ỦY BÂN NHÂN DÂN TP THỦ ĐỨC
TRƯỜNG TRUNG CẤP NGHỀ ĐÔNG SÀI GÒN

GIÁO TRÌNH
Tên mô đun: PLC cơ bản

NGHỀ: ĐIỆN TỬ CÔNG NGHIỆP
TRÌNH ĐỘ TRUNG CẤP

(Ban hành kèm theo Quyết định số: 431/QĐ-TCN ngày 18 tháng 10 năm 2022 của Hiệu trưởng Trường trung cấp nghề Đông Sài Gòn)

TP Thủ Đức, năm 2022

MỤC LỤC

TÊN MÔ ĐUN: PLC CƠ BẢN

Mã số mô đun: MĐ27

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

Mô đun PLC cơ bản học sau các môn học, mô đun: Kỹ thuật cơ sở, chuyên môn.

Là mô đun chuyên môn nghề.

Mô đun này nhằm trang bị cho học viên các trường dạy nghề những kiến thức về điều khiển lập trình, với những kiến thức này học viên có thể áp dụng trực tiếp vào lĩnh vực sản xuất cũng như đời sống. Mô đun này cũng có thể sử dụng làm tài liệu tham khảo tốt cho các cán bộ kỹ thuật, các học viên của ngành khác có quan tâm đến lĩnh vực lập trình điều khiển.

Mục tiêu mô đun:

- 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 cở nhỏ khác.
- Phân tích được cấu tạo phần cứng và nguyên tắc hoạt động của phần mềm trong hệ điều khiển lập trình PLC.
- Thực hiện được phương pháp kết nối dây giữa PC - CPU và thiết bị ngoại vi.
- 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 và lập được chương trình để thực hiện được một số bài toán ứng dụng đơn giản trong công nghiệp.
- Phân tích được một số chương trình đơn giản, phát hiện sai lỗi và sửa chữa khắc phục.
- Đảm bảo an toàn cho người và thiết bị khi thực hiện bài tập

Nội dung của mô đun:

Số TT	Tên các bài trong mô đun	Thời gian (giờ)			
		Tổng số	Lý thuyết	Thực hành	Kiểm tra*
1	Bài mở đầu: Giới thiệu chung về PLC và bài toán điều khiển	2	2		
2	1 Đại cương về điều khiển lập trình.	4	2	2	
3	4 Các phép toán nhị phân của PLC.	12	5	6	1
4	5 Các phép toán số của PLC.	12	6	6	
5	Xử lý tín hiệu Analog.	10	5	4	1
6	PLC của các hãng khác.	5	2	3	
7	Lắp đặt mô hình điều khiển bằng PLC.	30	9	20	1
	Cộng:	75	30	42	3

BÀI 1
ĐẠI CƯƠNG VỀ ĐIỀU KHIỂN LẬP TRÌNH

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.
- So sánh PLC với các hình thức điều khiển khác.
- Rèn luyện đức tính tích cực, chủ động và sáng tạo.

Nội dung chính:

1. Giới thiệu chung về PLC

Trong thực tiễn, ngành tự động hóa (TĐH) đã luôn có vai trò đặc biệt trong các lĩnh vực sản xuất 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.

PLC – thiết bị điều khiển logic lập trình, đã du nhập vào Việt nam trên 20 năm và nay đã trở thành khái niệm phổ cập trong lĩnh vực tự động hóa công nghiệp. Thị trường PLC luôn được coi là thị trường bền vững nhất, với mức tăng

trưởng là 4,6% liên tục từ 2003 đến 2008, và ngày càng phát triển cho đến nay. Thậm chí khái niệm PLC dã không còn bao hàm là chữ viết tắt của "Điều khiển logic khả trình nữa". Khả năng truyền thông, bộ nhớ lớn và tốc độ cao của CPU đã biến PLC trở thành một sản phẩm tự động hóa tiêu chuẩn. Một thiên đường mới với PAC (Program Automation Controller) sẽ làm thay đổi bộ mặt của tự động hóa công nghiệp ở lớp điều khiển.

2. Điều khiển nối cứng và điều khiển lập trình.

Điều khiển nối cứng và điều khiển lập trình là hai khái niệm khác nhau trong lĩnh vực điện tử. Điều khiển nối cứng là loại điều khiển mà các chức năng của nó được đặt cố định thông qua kết nối dây. Nếu muốn thay đổi chức năng điều đó có nghĩa là thay đổi kết nối dây.

Điều khiển lập trình là loại điều khiển mà chức năng của nó được đặt cố định thông qua một chương trình còn gọi là bộ nhớ chương trình. Các phần tử nhập tín hiệu được nối ở ngõ vào của bộ điều khiển. Các phần tử này khởi động các cuộn dây đặt ở ngõ ra. Quá trình điều khiển ở đây được thực hiện bằng một chương trình đã soạn thảo theo mục đích, yêu cầu của việc điều khiển thiết bị. Nếu chức năng điều khiển cần được thay đổi, thì chỉ phải thay đổi chương trình bằng thiết bị lập trình cho đối tượng điều khiển tương ứng.

3. So sánh PLC với các hình thức điều khiển khác.

PLC (Programmable Logic Controller) là một loại điều khiển lập trình được sử dụng rộng rãi trong các ứng dụng công nghiệp. So với các hình thức điều khiển khác như điều khiển nối cứng, điều khiển lập trình có nhiều ưu điểm:

- Điều khiển lập trình có thể được lập trình lại để thực hiện các chức năng khác nhau mà không cần phải thay đổi kết nối dây.

- Nó cũng có thể được sử dụng để thực hiện các chức năng phức tạp hơn so với điều khiển nối cứng.

- PLC cũng có thể được sử dụng để thực hiện các chức năng mà các hình thức điều khiển khác không thể thực hiện được, chẳng hạn như điều khiển các quá trình phức tạp hoặc các quá trình đòi hỏi độ chính xác cao.

Tóm lại, điều khiển lập trình là một công nghệ điều khiển tiên tiến và có nhiều ưu điểm so với các hình thức điều khiển khác Tuy nhiên, điều khiển lập trình cũng có nhược điểm:

- Nó có thể đòi hỏi một số kiến thức về lập trình để lập trình.

- Nó cũng có thể đòi hỏi một số chi phí để mua các thiết bị lập trình và phần mềm.

- Ngoài ra, nó cũng có thể đòi hỏi một số thời gian để lập trình và cài đặt.

. Tuy nhiên, nó cũng có nhược điểm của nó

4. **Các ứng dụng của PLC trong thực tế.**

PLC là một công nghệ điều khiển lập trình được sử dụng rộng rãi trong các ứng dụng công nghiệp. Nó được sử dụng để thực hiện các chức năng khác nhau trong các quá trình sản xuất, kiểm soát và giám sát các thiết bị công nghiệp. Dưới đây là một số ứng dụng của PLC trong thực tế:

- Hệ thống băng tải: PLC được sử dụng để kiểm soát hệ thống băng tải trong các nhà máy sản xuất.
- Hệ thống đóng gói và nhãn mác: PLC được sử dụng để kiểm soát các hệ thống đóng gói và nhãn mác trong ngành thực phẩm và đồ uống.
- Hệ thống đóng chai tự động: PLC được sử dụng để kiểm soát các hệ thống đóng chai tự động trong ngành thực phẩm và đồ uống1.
- Hệ thống điều khiển cầu thang cuốn và thang máy: PLC được sử dụng để kiểm soát các hệ thống điều khiển cầu thang cuốn và thang máy trong các tòa nhà cao tầng.
- Hệ thống điều khiển cần trục công nghiệp: PLC được sử dụng để kiểm soát các hệ thống cần trục công nghiệp trong các nhà máy sản xuất.

Ngoài ra, PLC cũng được sử dụng trong các ngành công nghiệp khác như ngành giấy, ngành dệt may, ngành sản xuất ô tô, ngành sản xuất điện tử và nhiều ngành công nghiệp khác.

Bài 2
CẤU TRÚC VÀ PHƯƠNG THỨC HOẠT ĐỘNG CỦA 1 PLC

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. Cấu trúc của một PLC

Mục tiêu:

- Trình bày chức năng, nguyên lý hoạt động, cấu trúc, thành phần của một PLC bất kỳ.

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

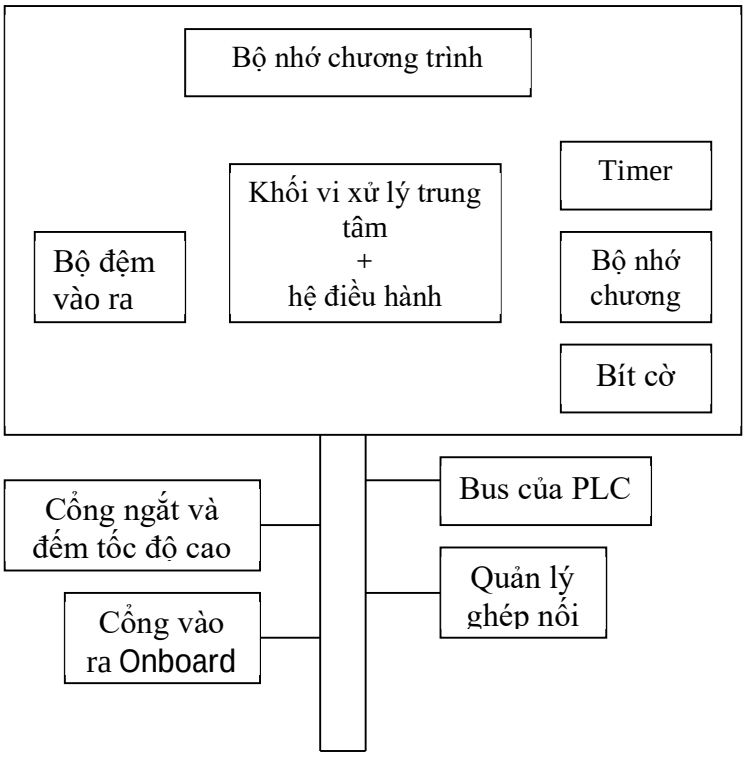
- Tốc độ xử lý

- 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.

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



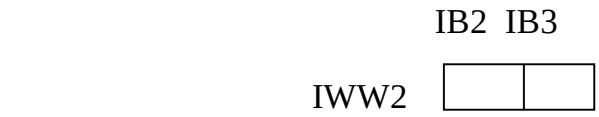
Hình 1.1: Cấu trúc của một PLC.

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 tiền có địa chỉ 0. Dữ liệu chứa trong byte nhớ gọi là nội dung.

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



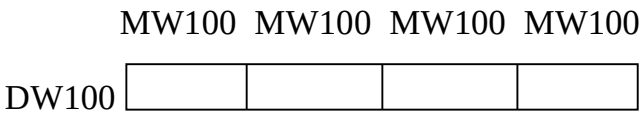
IW2 là từ đơn có địa chỉ 2

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.

2. Thiết bị điều khiển lập trình S7-200

Mục tiêu:

- Trình bày về thiết bị điều khiển lập trình của hãng Siemens.

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. Thành phần cơ bản của S7-200 là khối vi xử lý CPU212 và CPU214. Về hình thức bên ngoài, sự khác nhau của hai loại CPU này nhờ số đầu vào/ra và nguồn cung cấp.

- CPU 212 có 8 cổng vào và 6 cổng ra, có khả năng mở rộng thêm 2 modul.

- CPU 214 có 14 cổng vào và 10 cổng ra, có khả năng mở rộng thêm 7 modul.

* CPU 214 có những đặc điểm sau:

- 2048 từ nhớ chương trình

- 2048 từ nhớ dữ liệu

- 14 ngõ vào và 19 ngõ ra digital kèm theo trong khối trung tâm.

- Hỗ trợ tối đa 7 modul mở rộng kể cả modul analog.

- Tổng số cổng và/ra cực đại là 64 cổng vào/ra digital.

- 128 timer chia làm 3 loại theo độ phân giải khác nhau: 4 timers 1ms, 16 timer 10ms, 108 timer 100ms.

- 128 bộ đếm chia làm hai loại: 96 timer đếm lên và 32 timer đếm lên xuống.

- 256 ô nhớ nội bộ.

- 688 ô nhớ đặc biệt dùng để thông báo trạng thái và đặt chế độ làm việc.

- Có phép tính số học.

- Ba bộ đếm tốc độ cao với nhịp 2KHz và 7KHz.

- Hai bộ điều chỉnh tương tự.

- Toàn bộ vùng nhớ không bị mất dữ liệu trong khoảng thời gian 190 giờ khi PLC bị mất nguồn nuôi.

2.1. 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

2.2. 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

PQD:Chỉ ô nhớ có kích thước là 4 byte thuộc vùng Peripheral output, thường là cổng ra của các modul tương tự

DBX:Chỉ ô nhớ có kích thước là 1 bit trong khối dữ liệu DB, được mở bằng lệnh OPN DB (Open Data Block).

DBB: Chỉ ô nhớ có kích thước là 1 byte trong khối dữ liệu DB, được mở bằng lệnh OPN DB (Open Data Block).

DBW:Chỉ ô nhớ có kích thước là 2 byte trong khối dữ liệu DB, được mở bằng lệnh OPN DB (Open Data Block).

DBD:Chỉ ô nhớ có kích thước là 4 byte trong khối dữ liệu DB, được mở bằng lệnh OPN DB (Open Data Block).

DBx.DBX:Chỉ trực tiếp ô nhớ có kích thước là 1 bit trong khối dữ liệu DBx, với x là chỉ số của khối DB. Ví dụ DB3.DBX1.5

DBx.DBB:Chỉ trực tiếp ô nhớ có kích thước là 1 byte trong khối dữ liệu DBx, với x là chỉ số của khối DB. Ví dụ DB4.DBB1

DBx.DBW:Chỉ trực tiếp ô nhớ có kích thước là 2 byte trong khối dữ liệu DBx, với x là chỉ số của khối DB. Ví dụ DB3.DBW1

DBx.DBD:Chỉ trực tiếp ô nhớ có kích thước là 4 byte trong khối dữ liệu DBx, với x là chỉ số của khối DB. Ví dụ DB5.DBD1

DIX: Chỉ ô nhớ có kích thước là 1 bit trong khối dữ liệu DB, được mở bằng lệnh OPN DI (Open instance data block).

DIB: Chỉ ô nhớ có kích thước là 1 byte trong khối dữ liệu DB, được mở bằng lệnh OPN DI (Open instance data block).

DIW: Chỉ ô nhớ có kích thước là 2 byte trong khối dữ liệu DB, được mở bằng lệnh OPN DI (Open instance data block).

DID: Chỉ ô nhớ có kích thước là 4 byte trong khối dữ liệu DB, được mở bằng lệnh OPN DI (Open instance data block).

3. 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 1 byte (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.

4. 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.

5. Xử lý chương trình

Mục tiêu: trình bày cách xử lý chương trình trong PLC.

3.1. Vòng quét chương trình

PLC thực hiện chương trình theo chu trình lặp, mỗi vòng lặp được gọi là một vòng quét (Scan). Mỗi vòng quét được bắt đầu bằng giai đoạn chuyển dữ liệu từ các cổng vào số tới vùng bộ đệm ảo ngõ vào (I), tiếp theo là giai đoạn thực hiện chương trình. Trong từng dòng quét, chương trình được thực hiện từ lệnh đầu tiên đến lệnh kết thúc. Sau giai đoạn thực hiện chương trình là giai đoạn chuyển các

nội dung chủa bộ đệm ảo ngõ ra (Q) tới các cổng ra số. Vòng quét được kết thúc bằng giai đoạn truyền thông nội bộ và kiểm tra lỗi.

Thời gian cần thiết để PLC thực hiện được một vòng quét gọi là thời gian vòng quét (scan time). Thời gian vòng quét không cố định, tức là không phải vòng quét nào cũng thực hiện trong một khoảng thời gian như nhau. Có vòng quét thực hiện lâu, có vòng quét thực hiện nhanh tùy thuộc vào số lệnh trong chương trình được thực hiện, vào khối lượng dữ liệu truyền thông… trong vòng quét đó.

Như vậy việc đọc dữ liệu từ đối tượng để xử lý, tính toán và việc gủi tín hiệu điều khiển tới đối tượng có một khoảng thời gian trễ đúng bằng thời gian vòng quét. Nói cách khác, thời gian vòng quét quyết định tính thời gian thực của chương trình điều khiển trong PLC. Thời gian quét càng ngắn, tính thời gian thực của chương trình càng được nâng cao.

Tại thời điểm thực hiện lệnh vào/ra, thông thường lệnh không làm việc trực tiếp với cổng vào/ra mà chỉ thông qua bộ đệm ảo cả cổng trong vùng nhớ tham số. Viêc truyền thông giữa bộ đệm ảo với ngoại vi do hệ điều hành CPU quản lý. 5**.2. Cấu trúc chương trình của s7-200**

Có thể lập trình cho PLC s7-200 bằng cách sử dụng một trong các phần mềm sau:

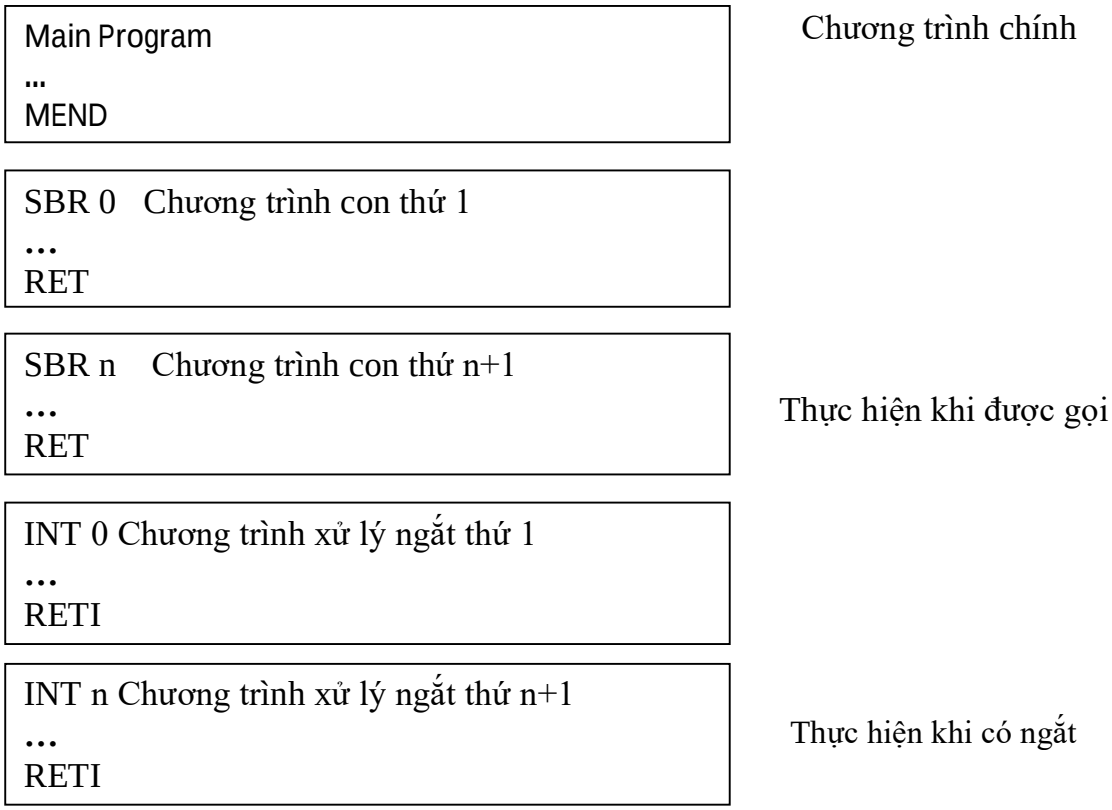
- STEP 7- Micro/DOS
- STEP 7 – Micro/WIN

Những phần mềm này đều có thể lập trình trên các máy tính lập trình họ PG7xx và các máy tính cá nhân (PC). Các chương trình cho S7-200 phải có cấu trúc bao gồm chương trình chính (main program) và sau đó đến các chương trình con và các chương trình xử lý ngắt được chỉ ra sau đây:

- Chương trình chính được kết thúc bằng lệnh kết thúc chương trình (MEND).
- Chương trình con là một bộ phận của chương trình. Các chương trình con phải được viết sau lệnh kết thúc chương trình chính, đó là lệnh MEND.
- Các chương trình xử lý ngắt là một bộ phận của chương trình. Nếu cần sử dụng chương trình xử lý ngắt phải được viết sau lệnh kết thúc chương trình chính MEND.

Các chương trình con được nhóm lại thành một nhóm ngay sau chương trình chính. Sau đó đến ngay các chương trình xử lý ngắt. Bằng cách viết như vậy, cấu trúc chương trình được rõ ràng và thuận tiện hơn trong việc đọc chương trình sau

này. Có thể tự do trộn lẫn các chương trình con và chương trình xử lý ngắt đằng sau chương trình chính.



Hình 1.2: Cấu trúc của chương một chương trình

5.3. Phương pháp lập trình

Cách lập trình cho S7-200 nói riêng và cho các PLC hãng Siemens nói chung dựa trên 3 phương pháp cơ bản:

- Phương pháp giản đồ thang (Ladder Logic, kí hiệu là LAD)
- Phương pháp liệt kê lệnh (Statement List, ký hiệu là STL)
- Phương pháp sơ đồ khối chức năng (Funtion Block Diagram)

Bài này sẽ giới thiệu chủ yếu các thành phần cơ bản cũng như cách sủ dụng trong lập trình của hai phương pháp phổ biến nhất là LAD và STL. Còn phương pháp FBD chỉ có từ Version 3.0 của phần mêm STEP 7 trở đi.

Nếu chương trình được viết theo kiểu LAD, thiết bị lập trình sẽ tự tạo ra một chương trình theo kiểu STL tương ứng. Nhưng ngược lại không phải mọi chương trình được viết theo kiểu STL đều có thể chuyển sang LAD.

Bộ lệnh của phương pháp STL được trình bày đều có một chức năng tương ứng với mỗi tiếp điểm, các cuộn dây va các hộp dùng trong LAD. Những lệnh này phải đọc và phối hợp các trạng thái của các tiếp điểm để đưa ra một quyết định vè giá rị trạng thái đầu ra hoặc một giá trị logic cho phép, hoặc không cho phép thực hiện chức năng của một (hay nhiều) hộp. Để dễ dàng làm quen với các thành phần cơ bản của LAD và của STL cần nắm được các định nghĩa cơ bản sau:

* *Định nghĩa về LAD*: LAD là một ngôn ngữ lập trình bằng đồ họa. Những thành phần cơ bản dùng trong LAD tương ứng với các thành phần của bảng điều khiển dùng rơle. Trong chương trình LAD, các phần tử cơ bản dùng để biểu diễn lệnh logic như sau:

- *Tiếp điểm*: là biểu tượng (symbol) mô tả các tiếp điểm của rowle. Các tiếp điểm đó có thể là thường đóng hay thường mở

- *Cuộn dây* (coil): là biểu tượng mô tả relay được mắc theo chiều dòng điện cung cấp cho relay.

- *Hộp (box)*: là biểu tượng mô tả các hàm khác nhau, nó làm việc khi có dọng điện chay đến hộp. Những dạng hàm thường được biểu diễn bằng hộp là bộ thời gian (timer), bộ đếm (counter) và các hàm toán học. Cuộn dây và các hộp mắc phải đúng chiều dòng điện.

- *Mạng LAD*: là đường nối các phần tử thành các mạch hoàn thiện, đi từ đường nguồn bên trái đến đường nguồn bên phải. Đường nguồn bên trái là dây nóng, đường nguồn bên phải là dây trung hòa (neutral) hay là đường trở về nguồn cung cấp,

* *Định nghĩa về STL*: Phương pháp liệt kê lệnh (STL) là phương pháp thể hiện chương trình dưới dạng tập hợp các câu lệnh. Mỗi câu lệnh trong chương trình, kết cả những câu lệnh hình thức biểu diễn một chức năng của PLC.

Định nghĩa về *ngăn xếp logic*(logic stack):

Bảng 1. Định nghĩa về ngăn xếp.

S0	Stack 0 - bit đầu tiên hay bit cuối cùng của ngăn xếp

S1	Stack 1 - bit thứ hai của ngăn xếp
S2	Stack 2 - bit thứ ba của ngăn xếp
S3	Stack 3 - bit thứ tư của ngăn xếp
S4	Stack 4 - bit thứ năm của ngăn xếp
S5	Stack 5 - bit thứ sáu của ngăn xếp
S6	Stack 6 - bit thứ bảy của ngăn xếp
S7	Stack 7 - bit thứ tám của ngăn xếp
S8	Stack 8 - bit thứ chín của ngăn xếp

Để tạo ra được một chương trình dạng STL, người lập trình phải hiểu rõ phương thức sử dụng 9 bit ngăn xếp logic của S7-200. Ngăn xếp logic là một khối gồm 9 bit chồng lên nhau. Tất cả các thuật toán liên quan đến ngăn xếp đều chỉ làm việc với bít đầu tiên hoặc bít đầu và thứ hai của ngăn xếp. Giá trị logic mới đều có thể được gửi (hoặc được nối thêm) vào ngăn xếp. Khi phối hợp hai bít đầu tiên của ngăn xếp được biểu diễn trong hình bên.

* *Định nghĩa về FBD*: Phương pháp sơ đồ khối sử dụng các "Khối" cho từng chức năng. Ký tự trong hộp cho biết chức năng (ví dụ kí tự & là phép toán logic AND). Ngôn ngữ lập trình này có ưu điểm là 1 người không chuyên lập trình như một kỹ thuật viên công nghệ cũng có thể sử dụng phương pháp soạn thảo này.

Bài 3

KẾT NỐI DÂY GIỮA PLC VÀ CÁC THIẾT BỊ NGOẠI VI

Mục tiêu:

Trình bày cách kết nối giữa PLC Siemens s7-200 với các thiết bị ngoại vi.

- 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.

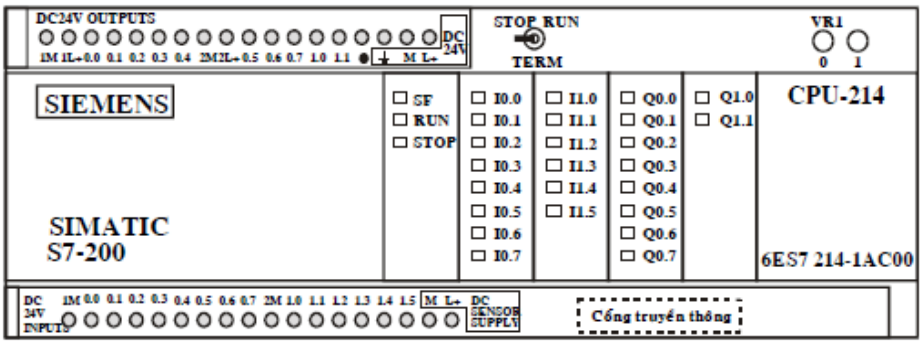
- Rèn luyện tính tỉ mỉ, cẩn thận trong công việc

Nội dung chính:

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.

1. Giới thiệu CPU 214 và cách kết nối với thiết bị ngoại vi

Sơ đồ bề mặt của bộ điều khiển lập trình S7-200 CPU 214 được cho như hình 3.



Hình 1.3: Cấu tạo của PLC S7-200.

