MỤC LỤC

LỜI NÓI ĐẦU

Tự động hoá trong công nghiệp và dân dụng ngày càng phát triển. Bộ não trong các hệ thống tự động hoá là các bộ điều khiển lập trình. Việc học tập và tìm hiểu về các bộ điều khiển lập trình cũng như vận hành nó cho tốt đang là nhu cầu cấp thiết cho các sinh viên ngành kỹ thuật.

Để đáp ứng được nhu cầu của sản xuất hiện nay tại các nhà máy, khu công nghiệp… thì một số mảng khi sinh viên ra trường vẫn chưa đáp ứng được; ví dụ như kỹ thuật điều khiển lập trình. Chính vì thế để trang bị cho Sinh viên kiến thức về kỹ thuật lập trình nên tập thể giáo viên Khoa Điện – TĐH hết sức quan tâm, bài giảng "Điều khiển logic lập trình" đã được viết với mong muốn góp phần nhỏ vào việc giảng dạy của giáo viên Tổ Tự Động Hoá và tự học điều khiển lập trình của giáo viên, học sinh, sinh viên quan tâm về PLC họ Simatic S7 – 200 của hãng SIEMENS.

Nội dung của bài giảng bao gồm:

Chương 1: Đại cương về điều khiển lập trình

Chương 2: Cấu trúc và phương thức hoạt động của PLC

Chương 3: Ngôn ngữ lập trình

Chương 4: Xử lý tín hiệu analog

Chương 5: Một số bài toán ứng dụng

Trong khi hoàn chỉnh nội dung bài giảng, các tác giả đã cố gắng rất nhiều để có được nội dung phong phú, cách trình bày thuyết phục, tuy nhiên không tránh khỏi những thiếu sót. Chúng tôi mong muốn tài liệu này ngày càng được hoàn thiện hơn để phục vụ thật tốt các yêu cầu của bạn đọc và phù hợp với xu thế phát triển nhà trường đề ra. Rất mong được những góp ý sửa đổi, bổ sung.

Các ý kiến xin gửi về: Tổ tự động hoá Khoa Điện – TĐH Trường Cao Đẳng Công nghiệp Phúc Yên

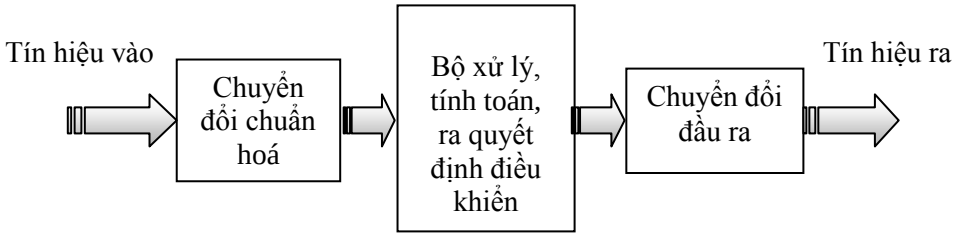
Vĩnh Phúc, tháng 5 năm 2013

Các tác giả

Chương 1
ĐẠI CƯƠNG VỀ ĐIỀU KHIỂN LẬP TRÌNH

1.1. Khái niệm về điều khiển lập trình

Một hệ thống công nghiệp có thể hoạt động được bao gồm các phần chính như là phần thu nhận thông tin, xử lý điều khiển và chấp hành. Tín hiệu ở đầu vào hệ thống sẽ được thu nhận bằng các thiết bị đo (sensor) và được chuẩn hoá tương thích với đầu vào của phần xử lý. Phần xử lý làm nhiệm vụ nhận các thông tin cần thiết, tính toán, xử lý, ra quyết định điều khiển để tác động tới đầu ra điều khiển thông qua chuyển đổi đầu ra. Tín hiệu ra từ phần xử lý có dạng số nên cần phải khuyếch đại, hoặc chuyển đổi dạng tín hiệu để điều khiển các thiết bị phần chấp hành của hệ thống. Một hệ thống như vậy được coi là một hệ thống đo và điều khiển cơ bản.



