang đợi thì ĐC3 khởi động, sau 3s ĐC2 khởi động, sau 3s ĐC1 khởi động, sau 2s van K2 mở để xả liệu cấp liệu cho xe đến khi S3 báo xe đầy thì đóng K2, sau 2s cắt ĐC1, sau 2s cắt ĐC2, sau 2s cắt ĐC3 và xe rời khỏi vị trí. Khi có xe mới vào quá trình lặp lại từ đầu.

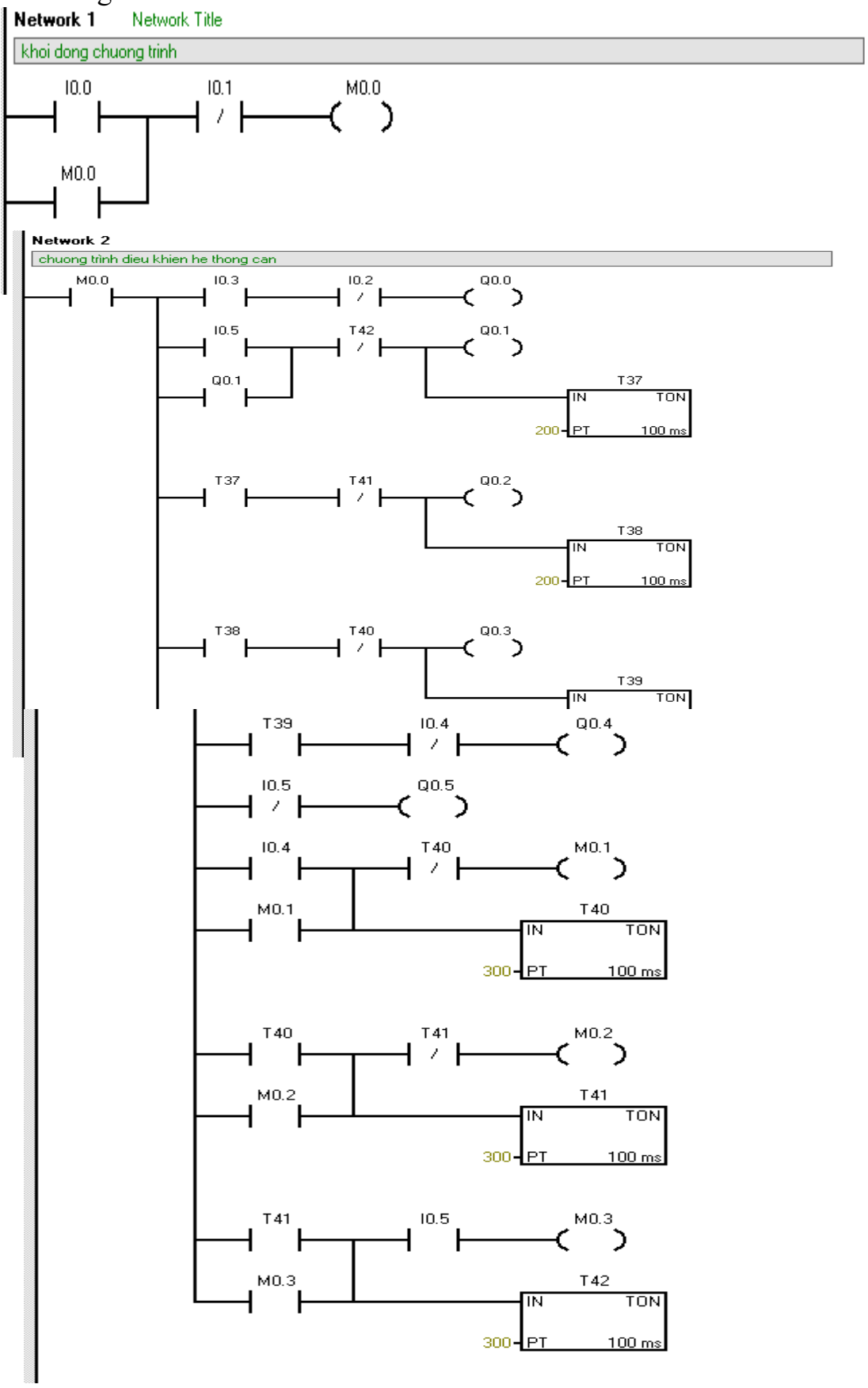
Ấn nút **Stop** dừng toàn bộ hệ thống.

2.2. Trình tự thực hiện:

Bảng ký hiệu:

Ký hiệu	Địa chỉ	Chú thích
Start	I0.0	Khởi động hệ thống
Stop	I0.1	Dừng hệ thống
S1	I0.2	Cảm biến báo đầy liệu trên thùng chứa
S2	I0.3	Cảm biến báo hết liệu trên thùng chứa
S3	I0.4	Cảm biến báo xe đầy liệu.
S4	I0.5	Cảm biến báo có xe chứa liệu
ĐC1	Q0.0	Động cơ 1 kéo băng tải cấp liệu
ĐC2	Q0.1	Động cơ 2 kéo băng tải cấp liệu
ĐC3	Q0.2	Động cơ 3 kéo băng tải cấp liệu
K1	Q0.3	Van cấp liệu.
K2	Q0.4	Van xả liệu.

Chương trình điều khiển:

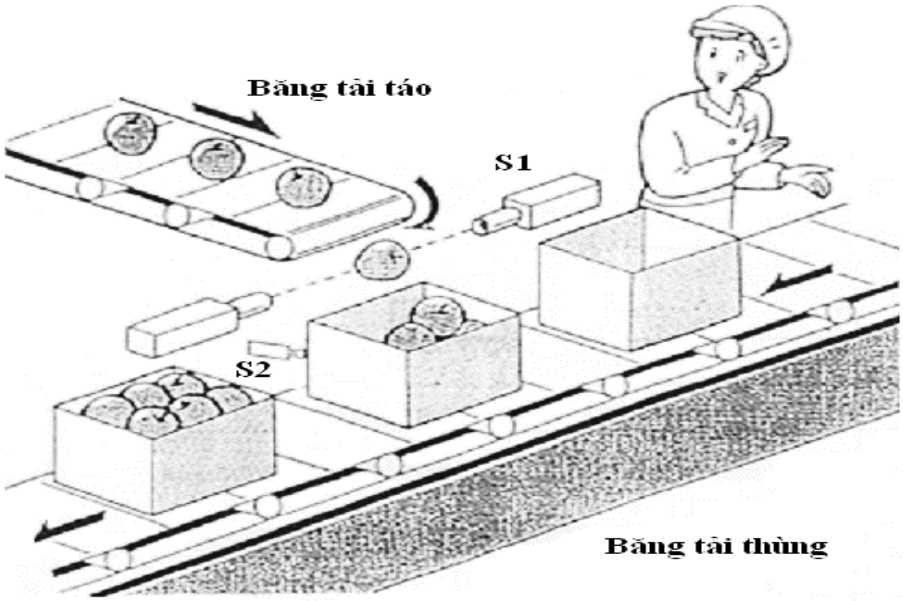


3. Lập trình điều khiển đếm sản phẩm

3.1. Yêu cầu công nghệ

Lập trình điều khiển hệ thống băng tải đếm sản phẩm bằng PLC S7-200

Sơ đồ công nghệ:



- Ấn Ol g táo vào. Khi vỏ thùng vào đến vị trí S2 thì ĐC1 dừng.

- Ngay khi ĐC1 dừng thì ĐC2 chạy để kéo băng tải táo hoạt động đưa táo rơi vào thùng. Táo được đếm bởi một cảm biến hồng ngoại S1.

- Khi số táo đưa vào thùng đủ 5 quả (mỗi hộp chứa 5 quả) thì ĐC2 dừng. Tiếp tục ĐC1 chạy lại để đưa thùng táo thành phẩm ra ngoài và đóng thùng táo mới.

- Hệ thống tự động hoạt động như trên cho đến khi ấn OFF thì dừng.