Hệ thống bao gồm các thiết bị :

1. Bộ điều khiển PLC-Station 1200 chứa :

- CPU-214 : AC Power Supply, 24VDC Input, 24VDC Output

- Digital Input / Output EM 223 : 4x DC 24V Input, 4x Relay Output

- Analog Input / Output EM 235 : 3 Analog Input, 1 Analog Output 12bit

2. Khối Contact LSW-16

3. Khối Relay RL-16

4. Khối Đèn LL-16

5. Khối AM-1 Simulator

6. Khối DCV-804 Meter

7. Khối nguồn 24V PS-800

8. Máy tính.

9. Các dây nối với chốt cắm 2 đầu

Mô tả hoạt động của hệ thống

1. Các lối vào và lối ra CPU cũng như của các khối Analog và Digital được nối ra các chốt cắm.

2. Các khối PLC STATION – 1200, DVD – 804 và PS – 800 sử dụng nguồn 220VAC

3. Khối RELAY – 16 dùng các RELAY 24VDC

4. Khối dèn LL – 16 dùng các đèn 24V

5. Khối AM – 1 dùng các biến trở 10KΩ

Dùng các dây nối có chốt cắm 2 đầu và tùy từng bài toán cụ thể để đấu nối các lối vào/ra của CPU 214, khối Analog EM235, khối Digital EM222 cùng với các đèn, contact, Relay, biến trở, và khối chỉ thị DCV ta có thể bố trí rất nhiều bài thực tập để làm quen với cách hoạt động của một hệ thống PLC, cũng như cách lập trình cho một hệ PLC.

Để cho bộ điều khiển lập trình này hoạt động được thì người ta phải kết nối PLC với nguồn cung cấp và các ngõ vào ra của nó với thiết bị ngoại vi.

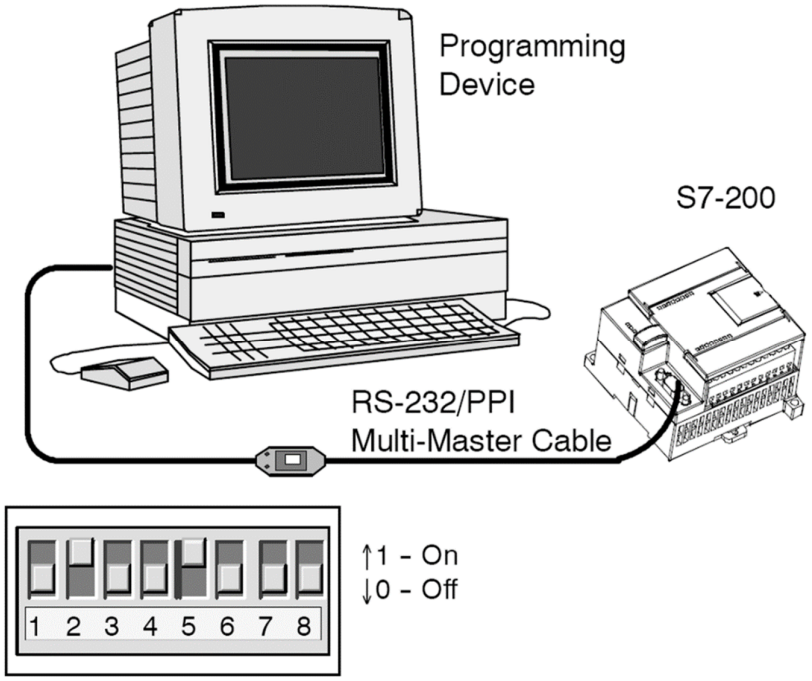
Muốn nạp chương trình vào CPU, người sử dụng phải soạn thảo chương trình bằng các thiết bị lập trình hặc máy tính với phần mềm tương ứng cho loại PLC đang sử dụng và có thể nạp trực tiếp vào CPU hoặc copy chương trình vào card nhớ để sử dụng và có thể nạp trực tiếp vào CPU của PLC.

Thông thường khi lập trình cũng như khi kiểm tra hoạt động của PLC thì người lập trình thường kết nối trực tiếp thiết bị lập trình hoặc máy tính cá nhân với PLC.

Như vậy, để hệ thống điều khiển có thể điều khiển và lập trình bằng PLC thì cần phải kết nối PLC với máy tính cũng như các ngõ vào/ra với thiết bị ngoại vi.

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 được cho như hình 4



Hình 1.4: Kết nối máy tính với CPU qua cổng truyền thông PPI

Sử dụng cáp PC/PPI.

Tùy theo tốc độ truyền giữa máy tính và CPU mà các công tắc 1,2,3 được để ở vị trí thích hợp. Thông thường đối với CPU 214 thì tốc độ truyền thường đặt là 9,6 Kbaud (tức công tắc 1,2,3 được đặt theo thứ tự là 010).

Tùy theo truyền thông là 10 Bit hay 11 Bit mà công tắc 4 được đặt ở vị trí thích hợp. Khi kết nối bình thường với máy tính thì công tắc 4 chọn ở chế độ truyền thông là 11 Bit. Công tắc 5 ở cáp PC/PPI được sử dụng để kết nối port truyền thông RS-232 của một modem với S7-200 CPU.

Khi kết nối bình thường với máy tính thì công tắc 5 được đặt ở vị trí data Comunications Equiment (DCE). Khi kết nối cáp PC/PPI với một modem thì port RS-232 của cáp PC/PPI được đặt ở vị trí Dât Terminal Equipment (DTE).

b. Kết nối vào/ra với ngoại vi

Các ngõ vào/ra của PLC cần thiết để điều khiển và giám sát quá trình điều khiển. Các ngõ vào và ra có thể được phân thành 2 loại cơ bản: số (digital) và tương tự (analog). Hầu hết các ứng dụng sử dụng các ngõ vào/ra số. Trong bài này chỉ đề cập đến việc kết nối các ngõ vào/ra số với ngoại vi, còn đối với ngõ vào/ra tương tự sẽ trình bày ở phần sau.

Đối với bộ điều khiển lập trình họ S7-200, hãng Siemens đã đưa ra rất nhiều loại CPU với điện áp cung cấp cho các ngõ vào/ra khác nhau.

Tùy thuộc vào từng loại CPU mà ta có thể nối dây khác nhau. Việc thực hiện nối dây cho CPU có thể tra cứu sổ tay kèm theo của hãng sản xuất.

* *Nối nguồn cung cấp cho CPU*

Tùy theo loại và họ PLC mà các CPU có thể là khối riêng hoặc có đặt sẵn các ngõ vào và ra cũng như một số chức năng đặc biệt khác. Hầu hết các PLC họ S7-200 được nhà sản xuất lắp đặt các khâu vào, khâu ra và CPU trong cùng một vỏ hộp. Nhưng nguồn cung cấp cho các khâu này hoàn toàn độc lập nhau. Nguồn cung cấp cho CPU của họ S7-200 có thể là:

Xoay chiều: 20...29 VAC , f = 47...63 Hz;

85...264 VAC, f = 47...63 Hz

***** *Kết nối các ngõ vào số với ngoại vi*

Các ngõ vào của PLC có thể được chế tạo là một khối riêng, hoặc kết hợp với các ngõ ra chung trong một khối hoặc được tích hợp trên khối CPU.

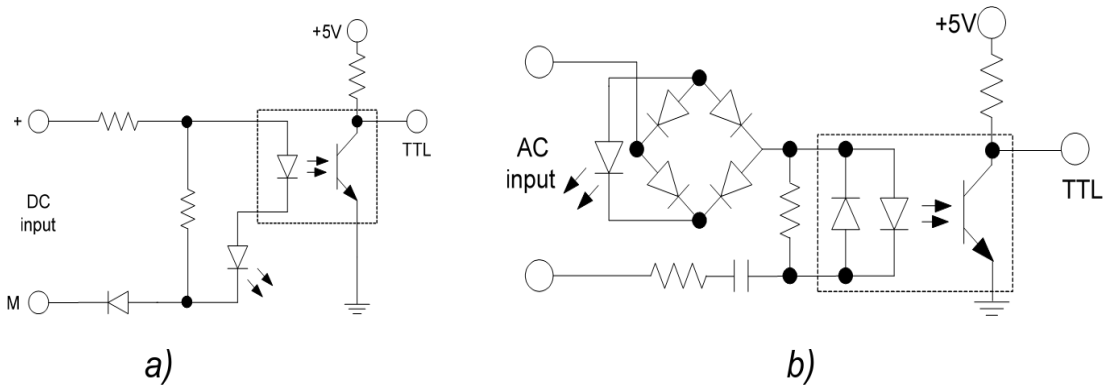
Trong trường hợp nào cũng vậy, các ngõ vào cũng phải được cung cấp nguồn riêng với cấp điện áp tùy thuộc vào loại ngõ vào. Cần lưu ý trong một khối ngõ vào cũng như các ngõ vào được tích hợp sẵn trên CPU có thể có các nhóm được cung cấp nguồn độc lập nhau.

Vì vậy cần lưu ý khi cấp nguồn cho các nhóm này. Nguồn cung cấp cho các khối vào của họ S7-200 có thể là:

Xoay chiều: 15…35 VAC, f = 47…63 Hz; dòng cần thiết nhỏ nhất là 4mA

79…135 VAC, f = 47…63 Hz; dòng cần thiết nhỏ nhất là 4mA

Sơ đồ mạch điện bên trong của các ngõ vào được cho như hình 5 a,b:



Hình 1.5: a) Mạch điện của 1 ngõ vào số sử dụng nguồn cung cấp DC.

b) Mạch điện của 1 ngõ vào số sử dụng nguồn cung cấp AC.

Tùy theo yêu cầu mà có thể quyết định sử dụng loại ngõ vào nào.

+ Ngõ vào DC:

- Điện áp DC thường thấp do đó an toàn hơn.
- Đáp ứng ngõ vào DC rất nhanh.
- Điện áp DC có thể được kết nối với nhiều phần tử trong hệ thống điện.

+ Ngõ vào AC:

- Ngõ vào AC yêu cầu cần phải có thời gian. Ví dụ đối với

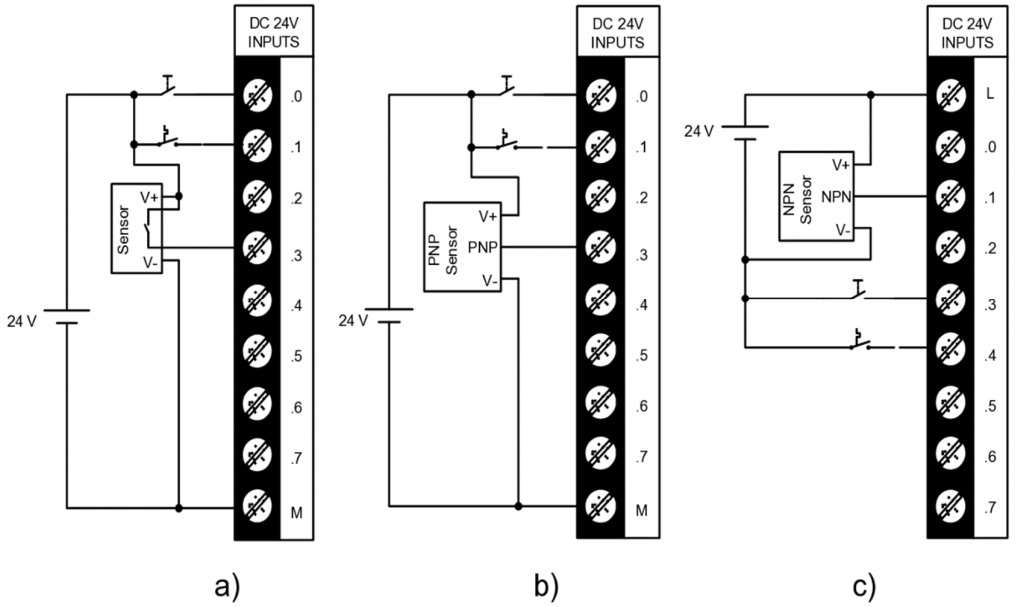
điện áp có tần số 50 Hz phải yêu cầu thời gian đến 1/50 giây mới nhận biết được.

- Tín hiệu AC ít bị nhiễu hơn tín hiệu DC, vì vậy chúng thích hợp với khoảng cách lớn và môi trường nhiễu (từ).
- Nguồn AC kinh tế hơn.
- Tín hiệu AC thường được sử dụng trong các thiết bị tự động hiện hữu.

Đối với các ngõ vào số, khi kết nối với ngoại vi, ngoại trừ các trường hợp đặc biệt thì thông thường mỗi một ngõ vào được kết nối với một bộ tạo tín hiệu nhị phân như: nút nhấn, công tắc, cảm biến tiếp cận .... Hình 6 a,b,c minh họa cách kết nối dây các ngõ vào PLC với các bộ tạo tín hiệu nhị phân khác nhau.

Cần lưu ý đến các loại cảm biến khi kết nối với các ngõ vào PLC.

Trong ví dụ hình 6 a có 3 ngõ vào, một là nút nhấn thường hở, hai là tiếp điểm của relay nhiệt, và ba là cảm biến tiếp cận với ngõ ra là relay. Cả ba bộ tạo tín hiệu này được cung cấp bởi một nguồn 24VDC. Khi tiếp điểm hở hoặc cảm biến phát tín hiệu "0" thì không có điện áp tại các ngõ vào. Nếu các tiếp điểm được đóng lại hoặc cảm biến phát tín hiệu "1" thì ngõ vào được cấp điện.



Hình 1.6: Kết nối ngõ vào với ngoại vi.

a. Nút nhấn và cảm biến có ngõ ra là relay nối với ngõ vào loại sinking.

b. Nút nhấn và cảm biến loại PNP nối với ngõ vào loại sinking.

c. Nút nhấn và cảm biến loại NPN nối với ngõ vào loại sourcing.

Đối với các ngõ vào ra của CPU 214 DC/DC/DC, CPU 224 AC/DC/Relay

** Kết nối các ngõ ra số với ngoại vi*

Các ngõ ra của PLC có thể được chế tạo là một khối riêng, hoặc kết hợp

với các ngõ ra chung trong một khối hoặc được tích hợp trên khối CPU. Trong trường hợp nào cũng vậy, các ngõ ra cũng phải được cung cấp nguồn riêng với cấp điện áp tùy thuộc vào loại ngõ ra. Cần lưu ý trong một khối ra cũng như các ngõ ra được tích hợp sẵn trên CPU có thể có các nhóm được

cung cấp nguồn độc lập nhau. Vì vậy cần lưu ý khi cấp nguồn cho các nhóm

này. Nguồn cung cấp cho các khối ra của họ S7-200 có thể là:

Xoay chiều: 20...264 VAC , f = 47...63 Hz.

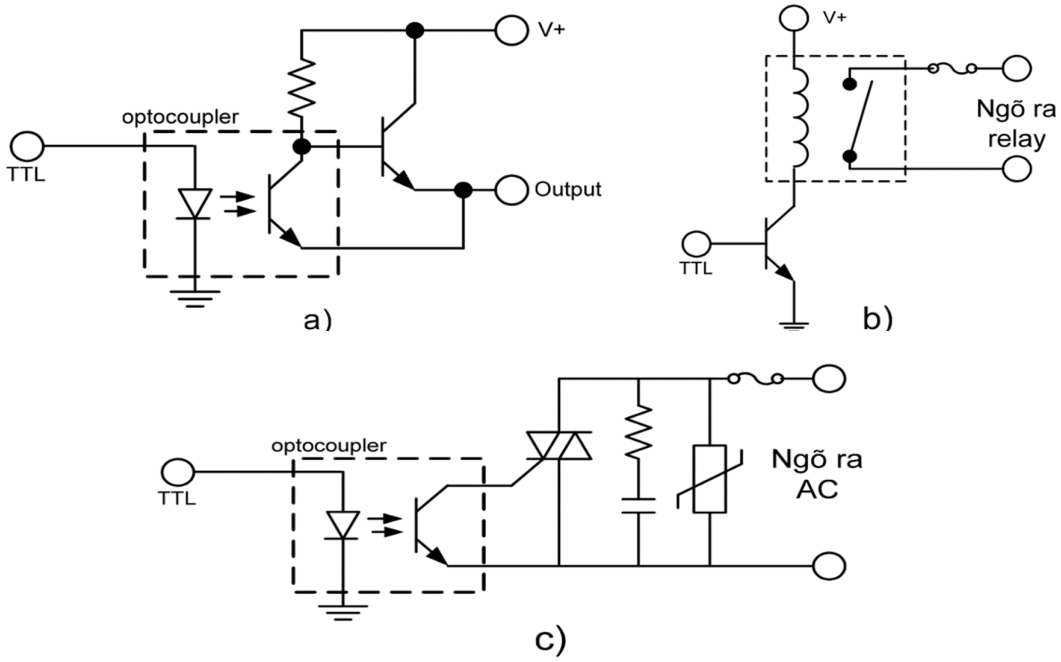
Một chiều: 5...30 VDC đối với ngõ ra rơ le; 20.4 ... 28.8 VDC đối với ngõ ra transistor.

Các khối ra tiêu chuẩn của PLC thường có 8 đến 32 ngõ ra theo cùng loại và có dòng định mức khác nhau. Ngõ ra có thể là relay, transistor hoặc triac. Relay là ngõ ra linh hoạt nhất. Chúng có thể là ngõ ra AC và DC. Tuy nhiên đáp ứng của ngõ ra relay chậm, giá thành cao và bị hư hỏng sau vài triệu lần đóng cắt. Còn ngõ ra transistor thì chỉ sử dụng với nguồn cung cấp là DC và ngõ ra triac thì chỉ sử dụng được với nguồn AC. Tuy nhiên đáp ứng của các ngõ ra này nhanh hơn.

Sơ đồ mạch điện bên trong của các ngõ ra được cho như hình 7.

Cần chú ý khi thiết kế hệ thống có cả hai loại ngõ ra AC và DC. Nếu nguồn AC nối vào ngõ ra DC là transistor, thì chỉ có bán kỳ dương của chu kỳ

điện áp được sử dụng và do đó điện áp ra sẽ bị giảm. Nếu nguồn DC được nối với ngõ ra AC là triac thì khi có tín hiệu cho ngõ ra, nó sẽ luôn luôn có điện cho dù có điều khiển tắt bằng PLC.

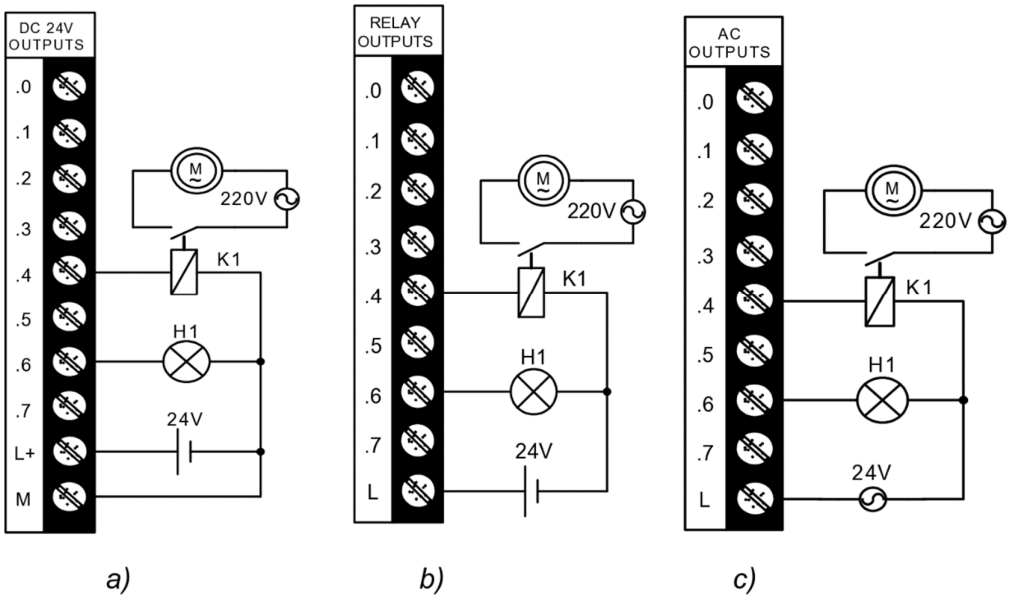


Hình 1.7: Mạch điện bên trong của các loại ngõ ra khác nhau.

a) Ngõ ra transistor ; b) Ngõ ra relay ; c) Ngõ ra triac

Cần lưu ý khi thiết kế hệ thống có cả hai loại ngõ ra AC và DC. Nếu nguồn AC nối ngõ vào, ngõ ra DC là transistor, thì chỉ có bán kỳ dương của chu kỳ điện áp được sử dụng và do đó điện áp ra sẽ bị giảm. Nếu nguồn DC được nối với ngõ ra là AC là triac thì khi có tín hiệu cho ngõ ra, nó sẽ luôn luôn có điện cho dù có điều khiển tắt bằng PLC.

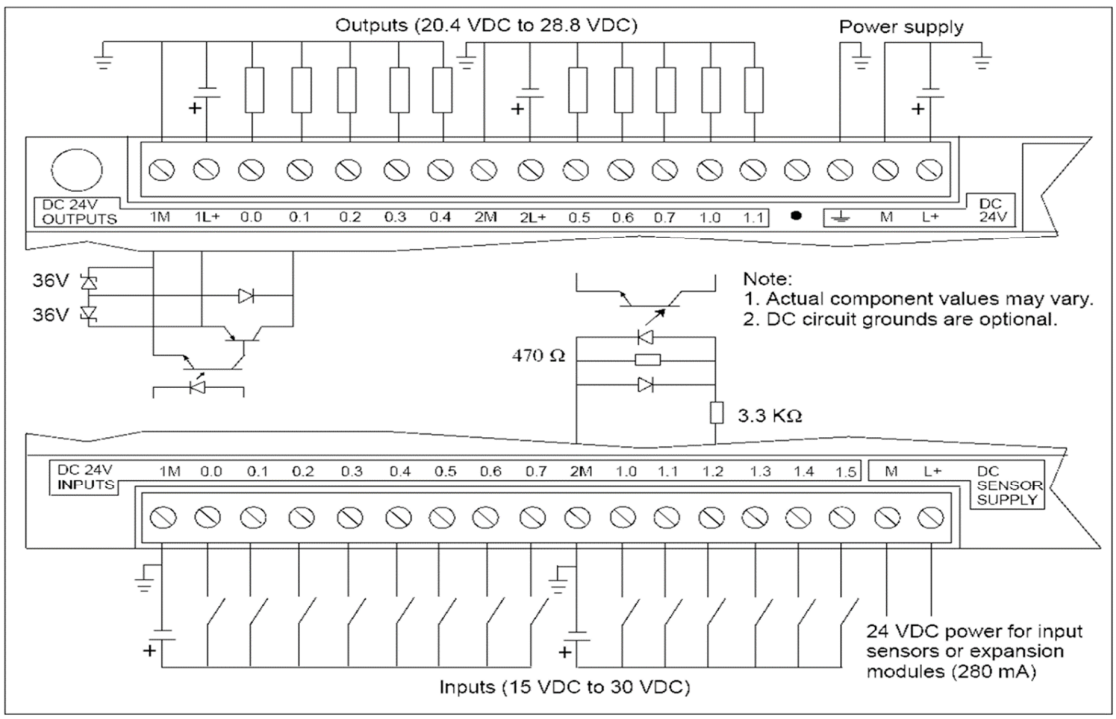
Đối với các ngõ ra số, khi kết nối với ngoại vi, ngoại trừ các trường hợp đặc biệt thì thông thường mỗi một ngõ ra được kết nối với một đối tượng điều khiển nhận tín hiệu nhị phân như: đèn báo, cuộn dây relay, chuông báo... Hình 8 minh họa cách kết nối dây các ngõ ra PLC với các cơ cấu chấp hành.



Hình 1.8: Kết nối dây ngõ ra PLC với cơ cấu chấp hành

Hình 8a là một ví dụ cho các khối ra sử dụng 24Vdc với mass chung. Tiêu biểu cho loại này là ngõ ra transistor. Trong ví dụ này các ngõ ra được kết nối với tải công suất nhỏ là đèn báo và cuộn dây relay. Quan sát mạch kết nối này, đèn báo sử dụng nguồn cung cấp là 24Vdc. Nếu ngõ ra 6 ở mức logic "1" (24Vdc) thì dòng sẽ chảy từ ngõ ra .6 qua đèn H1 và xuống Mass (M), đèn sáng. Nếu ngõ ra ở mức logic "0" (0V), thì đèn H1 tắt. Nếu ngõ ra 4 ở mức logic "1" thì cuộn dây relay có điện, làm tiếp điểm của nó đóng lại cung cấp điện 220 VAC cho động cơ.

Hình 8 b là một ví dụ ngõ ra relay sử dụng nguồn cấp là 24 VDC, và hình 8 c là ví dụ ngõ ra triac sử dụng nguồn xoay chiều 24 VAC.



Hình 1.9: Cách kết nối ngõ vào/ra của CPU 214 DC/DC/DC với nguồn và ngoại vi.

Một chú ý quan trọng khi kết nối các ngõ ra cần tra cứu sổ tay khối ngõ ra hiện có để có được thông tin chính xác tránh được những sự cố đáng tiếc xảy ra. Hình 9 là ví dụ của CPU 214 với nguồn cung cấp DC, ngõ vào DC và ngõ ra DC được nối dây với ngoại vi (trích từ sổ tay S7-200 Programmable Controller System Manual). Ta nhận thấy mỗi một nhóm ngõ vào cũng như một nhóm ngõ ra và CPU được cung cấp nguồn riêng là 24 VDC. Ngoài ra trên khối CPU còn có nguồn phụ 24 VDC (đến 280 mA) có thể được sử dụng để cung cấp cho các cảm biến hoặc khối mở rộng.

4.2. Ví dụ kết nối ngõ vào/ra của PLC từ một sơ đồ điều khiển có tiếp điểm

Trong nhiều trường hợp, cần cải tạo một hệ thống điều khiển với relay và contactor thành hệ thống điều khiển với PLC. Một câu hỏi đặt ra là chúng ta cần giữ lại những phần nào trong hệ thống điều khiển, còn phần nào sẽ loại bỏ đi?

Để dễ dàng trong việc chuyển đổi, có thể áp dụng phương pháp sau để chuyển đổi từ một hệ thống điều khiển cũ sang điều khiển với PLC.

** Về phần cứng:*

- Xác định các bộ tạo tín hiệu (ví dụ: nút nhấn, công tắc, cảm biến…) cần thiết nhất trong hệ thống điều khiển, mỗi bộ tạo tín hiệu tùy theo loại tạo ra tín

hiệu nào nên được kết nối với một ngõ vào cảu PLC tương ứng, ví dụ nếu bộ tạo ra tín hiệu nhị phân được thì được kết nối với ngõ vào số, còn bộ tạo ra tín hiệu tương tự thì kết nối với ngõ vào tương tự (analog). Còn các bộ tạo tín hiệu còn lại nếu không cần thiết thì có thể bỏ đi và sẽ được thực hiện bằng chương trình PLC.