Hình 1-1: Cấu trúc hệ thống đo - điều khiển cơ bản

Để cho hệ thống hoạt động theo mong muốn, chúng ta cần phải *lập trình* cho khâu xử lý tính toán, hay là viết các chương trình xử lý, tính toán cho hệ thống theo một quy luật logic nào đó. Việc thực hiện lập trình cho hệ thống sẽ được thông qua một vài phần mềm được tích hợp các quy luật lập trình, hay gọi là ngôn ngữ lập trình để tạo nên các chương trình hoạt động cho hệ thống.

Thuật ngữ: "Điều khiển lập trình" có nghĩa là thực hiện việc lập trình, viết các chương trình để hệ thống làm việc theo yêu cầu của con người đặt ra. Trải qua các giai đoạn phát triển của điều khiển công nghiệp, cho đến ngày nay, con người đã tích hợp được những bộ điều khiển (controller) khá hoàn chỉnh. Việc lập trình cho hệ thống được thực hiện một cách dễ dàng thông qua giao diện người – máy HMI (Human Machine Interface). Dòng sản phẩm có tính năng điều khiển – lập trình có tên là PLC (Programmable Logic Controller).

Như vậy, *PLC là bộ điều khiển có khả năng thích ứng với nhiều chương trình khác nhau do người lập trình tải vào bộ nhớ*. PLC được tích hợp trong đó phần nhận tín hiệu vào, phần chuyển đổi và truyền tín hiệu, lưu vào bộ nhớ, xử lý tính toán và ra quyết định điều khiển thông qua chuyển đổi tín hiệu ra và đầu ra của bộ điều khiển.

Hệ thống điều khiển có lập trình bao gồm giao diện người máy (hoặc máy tính, thiết bị lập trình), các mô đun chuyển đổi và truyền thông, PLC, các thiết bị phụ trợ, đo lường và chấp hành. Trong đó, PLC là khối chức năng đặc biệt, chứa các tiếp điểm vào/ra nối tới phần đệm cổng vào/ra. Phần quan trọng của PLC là lưu các thuật toán tính toán điều khiển, lưu trữ chương trình… Đó cũng là phần mà nội dung của cuốn sách quan tâm nhiều nhất.

1.2. Lịch sử phát triển của PLC

Trước khi có PLC đã có những bộ điều khiển tự động bằng các mạch rơle-công tắc tơ hoặc các mạch rơ le số/tương tự không tiếp điểm. Các bộ điều khiển này ngày nay được gọi là các bộ điều khiển cứng. Các bộ điều khiển cứng khi cần phải thay đổi hoặc mở rộng số lượng thiết bị, tiếp điểm trong hệ thống sẽ khó thực hiện vì phải thay đổi mạch cứng. Do đó người ta mong muốn chế tạo được các bộ điều khiển linh hoạt hơn.

Năm 1969, hãng sản xuất ôtô GM đề xuất thiết kế các bộ điều khiển có khả năng thích ứng với nhiều chương trình điều khiển khác nhau với các đặc điểm:

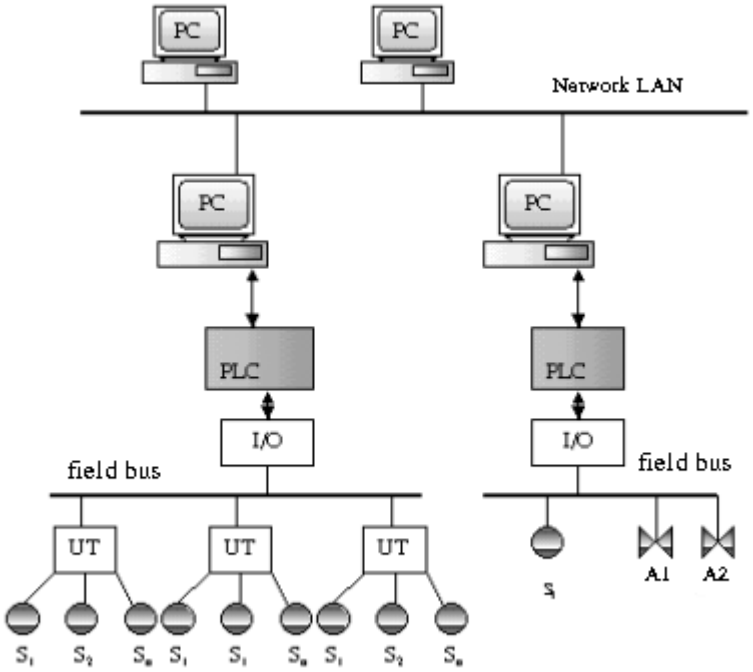
1. Dễ dàng thay đổi được chương trình điều khiển
2. Đơn giản cho việc thay thế và sửa chữa.
3. Độ tin cậy cao so với các bộ điều khiển cứng truyền thống.
4. Nhỏ gọn hơn so với các bộ điều khiển thuyền thống.
5. Dữ liệu vào/ra có thể được truyền tới phần điều khiển trung tâm.
6. Giá thành tốt hơn các bộ điều khiển rơ le.
7. Bộ điều khiển có tính năng mở.
8. Độ bền công nghiệp cao.

Do tính thích ứng với nhiều chương trình điều khiển, việc thay đổi chương trình dễ dàng và không đòi hỏi những chuyên gia lập trình và điều khiển có trình độ chuyên môn cao nên bộ điều khiển kiểu này ngày càng hấp dẫn giới điều khiển kỹ thuật, nó được phát triển và ứng dụng vào nhiều ngành công nghiệp và dân dụng.

1.3. Các hệ thống điều khiển công nghiệp

1.3.1. Hệ thống thu thập số liệu, giám sát và điều khiển (Supervisory Control And Data Aquirition - SCADA)

Hệ thống điều khiển kiểu thu thập, giám sát và điều khiển SCADA ra đời từ những năm 1980, song song với việc ra đời các thiết bị logic lập trình được (PLC). SCADA chủ yếu sử dụng PLC để điều khiển hệ thống. SCADA thích hợp cho việc quản lý và điều khiển hệ thống sản xuất cỡ nhỏ với cấu trúc cơ bản như sau:



Hình 1-2: Cấu trúc hệ thống SCADA

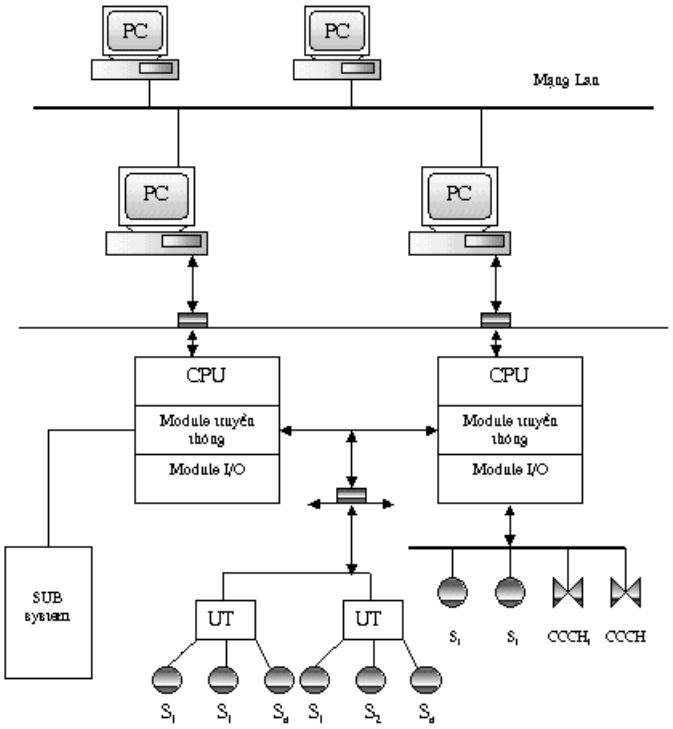
Trong đó:

- PC: Professional Computer (Máy tính chuyên dụng).
- LAN: Local Area Network (Mạng máy tính nội bộ).
- PLC: Programmable Logic Controller (Bộ điều khiển logic lập trình được).
- I/O: Input/Output (Thiết bị vào/ra).
- UT: Unit Terminator (Thiết bị đầu cuối – hoặc RTU-Remote Terminator Unit).
- S_i: Sensor (Thiết bị đo lường).
- A_i: Actuator (Cơ cấu chấp hành: Động cơ, van, rơ le, ...).
- Field bus: bus trường.