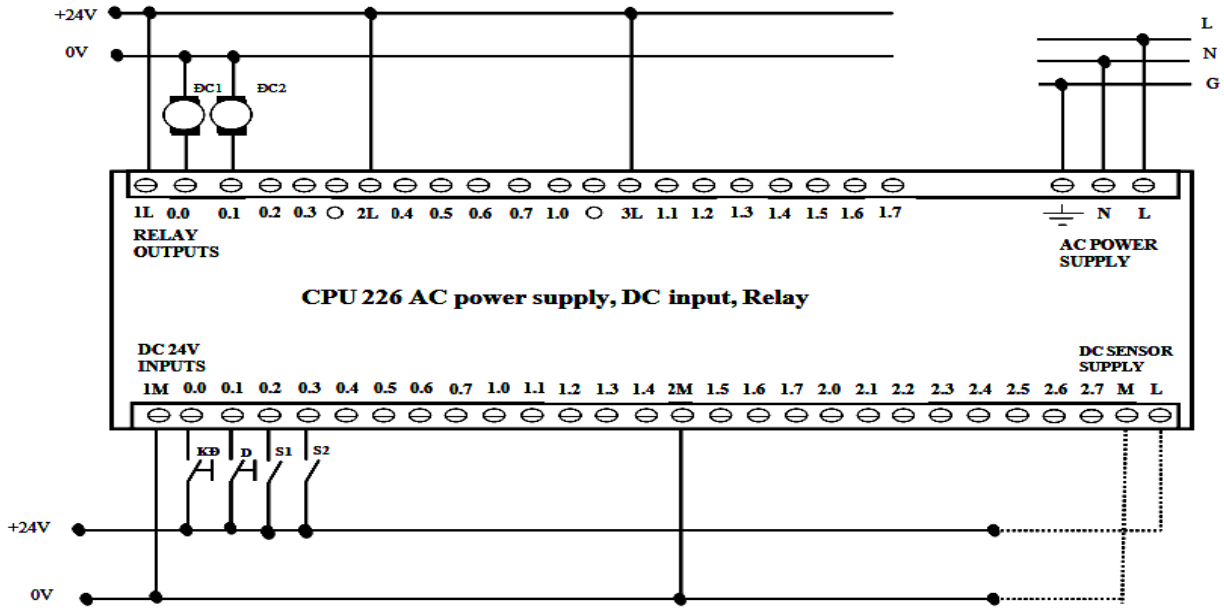
- Trong lúc hệ thống đang hoạt động mà có bất kỳ sự cố nào xảy ra thì dừng ngay và đưa tín hiệu nháy đèn với thời gian trong 1 chu kỳ là 5 giây.

3.2. Trình tự thực hiện:

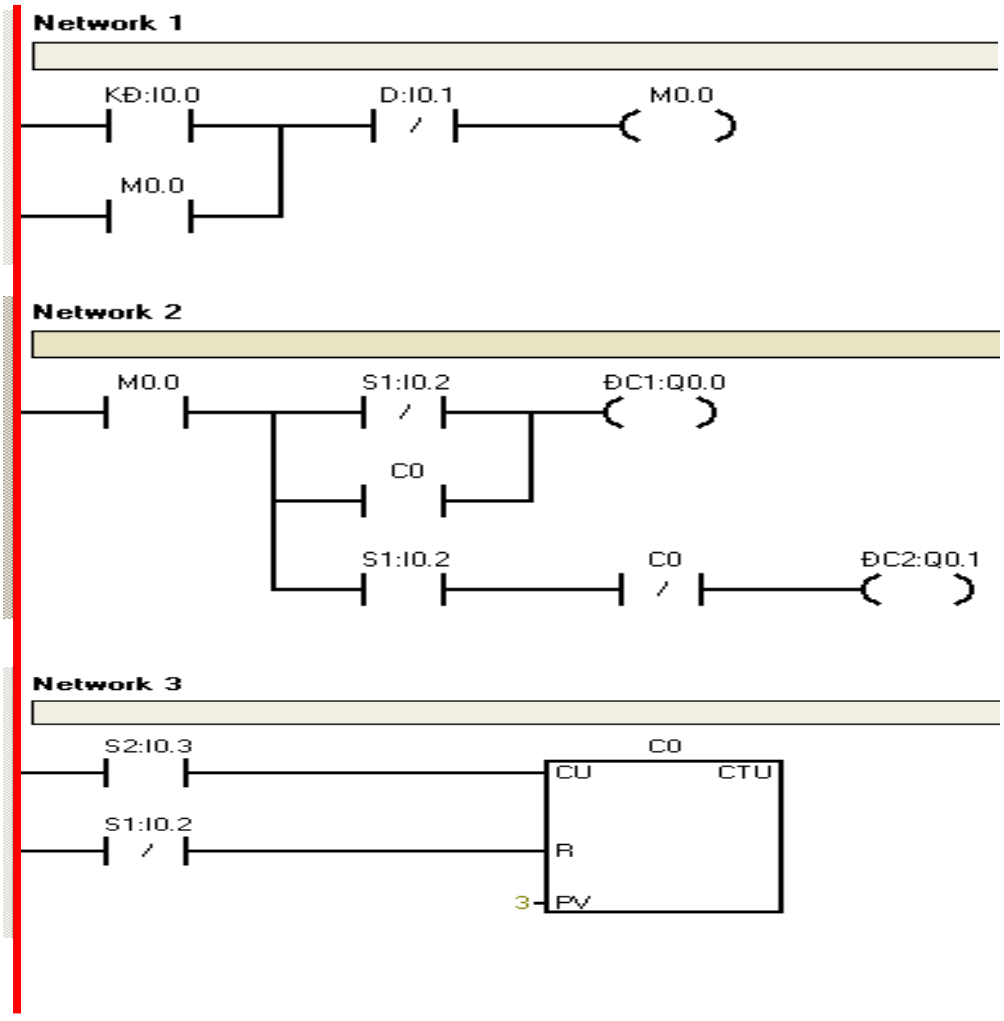
Bảng địa chỉ:

Ký hiệu	Địa chỉ	Chú thích
ON	I0.0	Nút ấn khởi động
OFF	I0.1	Nút ấn dừng
Sự cố	I0.2	Công tắc sự cố
ĐC1	Q0.0	Động cơ 1
ĐC2	Q0.1	Động cơ 2
Đèn sự cố	Q0.2	Đèn báo sự cố

Sơ đồ kết nối:



Chương trình điều khiển:

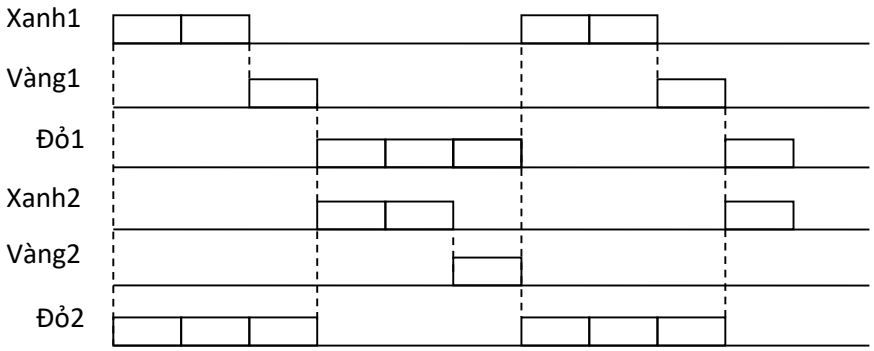


4. Lập trình điều khiển đèn giao thông

4.1. Yêu cầu công nghệ:

Lập trình điều khiển hệ thống đèn giao thông bằng PLC S7-200

Ấn nút **Start** để hệ thống làm việc theo giản đồ sau (mỗi ô tương ứng 5 giây):



Ấn nút **Stop** dừng toàn bộ hệ thống.

Ấn nút **Reset** hệ thống được reset

4.2. Trình tự thực hiện:

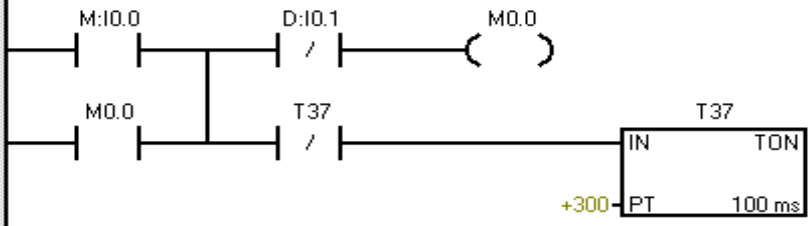
Bảng địa chỉ:

Ký hiệu	Địa chỉ	Chú thích
M	I0.0	Nút ấn khởi động
D	I0.1	Nút ấn dừng
X1	Q0.0	Đèn xanh 1
V1	Q0.1	Đèn vàng 1
Đ1	Q0.2	Đèn đỏ 1
X2	Q0.3	Đèn xanh 2
V2	Q0.4	Đèn vàng 2
Đ2	Q0.5	Đèn đỏ 2

Chương trình điều khiển:

Network 1 Network Title

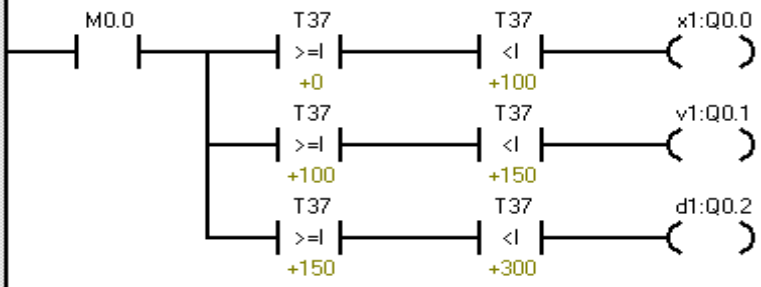
khoi dong chuong trinh



Symbol	Address	Comment
D	I0.1	
M	I0.0	

Network 2

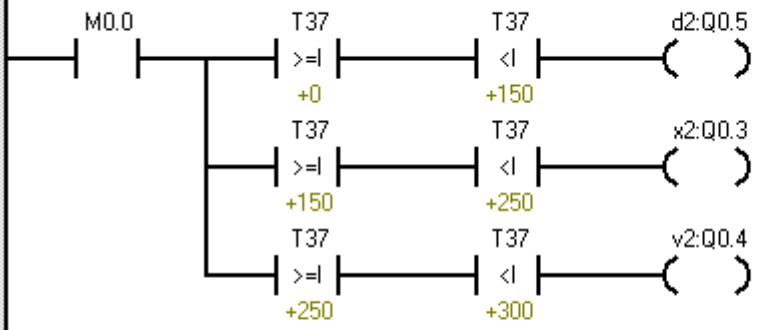
dieu khien huong 1



Symbol	Address	Comment
d1	Q0.2	
v1	Q0.1	
x1	Q0.0	

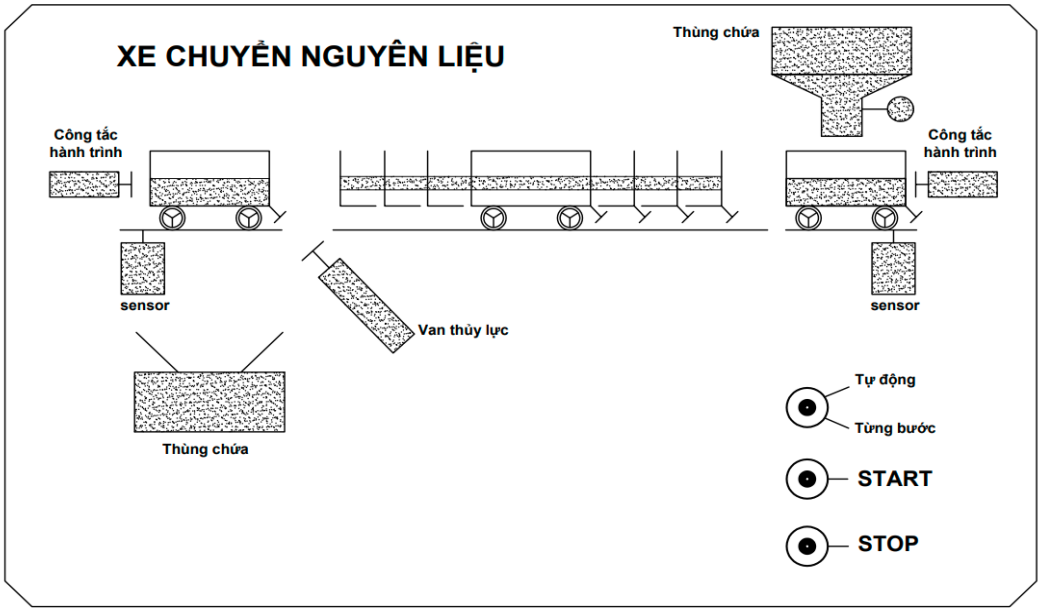
Network 3

dieu khien huong 2



Symbol	Address	Comment
d2	Q0.5	
v2	Q0.4	
x2	Q0.3	

5. Lập trình điều khiển xe chuyển nhiên liệu



Xe vận chuyển nguyên liệu hoạt động như sau:

Xe có thể thực hiện thông qua công tắc chuyển chế độ:

- Chế độ tự động I0.6
- Chế độ bước I0.7

Vị trí cơ bản: xe ở vị trí công tắc hành trình End 2 (I0.3 và xe chưa được làm đầy).

- Chế độ tự động:

Khi xe ở vị trí cơ bản và công tắc chọn chế độ đặt ở chế độ tự động, nhấn nút khởi động (I0.0) thì van xả Y1 mở, vật liệu được đổ vào xe, cảm biến Fill 2 dùng để nhận biết xe đã được đổ đầy. Khi xe đầy thì van xả Y1 mất điện và xe chạy về hướng B sau thời gian ổn định 5s, xe dừng lại tại B (trạm nhận nguyên liệu) khi chạm công tắc hành trình S2. Xy lanh thủy lực của thiết bị xả được điều khiển và tấm chắn trên xe được mở vật liệu được rót vào bồn chứa. Khi xe xả hết vật liệu cảm biến S4 phát ra tín hiệu 1, pít tông thủy lực của thiết bị xả mất điện, tấm chắn trở về vị trí cũ, xe dừng 5s sau đó chạy về hướng A. chu kỳ hoạt động được lặp lại.

Nếu trong chu kỳ hoạt động mà nút "dừng" được ấn thì quá trình vẫn tiếp tục cho đến khi xe trở về vị trí cơ bản (xe rỗng và ở trạm nhận nguyên liệu) và dừng hẳn.

- Chế độ bước:

Ở mỗi bước thực hiện phải thông qua nút ấn "Start"

Ví dụ: Khi ấn "start" xe đúng vị trí van xả được mở, khi xe đầy thì S3 tác động, van xả đóng lại. Nếu tiếp tục ấn "Start" thì xe chạy về hướng B.

Viết chương trình, kết nối và kiểm tra hoạt động theo hai cách:

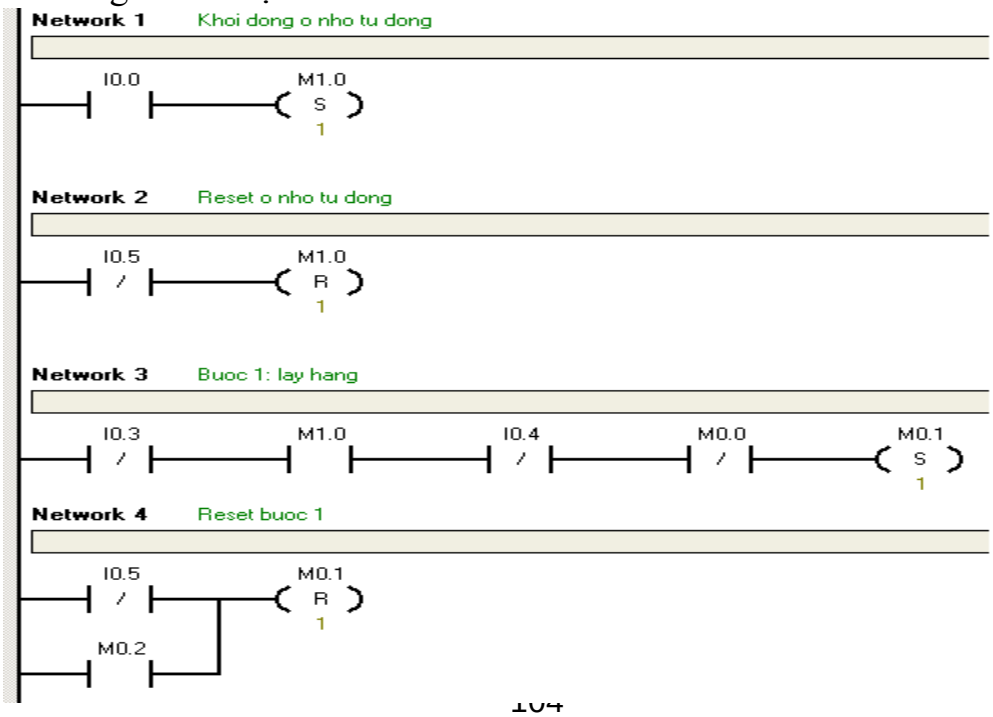
- Điều khiển dùng tổ hợp logic
- Điều khiển trình tự

Bảng ký hiệu:

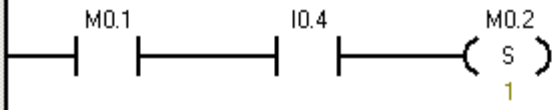
Ký hiệu	Địa chỉ	Chú thích
Start	I0.0	Khởi động hệ thống, thường hở
End 1	I0.1	Công tắc hành trình ở trạm xả, thường đóng
Fill 1	I0.2	Cảm biến báo xe rỗng, thường đóng
End 2	I0.3	Công tắc hành trình trạm nạp, thường đóng
Fill 2	I0.4	Cảm biến báo đầy, thường hở
Stop	I0.5	Dừng, thường đóng
Step	I0.6	Chế độ bước, thường hở
Auto	I0.7	Chế độ tự động, thường hở
Dir_A	Q0.0	Xe chạy về hướng A
Dir_B	Q0.1	Xe chạy về hướng B
Y1	Q0.2	Van xả nguyên liệu
Y2	Q0.3	Van thủy lực

Chương trình điều khiển:

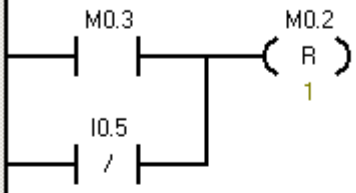
Chương trình được viết ở LAD:



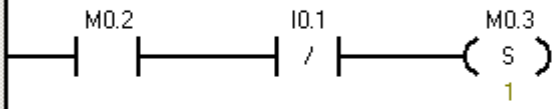
Network 5 Buoc 2: Chuyen hang



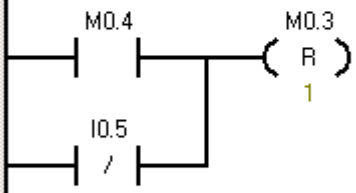
Network 6 Reset buoc 2



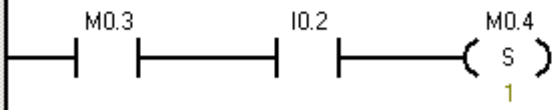
Network 7 Buoc 3: Xa hang



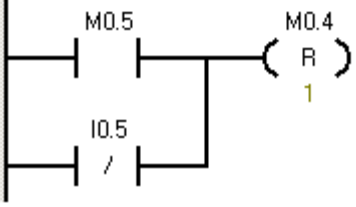
Network 8 Reset buoc 3



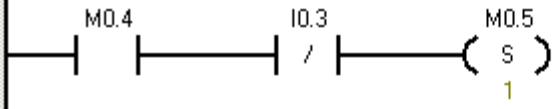
Network 9 Buoc 4: tro ve



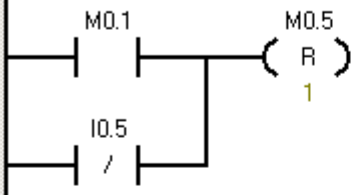
Network 10 Reset buoc 4



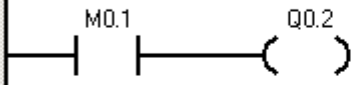
Network 11 Buoc 5: Ket thuc



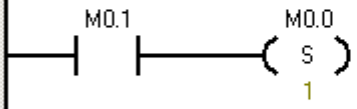
Network 12 Reset buoc 5



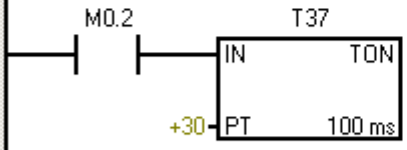
Network 13 lay hang



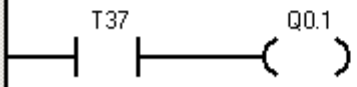
Network 14 Set o nho khoi dong



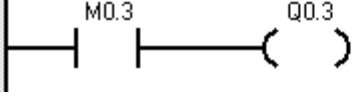
Network 15 Dinh thoi gian on dinh sau khi hang da do day xe

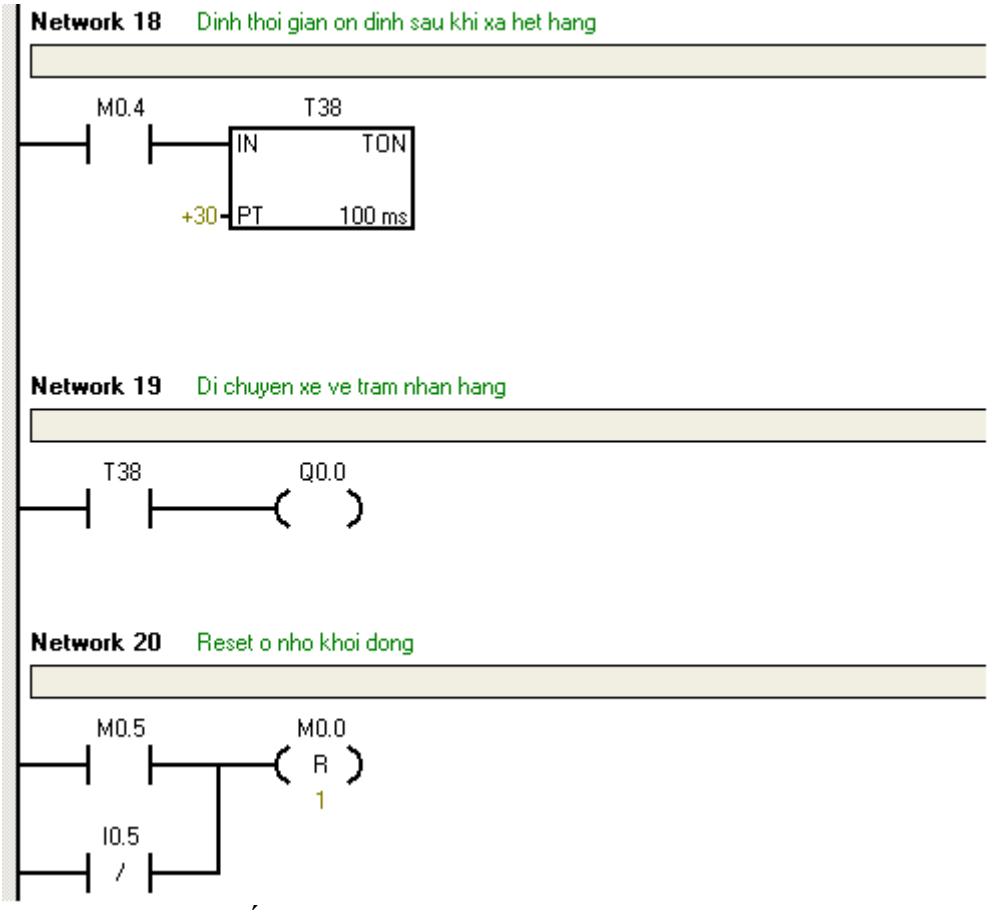


Network 16 Xe chuyen ve tram xa



Network 17 Xa hang





Chương trình được viết ở STL:

```
Network 1    Khoi dong o nho tu dong

LD      I0.0
S       M1.0, 1

Network 2    Reset o nho tu dong

LDN     I0.5
R       M1.0, 1

Network 3    Buoc 1: lay hang

LDN     I0.3
A       M1.0
AN      I0.4
AN      M0.0
S       M0.1, 1

Network 4    Reset buoc 1

LDN     I0.5
O       M0.2
R       M0.1, 1

Network 5    Buoc 2: Chuyen hang

LD      M0.1
A       I0.4
S       M0.2, 1
```

Network 5 Buoc 2: Chuyen hang

```
LD      M0.1
A       I0.4
S       M0.2, 1
```

Network 6 Reset buoc 2

```
LD      M0.3
ON      I0.5
R       M0.2, 1
```

Network 7 Buoc 3: Xa hang

```
LD      M0.2
AN      I0.1
S       M0.3, 1
```

Network 8 Reset buoc 3

```
LD      M0.4
ON      I0.5
R       M0.3, 1
```

Network 9 Buoc 4: tro ve

```
LD      M0.3
A       I0.2
S       M0.4, 1
```

Network 10 Reset buoc 4

```
LD      M0.5
ON      I0.5
R       M0.4, 1
```

Network 11 Buoc 5: Ket thuc

```
LD      M0.4
AN      I0.3
S       M0.5, 1
```

Network 12 Reset buoc 5

```
LD      M0.1
ON      I0.5
R       M0.5, 1
```

Network 13 lay hang

```
LD      M0.1
=       Q0.2
```

Network 14 Set o nho khoi dong

```
LD      M0.1
S       M0.0, 1
```

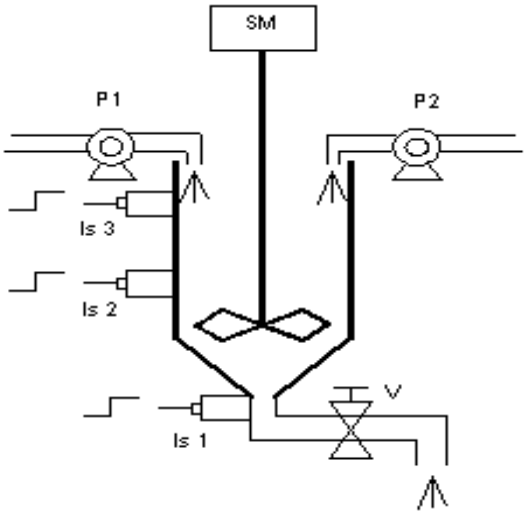
```
Network 15    Dinh thoi gian on dinh sau khi hang da do day xe

LD      M0.2
TON     T37, +30

Network 16    Xe chuyen ve tram xa

LD      T37
=       Q0.1

Network 17    Xa hang

LD      M0.3
=       Q0.3

Network 18    Dinh thoi gian on dinh sau khi xa het hang

LD      M0.4
TON     T38, +30

Network 19    Di chuyen xe ve tram nhan hang

LD      T38
=       Q0.0

Network 20    Reset o nho khoi dong

LD      M0.5
ON      I0.5
R       M0.0, 1
```

6. Lập trình điều khiển trộn liệu

6.1. Yêu cầu công nghệ:

Lập trình điều khiển hệ thống trạm trộn liệu bằng PLC S7-200

Sơ đồ công nghệ:



Khi nhấn nút **RUN** (ban đâu thùng rỗng tiếp điêm của các cảm biến mở) – PLC ra lệnh cấp điện cho P1 bơm liệu 1 vào bình. Khi liệu 1 đầy lên vị trí l**s1** (tiếp điểm ls1 đóng lại) – P1 vẫn tiếp tục bơm. Khi liệu 1 đầy lên đến vị trí l**s2** – PLC

ra lệnh dừng P1 đồng thời ra lệnh khởi động P2 và SM thực hiện khuấy. Khi liệu 2 được P2 bơm đầy đến vị trí l**s3** –PLC ra lệnh dừng P2 và SM vẫn tiếp tục khuấy. Sau 1 phút PLC ra lệnh dừng SM đồng thời ra lệnh mở V bắt đầu quá trình xả. Khi liệu xả ra ngoài thì lần lượt tiếp điểm của các cảm biến l**s3, ls2, ls1** mở ra. Khi l**s1** mở ra thì PLC ra lệnh đóng van V đồng thời ra lệnh đóng bơm P1 quá trình lặp lại như trên. Sau 3 mẻ trộn dừng hệ thống.