- Tương tự xác định các cơ cấu chấp hành (đối tượng điều khiển) cần thiết nhất, thông thường các đối tượng này đều là các đèn báo, contactor chính, van từ,.v.v… Tùy theo loại mà mỗi một đối tượng điều khiển có thể kết nối trực tiếp hoặc gián tiếp với các ngõ ra tương ứng, mỗi một đối tượng điều khiển cần một ngõ ra. Nếu các đối tượng điều khiển cần dòng điều khiển lớn thì yêu cầu phải sử dụng relay trung gian. Ví dụ như các contactor chính điều khiển các động cơ công suất lớn thì ngõ ra của PLC sẽ được nối với một relay trung gian và thông qua tiếp điểm của relay trung gian để điều khiển các contactor này. Còn các đối tượng điều khiển không tác động trực tiếp đến quá trình điều khiển mà chỉ đóng vai trò trung gian hỗ trợ cho quá trình điều khiển như relay trung gian thì có thể loại bỏ và được thay thế bằng một ô nhớ nào đó trong chương trình của PLC.

- Sau khi đã xác định được số lượng các ngõ vào/ra với các ngoại vi tương ứng và chú ý ghi chú lại càng chi tiết càng tốt.

- Thực hiện việc nối dây các ngõ vào, ngõ ra của PLC với các bộ tạo tín hiệu điều khiển và đối tượng điều khiển. Trong quá trình nối dây cần lưu ý đến các nguyên tắc an toàn trong hệ thống điều khiển.

- Tất cả việc kết nối dây trong hệ thống điều khiển trước đây sẽ được biến đổi thành chương trình trong PLC.

** Về phần mềm:*

Việc viết chương trình có thể thực hiện theo hai cách:

Cách 1: Tùy theo yêu cầu công nghệ mà có thể thiết lập thuật giải điều khiển và viết chương trình theo thuật giải điều khiển này.

Cách 2: Vẫn duy trì hoạt động của hệ thống như cũ, hay nói khác đi là không cần thiết phải lập lại thuật giải điều khiển vì tất cả đã được thiết kế trong sơ đồ điều khiển cứng trước đây mà chỉ cần biến đổi sơ đồ điều khiển này thành chương trình trong PLC. Cách này tương đối dễ dàng và có thể không bị lỗi khi lập trình.

Trong phần này, trình bày phương pháp chuyển đổi theo 2 cách theo các bước như sau:

- Thực hiện viết chương trình lần lượt cho mỗi đối tượng điều khiển, mỗi đối tượng điều khiển được viết ở một đoạn chương trình và có ghi chú cụ thể để dễ dàng sửa lỗi.

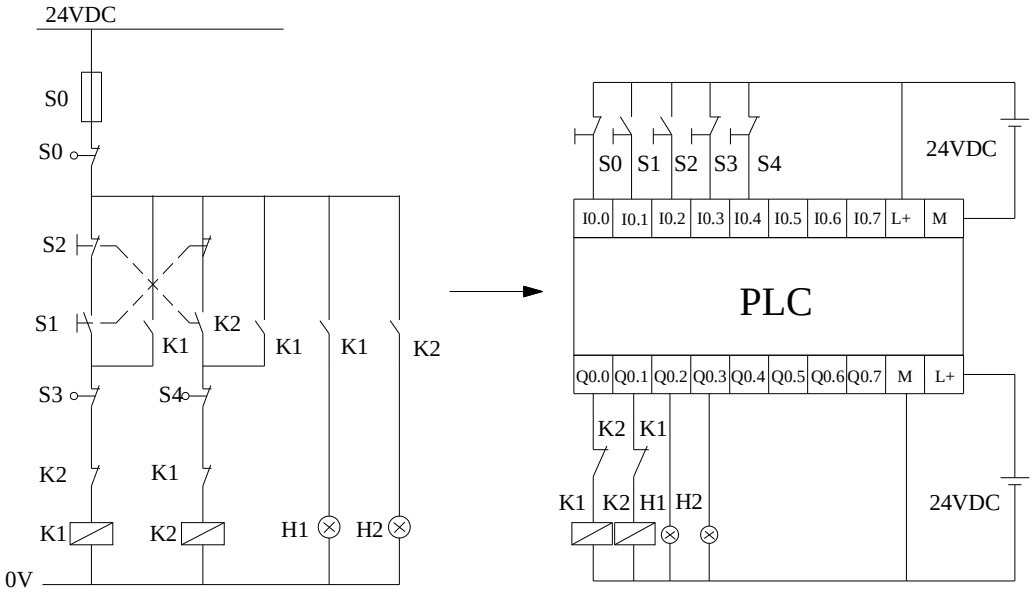
- Chỉ có các điều kiện cần thiết nhất cho đối tượng điều khiển mới được viết vào đoạn chương trình điều khiển nó.

- Nếu một số đối tượng điều khiển có cùng chung một nhóm điều kiện, thì nhóm điều kiện này nên được viết riêng ở một đoạn chương trình và cất kết quả vào một ô nhớ trong PLC. Nếu dối tượng điều khiển nào cần nhóm điều kiện này thì chỉ cần lấy kết quả được chứa trong ô nhứ. Điều này giúp cho cấu trúc chương trình mạch lạc và việc đọc chương trình trở nên dễ dàng hơn.

- Các đối tượng điều khiển không cần thiết (ví dụ contactor trung gian) sẽ được thay thế bằng một ô nhớ trong PLC. Nếu các đối tượng điều khiển nào cần đến tiếp điểm của relay trung gian thì chỉ cần thay thế bằng tiếp điểm của ô nhớ.

- Tùy theo hệ thống điều khiển có phức tạp hay không mà có thể phân chia thành nhiều khối chương trình để dễ dàng trong quá trình quản lý.

Hình 10 là một ví dụ về việ chuyển đổi một sơ đồ điều khiển cửa ra vào cơ quan bằng contactor thành hệ thống điều khiển bằng PLC.



Hình 1.10: Kết nối ngõ vào/ta của PLC trong sơ đồ điều khiển có tiếp điểm.

Dựa vào các bước trên, ta nhận thấy các nút ấn, contactor cần thiết được giữ lại như trong bảng xác định kết nối vào/ra với ngoại vi và PLC được chọn ở đây là loại CPU 214 DC/DC/relay. Do contactor K1 và K2 không được phép có điện đồng thời nên theo quan điểm an toàn cần phải khóa chéo hai contactor này lại với nhau.

Bảng 2: Bảng xác định kết nối vào/ra với ngoại vi.

Ký hiệu	Địa chỉ	Chú thích
S0	I0.0	Nút nhấn dừng, thường đóng
S1	I0.1	Nút nhấn mở cửa, thường hở
S2	I0.2	Nút nhấn đóng cửa, thường hở
S3	I0.3	Công tắc hành trình giới hạn cửa mở, thường đóng
S4	I0.4	Công tắc hành trình giới hạn cửa đóng, thường đóng
K1	Q0.0	Cuộn dây contactor K1, điều khiển mở cửa
K2	Q0.1	Cuộn dây contactor K2, điều khiển đóng cửa
H1	Q0.2	Đèn báo cửa đang mở
H2	Q0.3	Đèn báo cửa đang đóng

5. Kiểm tra việc kết nôi dây bằng phần mềm

Mục tiêu: Trình bày cách sử dụng phần mềm để kiểm tra việc kết nối trong chương trình điều khiển.

Một công việc quan trọng cho người lắp đặt và vận hành là biết được các kết nối của các ngõ vào/ra với ngoại vi có đúng hay không trước khi nạp chương trình điều khiển vào PLC. Hoặc khi một hệ thống đang hoạt động bình thường nhưng một sự cố hư hỏng xảy ra thì các phần ngoại vi nào bị hư và phát hiện nó bằng cách nào. Các phần mềm cho các bộ điều khiển bằng PLC thường có trang bị thêm công cụ để kiểm tra việc kết nối dây ngõ vào/ra với ngoại vi. Trong phần mềm Step 7 Micro/Win (phần mềm lập trình cho họ S7- 200 có trang bị thêm phần này đó là mục Status Chart.

5.1. Statuschart

Chúng ta có thể sử dụng status chart để đọc, ghi hoặc cưỡng bức các biến trong chương trình.

Chúng ta có thể sử dụng Status Chart để đọc, ghi hoặc cưỡng bức các biến trong chương trình theo mong muốn. Để có thể mở Status Chart, ta nhấp đúp chuột vào biểu tượng Status 

Chart trong cửa sổ Navigation Bar trên màn hình Step 7-Micro/Win32 hoặc vào mục View → Component → Status Chart.

5.2. Đọc và thay đổi biến với status chart

Để đọc hay ghi các biến trong status chart chúng ta thực hiện theo các bước sau:

Bước 1: Ở ô đầu tiên trong cột Address ta nhập vào địa chỉ hay tên ký hiệu của một biến trong chương trình ứng dụng mà muốn giám sát hoặc điều khiển, sau đó ấn ENTER. Lặp lại bước này cho tất cả các biến được thêm vào biểu đồ.

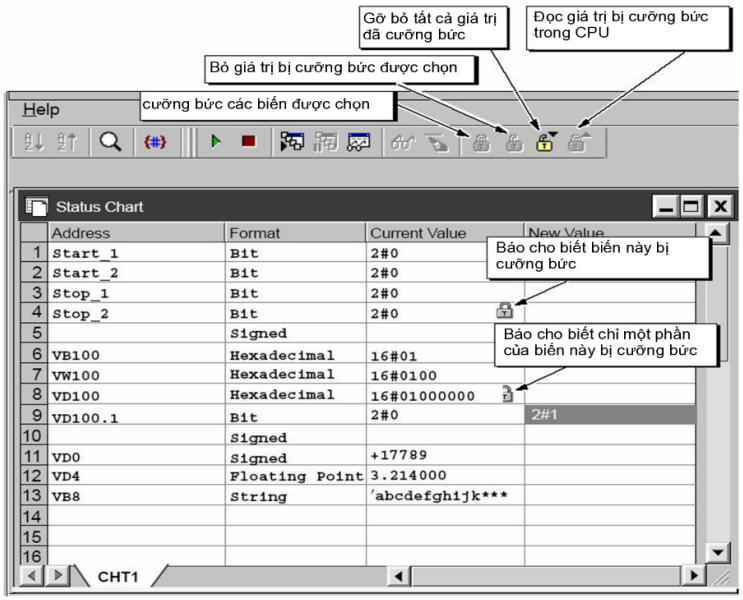
Bước 2: Nếu biến là 1 Bit (ví dụ:I, Q, hoặc M), thì kiểu biến đặt ở cột Format là bit. Nếu biến là một byte, word, hay double word thì chọn ở cột Format và nhấp đúp chuột để tìm kiểu biến mong muốn.

Bước 3: Để xem giá trị hiện hành của các biến trong PLC trong biểu đồ, hãy nhấp chuột vào biểu tượng  hoặc chọn Debug → Chart Status. Để chụp được một giá trị của các biến tại thời điểm nhấp chuột sử dụng Debug → Single Read hoặc nhấp chuột vào biểu tượng 

Bước 4: Để dừng việc giám sát thì nhấp chuột vào biểu tượng  hoặc chọn Debug → Chart Status.

Bước 5: Để thay đổi giá trị của một biến hoặc nhiều biến, hãy nhập giá trị mới vào cột "New Value" cho các biến mong muốn và nhấp chuột vào biểu tượng  hoặc chọn Debug → Write All để ghi tất cả giá trị này vào các biến tương ứng trong CPU.

Ví dụ về Status Chart được thể hiện trong hình 11.



Hình 1.11: Ví dụ về status chart..

Trong một số trường hợp cần phải ép buộc một ngõ vào hoặc bất kỳ một biến nào trong đó trong chương trình theo một giá trị mong muốn cho phù hợp với hoàn cảnh hoạt động hiện tại của hệ thống hoặc để kiểm tra các lỗi xảy ra trong hệ thống điều khiển, ta có thể sử dụng công cụ cưỡng bức biến (Force).

Để cưỡng bức biến trong Status Chart với một giá trị xác định, thực hiện các bước sau:

Bước 1: Chọn một ô trong cột Address, vào địa chỉ hay hay tên của biến cần cưỡng bức.

Bước 2: Nếu biến là 1 Bit (ví dụ:I0.0, Q0.1), thì kiểu biến ở cột Format luôn luôn là bit. Nếu biến là một byte, word, hay double word thì chọn ở cột Format và nhấp đúp chuột để tìm kiểu biến mong muốn.

Bước 3: Để cưỡng bức biến với giá trị hiện hành, trước tiên hãy đọc giá trị hiện hành trong PLC bằng cách nhấp chuột vào biểu tượng  hoặc chọn Debug → Chart Status.

Nhấp hoặc cuộn ô chứa giá trị hiện hành muốn cưỡng bức. Nhấp chuột vào biểu tượng  hoặc chọn Debug → Force ở trên vị trí giá trị hiện hành để cưỡng bức biến giá trị đó.

Bước 4: Để cưỡng bức một giá trị mới cho một biến, nhập giá trị vào cột "New Value" và nhấp chuột vào biểu tượng  hoặc chọn Debug → Force.

Bước 5: Để xem giá trị hiện hành của tất cả các biến bị cưỡng bức, kích chuột vào biểu tượng Read All Forced  hoặc chọn Debug → Read All Forced.

Bước 6: Để cho tất cả các biến trở lại trạng thái bình thường, hãy kích chuột vào biểu tượng Unforce All  hoặc chọn Debug → Unforce All.

Muốn gỡ bỏ cưỡng bức một biến, hãy chọn biến mong muốn và nhấp chuột vào biểu tượng  hoặc chọn Debug → Unforce.

6. Cài đặt và sử dụng với phần mềm step 7 Micro/win

Mục tiêu: Hướng dẫn cách cài đặt và sử dụng phần mềm chuyên dụng.

6.1. Những yêu cầu đối với máy tính PC:

- Tối thiểu phải có 6640 Kbyte RAM (với 500kB bộ nhớ còn trống).

- Màn hình 24 dòng, 80 cột ở chế độ văn bản.

- Còn 2Mbyte trống trong ổ đĩa cứng.

- Có hệ điều hành MS-DOS ver 5.0 hoặc cao hơn.

- Bộ chuyển đổi RS 232 –RS 485 phục vụ ghép nối truyền thông trực tiếp giữa PC và PLC

Truyền thông giữa Step 7 – Micro/win với CPU s7-200 qua cổng truyền thông ở phía đáy của PLC. Sử dụng cáp có bộ chuyển đổi RS232-RS485, được gọi là cáp PC/PPI, để nối với máy tính tạo thành mạch truyền thông trực tiếp.

Cắm một đầu của cáp PC/PPI với cổng truyền thông 9 chân của PLC, còn đầu kia nối với cổng truyền thông nối tiếp RS232 của máy PC. Nếu máy PC có cổng

truyền thông nối tiếp RS232 với 25 chân , thì phải qua bộ chuyển đổi chân.

6.2. Cài đặt phần mềm lập trình SEP 7-Micro/win 32.

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.

Bài 4
CÁC PHÉP TOÁN NHỊ PHÂN CỦA PLC

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

Mục tiêu: Trình bày các lệnh logic cơ bản trong PLC.

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. Một vòng như vậy được gọi là một vòng quét (scan). Một vòng quét 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.

Để tạo ra một chương trình dạng STL, người lập trình phải hiểu rõ phương thức sử dụng 9 bit ngăn xếp logic của S7-200. Ngăn xếp logic là một khối gồm 9 bit chồng lên nhau. Tất cả các thuật toán liên quan đến ngăn xếp đều chỉ làm việc với bit đầu tiên hoặc bit đầu và bit thứ hai của ngăn xếp. Khi phối hợp hai bit đầu tiên của ngăn xếp thì ngăn xếp sẽ được kéo lên 1 bit. Ngăn xếp và tên từng bít được ký hiệu như hình 1.

S0	Stack 0 - Bit thứ 1 của ngăn xếp
S1	Stack 1 - Bit thứ 2 của ngăn xếp
S2	Stack 2 - Bit thứ 3 của ngăn xếp
S3	Stack 3 - Bit thứ 4 của ngăn xếp
S4	Stack 4 - Bit thứ 5 của ngăn xếp
S5	Stack 5 - Bit thứ 6 của ngăn xếp
S6	Stack 6 - Bit thứ 7 của ngăn xếp
S7	Stack 7 - Bit thứ 8 của ngăn xếp
S8	Stack 8 - Bit thứ 9 của ngăn xếp

Hình 2.1: Ngăn xếp trong S7-200.

Đối với từng loại CPU thì khả năng quản lý không gian nhớ cũng khác nhau do vậy trước khi lập trình cần nắm vững giới hạn của các toán hạng để sử dụng cho đúng. Bảng sau trình bầy giới hạn toán hạng của CPU 212 và CPU214.

Bảng 1: Giới hạn toán hạng của CPU 212 và CPU 214.

Phương pháp truy nhập	Giới hạn cho phép của toán hạng	
	CPU 212	CPU 214
Truy nhập bit (địa chỉ byte.chỉ số bit)	V (0.0 – 1023.7) I (0.0 – 7.7) Q (0.0 – 7.7) M (0.0 – 15.7) SM (0.0 – 45.7) T (0 – 63) C (0 – 63)	V (0.0 – 4095.7) I (0.0 – 7.7) Q (0.0 – 7.7) M (0.0 – 31.7) SM (0.0 – 85.7) T (0 – 127) C (0 – 127)
Truy nhập byte	VB (0 – 1023) IB (0 – 7) QB (0 – 7) MB (0 – 15) SMB (0 – 45) AC (0 – 3) Hằng số	VB (0 – 4095) IB (0 – 7) QB (0 – 7) MB (0 – 31) SMB (0 – 85) AC (0 – 3) Hằng số
Truy nhập từ đơn (địa chỉ byte cao)	VW (0 – 1022) T (0 – 63) C (0 – 63) IW (0 – 6) QW (0 – 6)	VW (0 – 4095) T (0 – 127) C (0 – 127) IW (0 – 6) QW (0 – 6)

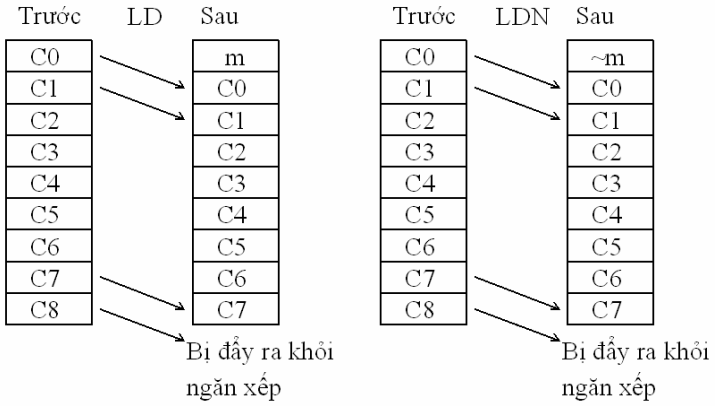
	SMW (0 – 44) AC (0 – 3) AIW (0 – 30) AQW (0 – 30) Hằng số	SMW (0 – 84) AC (0 – 3) AIW (0 – 30) AQW (0 – 30) Hằng số
Truy nhập từ kép (địa chỉ byte cao)	VD (0 – 1020) ID (0 – 4) QD (0 – 4) MD (0 – 12) SMD (0 – 42) AC (0 – 3) HC (0) Hằng số	VD (0 – 4092) ID (0 – 4) QD (0 – 4) MD (0 – 28) SMD (0 – 82) AC (0 – 3) HC (0 – 2) Hằng số

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

- *Các lệnh thay đổi ngăn xếp*

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 2.2: 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.

- *Các lệnh trong đại số Boolean*

Trong LAD các lệnh này được biểu diễn qua cấu trúc mạch, mắc nối tiếp hay song song các tiếp điểm thường đóng và các tiếp điểm thường mở. STL có thể sử dụng các lệnh A (And) hay o (Or) cho các hàm hở hoặc lệnh AN (And Not), ON (Or Not) cho các hàm kín. Giá trị ngăn xếp thay đổi phụ thuộc vào từng lệnh (*Bảng 2*).

Ngoài những lệnh làm việc trực tiếp với tiếp điểm, s7-200 còn 5 lệnh đặc biệt biểu diễn các phép tính của đại số Boolean cho các bit trong ngăn xếp, được gọi là các lệnh *Stack Logic*. Đó là các lệnh ALD (And load), OLD (Or Load), LPS (Logic push), LRD (Logic read) và LPP (logic Pop). Lệnh *Stack Logic* được dùng để tổ hợp, sao chụp hoặc xoá các mệnh đề logic. LAD không có bộ đếm dành cho lệnh *Stack Logic*. STL sử dụng các lệnh *Stack Logic* để thực hiện phương trình tổng thể có nhiều biểu thức con (*Bảng 2.3).*

Bảng 2: Các lệnh đại số trong STL.

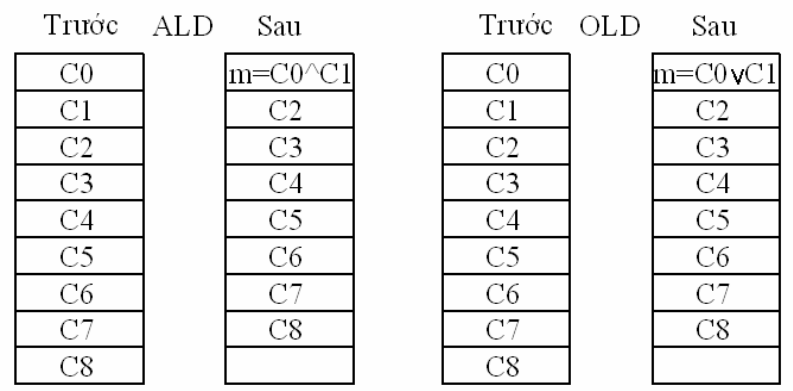
Lệnh	Mô tả lệnh	Toán hạng
O n A n	Lệnh thực hiện toán tử A và O giữa giá trị logic của tiếp điểm n và giá trị bit đầu tiên trong ngăn xếp. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp	n: I, Q, M, SM, T, C, V (bit)
ON n AN n	Lệnh thực hiện toán tử A và O giữa giá trị logic nghịch đảo của tiếp điểm n và giá trị bit đầu tiên trong ngăn xếp. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp	
OI n AI n	Lệnh thực hiện tức thời toán tử A và O giữa giá trị logic của tiếp điểm n và giá trị bit đầu tiên trong ngăn xếp. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp	n: I (bit)
ONI n ABI n	Lệnh thực hiện tức thời toán tử A và O giữa giá trị logic nghịch đảo của tiếp điểm n và giá trị bit đầu tiên trong ngăn xếp. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp	

Bảng 3: Một số lệnh thường gặp.

Lệnh	Mô tả lệnh	Toán hạng
ALD	Thực hiện phép A giữa bít thứ 1 và bít thứ 2 của ngăn xếp. Kết quả được ghi vào bít thứ 1. Giá trị còn lại của ngăn xếp được kéo lên 1 bít.	không có

OLD	Thực hiện phép O giữa bít thứ 1 và bít thứ 2 của ngăn xếp. Kết quả được ghi vào bít thứ 1. Giá trị còn lại của ngăn xếp được kéo lên 1 bít.	không có
LPS	Lệnh logic Push, sao chép bít đầu tiên của ngăn xếp	không có
LRD	Sao chép giá trị thứ hai của ngăn xếp lên giá trị thứ 1.	không có
LPP	Lệnh kéo ngăn xếp lên 1 bit. Giá trị bit sau được	không có

Trên hình 3 là kết quả khi thực hiện hai lệnh ALD và OLD.



Hình 2.3: Thực hiện lệnh ALD và OLD.

Ví dụ 1: Phân tích sự thay đổi nội dung ngăn xếp cho

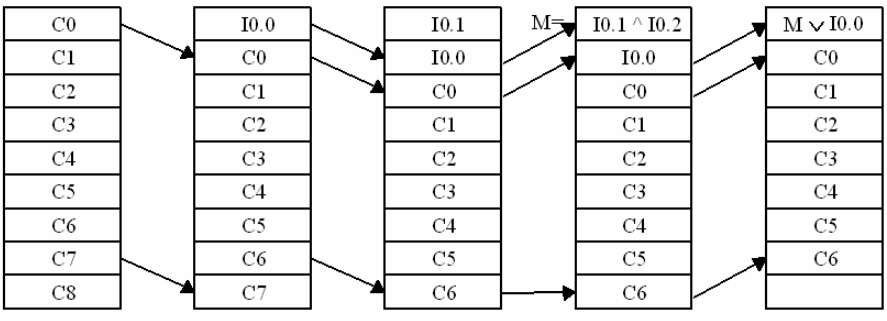
đoạn mã lệnh sau:

LD I0.0

LD I0.1

A I0.2

OLD = Q0.0



2.4: Hình Nội dung ngăn xếp khi thực hiện đoạn mã lệnh.

1.2. Các lệnh liên kết logic và các cổng logic cơ bản:

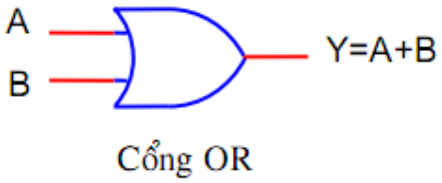
a, Phép toán OR và cổng OR

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:

$$X = A + B$$

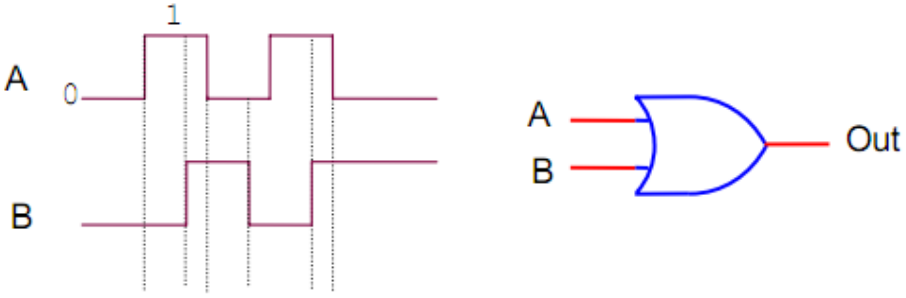
Trong biểu thức này, dấu "+" không có nghĩa là phép cộng thuần túy. Nó là phép toán OR, kết quả của phép toán OR được cho trong bảng sự thật sau:

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



Hình 2.5: 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 2.6: Giản đồ xung của phép toán .

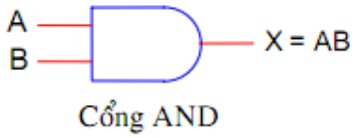
b, Phép toán AND và cổng AND

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 2.7: 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 2.8: 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.

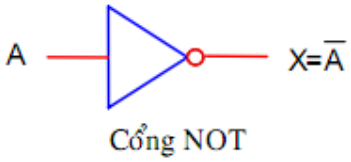
c, Phép toán NOT và cổng NOT

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

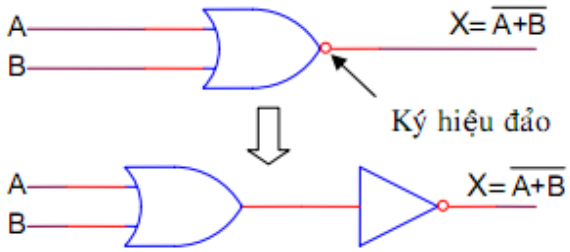


Hình 2.9: Bảng sự thật của phép toán NOT.

d, Phần thử NOR và cổng NOR

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

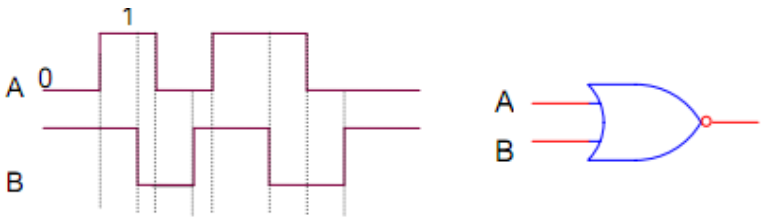
		OR	NOR
A	B	A+B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



Hình 2.10: 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.