Trong hệ thống này, các bộ PLC thu thập số liệu, xử lý kết quả đo và đưa ra quyết định điều khiển, đồng thời gửi kết quả đo về máy tính trung tâm. Máy tính trung tâm có nhiệm vụ hiển thị kết quả đo và cho phép vận hành hệ thống với yêu cầu từ máy tính. Người điều khiển thông qua bàn phím và chuột có thể điều khiển hệ thống, máy tính truyền lệnh điều khiển xuống PLC thông qua các module vào ra (I/O), hệ thống thực hiện các công đoạn cần thiết để điều khiển quá trình sản xuất. Hệ thống kiểu này giá thành rẻ, thích hợp cho các hệ thống vừa và nhỏ. Tuy nhiên hạn chế ở chỗ: khó thực hiện cho hệ thống lớn; không có phần mềm chuyên dụng cho dự phòng; khả năng cho phép mở rộng các điểm đo bị hạn chế.

1.3.2. Hệ thống điều khiển phân tán (DCS)

Hệ DCS (Distributed Control System) khắc phục được các nhược điểm của hệ SCADA trên, đặc biệt là việc xử lý tập trung thông tin ở trung tâm điều khiển, do đó lượng thông tin truyền đi và kênh truyền sẽ rất lớn đòi hỏi phần xử lý trung tâm phải có dung lượng cùng với tốc độ cao làm cho toàn hệ thống cồng kềnh phức tạp, chi phí lớn. Cấu trúc của hệ DCS về cơ bản được bố trí như *hình 1.3* sau đây.



Hình 1-3: Cấu trúc cơ bản của hệ DCS

Phân cấp của hệ thống như sau:

- Cấp tiếp xúc gần nhất với đối tượng điều khiển: Gồm các cảm biến, Module chuẩn hoá tín hiệu, các van điều khiển, các Module I/O, các Module truyền thông và các khối xử lý trung tâm của từng nhóm tín hiệu và thường gọi là các khối xử lý phân tán. Tập hợp của nhóm các thiết bị đó gọi là các thiết bị hiện trường.
- Cấp điều khiển cục bộ (Local Control): Gồm các Module I/O, PLC, PC công nghiệp...
- Cấp điều khiển giám sát: Gồm các máy tính với giao diện quan sát lớn, các bảng hiển thị thông số lớn, các thiết bị giám sát khác và máy in. Cấp này có nhiệm vụ giám sát, điều khiển, lưu giữ, in ấn, hiển thị tức thời (động) các sơ đồ công nghệ và các thông số chính của quá trình sản xuất...
- Cấp quản lý: Gồm các máy tính được nối mạng, làm nhiệm vụ thống kê số liệu sản xuất, lập bảng biểu, lưu trữ, tính toán tối ưu quá trình sản xuất...

Hệ thống có ưu điểm như sau:

- Giao diện người dùng và các thông tin hiển thị rõ ràng.
- Có chức năng dự phòng linh hoạt.
- Có thể thay đổi quy trình công nghệ bằng phần mềm tương đối dễ.
- Tính năng tác động nhanh được cải thiện.
- Độ ổn định khá cao.

- Thuận tiện cho việc kết nối với các hệ thống khác và dễ sử dụng.

Tuy nhiên, nhược điểm của hệ thống:

- Giá thành đắt.

- Yêu cầu kỹ thuật viên phải có trình độ cao, hiểu biết về công nghệ DCS, PC, Controller, profibus, ...