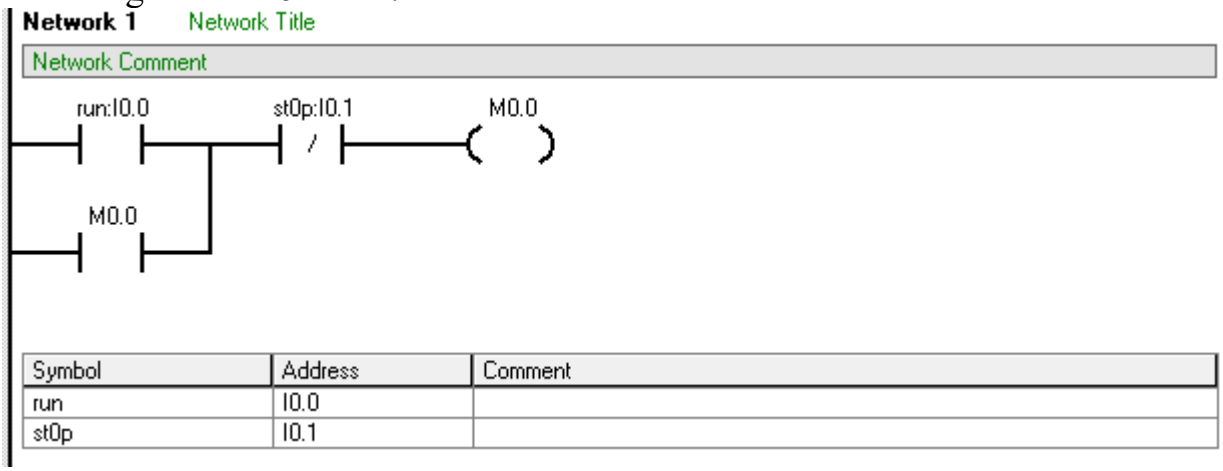
Khi nhấn nút STOP dừng toàn bộ hệ thống.

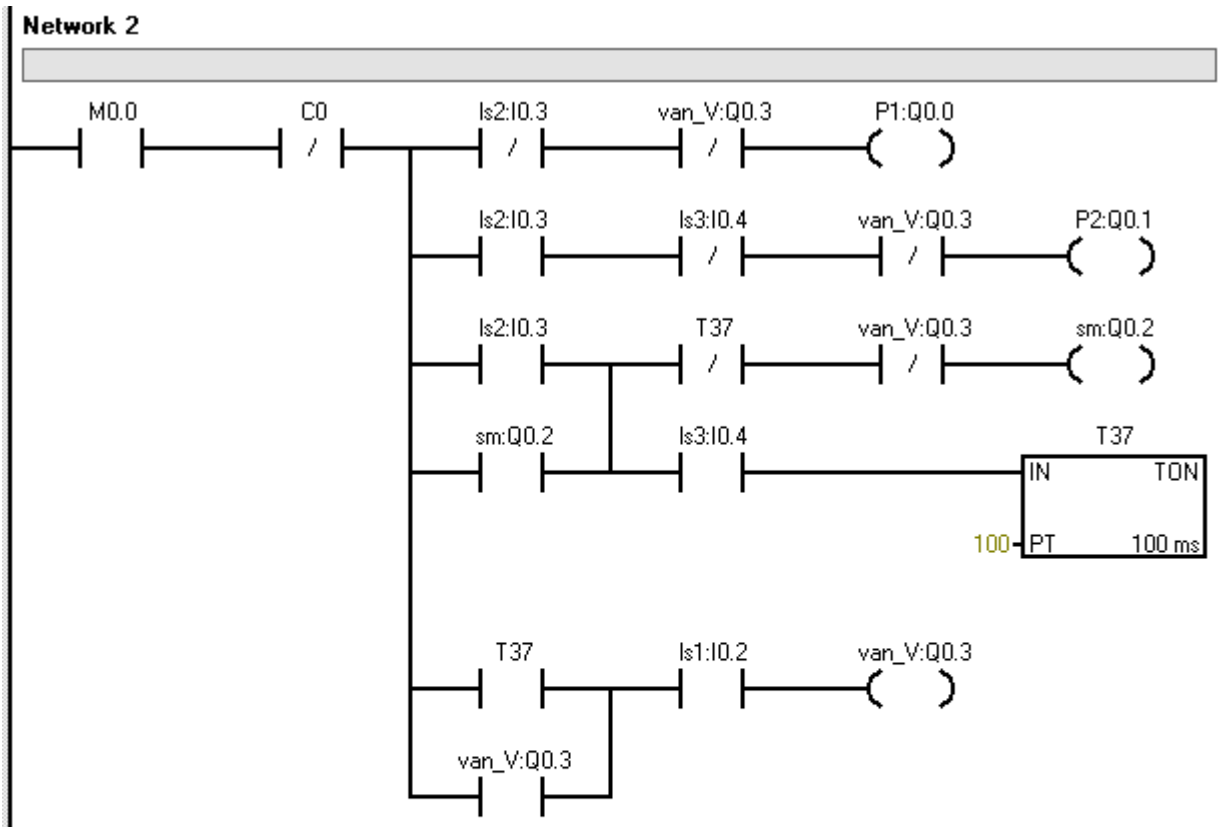
6.2. Trình tự thực hiện:

Bảng ký hiệu:

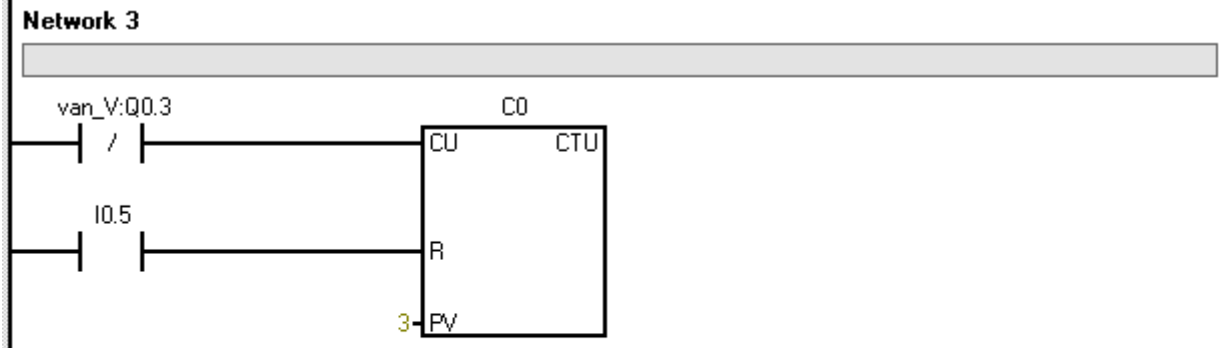
Ký hiệu	Địa chỉ	Chú thích
Run	I0.0	Khởi động hệ thống
Stop	I0.1	Dừng hệ thống
Ls1	I0.2	Cảm biến báo hết liệu trong thùng
Ls2	I0.3	Cảm biến báo đủ liệu để P2 và SM hoạt động
Ls3	I0.4	Cảm biến báo đầy liệu trong thùng
P1	Q0.0	Bơm liệu 1
P2	Q0.1	Bơm liệu 2.
SM	Q0.2	
Van_V	Q0.3	Động cơ trộn Van xả hỗn hợp đã trộn

Chương trình điều khiển:





Symbol	Address	Comment
ls1	I0.2	
ls2	I0.3	
ls3	I0.4	
P1	Q0.0	
P2	Q0.1	
sm	Q0.2	
van_V	Q0.3	



Symbol	Address	Comment
van_V	Q0.3	

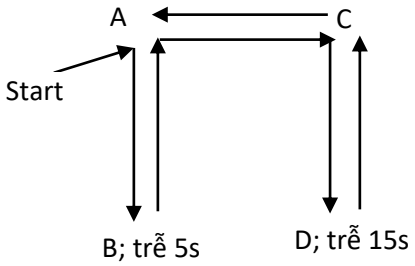
7. Lập trình điều khiển cầu trục

7.1. Yêu cầu công nghệ:

Lập trình điều khiển hệ thống cầu trục tự động bằng PLC S7_200

Vị trí ban đầu cầu trục đang ở A. Nhấn Start: Cầu trục đi xuống, chạm B: dừng xuống, chờ 5s sau đi lên. Chạm A: dừng lên, đi sang phải. Chạm C: dừng sang phải, đi xuống. Chạm D: dừng xuống, chờ 15s sau đi lên. Chạm C: dừng lên, đi sang trái. Chạm A: dừng sang trái, bắt đầu một chu trình mới.

Nhấn Stop: Hệ thống hoạt động hết chu kỳ thì dừng .