Ví dụ 4: Xác định sóng ngõ ra của cổng NOR ứng với ngõ vào như sau:

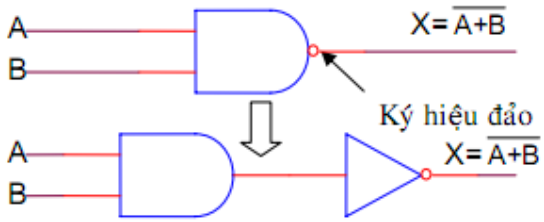


Hình 2.11: Giản đồ xung của phép toán NOR .

e, Phần tử NAND và cổng NAND

Cổng NAND tương ứng với cổng AND và NOT:

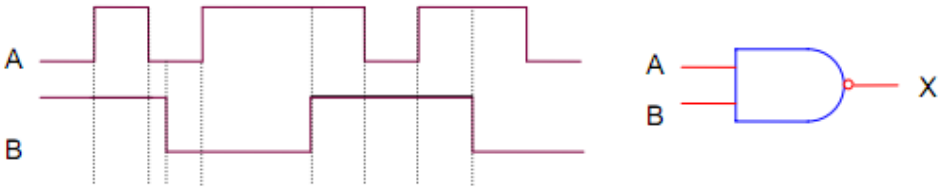
		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0



Hình 2.12: Bảng sự thật của phép toán NAND.

Ví dụ 5: Xác định sóng ngõ ra của phần tử NAND khi biến sóng ngõ vào như sau:

Hình 2.13: Giản đồ xung của phép toán NAND .

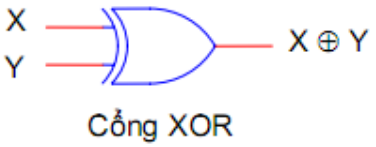


Đầu vào của phần tử NAND có 2 xung với xung cao tương ứng với "1" logic, xung ở mức thấp tương ứng với mức "0" logic. Dựa vào bảng chân lý ứng với phần tử NAND chúng ta có thể xác định được dạng sóng đầu ra.

f, Phép toán XOR và cổng XOR

Phép toán XOR có bảng sự thật như sau:

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0



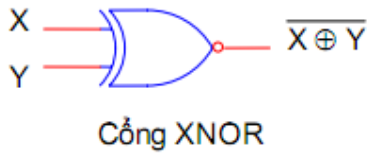
Hình 2.14: Bảng sự thật của phép toán XOR.

Biểu thức toán: $X \oplus Y = \overline{X}Y + \overline{Y}X$

g, Phép toán tương đương và cổng XNOR

Bảng sự thật:

X	Y	X ≡ Y
0	0	1
0	1	0
1	0	0
1	1	1



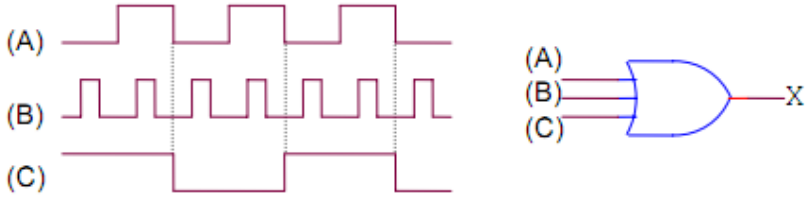
Hình 2.15: Bảng sự thật của phép toán XNOR.

Từ bảng sự thật thấy rằng: X≡Y=0 khi X≠Y, và X≡Y=0 khi X=Y

Biểu thức toán: $X \equiv Y = \overline{X \oplus Y} = XY + \overline{X}.\overline{Y}$

1.3. Bài tập ứng dụng

1, Vẽ lại sóng ngõ ra cho mạch hình sau:



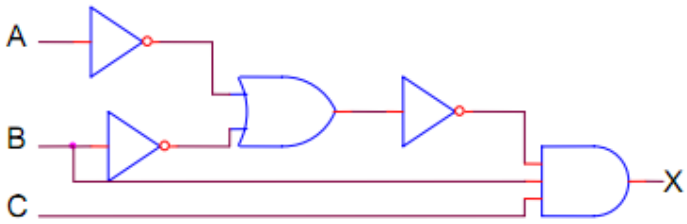
a, Giả sử ngõ vào A=0, vẽ dạng sóng ngõ ra.

b, Giả sử ngõ ra A=1, vẽ dạng sóng ngõ ra.

c, Thay cổng OR thành cổng AND rồi vẽ sóng ngõ ra

2, Có bao nhiêu tổ hợp ngõ vào của cổng OR 5 ngõ vào làm cho ngõ ra ở mức cao?

3, Viết biểu thức Boolean cho ngõ ra X. Xác định giá trị của X Ứng với



các điều kiện ngõ vào có thể.

2. Các lệnh ghi/xoá giá trị cho tiếp điểm

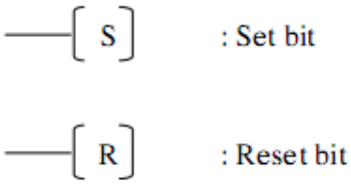
Mục tiêu: Trình bày lệnh ghi/xóa giá trị cho tiếp điểm.

2.1. 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.

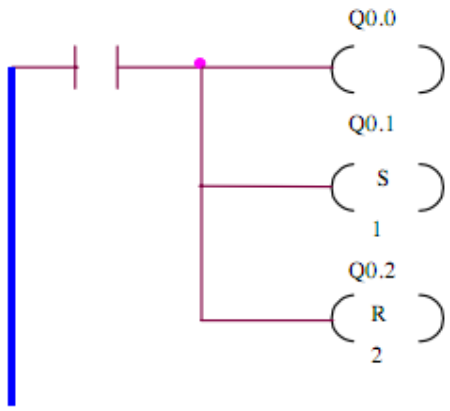
2.2. Lệnh Set (S) và Reset (R) trong PLC S7-200

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:

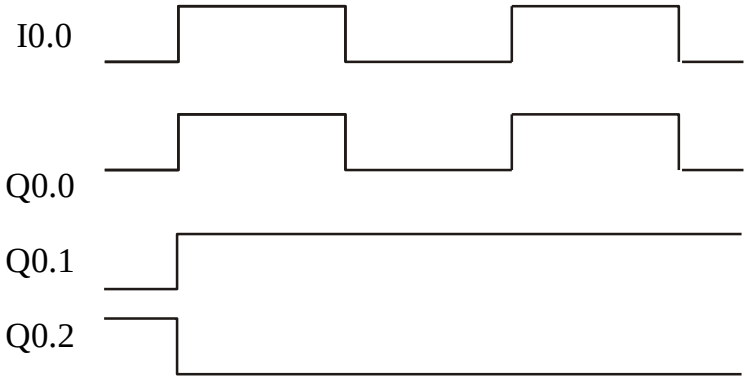


Hình 2.16: Lệnh SET và RESET trong S7-200.

Ví dụ 6: Sử dụng lệnh set và reset:



a)



b)

Hình 2.17: a) Đoạn lệnh sử dụng ngôn ngữ LAD.

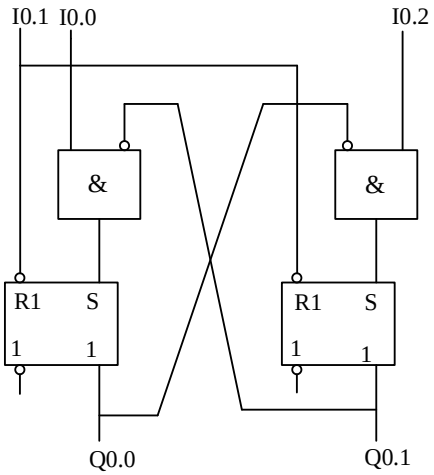
b) Giản đồ xung lối ra.

Trong chương trình I0.0 là đầu vào. Q0.0, Q0.1, Q0.2 là đầu ra.

2.3. Các ví dụ ứng dụng dùng bộ nhớ

a, Mạch chốt lẫn nhau của 2 van từ

Sơ đồ logic và bảng xác lập vào ra.

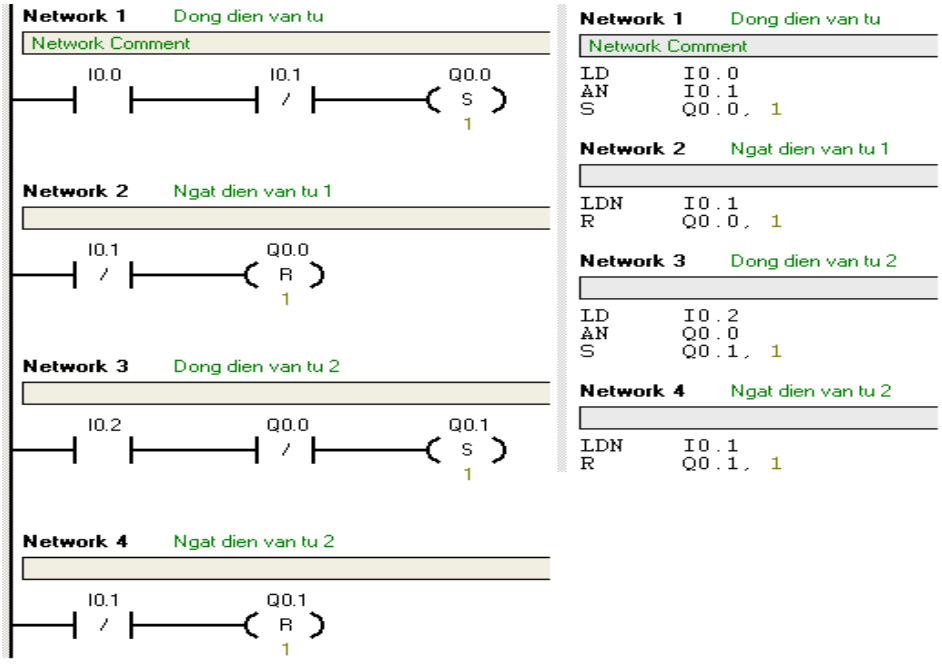


Xác lập vào/ra		
Kí hiệu	Địa chỉ	Chú thích
S1	I0.0	Nút ấn thường mở
S2	I0.1	Nút ấn thường đóng
S3	I0.2	Nút ấn thường mở
Y1	Q0.0	Van từ 1
Y2	Q0.1	Van từ 2

Mô tả hoạt động:

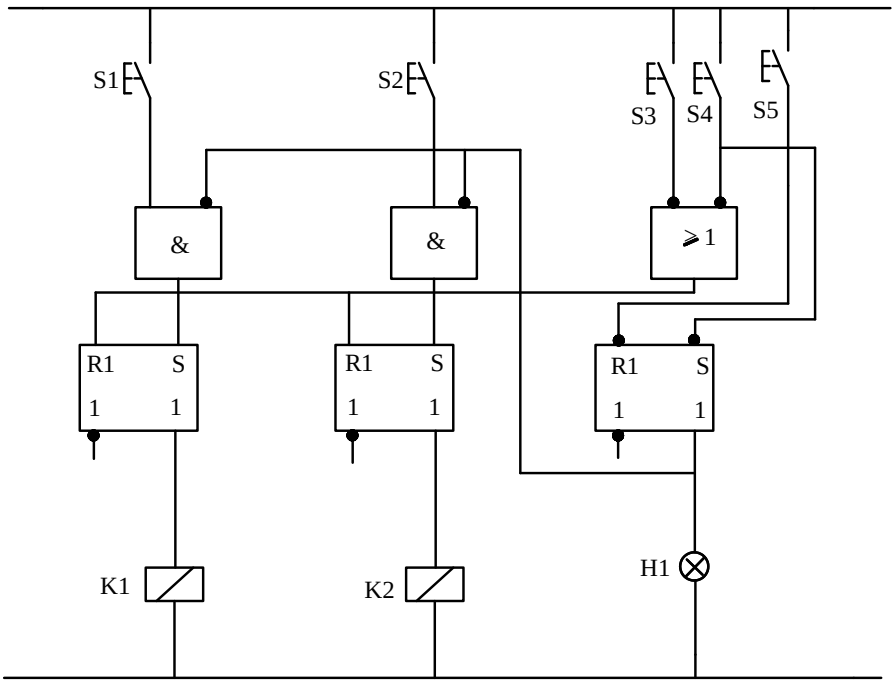
Qua việc khởi động S1 hoặc S3 các bộ phận nhớ một (van từ 1) hoặc bộ nhớ hai (van từ 2) sẽ được đặt. Nút ấn S2 làm nhiệm vụ cắt mạch.

Chương trình được viết ở LAD như sau:



b, Mạch tuần tự cưỡng bức có báo lỗi

Sơ đồ logic:



Mô tả hoạt động:

Qua việc khởi động ấn S1 thì K1 có điện. Khi ấn S2 thì K2 có điện. Khi ấn S3 cả K1 và K2 mất điện. Khi có lỗi thì cả K1 và K2 có thể bị ngắt điện bằng cách nhấn S4. Nút ấn S5 để phục hồi mạch, khi đo quá trình mới có thể được bắt đầu.

Nhiệm vụ

- Lập bảng xác định vào/ra
- Vẽ sơ đồ kết nối với PLC
- Vẽ sơ đồ STL và sơ đồ LAD
- Viết bảng câu lệnh mô tả mạch
- Viết và thử chương trình

c, Bộ chọn theo bước

Mô tả hoạt động

- Qua việc khởi động nút ấn S1, tín hiệu được thay đổi từ 0 đến 1và đèn H1 sáng, khi dời tay không ấn S1thì đèn tín hiệu thay đổi từ 1 đến 0, H2 sáng. Lập lại việc nhấn S1, tín hiệu thay đổi từ 0 đến 1 và đèn H3 sáng. Khi nhấn S2, tất cả các bộ nhớ được reset. Quá trình có thể được lặp lại từ đầu.

Nhiệm vụ

- Lập bảng xác định vào/ra
- Vẽ sơ đồ kết nối với PLC
- Vẽ sơ đồ STL và sơ đồ LAD
- Viết bảng câu lệnh mô tả mạch
- Viết và thử chương trình

3. Timer

Mục tiêu: Trình bày chức năng, cấu trúc, nguyên lý hoạt động của bộ định thời trong PLC.

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 xánh 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 4 mô tả các kiểu timer và độ phân giải ứng với CPU 212 và CPU 214.

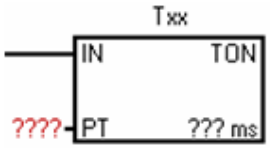
Bảng 4: Các loại Timer của CPU 212 và CPU 214.

Lệnh	**Độ phân giải**	**Giá trị cực đại**	**CPU 212**	**CPU 214**
TON	1ms	32,767s	T32	T32, T96
	10ms	327,67s	T33-T36	T33-T36, T97-T100
	100ms	3276,7s	T37-T63	T37-T63, T101-T127
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

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

3.1. On - delay Timer (TON)

- Cú pháp:

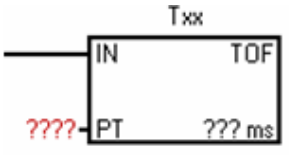


Hình 2.18: Khai báo sử dụng TON.

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 2.19: 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.

Bảng 5: Các bước khai báo sử dụng Timer.

STT	**Mô tả**
TON Txx n	Khai báo timer số hiệu xx kiểu TON để tạo thời gian trễ tính từ khi bit đầu trong ngăn xếp có giá trị logic 1. Nếu như giá trị đếm tức thời lớn hơn hoặc bằng giá trị đặt trước n thì T-bit có giá trị 1. Có thể reset timer kiểu TON bằng lệnh R hoặc bằng giá trị logic 0 tại đầu vào.
TONR Txx n	Khai báo timer số hiệu xx kiểu TONR để tạo thời gian trễ tính từ khi bit đầu tiên trong ngăn xếp có giá trị logic 1. Nếu như giá trị đếm tức thời lớn hơn hoặc bằng giá trị đặt trước n thì T-bit có giá trị logic bằng 1. Chỉ có thể reset timer kiểu TONR bằng lệnh R cho T-bit.

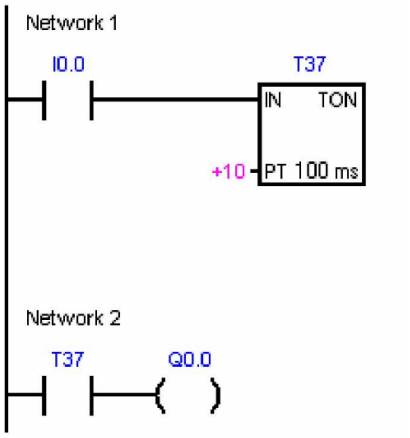
Khi sử dụng timer kiểu TONR, giá trị đếm tức thời được lưu lại và không bị thay đổi trong khoảng thời gian tín hiệu đầu vào logic 0. Giá trị của T-bit không được nhớ mà hoàn toàn phụ thuộc vào kết quả so sánh giữa giá trị đếm tức thời và giá trị đặt trước.

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.

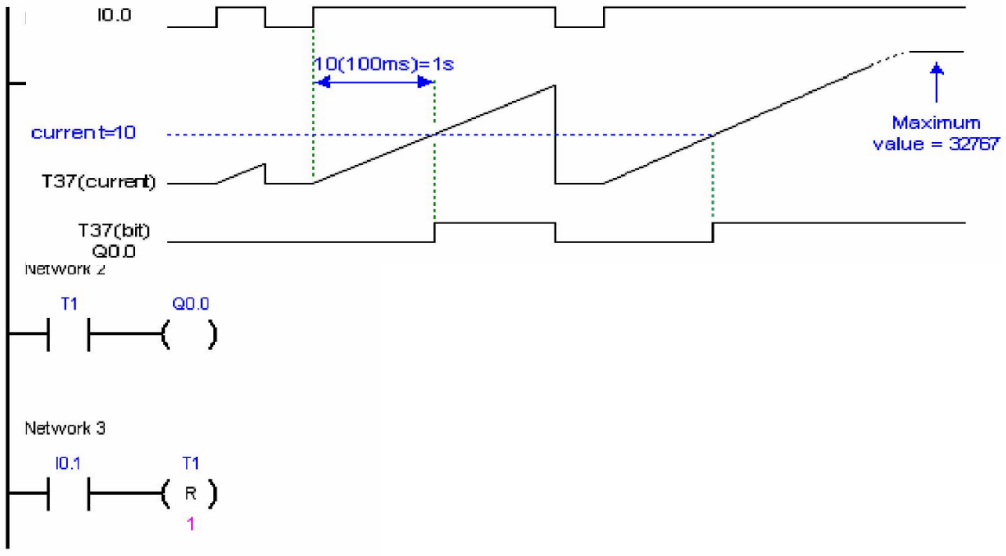
Khi timer được reset, T-word và T-bit của nó đồng thời được xoá và có giá trị 0. Đối với timer TON có hai phương pháp reset là xoá tín hiệu đầu vào hoặc dùng lệnh R. Còn đối với timer TONR chỉ có một phương pháp reset duy nhất là dùng lệnh R (R Txx K1).

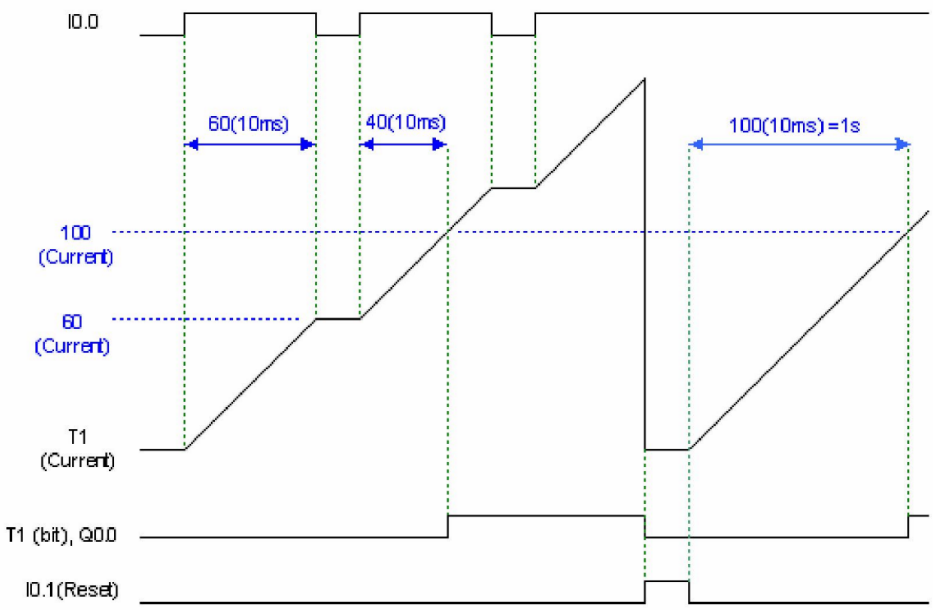
Ví dụ 7: Cách khai báo và sử dụng sử dụng timer kiểu TON.



Khi ngõ vào I0.0 =1 Timer 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.

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

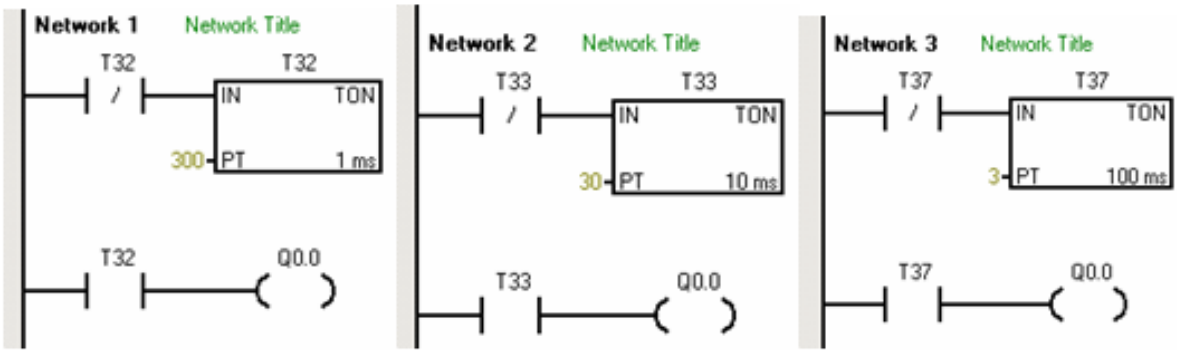
- **Hiệu quả của việc cập nhật giá trị đếm tức thời của timer 1ms, 10ms, 100ms.**

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.

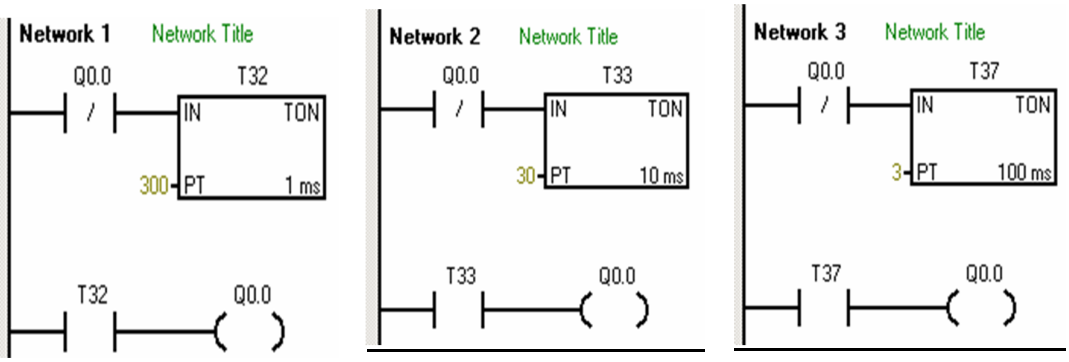
- Trong trường hợp sử dụng timer có độ phân giải 10ms, Q0.0 sẽ luôn có giá trị logic 0 vì khi bít T33 có giá trị logic 1 ở đầu vòng quét thì ngay sau đó sẽ bị chuyển về trạng thái 0.

- 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 2.20: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 2.21: Khắc phục ảnh hưởng của độ phân giải đến đầu ra của timer.

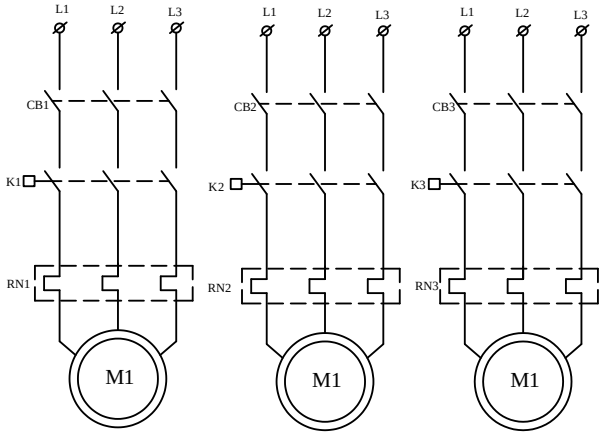
3.3. Bài tập ứng dụng timer

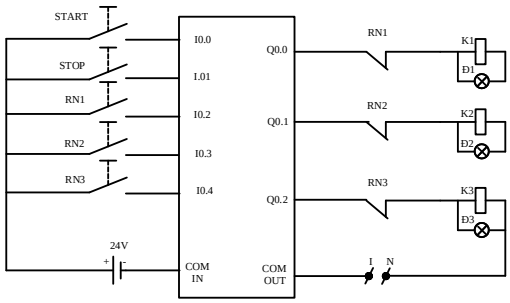
Khởi động hệ thống băng tải gồm 3 động cơ:

Khi khởi động START thì động cơ 1 chạy, sau 3s thì tự động động cơ 2 chạy, tiếp theo 3s kể từ động cơ chạy. Tương ứng mỗi động cơ chạy thì có đèn sáng.

Khi ấn STOP thì động cơ thứ 1 dừng trước, sau 5s thì tự động động cơ thứ 2 dừng và sau 5s thì tự động động cơ thứ 3 dừng hẳn.