1.3.3 Các hệ thống điển hình khác

- *Hệ thống tích hợp:* Từ năm 1998 đến nay trên thị trường công nghệ quốc tế và trong nước đã dần dần triển khai hệ thống điều khiển công nghiệp kiểu tích hợp với tên gọi là Hệ thống thông tin tích hợp (Integrated Information Systems – IIS). Hệ này có cấu trúc gần tương tự với kiểu DCS nhưng được tích hợp nhiều chức năng hơn. Ngoài chức năng điều khiển phân tán và tính năng mở còn có chương trình điều khiển theo quy trình công nghệ đảm bảo sản xuất tối ưu. Trên hệ thống còn tích hợp các chương trình tổ chức, lập kế hoạch sản xuất, tính toán lỗ lãi, marketing, thương mại điện tử,... nhằm đem lại lợi nhuận cao cho sản xuất.

- *Các ứng dụng khác:* Ngoài những ứng dụng của PLC trong các hệ thống điều khiển công nghiệp với quy mô lớn mà chúng ta đã xét, PLC còn có thể ứng dụng vào các công đoạn tự động hoá từng phần, từng mảng công việc khác nhau tuỳ từng điều kiện cụ thể về tính chất công việc, kinh tế,... Chẳng hạn, PLC ứng dụng điều khiển hoạt động cửa tự động, tự động hoá toà nhà, cầu thang máy, trạm trộn bê tông, điều khiển gara tự động, điều khiển robot, điều khiển đèn đường giao thông, điều khiển hệ thống báo động, ...

1.4. Ưu nhược điểm của PLC

Các điều kiện đưa ra để chế tạo PLC chính là các đặc điểm mang tính ưu việt của PLC so với các bộ điều khiển truyền thống, trong đó ưu điểm lớn nhất là khả năng thích ứng với các chương trình điều khiển khác nhau của PLC. Trong PLC khi thay đổi chương trình điều khiển, do dùng các vi mạch để xử lý thông tin cho nên các ghép nối cần thiết trong quá trình lập chương trình điều khiển không phải là các ghép nối cơ học mà là các ghép nối logic được người lập trình tạo ra bằng phần mềm (Software) và được cài đặt vào bộ nhớ.

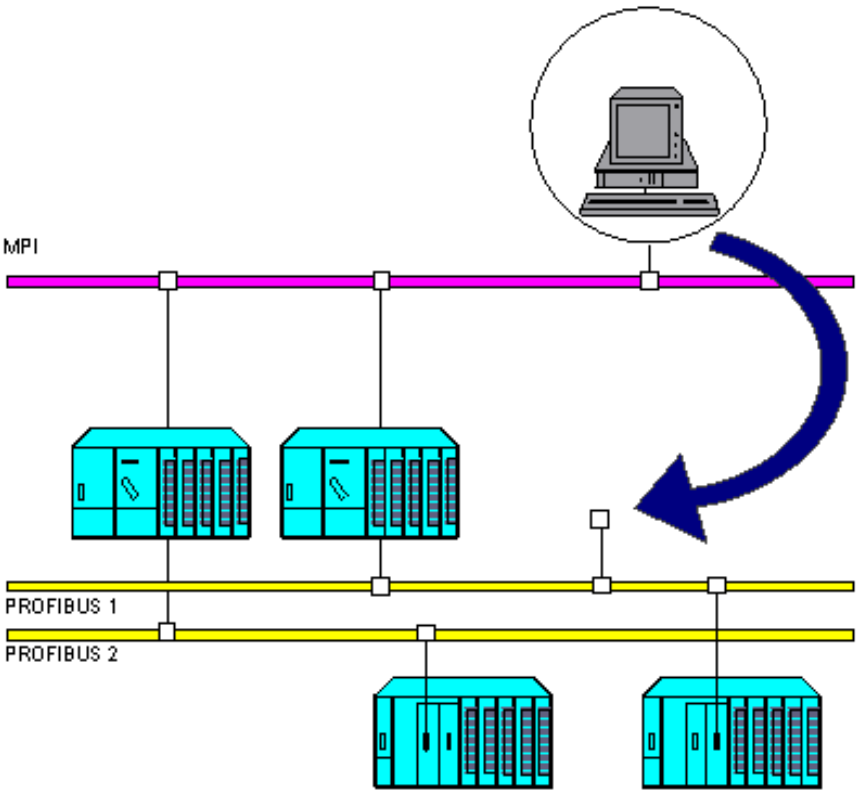
PLC có tốc độ xử lý cao, thường xử lý một lệnh trong khoảng thời gian 0,64μs. Nó còn là thiết bị tiêu tốn ít năng lượng so với các bộ điều khiển truyền thống. Nó nhỏ, gọn, trọng lượng nhẹ, dễ dàng lắp đặt trong các tủ điều khiển, dễ dàng ghép nối với các thiết bị khác của hệ thống.