7.2. Trình tự thực hiện:

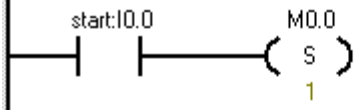
Bảng ký hiệu:

Ký hiệu	Địa chỉ	Chú thích
Start	I0.0	Khởi động hệ thống
Stop	I0.1	Dừng hệ thống
A	I0.2	Công tắc hành trình A
B	I0.3	Công tắc hành trình B
C	I0.4	Công tắc hành trình C
D	I0.5	Công tắc hành trình D
Xuống	Q0.0	Cầu trục chạy xuống
Lên	Q0.1	Cầu trục chạy lên
Phải	Q0.2	Cầu trục chạy sang phải
Trái	Q0.3	Cầu trục chạy sang trái.

Chương trình điều khiển:

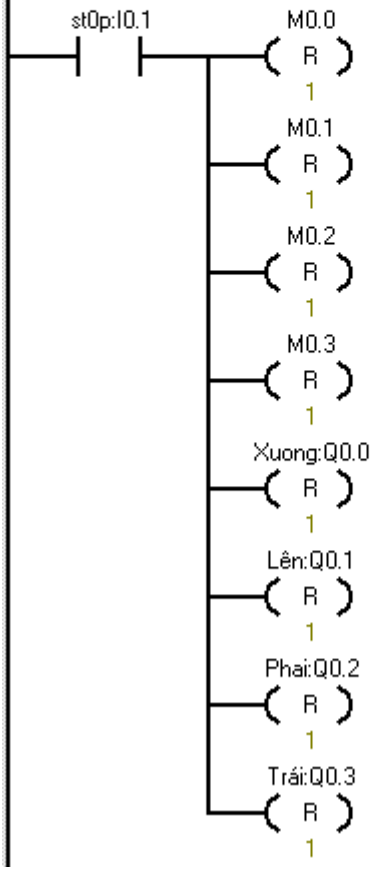
Network 1 Network Title

Network Comment



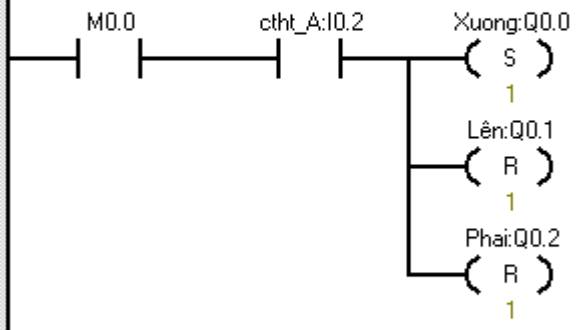
Symbol	Address	Comment
start	I0.0	

Network 2



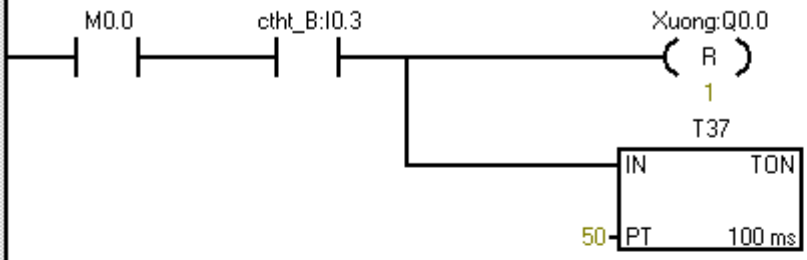
Symbol	Address	Comment
Lên	Q0.1	
Phai	Q0.2	
st0p	I0.1	
Trái	Q0.3	
Xuong	Q0.0	

Network 3



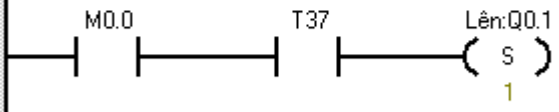
Symbol	Address	Comment
ctht_A	I0.2	
Lên	Q0.1	
Phai	Q0.2	
Xuong	Q0.0	

Network 4



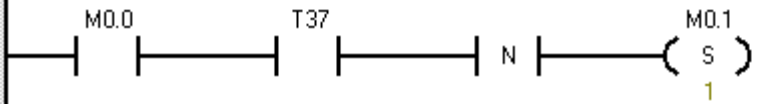
Symbol	Address	Comment
ctht_B	I0.3	
Xuong	Q0.0	

Network 5

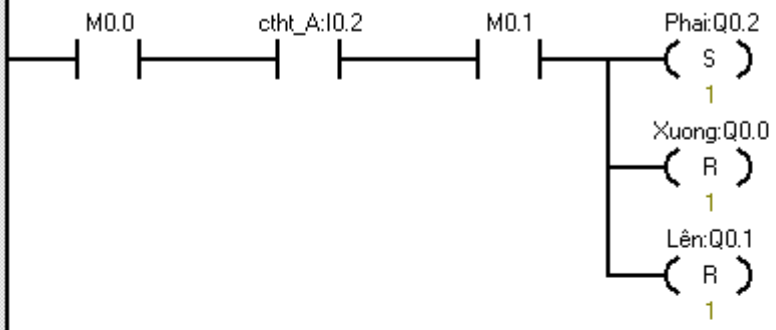


Symbol	Address	Comment
Lên	Q0.1	

Network 6

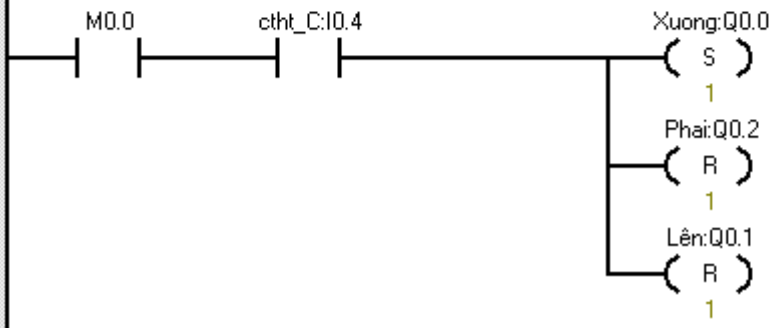


Network 7



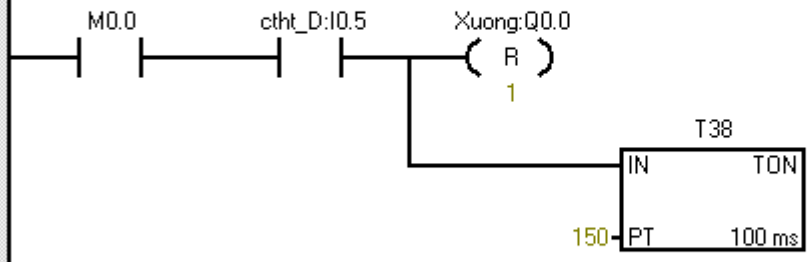
Symbol	Address	Comment
ctht_A	I0.2	
Lên	Q0.1	
Phai	Q0.2	
Xuong	Q0.0	

Network 8



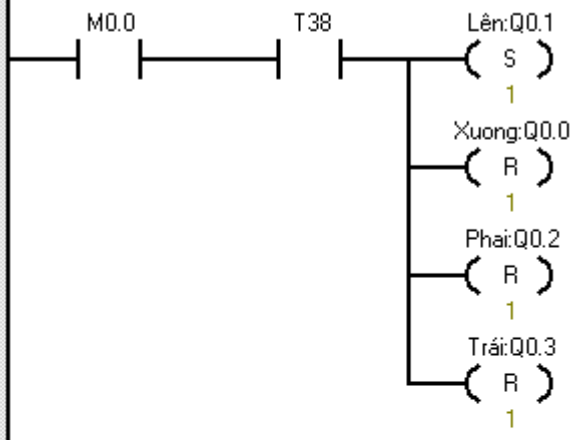
Symbol	Address	Comment
ctht_C	I0.4	
Lên	Q0.1	
Phai	Q0.2	
Xuong	Q0.0	

Network 9



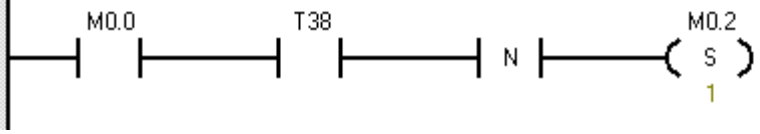
Symbol	Address	Comment
ctht_D	I0.5	
Xuong	Q0.0	