Mạch động lực:

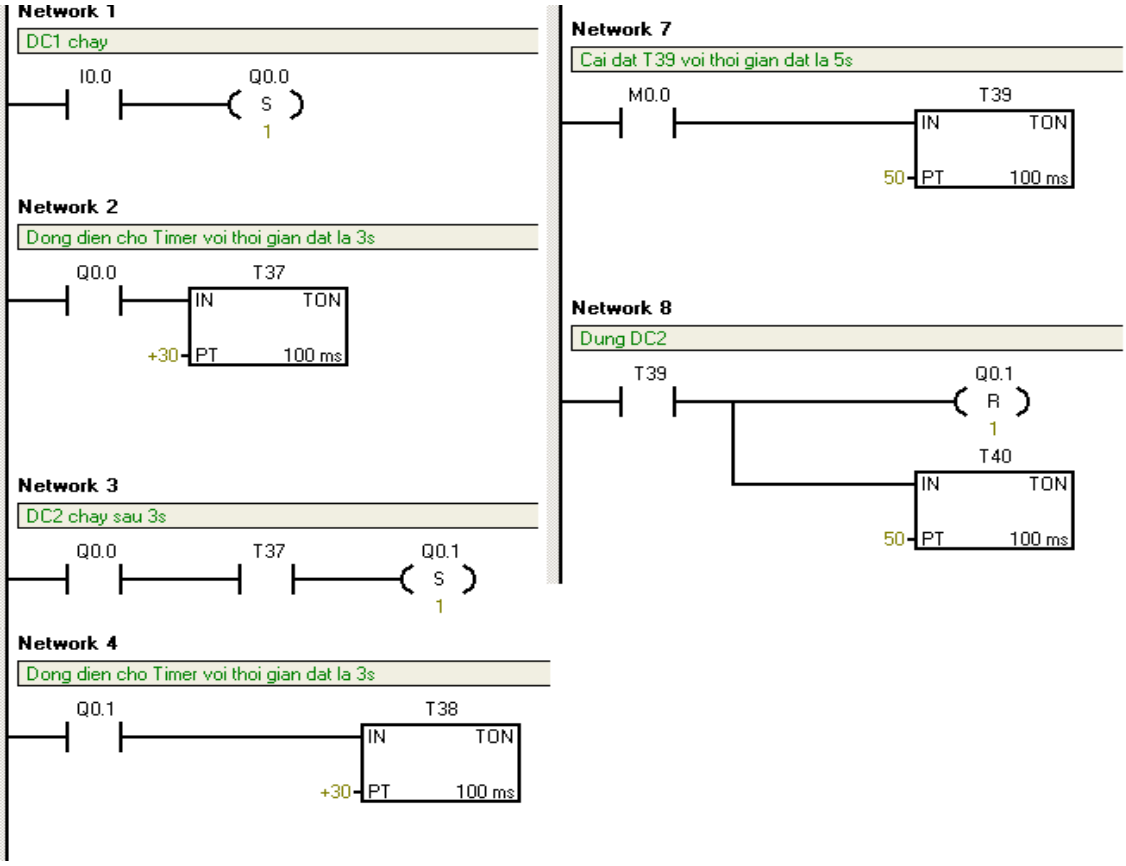




Bảng đầu vào/ra

Kí hiệu	Địa chỉ	Giải thích
START	I0.0	Nút ấn khởi động
STOP	I0.1	Nút ấn dừng
K1	Q0.0	Điều khiển động cơ 1
K2	Q0.1	Điều khiển động cơ 2
K3	Q0.2	Điều khiển động cơ 3

Chương trình viết bằng ngôn ngữ LAD



Chương trình viết bằng ngôn ngữ STL

Network 1

DC1 chay

```
LD      I0.0
S       Q0.0, 1
```

Network 2

Dong dien cho Timer voi thoi gian dat la 3s

```
LD      Q0.0
TON     T37, +30
```

Network 3

DC2 chay sau 3s

```
LD      Q0.0
A       T37
S       Q0.1, 1
```

Network 4

Dong dien cho Timer voi thoi gian dat la 3s

```
LD      Q0.1
TON     T38, +30
```

Network 5

DC 3 chay

```
LD      Q0.0
A       T38
=       Q0.2
```

Network 6

Dung DC1

```
LD      I0.1
R       Q0.0, 1
S       M0.0, 1
```

Network 7

Cai dat T39 voi thoi gian dat la 5s

```
LD      M0.0
TON     T39, 50
```

Network 8

Dung DC2

```
LD      T39
R       Q0.1, 1
TON     T40, 50
```

Network 9

Dung DC3

```
LD      T40
R       Q0.2, 1
R       M0.0, 1
```

Nguyên lý hoạt động của hệ thống:

Khi nhấn nút I0.0 (START) ở Network 1 Q0.0 có điện → công tắc tơ K1 có điện, đóng tiếp điểm trên mạch động lực → động cơ 1 (M1) kéo băng tải 1 chạy. Khi Q0.0 có điện sẽ đóng tiếp điểm thường mở Q0.0 ở network 2, sau khoảng thời gian là 3s, T1 có điện → Q0.1 có điện → công tắc tơ K2 có điện, đóng tiếp điểm K2 trên mạch động lực → động cơ 2 chạy và tiếp điểm thường mở của T1 ở network 3 đóng lại, sau khoảng thời gian 3s, T2 có điện → Q0.2 có điện → công tắc tơ K3 có điện, đóng tiếp điểm K3 trên mạch động lực → động 3 có điện kéo băng tải 3 chạy.

Khi ấn nút I0.1 ở network 1, Q0.0 mất điện, công tắc tơ K1 mất điện, mở tiếp điểm K1 trên mạch động lực → M1 dừng, đồng thời đóng điện cho T39. Sau 5s, ngắt điện động cơ 2 và đóng điện cho T40. Sau 5s dừng động cơ 3.

4. Counter

Mục tiêu: Trình bày chức năng, cấu trúc, nguyên lý hoạt động của bộ đếm trong PLC.

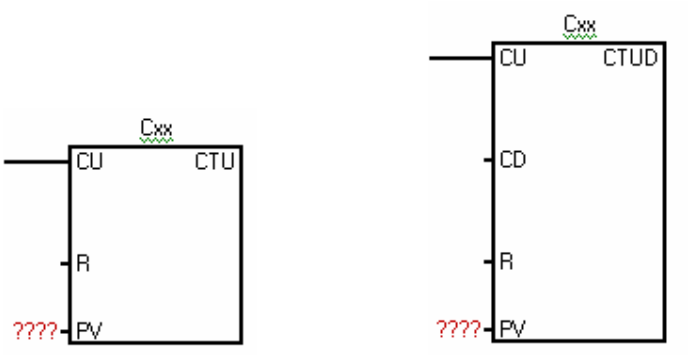
Counter là bộ đếm thực hiện chức năng đếm sườn xung trong S7-200. các bộ đếm của S7-200 được chia ra làm hai loại: bộ đếm tiến (CTU) và bộ đếm tiến/lùi (CTUD).

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 2.22: 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

CxxCPU 212: 0-47

(word) CPU214: 0-47, 80-127.