Sử dụng PLC trong điều khiển tự động chúng ta dễ dàng thiết lập được sự trao đổi thông tin với các PLC khác thông qua các mạng như Profibus DP, LAN (Local Area Network), Asi, Profinet.

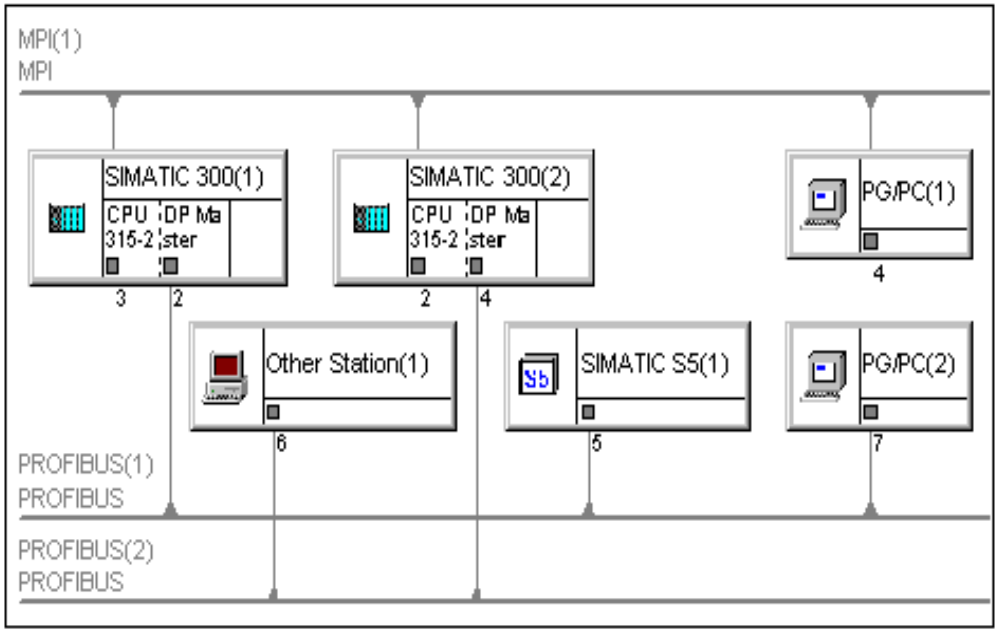
Tuy nhiên, hiện nay do chưa được chuẩn hoá trong phạm vị quốc tế nên mỗi hãng sản xuất PLC lại đưa ra một ngôn ngữ lập trình riêng dẫn đến thiếu tính thống nhất toàn cục.

Ngày nay, PLC được sử dụng rộng rãi trong các hệ thống điều khiển nhà máy, dây chuyền công nghệ sản xuất. Trong đó, với các hệ thống có quy mô lớn, người ta sử dụng nhiều PLC ghép nối với nhau và được quản lý giám sát bằng máy tính hoặc giám sát trực tiếp bằng màn hình tại các trạm. Đó là ưu điểm vượt trội về công nghệ điều khiển phân

tán. *Hình 1.4* và *hình 1.5* mô tả khả năng nối mạng công nghiệp của bộ điều khiển lập trình.