Network 10

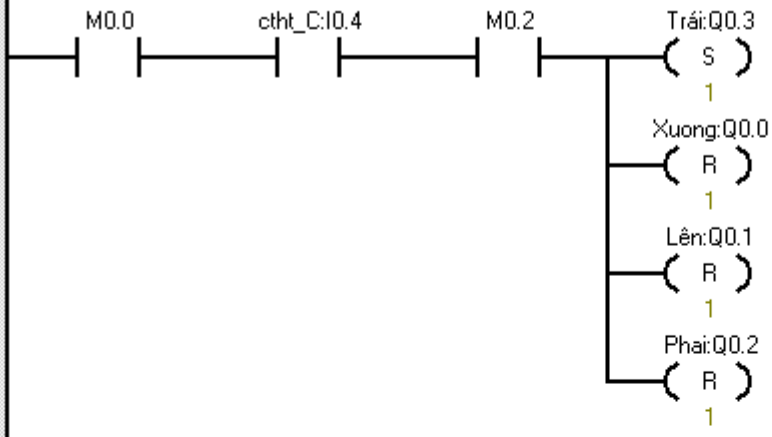


Symbol	Address	Comment
Lên	Q0.1	
Phai	Q0.2	
Trái	Q0.3	
Xuong	Q0.0	

Network 11

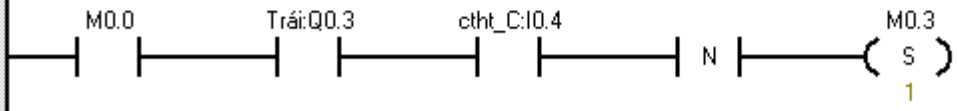


Network 12



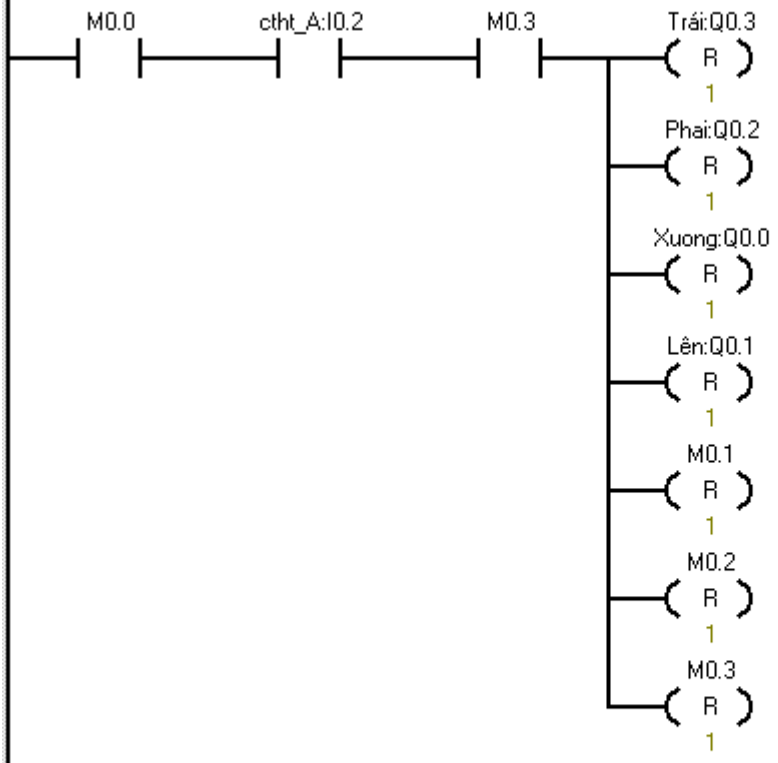
Symbol	Address	Comment
ctht_C	I0.4	
Lên	Q0.1	
Phai	Q0.2	
Trái	Q0.3	
Xuong	Q0.0	

Network 13



Symbol	Address	Comment
ctht_C	I0.4	
Trái	Q0.3	

Network 14



Symbol	Address	Comment
ctht_A	I0.2	
Lên	Q0.1	
Phai	Q0.2	
Trái	Q0.3	
Xuong	Q0.0	

Chương trình STL:

Network 1 Network Title

Network Comment

```
LD      start:I0.0
S       M0.0, 1
```

Symbol	Address	Comment
start	I0.0	

Network 2

```
LD      st0p:I0.1
R       M0.0, 1
R       M0.1, 1
R       M0.2, 1
R       M0.3, 1
R       Xuong:Q0.0, 1
R       Lên:Q0.1, 1
R       Phai:Q0.2, 1
R       Trái:Q0.3, 1
```

Symbol	Address	Comment
Lên	Q0.1	
Phai	Q0.2	
st0p	I0.1	
Trái	Q0.3	
Xuong	Q0.0	

Network 3

```
LD      M0.0
A       ctht_A:I0.2
S       Xuong:Q0.0, 1
R       Lên:Q0.1, 1
R       Phai:Q0.2, 1
```

Symbol	Address	Comment
ctht_A	I0.2	
Lên	Q0.1	
Phai	Q0.2	
Xuong	Q0.0	

Network 4

```
LD      M0.0
A       ctht_B:I0.3
R       Xuong:Q0.0, 1
TON     T37, 50
```

Symbol	Address	Comment
ctht_B	I0.3	
Xuong	Q0.0	

Network 5

```
LD      M0.0
A       T37
S       Lên:Q0.1, 1
```

Symbol	Address	Comment
Lên	Q0.1	

Network 6

```
LD      M0.0
A       T37
ED
S       M0.1, 1
```

Network 7

```
LD      M0.0
A       ctht_A:I0.2
A       M0.1
S       Phai:Q0.2, 1
R       Xuong:Q0.0, 1
R       Lên:Q0.1, 1
```

Symbol	Address	Comment
ctht_A	I0.2	
Lên	Q0.1	
Phai	Q0.2	
Xuong	Q0.0	

Network 8

```
LD      M0.0
A       ctht_C:I0.4
S       Xuong:Q0.0, 1
R       Phai:Q0.2, 1
R       Lên:Q0.1, 1
```

Symbol	Address	Comment
ctht_C	I0.4	
Lên	Q0.1	
Phai	Q0.2	
Xuong	Q0.0	

Network 9

```
LD      M0.0
A       ctht_D:I0.5
R       Xuong:Q0.0, 1
TON     T38, 150
```

Symbol	Address	Comment
ctht_D	I0.5	
Xuong	Q0.0	

Network 10

```
LD      M0.0
A       T38
S       Lên:Q0.1, 1
R       Xuong:Q0.0, 1
R       Phai:Q0.2, 1
R       Trái:Q0.3, 1
```

Symbol	Address	Comment
Lên	Q0.1	
Phai	Q0.2	
Trái	Q0.3	
Xuong	Q0.0	

Network 11

```
LD      M0.0
A       T38
ED
S       M0.2, 1
```

Network 12

```
LD      M0.0
A       ctht_C:I0.4
A       M0.2
S       Trái:Q0.3, 1
R       Xuong:Q0.0, 1
R       Lên:Q0.1, 1
R       Phai:Q0.2, 1
```

Symbol	Address	Comment
ctht_C	I0.4	
Lên	Q0.1	
Phai	Q0.2	
Trái	Q0.3	
Xuong	Q0.0	

Network 13

```
LD      M0.0
A       Trái:Q0.3
A       ctht_C:I0.4
ED
S       M0.3, 1
```

Symbol	Address	Comment
ctht_C	I0.4	
Trái	Q0.3	

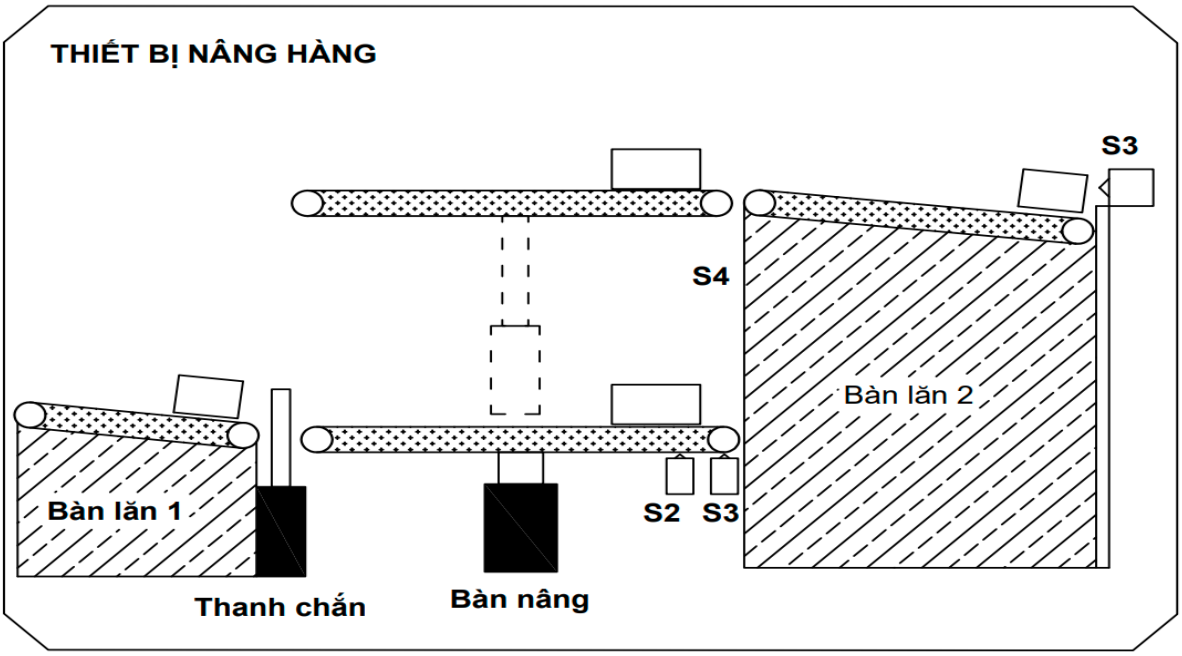
Network 14

```
LD      M0.0
A       ctht_A:I0.2
A       M0.3
R       Trái:Q0.3, 1
R       Phai:Q0.2, 1
R       Xuong:Q0.0, 1
R       Lên:Q0.1, 1
R       M0.1, 1
R       M0.2, 1
R       M0.3, 1
```

Symbol	Address	Comment
ctht_A	I0.2	
Lên	Q0.1	
Phai	Q0.2	
Trái	Q0.3	
Xuong	Q0.0	

8. Lập trình điều khiển hệ thống nâng hàng

Thiết bị nâng hàng:



Mô hình thiết bị nâng hàng

Hàng hóa từ bàn lăn thấp được đưa lên cao sang bàn lăn 2 nhờ vào bàn nâng. Hệ thống này thường thấy trong việc sắp xếp hàng hóa trong kho hoặc đưa hàng hóa vào các khoang chứa hàng của máy bay.