PVVW, 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 :CxxCPU 212: 48-63

(word) CPU 214: 48-79

PVVW, 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
```

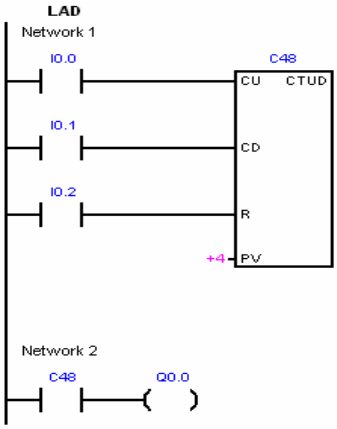
LD I0.2

CTUD C48 +4

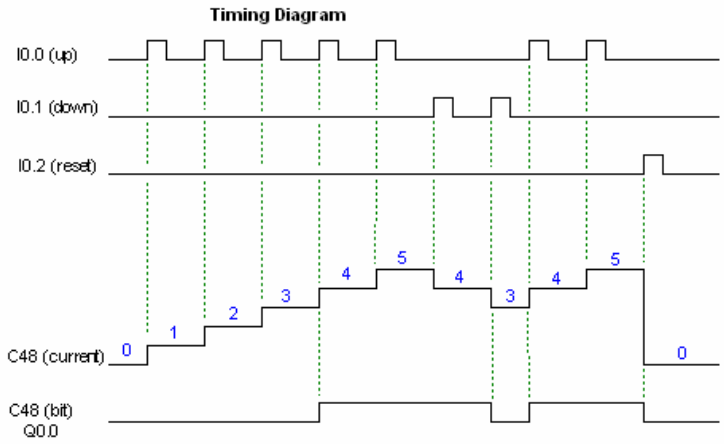
NETWORK 2 // Count Up/Down counter C48 turns on C48 bit when current value >= 4

LD C48

= Q0.0



- Viết bằng LAD
- Giản đồ thời gian



** Sử dụng bộ đếm tốc độ cao HSC*

Bộ đếm tốc độ cao được theo dõi và điều khiển các quá trình có tốc độ cao mà PLC không thể khống chế được do hạn chế của thời gian vòng quét. Trong CPU 212 có một bộ đếm tốc độ cao (HSC0) và CPU 214 có 3 bộ (HSC0,

HSC1, HSC2). Nguyên tắc hoạt động của bộ đếm tốc độ cao tương tự như các bộ đếm khác của PLC, đầu vào đếm theo sườn lên của tín hiệu đầu vào. Số đếm được được lưu vào trong một ô nhớ đặc biệt kiểu từ kép và được gọi là giá trị đếm tức thời của bộ đếm, ký hiệu CV (Current Value). Khi giá trị đếm tức thời bằng giá trị đặt trước thì bộ đếm phát ra một ngắt. Giá trị đặt trước là một số nguyên 32 bit cũng được lưu trong một ô nhớ kiểu từ kép và ký hiệu PV (Preset Value).

Nếu chế độ ngắt vào/ra với bộ đếm tốc độ cao được khai báo sử dụng, các tín hiệu sau đây sẽ được phát:

- Ngắt khi PV=CV (đối với HSC0, HSC1, HSC2)

- Ngắt khi có tín hiệu báo thay đổi hướng đếm từ cổng vào (với HSC1, HSC2).

- Ngắt khi có tín hiệu báo xoá (reset) từ cổng vào (với HSC1, HSC2).

Mỗi bộ đếm được chọn chế độ làm việc bằng lệnh HDEF. Từng chế độ làm việc lại có các kiểu hoạt động khác nhau. Kiểu hoạt động của mỗi bộ đếm được xác định bằng nội dung của một byte điều khiển trong vùng nhớ đặt biệt sau đó được khai báo với bộ đếm bằng lệnh HSC.

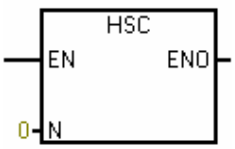
Bộ đếm tốc độ cao HSC0

Bộ đếm HSC0 có một cổng vào là I0.0. Nó chỉ chạy một chế độ làm việc duy nhất là đếm tiến hoặc đếm lùi theo sườn lên của I0.0. Chiều đếm được quy định bởi trang thái logic của bit SM37.3 như sau:

SM37.7 =0 Đếm lùi theo sườn lên của I0.0

=1 Đếm tiến theo sườn lên của I0.0

Tần số đếm của HSC0 là 2kHz. HSC0 sử dụng từ kép SMD38 để lưu giá trị đếm tức thời CV. Giá trị đặt trước PV được ghi vào từ kép SMD42. Cả giá trị PV và CV đều là số nguyên 32 bit có dấu.



Hình 2.23: Khai báo sử dụng bộ đếm HSC.

HSC0 sử dụng byte SMB37 để xác định kiểu hoạt động như: đếm tiến hay lùi; cho phép hay không cho phép thay đổi giá trị đếm tức thời CV, PV và cho

phép/không cho phép bộ đếm. Kiểu hoạt động của HSC0 phải được định nghĩa trong SMB37 trước khi thực hiện lệnh HDEF.

Cấu trúc SMB37, còn được gọi là byte điều khiển HSC0, như sau:

Bảng 5: Chức năng của byte điều khiển HSC0.

Bit	Chức năng
SM37.0	không sử dụng
SM37.1	không sử dụng
SM37.2	không sử dụng
SM37.3	chiều đếm: 0 - đếm tiến, 1 - đếm lùi
SM37.4	Cho phép sửa đổi giá trị đặt trước: 0 – không c phép, 1 – cho phép
SM37.5	Cho phép sửa đổi giá trị đặt trước: 0 – không cho phép, 1 – cho phép
SM37.6	Cho phép sửa giá trị đếm tức thời: 0 – không cho phép, 1 – cho phép
SM37.7	1 – cho phép kích HSC0, 0 – không cho phép kíchHSC0

Các bước khai báo sử dụng HSC0 (nên thực hiện tại vòng quét đầu tiên):

- Nạp giá trị điều khiển phù hợp cho SMB37

- Xác định chế độ làm việc cho bộ đếm bằng lệnh HDEF. Do HSC0 chỉ có một chế độ làm việc nên lệnh xác định kiểu sẽ là: HDEF K0 K0.

- Nạp giá trị đếm tức thời ban đầu và giá trị đặt trước và SMD38 và SMD42.

- Khai báo sử dụng chế độ ngắt vào ra và kích tín hiệu báo ngắt cho HSC0 bằng lệnh ATCH.

- Kích hoạt bộ đếm bằng lệnh HSC K0.

Sau khi được kích, bộ đếm HSC0 bắt đầu làm việc và sử dụng byte SMB36 để thông báo trạng thái hoạt động của nó như sau:

Khi sử dụng bộ đếm HSC0 cùng với chế độ ngắt vào/ra, tín hiệu báo ngắt HSC0 sẽ xuất hiện khi CV=PV nếu tín hiệu báo ngắt đã được khai báo.

Bảng 6: Chức năng của byte thông báo trạng thái.

Bit	Chức năng
SM36.0	không sử dụng
SM36.1	không sử dụng
SM36.2	không sử dụng
SM36.3	không sử dụng
SM36.4	không sử dụng
SM36.5	chiều đang đếm: 0 - đếm lùi, 1 - đếm tiến
SM36.6	So sánh kết quả tức thời: 0 - CV ≠ PV, 1 – CV = PV
SM36.7	So sánh kết quả tức thời: 0 – VC < PV, 1 – CV > PV

Thủ tục khai báo sử dụng bộ đếm tốc độ cao

Khai báo sử dụng các bộ đếm nên được thực hiện tại vòng quét đầu tiên khi bit SM0.1=1. Thủ tục khai báo sử dụng bộ đếm tốc độ cao bao gồm:

- Nạp giá trị về kiểu hoạt động phù hợp cho byte điều khiển. Ví dụ như khai báo kiểu hoạt động choHSC1 với:

  - o Tín hiệu xoá ngoài tích cực khi có logic 1 thì phải ghi 0 vào SM47.0
  - o Tín hiệu kích start tích cực khi có giá trị logic 1 thì thi 0 vào SM47.1
  - o Cho phép kích HSC1 thì ghi 1 vào SM47.7

- Xác định chế độ làm việc cho bộ đếm bằng lệnh HDEF. Ví dụ muốn xác định chế độ làm việc số 3 cho HSC1 thì thực hiện. (HSC0 chỉ có một chế độ đếm là mode0, còn HSC1 và HSC1 có 12 chế độ đếm từ mode0 → mode11)

HDEF K1 K3

- Nạp giá trị đếm tức thời ban đầu và giá trị đặt trước. Ví dụ nạp giá trị tức thời ban đầu là 0 và giá trị đặt trước là 3 cho HSC1

MOVD K0 SMD48

MOVD K3 SMD52

- Khai báo sử dụng chế độ ngắt vào ra và kích tín hiệu báo ngắt. Ví dụ như sử dụng HSC1 làm tín hiệu báo ngắt vào ra mã 13 (khi PV=CV) và mã tín hiệu 14 (*khi đổi chiều bộ đếm*) với các chương trình xử lý ngắt tương ứng có nhãn la 0

và 1 thì thực hiện trong STL như sau

ATCH K0 K13

ATCH K0 K14

- Kích bộ đếm với kiểu làm việc đã ghi trong byte điều khiển bằng lệnh HSC. Ví dụ kích bộ đếm HSC1 theo SMB47 bằng cách thực hiện lệnh trong STL

HSC K1

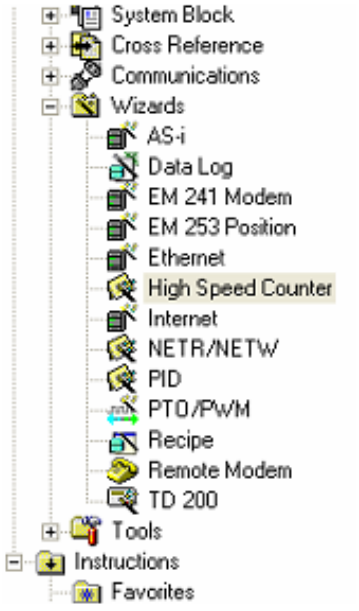
HDEF: lệnh xác định chế độ làm việc cho bộ đếm tốc độ cao.Tên bộ đếm được chỉ định bằng toán hạng HSC. Chế độ làm việc được chọn là nội dung của toán hạng trong MODE.

HSC: lệnh đặt kiểu làm việc cho bộ đếm tốc độ cao. Tên của bộ đếm được chỉ định bằng toán hạng N. Kiểu làm việc được đặt là nội dung của byte

điều khiển bộ đếm.

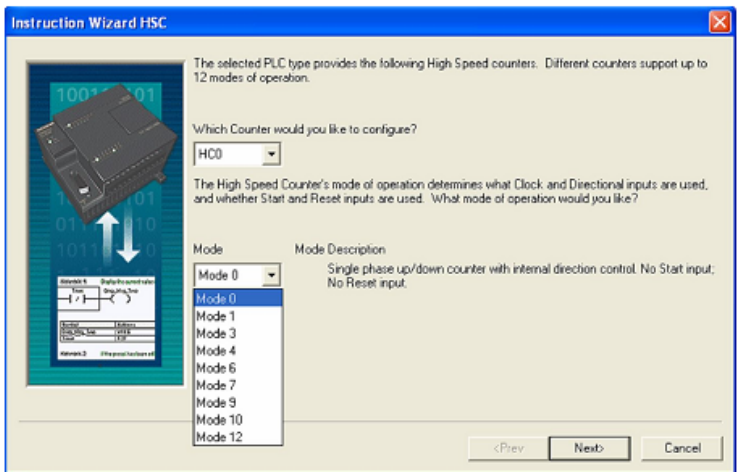
Để khai báo bộ đếm ta có thể dùng chương trình Winzard của PLC như sau:

- Vào Winzard chọn Hight Speed Counter



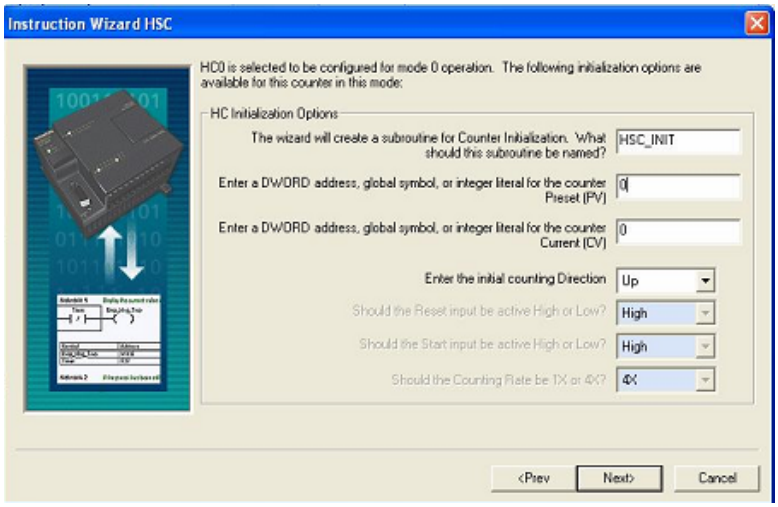
Hình 2.24: Lựa chọn bộ đếm tốc độc cao.

Chọn loại bộ đếm tốc độ cao và chế độ làm việc của bộ đếm



Hình 2.25. Chọn loại bộ đếm.

Chọn ngắt cho bộ đếm, đặt giá trị cho giá trị đặt trước PV và giá trị đếm tức thời VC.



Hình 2.26: Cài đặt tham số cho bộ đếm.

4.3. Bài tập ứng dụng bộ đếm

Chương trình điều khiển máy trộn.

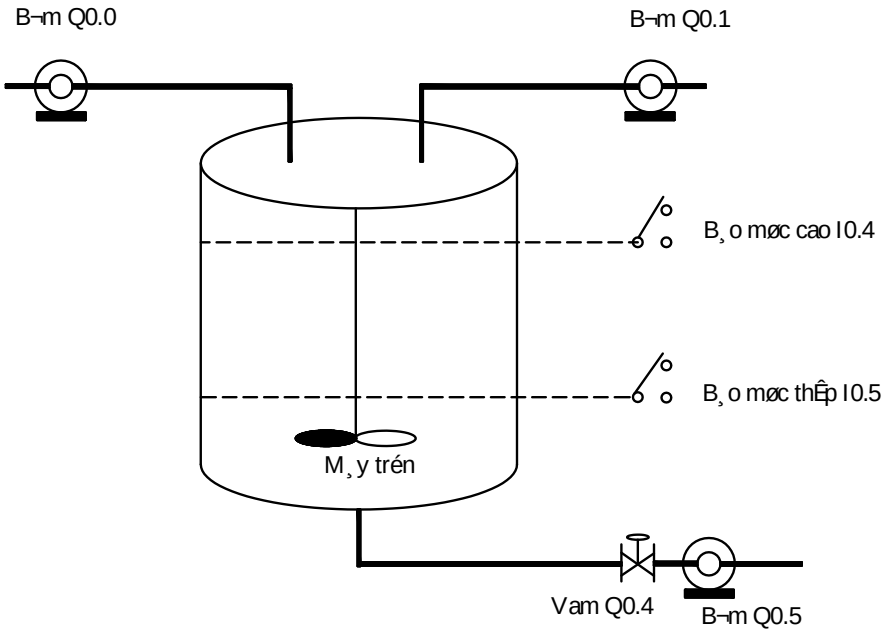
Trong hình dưới là sơ đồ bình trộn để tạo ra các màu khác nhau có 2 cảm biến:

Báo mức cao : I0.4

Báo mức thấp :I0.5

Đông cơ trộn điều khiển bởi Q0.2

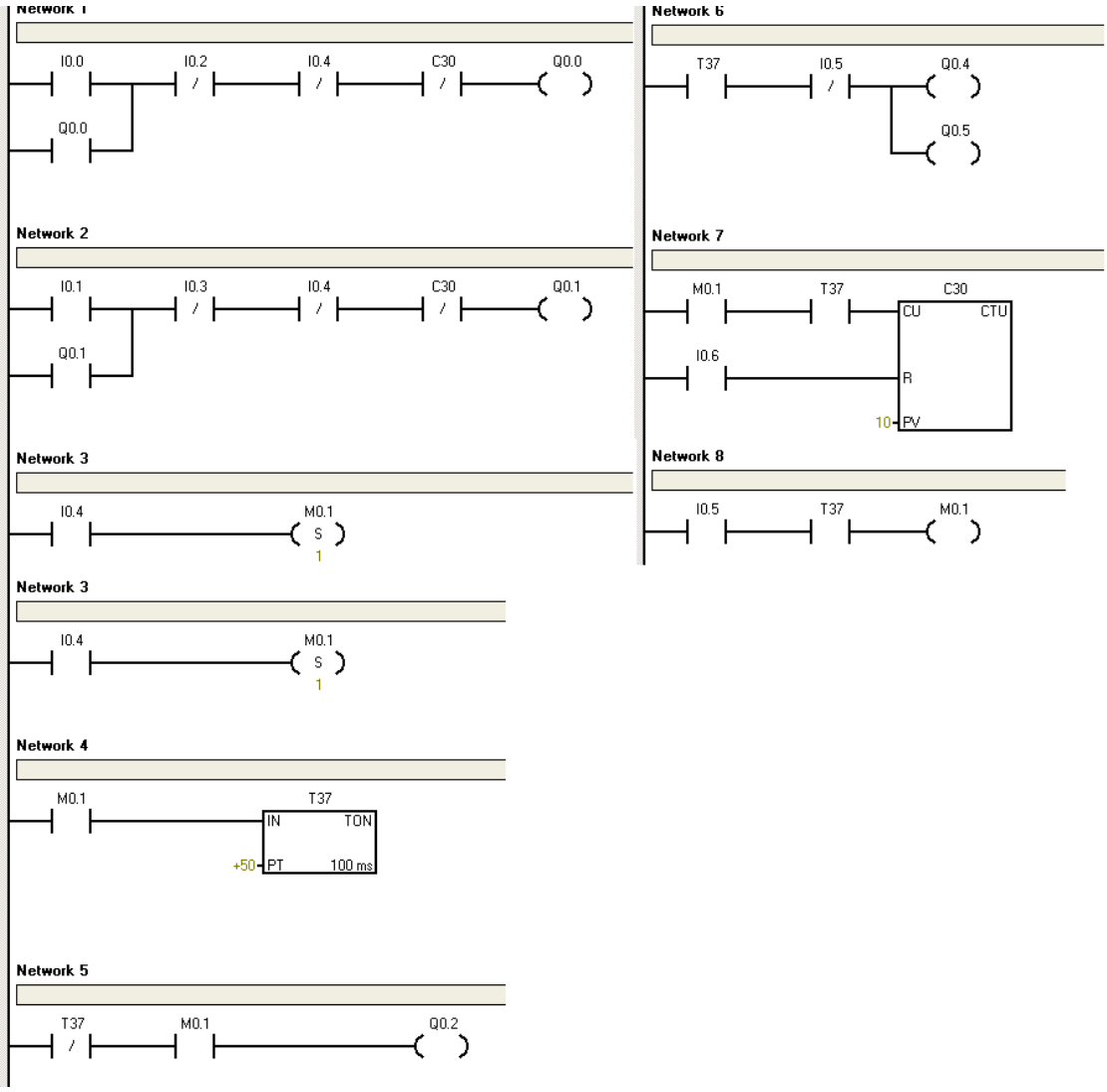
Quá trình được thực hiện như sau: trước tiên bơm 2 loại sơn khác nhau vào bình. Loại sơn thứ nhất được đưa vào bình máy bơm được điều khiển qua Q0.0. Loại sơn thứ hai được đưa vào bình nhờ máy bơm thứ 2 được điều khiển qua Q0.1.



Sau khi dung dịch trong bình đạt mức cực đại (I0.4=1) thì dừng hai máy bơm và băt đầu quá trình trộn, quá trình này được điều khiển bởi động cơ trộn (Q0.2) và thời gina trộn cần thiết là 5s. Sau khi trộn xong, sản phẩm được đưa ra để rót vào các hộp đựng sơn qua van (Q0.4) và máy bơm (Q0.5). Có thể tóm tắt quá trình trộn như sau:

- Bước 1: Rót loại sơn thứ nhất và loại sơn thứ hai vào bình.
- Bước 2: Điều hành quá trình làm việc khi đạt mức cao (I0.4=1).
- Bước 3: Điều khiển động cơ trộn và thời gian trộn.
- Bước 4: Đưa sản phẩm ra khỏi bình trộn.
- Bước 5: Đếm số lần trộn. Nếu đã đủ 10 lần thì dừng sản xuất.
- Bước 6: Quay lại chế độ làm việc ở bước 1.

Chương trình được viết trong PLC ở dạng LAD:



5. Bài tập ứng dụng:

Mục tiêu: Nêu một số bài toán thường gặp khi sử dụng PLC.

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)

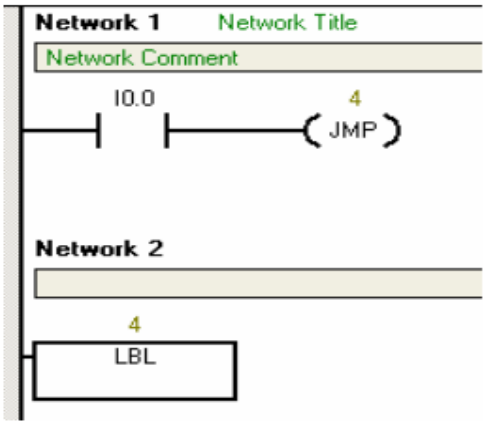
Trong quá trình hoạt động gặp sự cố ta có thể nhấn nút dừng khẩn cấp để dừng toàn bộ hệ thống.

3, Phát hiện chiều di chuyển của vật: Để phát hiện chiều di chuyển của vật, ta phải sử dụng 2 sensor1,2 kế tiếp nhau.

- Trường hợp vật di chuyển theo chiều thuận: sensor 1 tác động rồi đến sensor 2.
- Chiều ngược thì sensor tác động theo thứ tự ngược lại.

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

6. Lệnh nhảy và lệnh gọi chương trình con



Mục tiêu: Trình bày lệnh nhảy và gọi chương trình con.

Hình 2.27: Mô tả về lệnh nhảy và gọi chương trình con.

Khi I0.0 lên 1, chương trình sẽ thực hiện lệnh nhảy: nhảy tới nhãn tương ứng, khi đó đoạn chương trình ở giữa lệnh nhảy và nhãn dẽ được bỏ qua ở chu kì đó.

Kí hiệu của nhãn nhảy phải là một số nguyên n. Việc đặt nhãn cho lệnh nhảy phải nằm trong chương trình. Nhãn của chương trình con hoặc chương trình xử lý ngắt được khai báo ở đầu chương trình. Không thể dùng lệnh nhảy JMP để điều khiển chương trình chính vào một nhãn bất kỳ trong chương trình con hoặc trong chương trình xử lý ngắt.

Bài 5

CÁC PHÉP TOÁN SỐ CỦA PLC

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

Mục tiêu: Trình bày các lệnh truyền dẫn.

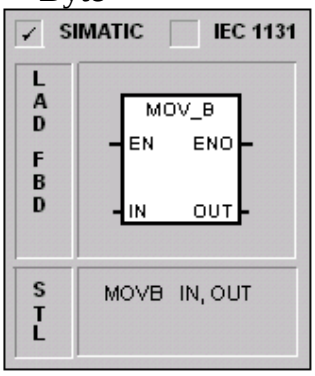
1.1. Truyền Byte, Word, Doubleword:

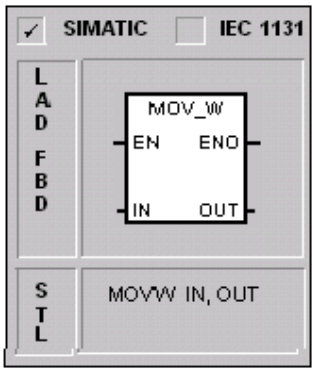
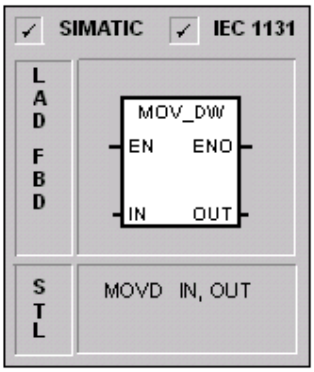
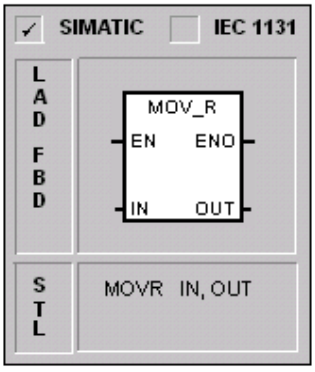
- Phép truyền Move Byte sẽ được thực hiện copy dữ liệu Byte tại ngõ vào IN và truyền tới Byte tại ngõ ra OUT.
- Phép truyền Move Word sẽ thực hiện copy dữ liệu Word tại ngõ vào In và truyền tới Word tại ngõ ra OUT
- Phép truyền Move DoubleWord sẽ thực hiện copy dữ liệu DoubleWord tại ngõ vào In và truyền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



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

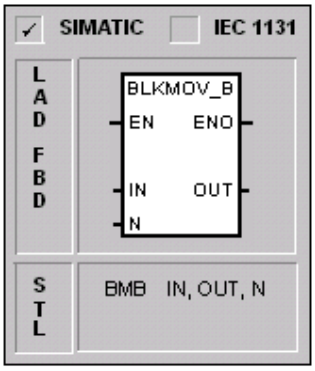
		AC, *VD, *LD, *AC	

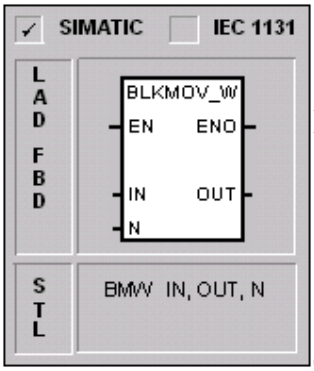
1.2. Truyền một vùng nhớ dữ liệu

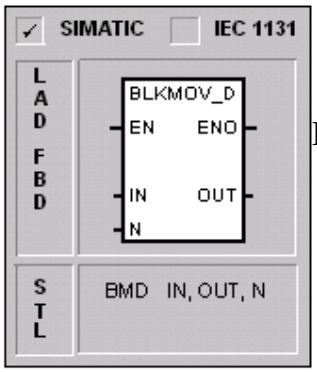
Phép truyền Block Move Byte, Block Move Word, Block Move Doubleword sẽ thực hiện truyền một số lượng Byte (N) có địa chỉ Byte đầu tại ngõ vào IN sang vùng nhớ có địa chỉ tại ngõ ra OUT. N là số lượng Byte có giới hạn từ 1 đến 255.

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

Block Move	Inputs/Output	Toán hạng	Dạng dữ liệu
Byte	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



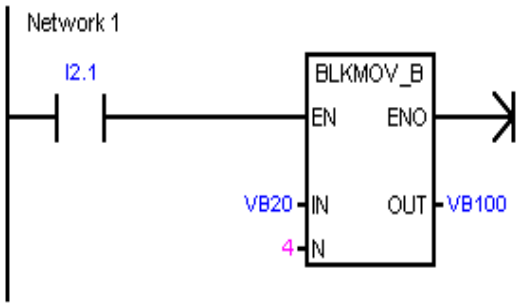


DoubleWord	IN, OUT	VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD	Doubleword
	N	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD	Byte

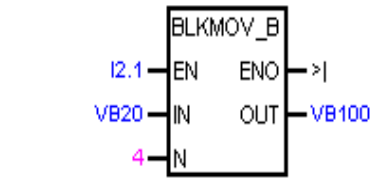
Ví dụ về truyền một mảng dữ liệu BLKMOVE:

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

Mục tiêu: Trình bày các lệnh so sánh trong PLC.

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.

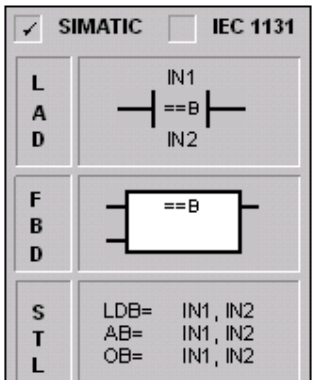
Dữ liệu tại ngõ vào IN1 được so sánh với dữ liệu tại ngõ vào IN2.

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



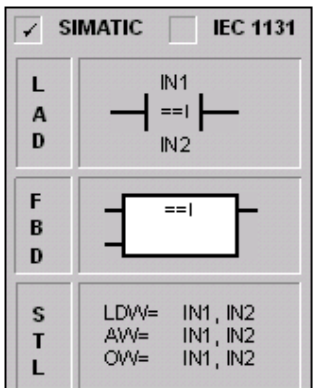
Lệnh so sánh bao gồm: IN1 = IN2, IN1 > IN2
IN1>=IN2, IN1 < IN2
IN1<=IN2, IN1<>IN2

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



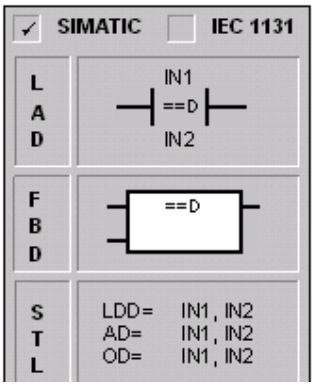
Lệnh so sánh bao gồm: IN1 = IN2, IN1 > IN2
IN1>=IN2, IN1 < IN2
IN1<=IN2,IN1<>IN2

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



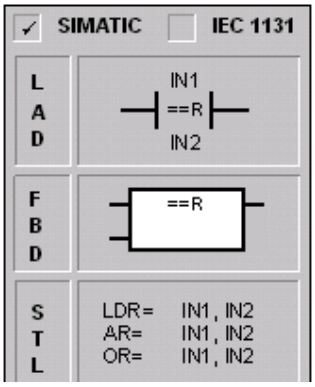
Lệnh so sánh bao gồm: IN1 = IN2, IN1 > IN2
IN1>=IN2, IN1 < IN2
IN1<=IN2,IN1<>IN2

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:

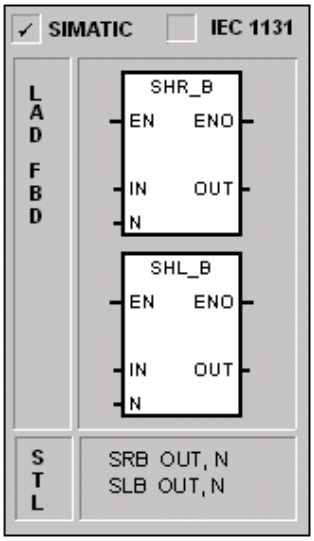
Inputs/Outputs	Operands	Data Types
Inputs	ID, QD, MD, SD, SMD, VD, LD, AC, Constant, *VD, *LD, *AC	REAL
Output	I, Q, M, SM, T, C, V, S, L, Power Flow	BOOL

3. Chức năng dịch chuyển

Mục tiêu: Trình bày các lệnh dịch chuyển trong PLC

3.1. 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

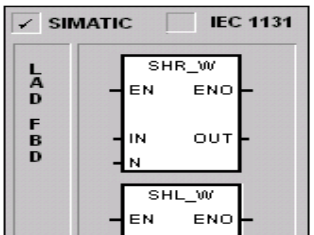
SM1.1: Bit cao được set tới giá trị cuối cùng của bit được dịch

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

3.2. 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

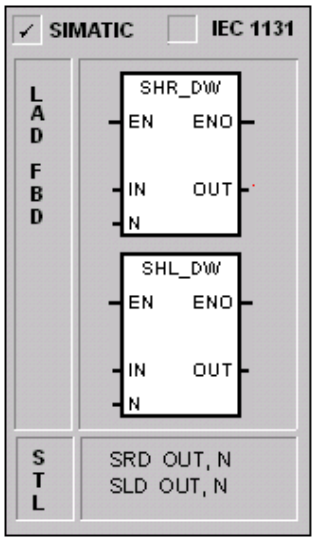
SM1.1: Bit cao được set tới giá trị cuối cùng của bit được dịch

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

3.3. 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

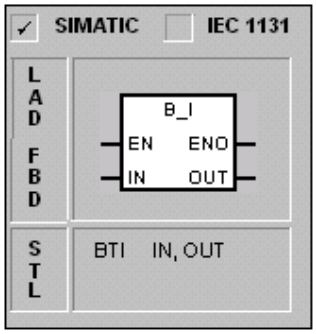
Bảng giới hạn toán hạng và vùng dữ liệu hợp lệ:

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

4. Chức năng chuyển đổi (Converter)

Mục tiêu: Trình bày các lệnh chuyển đổi loại dữ liệu số.

4.1. Chuyển đổi Byte sang Integer

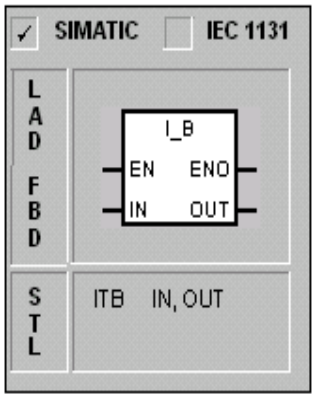


Lệnh chuyển đổi B_I chuyển đổi dữ liệu chứa trong Byte có địa chỉ tại ngõ IN sang giá trị số nguyên, kết quả chứa vào biến xác định tại ngõ ra OUT.

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

4.2. Chuyển đổi Integer sang Byte

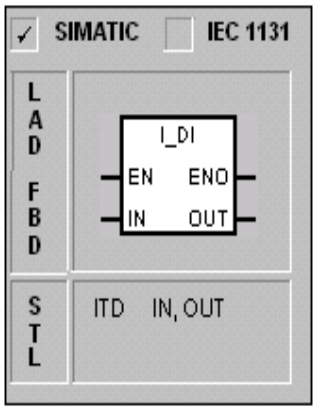


Lệnh chuyển đổi I_B chuyển đổi dữ liệu chứa trong Word có địa chỉ tại ngõ IN sang giá trị Byte, kết quả chứa vào biến xác định tại ngõ ra OUT. Các số nguyên có thể chuyển đổi là 0 đến 255.

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

4.3. Chuyển đổi Integer sang Double Integer

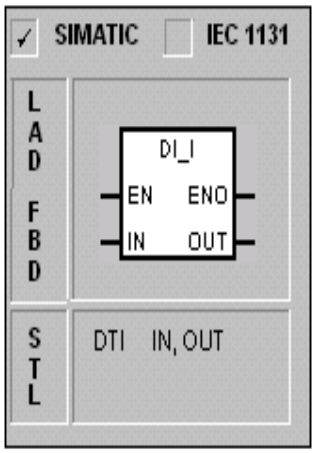


Lệnh chuyển đổi I_DI chuyển đổi giá trị số I tại ngõ IN sang một giá trị số nguyên kép DI, kết quả chứa vào biến xác định tại ngõ ra OUT.

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

4.4. Chuyển đổi Double Integer sang Integer

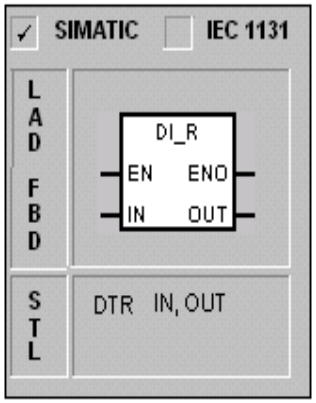


Lệnh chuyển đổi DI_I chuyển đổi giá trị số nguyên kép DI tại ngõ IN sang một giá trị số nguyên I, kết quả chứa vào biến xác định tại ngõ ra OUT. Nếu phép biến đổi bị tràn (kết quả lớn hơn khả năng chứa của ngõ OUT) thì ngõ ra không thay đổi và trạng thái EN)=0.

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

4.5. 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.

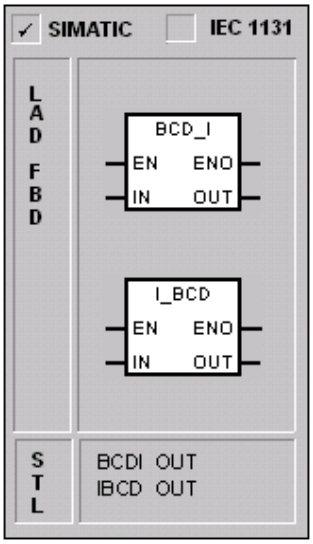
Bảng giới hạn vùng toán hạng và dạng dữ liệu hợp lệ:

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

4.6. 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

5. Chức năng toán học

Mục tiêu: Trình bày về các lệnh 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

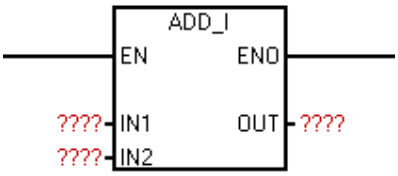
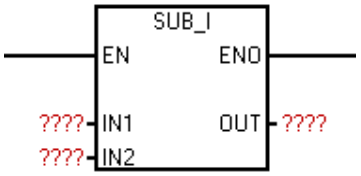
SUB_I(LAD)

-I(STL)

Lệnh thực hiện phép 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.

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

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

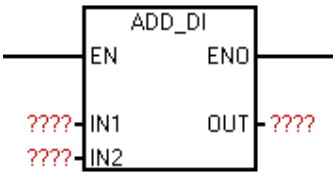
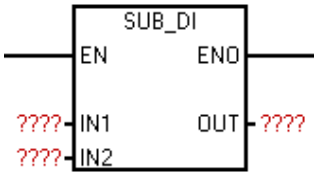
SUB_DI(LAD)

-D(STL)

Lệnh thực hiện phép 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.

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

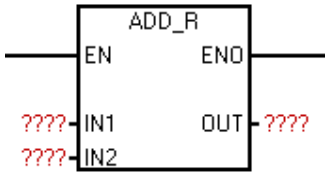
SUB_R(LAD)

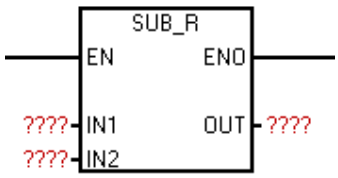
-R(STL)

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 (REAL): VD, ID, QD, MD, SD, SMD, LD, AC, Constant, *VD, *LD, *AC



-D IN1

IN2

OUT VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

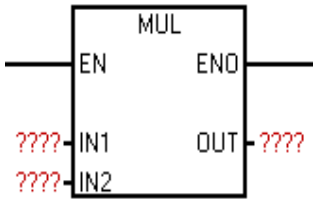
5.2. Phép nhân chia (MUL và DIV).

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:

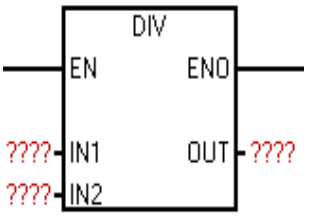
LAD	STL	Toán hạng
	MUL n1 n2	n1, IN1, IN2:VW, IW, QW, MW, SW, SMW, T, C, LW, AC, AIW, Constant, *VD, *LD, *AC (INT) n2, OUT: VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC (DINT)

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

Tương tự, ta có các lệnh nhân chia sau (*Tham khảo cú pháp và toán hạng trong mục trợ giúp*).

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.

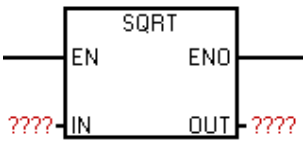
MUL_R: Nhân hai số thực.

DIV_R: Chia hai số thực.

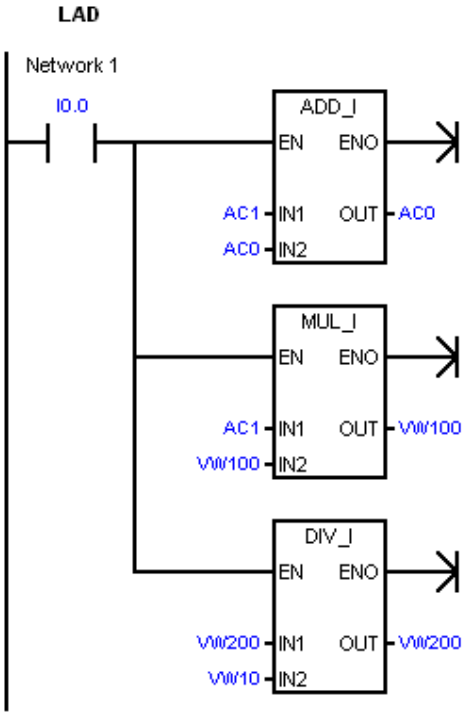
5.3. Phép lấy căn bậc hai (SQRT)

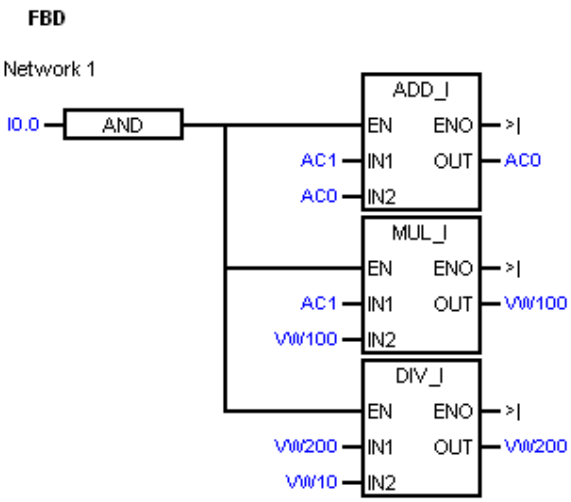
Lệnh thực hiện phép lấy căn bậc hai của số thực 32 bit IN. Kết quả cũng là một số 32 bit được ghi vào từ kép OUT (4 bytes).

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: (REAL) VD, ID, QD, MD, SMD, SD, LD, AC, Constant, *VD, *LD, *AC OUT: (REAL) VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

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





STL

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

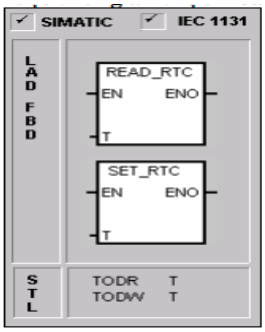
Khi I0.0 ON, chương trình thực thi:

	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

Mục tiêu: Trình bày chức năng thời gian thực trong PLC.

6.1. Lệnh đọc thời gian thực Read_RTC

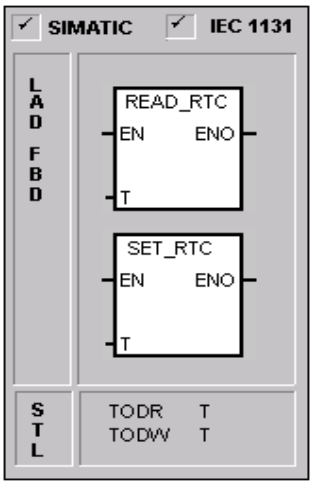


Bit EN : Bit cho phép đọc thời gian thực
T (8byte): VB,IB,QB,MB,SB,LB,*AC,*VD,*LD
Được định dạng như sau:

T (byte)	Giá trị (định dạng BCD)
0 (năm)	0-99
1 (tháng)	0 -12
2 (ngày)	0 - 31
3 (giờ)	0 - 23
4 (phút)	0 - 59
5 (giây)	0 - 59
6 (00)	00
7 (ngày trong tuần)	1 – 7; 1: Sunday

6.2. 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)	0-12
2 (ngày)	0-31
3 (giờ)	0-23
4 (phút)	0-59
5 (giây)	0-59
6 (00)	00
7 (ngày trong tuần)	1-7;
1: Sunday	1

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.

Bài 6

CÁC BÀI TẬP ỨNG DỤNG TRONG ĐIỀU KHIỂN ĐỘNG CƠ

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

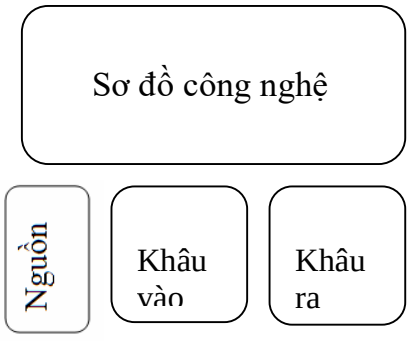
1. Giới thiệu

Mục tiêu: Giới thiệu sơ lược về mô hình bố trí và sử dụng thiết bị thực hành.

Việc nâng cao chất lượng giảng dạy trong kỹ thuật luôn luôn gắn liền với việc học đi đôi với hành. Hiện nay, trong thực hành của học sinh, các đồ dùng với hợp lý, gọn gàng, đảm bảo an toàn đã giúp cho học sinh có một cái nhìn khái quát về những ứng dụng trong thực thế. Nó không những giúp cho học sinh có hứng thú trong học tập mà còn có thêm những sáng kiến mới, cũng như cách thức tổ chức trong thực tế.

Từ lý do đó, việc có được những mô hình đáp ứng được những yêu cầu trên là vô cùng cần thiết, có thể dùng cho môn học PLC từ cơ bản đến nâng cao để mô phỏng các quy trình công nghệ trong thực tế sau này.

Nhìn chung, mô hình được bố trí như sau:



Hình 6.1: Cấu trúc mô hình điều khiển.

Mục đích của việc phần thành từng /cụm riêng để giúp học sinh tránh nhầm lẫn đáng tiếc trong quá trình thực hành, đồng thời tiếp thu thêm cách thức tổ chức thực hành.

Các mô hình thực tập gồm có:

- Mô hình thang máy xây dựng
- Mô hình điều khiển động cơ sao – tam giác
- Mô hình xe chuyển nguyên liệu
- Mô hình đo chiều dài và sắp xếp vật liệu
- Mô hình thiết bị nâng hàng hóa
- Mô hình thiết bị vô nước chai
- Mô hình thiết bị trộn hóa chất

Các mô hình này đã được sắp xếp theo thứ tự và có các bài tập kèm theo.

Toàn bộ các mô hình đều sử dụng điện áp 24VDC, được cấp từ nguồn riêng hoặc nguồn có sẵn cung cấp cho PLC. Đối với các PLC có ngõ ra là relay thi trên mô hình có thiết kế sẵn nguồn U_s dùng làm nguồn cung cấp cho các ngõ ra này.

Các mô hình cũng có thể được ứng dụng cho các bộ lập trình cơ nhỏ như LOGO của hãng Siemens, EASY của hãng Moeller, ZEN của omron…

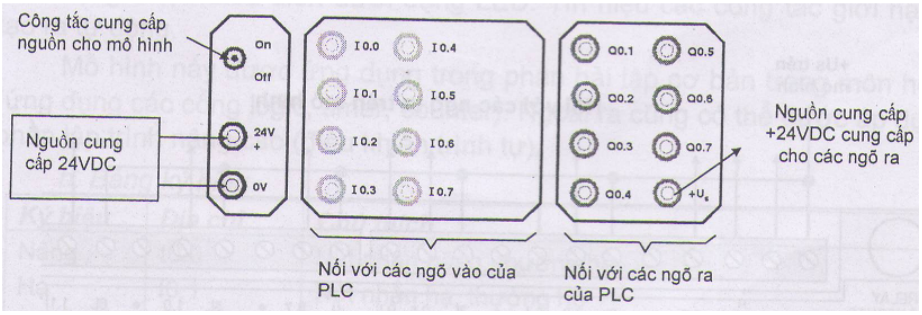
Tùy theo nội dung bài học mà có thể chọn mô hình thích hợp cho bài tập ứng dụng. Một mô hình có thể sử dụng với nhiều bài tập ứng dụng khác nhau.

VD: Mô hình thang máy xây dựng có thể được sử dụng trong các bài học như điều khiển theo tổ hợp logic, điều khiển với các lệnh ghi/xóa tiếp điểm, sử dụng timer, counter, và ứng dụng trong điều khiển trình tự.

2. Cách kết nối dây

Mục tiêu: Trình bày cách kết nối dây của PLC với các thiết bị ngoại vi trong mô hình thực hành.

Cách kết nối dây từ PLC đến mô hình được cho như hình vẽ:



Hình 6.2: Cách kết nối với mô hình.

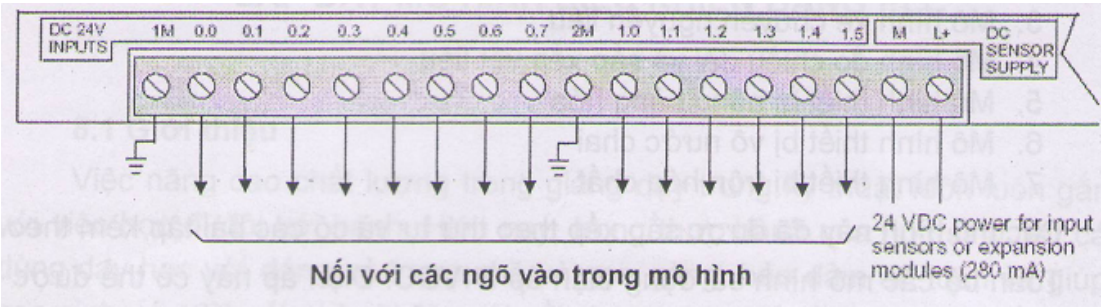
Để kết nối được vi PLC, yêu cầu các modul vào/ra của PLC như sau:

-Sử dụng nguồn áp 24VDC (ổn áp).

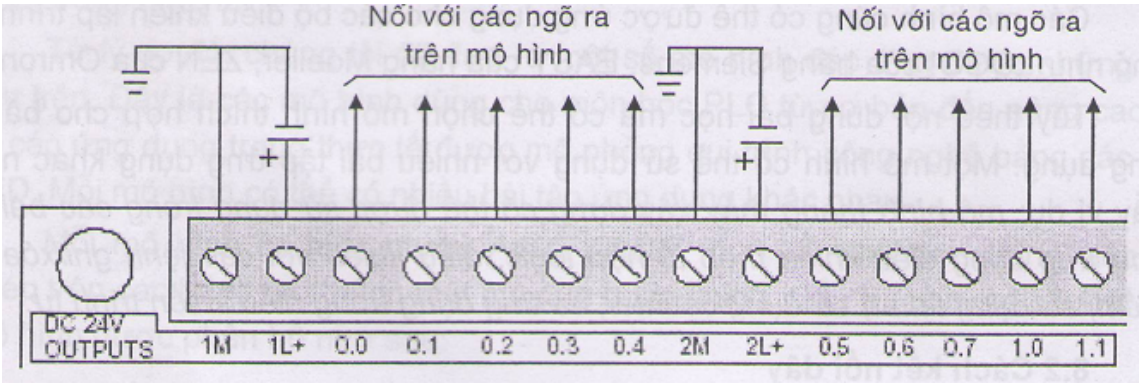
-Nguồn cung cấp cho modul vào/ra phải được kết nối.

-Nếu các ngõ ra là rơle và chưaq có nguồn cung cấp thì đấu chung một đầu lại rồi nối với nguồn U_s ở trên mô hình (hoặc nguồn +24VDC ngoài). Còn các đầu còn lại của rơle nối với ngõ ra trên mô hình.

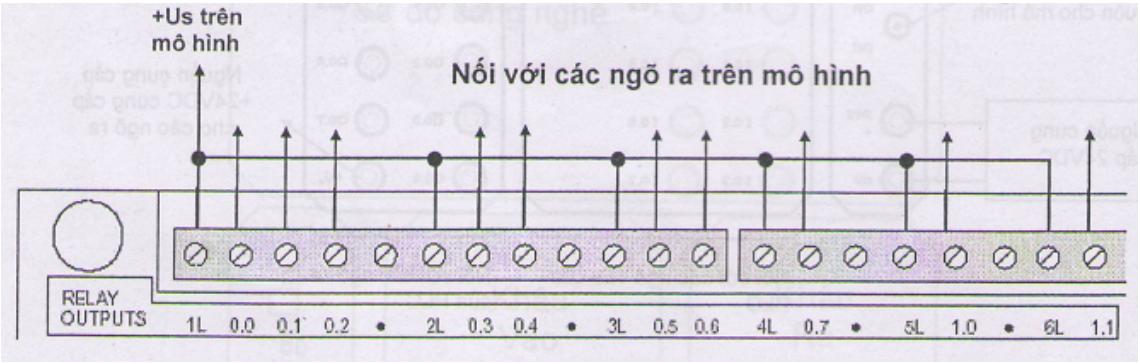
Các kết nối có thể thực hiện như ví dụ sau:



Hình 6.3: Cách kết nối với các ngõ vào trong mô hình.



Hình 6.4: Cách kết nối ngõ ra 24VDC của PLC với các ngõ ra trong mô hình.

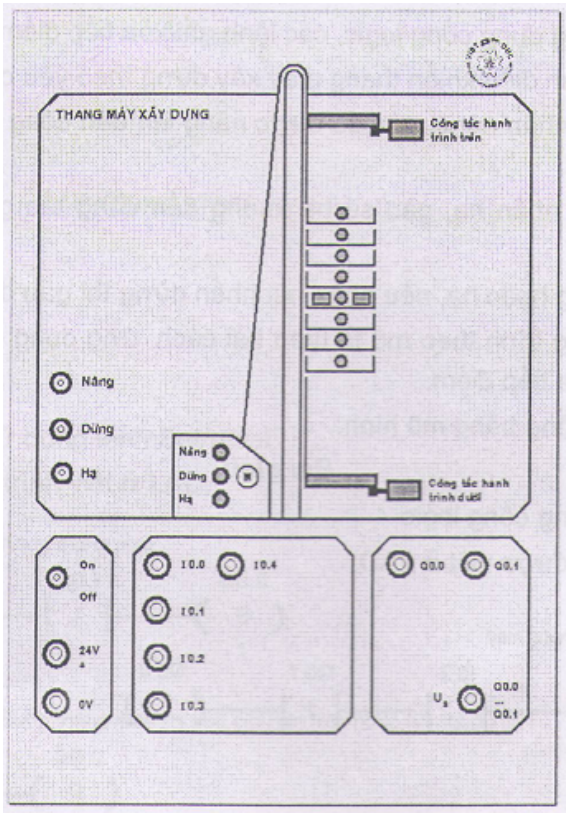


Hình 6.5: Cách kết nối các ngõ ra của PLC với các ngõ ra trong mô hình.

3. Các mô hình và bài tập ứng dụng.

Mục tiêu: Trình bày một số mô hình cụ thể thường sử dụng PLC và các bài toán thông dụng.

3.2. Mạch đổi chiều quay trực tiếp.



Hình 6.6: Mô hình thang máy xây dựng.

a, Mô tả

Mô tả quy trình công nghệ của một thang máy xây dựng. Sự chuyển động của thang được biểu diễn dưới dạng LED. Tín hiệu các công tắc giới hạn được tạo ra tự động.

Mô hình này được ứng dụng trong phần bìa tập cơ bản trong môn học PLC (ứng dụng các cổng logic, timer, counter). Ngoài ra cũng có thể được áp dụng cho phần mềm lập trình nâng cao (điều khiển trình tự).

b, Bảng ký hiệu:

Ký hiệu	Địa chỉ	Ghi chú
Nâng	I0.0	Nút nhấn nâng, thường mở
Hạ	I0.1	Nút nhấn hạ, thường mở
Dừng	I0.2	Nút nhấn dừng, thường đóng
GH trên	I0.3	Công tắc hành trình trên, thường đóng
GH dưới	I0.4	Công tắc hành trình dưới, thường đóng
K1	Q0.0	Cuộn dây khởi động từ K1, nâng gầu
K2	Q0.1	Cuộn dây khởi động từ K2, hạ gầu

c, Bài tập mẫu:

Các bài tập mẫu này được giải với phần mềm step 7 Micro/win 32 V3.01.

Bài tập 1: Ứng dụng cổng logic, các lệnh ghi/xóa tiếp điểm.

Viết chương trình điều khiển thang máy xây dựng theo yêu cầu sau:

- Khi nhấn nút nâng hàng, gàu sẽ được nâng lên đến công tắc giới hạn trên thì dừng lại.

- Khi nhấn nút hạ, gàu sẽ hạ xuống đến công tắc giới hạn dưới thì dừng lại.

- Khi đang nâng hoặc hạ, nếu nhấn nút dừng thì gàu dừng lại.

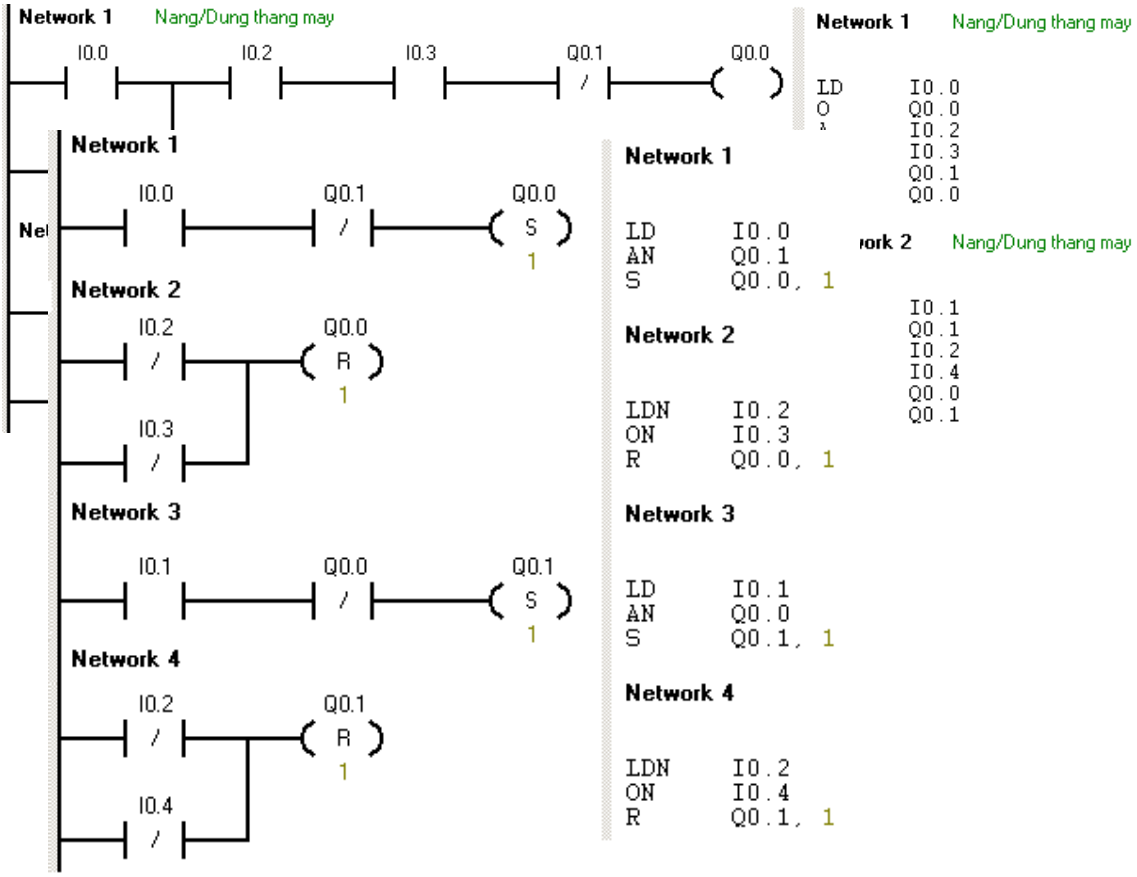
- Hãy viết chương trình theo mô tả với hai cách: ứng dụng cổng logic và sử dụng các lệnh ghi xóa tiếp điểm.

Kiểm tra hoạt động bằng mô hình.

Bài giải

Cách 1: Ứng dụng cổng logic

- Chương trình được viết ở LAD và STL:



Cách 2: Sử dụng lệnh set/reset:

- Chương trình viết ở LAD và STL:

Bài tập 2: Sử dụng timer.

Viết chương trình điều khiển thang máy xây dựng theo yêu cầu sau:

a/. Khi ấn nút nâng thì gàu được nâng lên, đến giới hạn trên thì dừng lại 5s, sau đó tự động hạ xuống. Đến giới hạn dưới thì dừng.

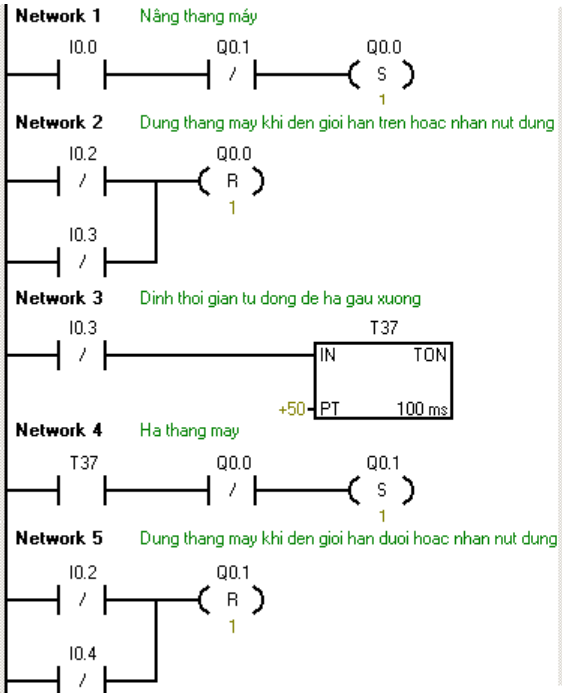
Trong quá trình nâng lên hoặc hạ xuống cũng có thể dừng.

b/. Trong khi ấn nút nâng thì gàu được nâng lên, đến giới haj trên thì dừng lại 5s, sau đó tự động hạ xuống đến giới hạn dưới thì dừng lại 10s, sau đó tự động nâng lên.

Thang cũng có thể nâng lên khi chưa hết 10s chờ tự động mà có người ấn nút nâng.

Bài giải:

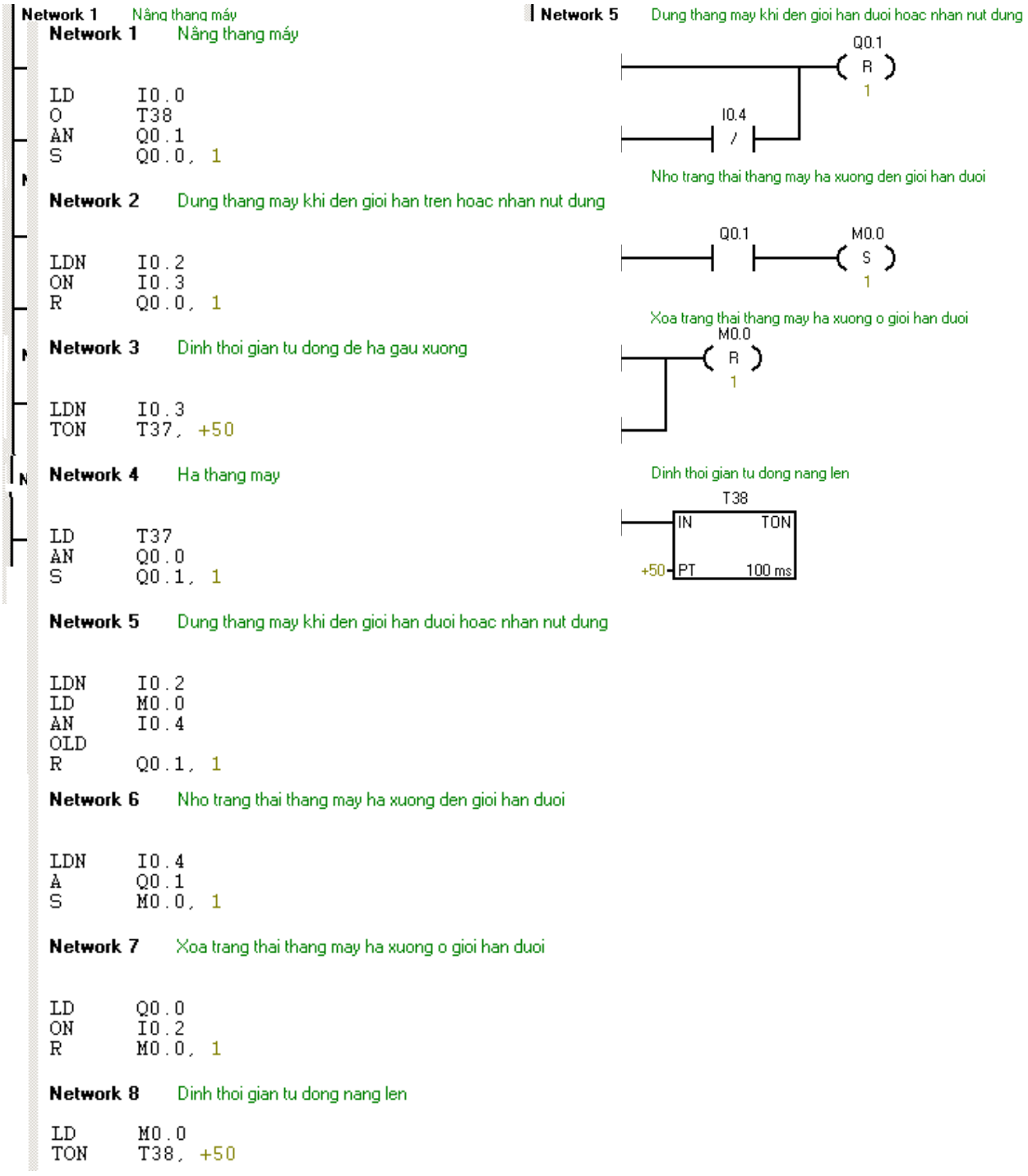
Câu a: Chương trình được viết ở LAD và STL:




```
Network 1    Nâng thang máy
LD      I0.0
AN      Q0.1
S       Q0.0, 1
Network 2    Dung thang may khi den gioi han tren hoac nhan nut dung
LDN     I0.2
ON      I0.3
R       Q0.0, 1
Network 3    Dinh thoi gian tu dong de ha gau xuong
LDN     I0.3
TON     T37, +50
Network 4    Ha thang may
LD      T37
AN      Q0.0
S       Q0.1, 1
Network 5    Dung thang may khi den gioi han duoi hoac nhan nut dung
LDN     I0.2
ON      I0.4
R       Q0.1, 1
```

Ghi chú: Nếu bài toán có yêu cầu khi nhấn nút hạ thì gầu cũng hạ, lúc này chèn thêm một tiếp điểm "NO" của I0.1 song song với tiếp điểm "NO" của T37 ở Network 4.

Câu b: Chương trình được viết ở LAD



```
Network 1    Nâng thang máy

LD      I0.0
O       T38
AN      Q0.1
S       Q0.0, 1

Network 2    Dung thang may khi den gioi han tren hoac nhan nut dung

LDN     I0.2
ON      I0.3
R       Q0.0, 1

Network 3    Dinh thoi gian tu dong de ha gau xuong

LDN     I0.3
TON     T37, +50

Network 4    Ha thang may

LD      T37
AN      Q0.0
S       Q0.1, 1

Network 5    Dung thang may khi den gioi han duoi hoac nhan nut dung

LDN     I0.2
LD      M0.0
AN      I0.4
OLD
R       Q0.1, 1

Network 6    Nho trang thai thang may ha xuong den gioi han duoi

LDN     I0.4
A       Q0.1
S       M0.0, 1

Network 7    Xoa trang thai thang may ha xuong o gioi han duoi

LD      Q0.0
ON      I0.2
R       M0.0, 1

Network 8    Dinh thoi gian tu dong nang len

LD      M0.0
TON     T38, +50
```

Chương trình viết ở STL:

Ghi chú: Nếu bài toán có cho yêu cầu khi nhấn nút hạ thì gàu cũng hạ, lúc này chèn thêm một tiếp điểm "NO" của I0.1 song song với tiếp điểm "NO" của T37 ở Network 4.

Bài tập 3: Sử dụng bộ đếm

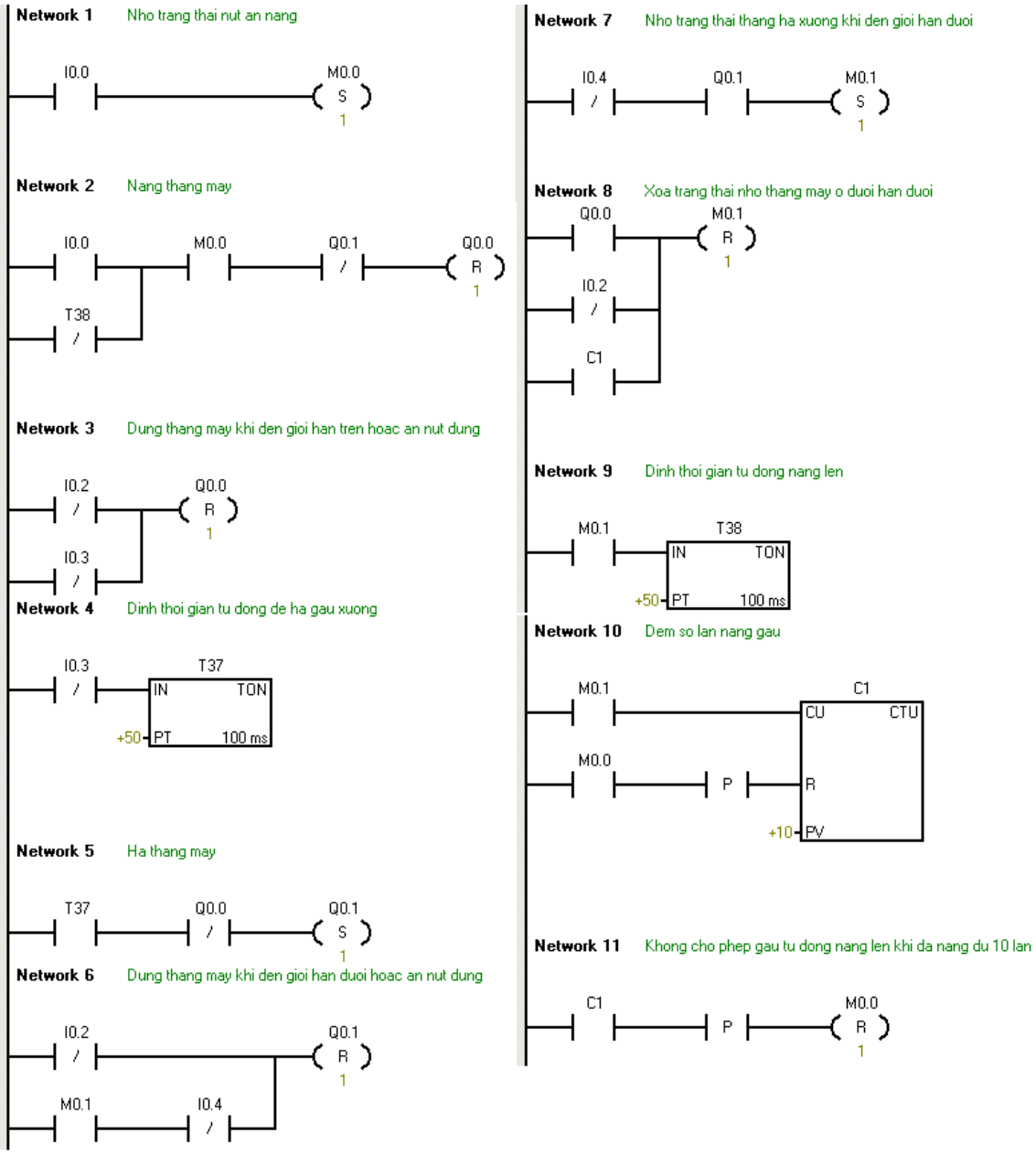
Viết chương trình điều khiển thang máy xây dựng theo yêu cầu sau:

Khi ấn nút nâng thì gàu được nâng lên, đến giới hạn trên thì dừng lại 5s, sau đó tự động hạ gàu xuống đến giới hạn dưới thì dừng lại 10s, sau đó tự động nâng lên. Khi gàu nang lên được 10 lần thì không nâng lên nữa và sau đó hạ xuống trở về vị trí cơ bản và quá trình lặp lại.

Trong quá trình đang nâng hoặc hạ thì cũng có thể dừng gàu.

Giải:

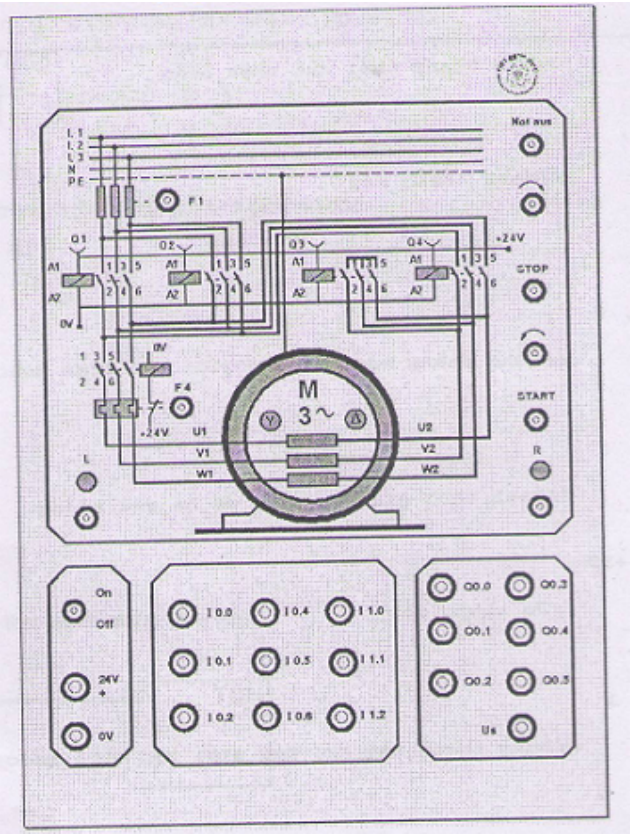
Chương trình viết ở LAD:



Chương trình viết ở STL:

```
Network 1      Nho trang thai nut an nang

LD      I0.0
S       M0.0, 1

Network 2      Nang thang may

LD      I0.0
ON      T38
A       M0.0
AN      Q0.1
R       Q0.0, 1

Network 3      Dung thang may khi den gioi han tren hoac an nut dung

LDN     I0.2
ON      I0.3
R       Q0.0, 1

Network 4      Dinh thoi gian tu dong de ha gau xuong

LDN     I0.3
TON     T37, +50

Network 5      Ha thang may

LD      T37
AN      Q0.0
S       Q0.1, 1

Network 6      Dung thang may khi den gioi han duoi hoac an nut dung
LDN     I0.2
LD      M0.1
AN      I0.4
OLD
R       Q0.1, 1

Network 7      Nho trang thai thang ha xuong khi den gioi han duoi
LDN     I0.4
A       Q0.1
S       M0.1, 1

Network 8      Xoa trang thai nho thang may o duoi han duoi
LD      Q0.0
ON      I0.2
O       C1
R       M0.1, 1

Network 9      Dinh thoi gian tu dong nang len

LD      M0.1
TON     T38, +50

Network 10     Dem so lan nang gau

LD      M0.1
LD      M0.0
EU
CTU     C1, +10

Network 11     Khong cho phep gau tu dong nang len khi da nang du 10 lan

LD      C1
EU
R       M0.0, 1
```

3.3. Mô hình điều khiển động cơ Y-Δ



Hình 6.7: Điều khiển khởi đông động cơ.

a, Mô tả:

Mô hình này mô phỏng một động cơ mở máy Y/Δ. Sự chuyển động của rotor và sự đóng cắt của các contactor được mô tả bằng LED.

Mô hình này được ứng dụng trong phần bài tập cơ bản của môn PLC (ứng dụng PLC điều khiển động cơ, các cổng logic, timer).

Có thể dựa vào mô hình băng tải để lập thành các bài tập khác nhau như: mở máy động cơ Y/Δ, đảo chiều quay động cơ, điều khiển động cơ ở các tốc độ khác nhau, bồn sấy.

Cách thức nối dây tương tự như trên.

Sau khi đã nối dây với mô hình PLC xong, thực hiện viết chương trình theo bài tập đưa ra (có thể tự kiểm tra các ngõ vào/ra bằng phần mềm, ví dụ như s7-200 dùng bảng status chart) và sau đó thực hiện mô phỏng với mô hình.

Khi động cơ quay phải (trái), các đèn LED sẽ chuyển động theo chiều phải (trái) theo các trạng thái tương ứng. nếu đèn chuyển động nhanh báo động cơ

quay tốc độ nhanh và ngược lại. Các contactor khi được đóng điện báo bởi các đèn báo đặt ở ký hiệu cuộn dây. Trong động cơ có đặt 3 đèn báo tượng trưng cho 3 cuộn dây của động cơ. Nếu động cơ có điện thì các dãy đèn này sáng.

** Cách vận hành mô hình:*

Sau khi đã nối dây với mô hình PLC xong, thực hiện viết chương trình theo bài tập đưa ra (có thể tự kiểm tra các ngõ vào/ra bằng phần mềm, ví dụ như s7-200 dùng bảng status chart) và sau đó thực hiện mô phỏng với mô hình.

Khi động cơ quay phải (trái), các đèn LED sẽ chuyển động theo chiều phải (trái) theo các trạng thái tương ứng. nếu đèn chuyển động nhanh báo động cơ quay tốc độ nhanh và ngược lại. Các contactor khi được đóng điện báo bởi các đèn báo đặt ở ký hiệu cuộn dây. Trong động cơ có đặt 3 đèn báo tượng trưng cho 3 cuộn dây của động cơ. Nếu động cơ có điện thì các dãy đèn này sáng.

b. Bảng ký hiệu:

Ký hiệu	Địa chỉ	Chú thích
Start	I0.0	Nhấn nút khởi động, thường hở
Stop	I0.1	Nhấn nút dừng, thường đóng
F1	I0.2	CB 3 pha, công tắc
F4	I0.4	Relay nhiệt, công tắc
Right	I0.5	Chọn chiều quay phải, thường hở
Left	I0.6	Chọn chiều quay trái, thường hở
H_Right	I1.0	Quay phải tốc độ nhanh, thường hở
H-Left	I1.1	Quay trái tốc độ nhanh, thường hở
Not aus	I1.2	Dừng khẩn cấp, công tắc
Q1	Q0.0	Khởi động từ K1, quay phải
Q2	Q0.1	Khởi động từ K2, quay trái
Q3	Q0.2	Khởi động từ K3, chạy Y
Q4	Q0.3	Khởi động từ K1, chạy Δ

R	Q0.4	Đèn báo quay phải
L	Q0.5	Đèn báo quay trái

c. Bài tập mẫu:

Các bài tập về điều khiển động cơ có thể sử dụng được với mô hình này. Trong phần này đưa ra bốn bài tập mẫu như sau:

Bài 1: (Sử dụng các cổng logic, các lệnh ghi/xóa tiếp điểm): Viết chương trình điều khiển động cơ Y/Δ bằng tay.

Động cơ được điều khiển theo yêu cầu sau:

Khi ấn nút khởi động "start" (I0.0), các contactor Q1 và Q3 đóng lại. Động cơ chạy ở chế độ tam Y. Sau đó nếu ấn nút right (I0.5) thì contactor Q3 tắt và contactor Q4 có điện. Động cơ chạy ở chế độ D.

Động cơ được cung cấp điện bởi CB 3 pha, và được bảo vệ bởi rowle nhiệt và nút dừng stop.

Bài 2: (Sử dụng các cổng logic, các lệnh ghi/xóa tiếp điểm): Viết chương trình điều khiển tốc độ và đảo chiều quay động cơ.

Sau khi được cấp điện, động cơ hoạt động như sau:

Trước tiên chọn chiều quay của động cơ bằng các nút ấn "Right" hoặc "Left". Khi ấn nút này tương ứng thì các contactor Q1, Q2 đóng lại và các đèn báo quay phải "R" hoặc quay trái "L" sáng. Sau đó ấn nút khởi động "Start" thì động cơ sẽ quay ở tốc độ thấp trước (contactor Q3 có điện). Bây giờ có thể cho động cơ quay ở tốc độ cao hơn bằng cách nhấn các nút ấn tương ứng với các chiều quay "H_Right" hoặc "H_Left". Khi ấn các nút này thì contactor Q4 có điện. Động cơ được bảo vệ quá nhiệt và dừng bằng nút ấn "stop".

Chú ý: không được phép đảo chiều trực tiếp, cũng như chuyển đổi tốc độ có thể từ thấp sang cao.

Bài 3: (Sử dụng timer) Viết chương trình điều khiển mở máy động cơ Y/Δ.

Viết chương trình điều khiển mở máy động cơ Y/Δ theo yêu cầu sau:

a/. Sau khi CB 3 pha được đóng, ấn nút khởi động “Start” thì động cơ hoạt động ở chế độ Y. Sau thời gian 10s thì động cơ được chuyển sang hoạt động ở chế độ tam giác.

(chú ý: động cơ quay theo chiều phải).

b/. Khi ấn nút right hoặc left thì động cơ hoạt động ở chế độ sao/tam giác với chiều quay là chiều đã chọn. Tương ứng các đèn báo quay phải hoặc quay trái sáng.

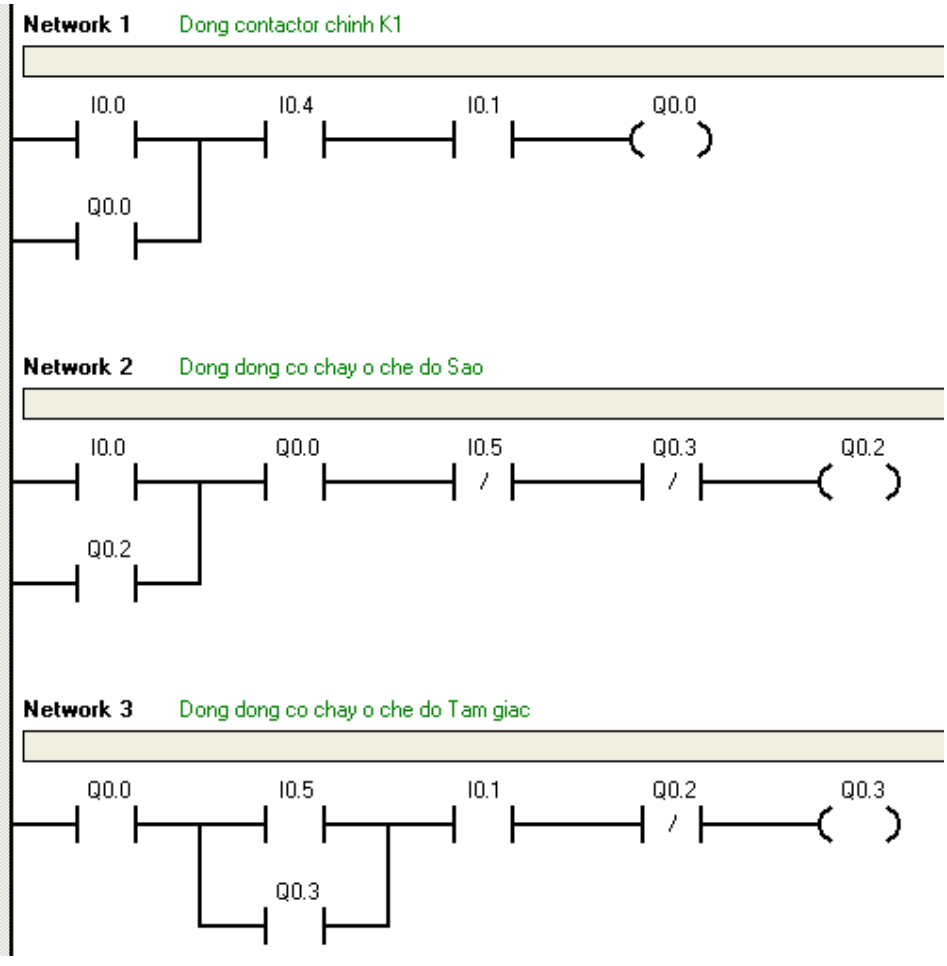
Bài 4: (Điều khiển trình tự) Viết chương trình điều khiển bồn trộn.

Viết chương trình điều khiển bồn trộn theo yêu cầu sau: Khi ấn nút “start”, thì động cơ quay phải ở tốc độ thấp trong thời gian 10s, sau đó dừng 5s, sau đó quay trái 10s, tiếp đó dừng 5s. Quá trình cứ thế lặp lại. Sau khoảng 20 lần thì động cơ dừng 10s và sau đó quay phải ở tốc độ thấp đượcv 5s thì chuyển sang quay ở tốc độ cao khoảng 30s thì dừng hẳn.

Bồn trộn được bảo vệ bởi nút dừng “stop”.

Giải mẫu: (bài tập 1)

Chương trình viết ở LAD:



Chương trình viết ở STL:

Network 1 Dong contactor chinh K1

```
LD      I0.0
O       Q0.0
A       I0.4
A       I0.1
=       Q0.0
```

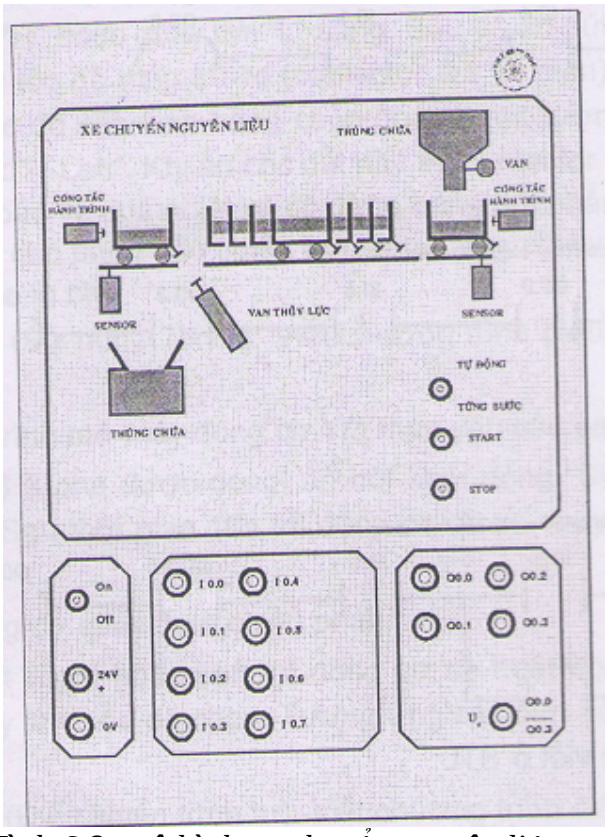
Network 2 Dong dong co chay o che do Sao

```
LD      I0.0
O       Q0.2
A       Q0.0
AN      I0.5
AN      Q0.3
=       Q0.2
```

Network 3 Dong dong co chay o che do Tam giac

```
LD      Q0.0
LD      I0.5
O       Q0.3
ALD
A       I0.1
AN      Q0.2
=       Q0.3
```

6.3.Mô hình xe chuyển nguyên liệu:



Hình 6.8: mô hình xe chuyển nguyên liệu

a. Mô tả:

Mô phỏng một xe chuyển nguyên liệu từ nơi này đến nơi khác với việc lấy nguyên liệu từ bồn chứa và xả nguyên liệu vào bồn chứa khác bằng các đèn LED với nhiều màu sắc khác nhau. Cũng như các cảm biến và công tắc hành trình đều tạo ra tự động.

Ứng dụng trong PLC cơ bản: điều khiển tổ hợp logic

Ứng dụng trong PLC nâng cao: điều khiển trình tự

Cách thức nối dây tương tự như trên.

b. Cách vận hành mô hình:

Sau khi đã nối dây mô hình với PLC xong, thực hiện viết chương trình theo bài tập đã đưa ra (có thể tự kiểm tra các ngõ vào/ra bằng phần mềm sau đó thực hiện mô phỏng chương trình.

Nguyên liệu trong bồn chứa khi đang được xả hoặc được rót vào biểu thị bởi những dòng LED chạy, các dòng LED chạy đuổi tượng trưng cho xe chạy ở chế độ tự động, chế độ tay, hoặc hoạt động ở cả hai chế độ.

d. 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

d. Bài tập mẫu:

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"

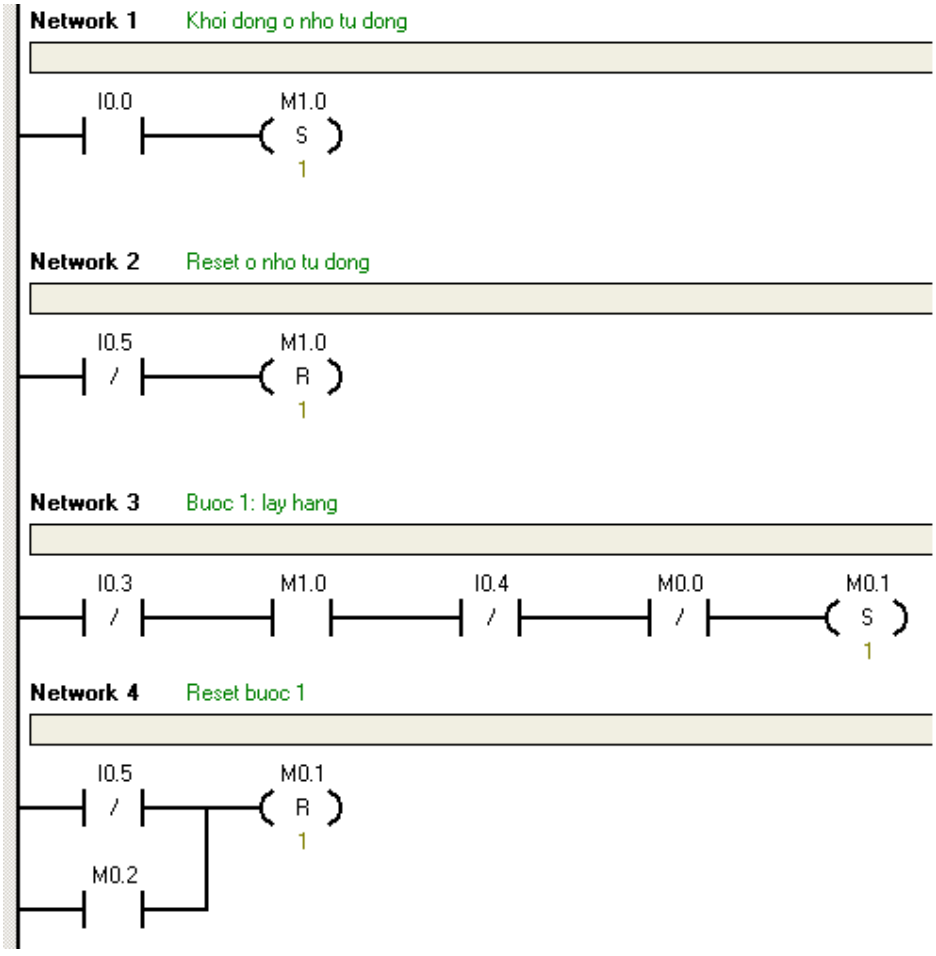
  Ví dụ: Khi ấn "start" xe đúng vị trí van xả được mở, khi xe đầy thì S3 tác động, van xả đóng lại. Nếu tiếp tục ấn "Start" thì xe chạy về hướng B.

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

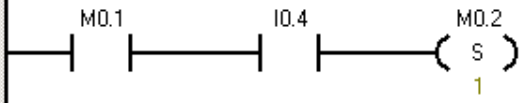
  - Điều khiển dùng tổ hợp logic
  - Điều khiển trình tự

Giải bài tập mẫu: Chương trình được viết theo kiểu trình tự

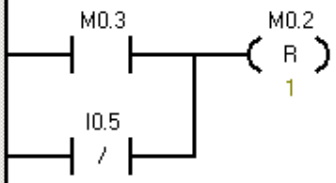
Chương trình được viết ở LAD:



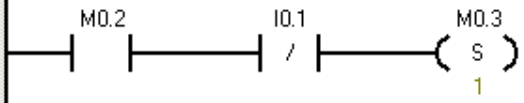
Network 5 Buoc 2: Chuyen hang



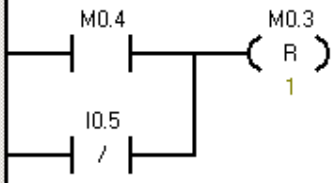
Network 6 Reset buoc 2



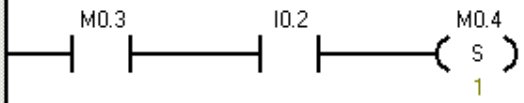
Network 7 Buoc 3: Xa hang



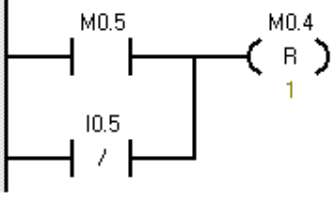
Network 8 Reset buoc 3



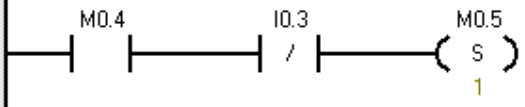
Network 9 Buoc 4: tro ve



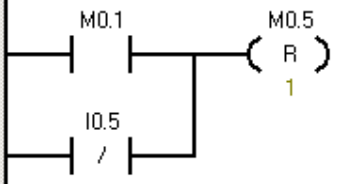
Network 10 Reset buoc 4



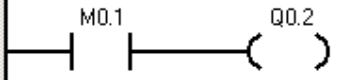
Network 11 Buoc 5: Ket thuc



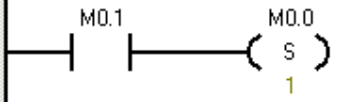
Network 12 Reset buoc 5



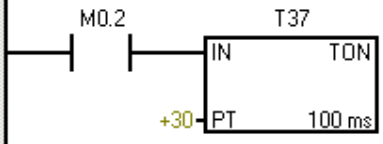
Network 13 lay hang



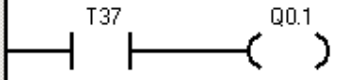
Network 14 Set o nho khoi dong



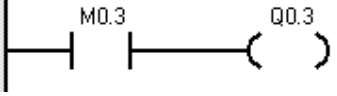
Network 15 Dinh thoi gian on dinh sau khi hang da do day xe

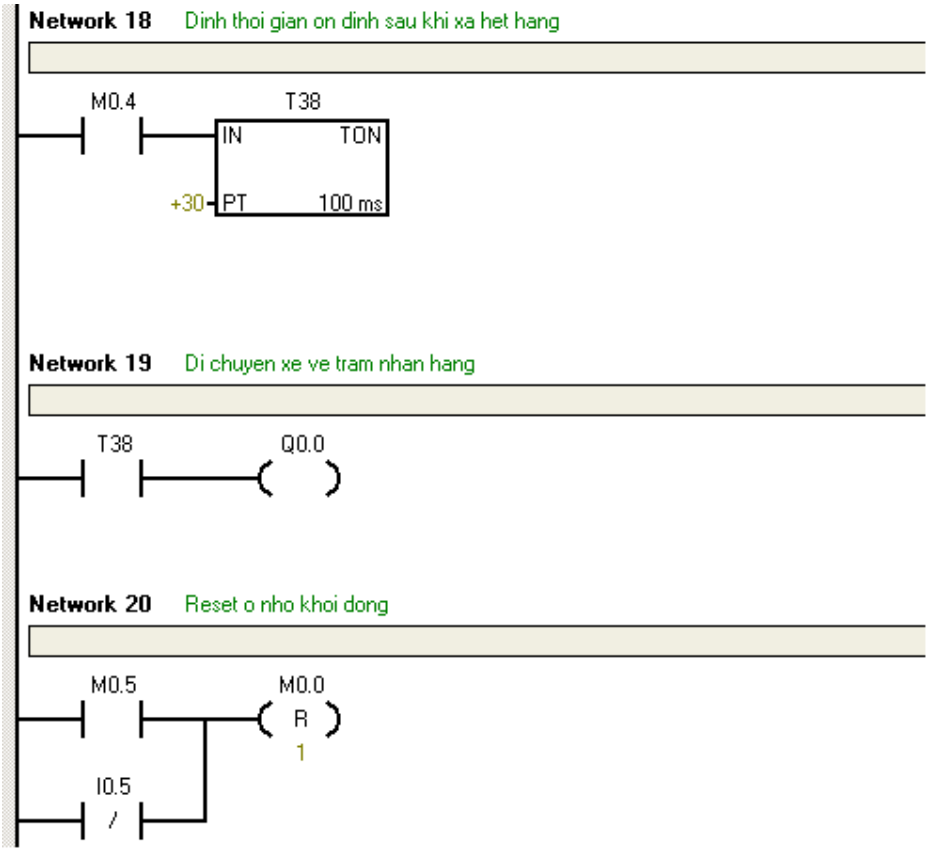


Network 16 Xe chuyen ve tram xa



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
```

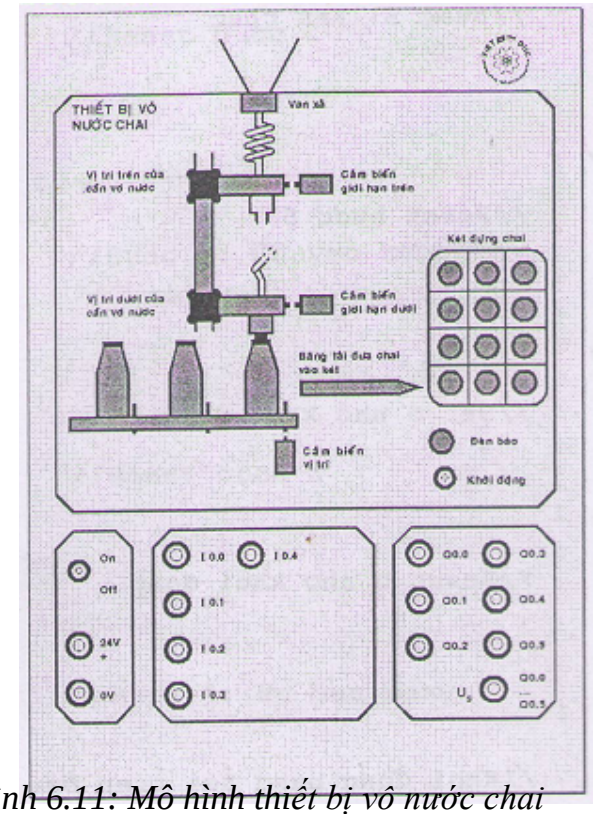
Network 19 Di chuyen xe ve tram nhan hang

```
LD      T38
=       Q0.0
```

Network 20 Reset o nho khoi dong

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

6.4. Thiết bị vô nước chai:



Hình 6.11: Mô hình thiết bị vô nước chai

a. Mô tả:

Mô phỏng một thiết bị vô nước chai có các cảm biến, công tắc hành trình và sự chuyển động bằng các LED.

Ứng dụng trong PLC cơ bản: điều khiển tổ hợp logic

Ứng dụng trong PLC nâng cao: điều khiển trình tự

b. Vận hành mô hình:

Sau khi đã nối dây mô hình với PLC xong, thực hiện viết chương trình theo bài tập đưa ra (có thể tự kiểm tra các ngõ vào/ra bằng phần mềm và thực hiện mô phỏng chương trình.

Hai chai bia trên phải được xem là chai rỗng. Chai ở vị trí thứ 3 được xem như chai đã được đưa đến đúng vị trí. Nước trong chai dâng lên được mô phỏng bằng đèn LED sáng dần. Tùy theo sự sáng dần này mà có thể định thời gian làm đầy chai. Khi chai đã được đổ đầy nước và nếu băng tải vận chuyển chai hoạt động thì chai đầy tự động được chuyển sang băng tải đưa chai vào két. Một tín hiệu sẽ phát ra nếu chai đã đúng vị trí trong két. Khi két đạt đến 12 thì nó không thể tự reset được. Để có thể xóa các LED trong két này phải ấn nút "khởi động". Để

hoạt động thực tế thì khi cần vô nước đến miệng chai phải dừng lại 1s để ổn định.

c. Bảng ký hiệu

Ký hiệu	Địa chỉ	Chú thích
S1	I0.0	Giới hạn trên của cần vô nước, thường đóng
S2	I0.1	Giới hạn dưới của cần vô nước, thường đóng
S3	I0.2	Cảm biến vị trí chai, thường hở
S4	I0.3	Khởi động hệ thống, thường hở
S5	I0.4	Chai đúng vị trí trong két, thường hở
K1	Q0.0	Van xả nước
K2	Q0.1	Hạ cần vô nước xuống
K3	Q0.2	Nâng cần vô nước lên
K4	Q0.3	Băng tải vận chuyển chai rỗng
K5	Q0.4	Đèn báo két đầy

d. Bài tập mẫu:

Thiết bị vô nước chai hoạt động như sau:

Trước khi vận hành thiết bị vô nước chai thì các chai rỗng phải được đặt lên băng tải. Nếu sau đó nút nhấn khởi động (I0.3) được tác động, thì băng tải sẽ vận chuyển chai rỗng với thời gian trì hoãn ban đầu là 1s. Băng tải dừng lại khi có một chai đến cảm biến vị trí (I0.2).

Bây giờ cần vô nước sẽ hạ từ trên xuống, khi đến giới hạn dưới (I0.1) thì dừng lại, sau đó 1s thì van xả sẽ được mở đổ nước vào chai, van xả sẽ được đóng lại khi chai đầy. Thời gian làm đầy kéo dài khoảng 3s.

Sau khi van xả đóng lại 1s thì cần vô nước được nâng lên, đến giới hạn trên (I0.0) thì dừng lại, sau đó 1s thì băng tải vận chuyển chai rỗng lại tiếp tục và quá trình cứ thế lặp lại.

Chai đã đổ đầy nước được đưa sang băng tải đưa chai vào két khi băng tải chai rỗng hoạt động, khi chai đúng vị trí trong két thì có một tín hiệu phát ra (I0.4).

Quá trình được lặp đi lặp lại cho đến khi nào số lượng chai trong két đủ 12 thì đèn báo sáng lên và hệ thống dừng lại. Quá trình mới lại bắt đầu khi nhấn nút khởi động.

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

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

Yêu cầu về đánh giá kết quả học tập:

Áp dụng hình thức kiểm tra tích hợp giữa lý thuyết với thực hành. Các nội dung trọng tâm cần kiểm tra là:

- Giải thuật phù hợp đơn giản, ngắn gọn.
- Nạp trình thành thạo, kiểm tra sửa chữa lỗi khi nạp trình.
- Sử dụng đúng các khối chức năng, các lệnh cơ bản (các phép toán nhị phân các phép toán số của PLC, xử lý tín hiệu analog).
- Sử dụng, khai thác thành thạo phầm mềm mô phỏng. Thực hiện kết nối tốt với PC.
- Lắp ráp thành thạo mạch động lực đảm bảo kỹ thuật và an toàn.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Trọng Thuần, *Điều khiển logic và ứng dựng*, NXB Khoa học kỹ thuật 2006.

[2] Trần Thế San (biên dịch), *Hướng dẫn thiết kế mạch và lập trình PLC*, NXB Đà Nẵng 2005.

[3] Tăng Văn Mùi (biên dịch), *Điều khiển logic lập trình PLC*, NXB Thống kê 2006.

[4] Nguyễn Doãn Phước – Phan Xuân Minh – Vũ Văn Hà, *Tự động hóa với Simatic s7_300*, Nhà xuất bản Khoa học kỹ thuật 2010