Yêu cầu: Thiết bị nâng hàng hoạt động như sau:

Hàng hóa được đặt trên bàn lăn 1. Bàn nâng ở vị trí giới hạn dưới thì khi ấn nút khởi động "ON", băng tải trên bàn nâng hoạt động, đồng thời thanh chắn hạ xuống (sử dụng khí nén) khoảng 2s để hàng hóa được đưa sang bàn nâng. Sau đó thanh chắn trở về vị trí cũ.

Khi hàng hóa đến vị trí cuối bàn nâng (S2), thì băng tải dừng. Khởi động từ K1 của động cơ M1 có điện kéo bàn nâng lên. Khi đến giới hạn trên thì bàn nâng dừng lại. Băng tải bắt đầu chuyển động đưa hàng sang bàn lăn 2. Khi hàng đến công tắc hành trình S5 thì băng tải dừng. Khởi động từ K2 của động cơ M1 có điện hạ bàn nâng xuống, đến giới hạn dưới thì dừng.

Quá trình mới lại bắt đầu cho đến khi nào ấn nút dừng "OFF".

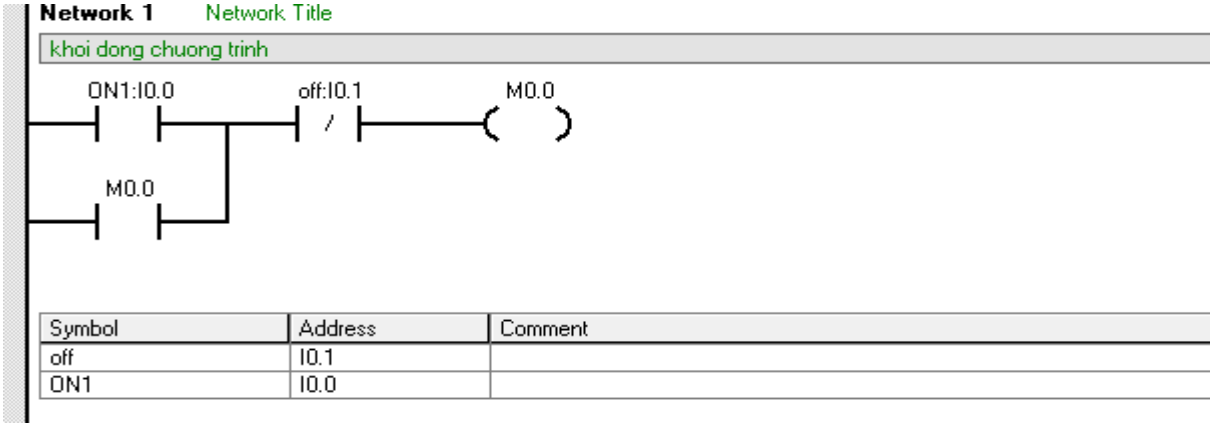
Bảng ký hiệu

Ký hiệu	Địa chỉ	Chú thích
ON	I0.0	Khởi động hệ thống, thường hở

OFF	I0.1	Dừng hệ thống, thường đóng
S2	I0.2	Báo hàng ở vị trí cuối bàn nâng, thường đóng
S3	I0.3	Giới hạn dưới bàn nâng, thường đóng
S4	I0.4	Cảm trên bàn nâng, thường đóng
S5	I0.5	Báo hàng ở cuối bàn lăn 2
Thanh chắn	Q0.0	Chặn hàng hóa ở bàn nâng 1
Băng tải	Q0.1	Băng tải chuyển hàng
K1	Q0.2	Nâng hàng hóa lên
K2	Q0.3	Hạ bàn nâng xuống

Viết chương trình, kết nối và kiểm tra hoạt động theo hai cách:

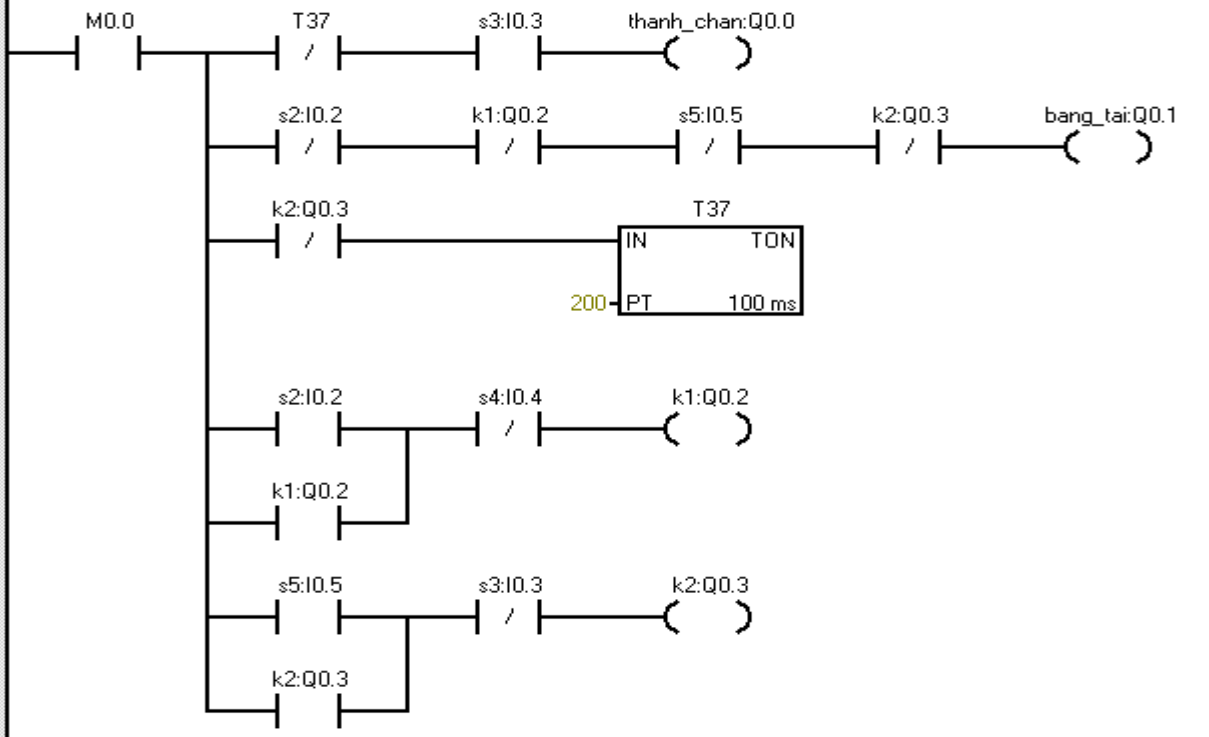
- Điều khiển dùng tổ hợp logic
- Điều khiển trình tự



Symbol	Address	Comment
off	I0.1	
ON1	I0.0	

Network 2

chuong trinh dieu khien he thong



Symbol	Address	Comment
bang_tai	Q0.1	
k1	Q0.2	
k2	Q0.3	
s2	I0.2	
s3	I0.3	
s4	I0.4	
s5	I0.5	
thanh_chan	Q0.0	

TÀI LIỆU THAM KHẢO

1. Nguyễn Doãn Phước – Phan Xuân Minh – Vũ Văn Hà: Tự động hóa với Simatic s7_300, Nhà xuất bản Khoa học kỹ thuật
2. Trung tâm Việt - Đức: Kỹ thuật điều khiển lập trình, Trung tâm Việt - Đức.
3. Nguyễn Trọng Thuần : Điều khiển LOGIC và ứng dụng, Nhà xuất bản Khoa học kỹ thuật 2006.
4. Trần Thế San (biên dịch): Hướng dẫn thiết kế mạch và lập trình PLC, NXB Đà Nẵng 2005.
5. Guenter – Wellenreuther – Dieter Zastrow: Automaticsieren mit SPS Theorie und Phraxsis, Viweg.
6. Siemens: LAD and FBD, fourth edition, siemens.
7. Siemens: Workshop to Promote the S7-300 automation platform, Siemens.