Hình 1-4: Khả năng nối mạng Profibus của PLC



Hình 1-5: Khả năng quản lý nhiều trạm của PLC

1.5. Phạm vi ứng dụng PLC

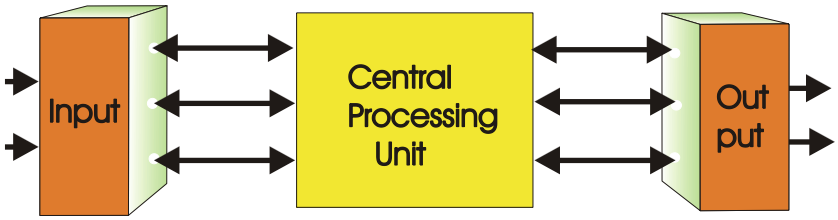
Hiện nay PLC đã được ứng dụng thành công trong nhiều lĩnh vực sản xuất cả trong công nghiệp và dân dụng. Từ những ứng dụng để điều khiển các hệ thống đơn giản, chỉ có chức năng đóng mở (ON/OFF) thông thường đến các ứng dụng cho các lĩnh vực phức tạp, đòi hỏi tính chính xác cao, ứng dụng các thuật toán trong quá trình sản xuất. Các lĩnh vực tiêu biểu ứng dụng PLC hiện nay bao gồm:

- Hóa học và dầu khí: Định áp suất (dầu), bơm dầu, điều khiển hệ thống ống dẫn, cân động trong ngành hóa …
- Chế tạo máy và sản xuất: Tự động hoá trong chế tạo máy, cân động, quá trình lắp đặt máy, điều khiển nhiệt độ lò kim loại…
- Bột giấy, giấy, xử lý giấy. Điều khiển máy băm, quá trình ủ bột, cán, gia nhiệt …
- Thủy tinh và phim ảnh: quá trình đóng gói, thử nghiệm vật liệu, cân đong, các khâu hoàn tất sản phẩm, đo cắt giấy .
- Thực phẩm, rượu bia, thuốc lá: đếm, kiểm tra sản phẩm, kiểm soát quá trình sản xuất, bơm (bia, nước trái cây …), cân đong, đóng gói, hòa trộn …
- Kim loại: Điều khiển quá trình cán, cuốn (thép), qui trình sản xuất, kiểm tra chất lượng sản phẩm.
- Năng lượng: Điều khiển nguyên liệu (cho quá trình đốt, xử lý trong các turbin …), các trạm cần hoạt động tuần tự khai thác vật liệu một cách tự động (than, gỗ, dầu mỏ).

Chương 2
CẤU TRÚC VÀ PHƯƠNG THỨC HOẠT ĐỘNG CỦA PLC

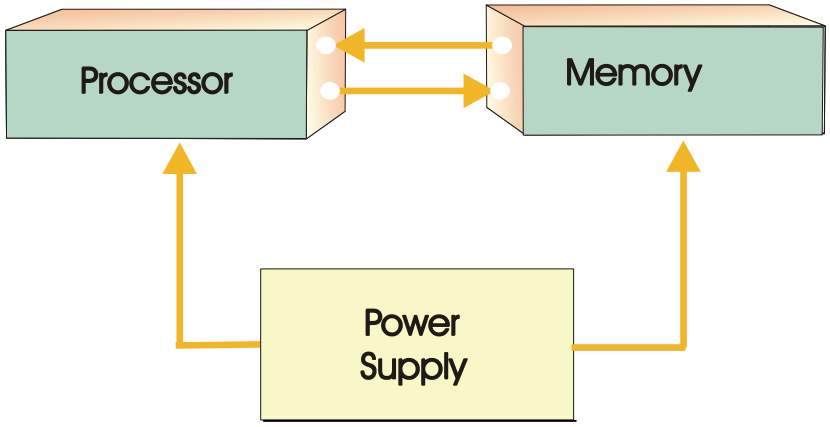
2.1. Cấu trúc của một PLC

Một hệ thống điều khiển lập trình cơ bản phải gồm có hai phần: Khối xử lý trung tâm (CPU: Central Processing Unit : CPU) và hệ thống giao tiếp vào/ra (I/0).



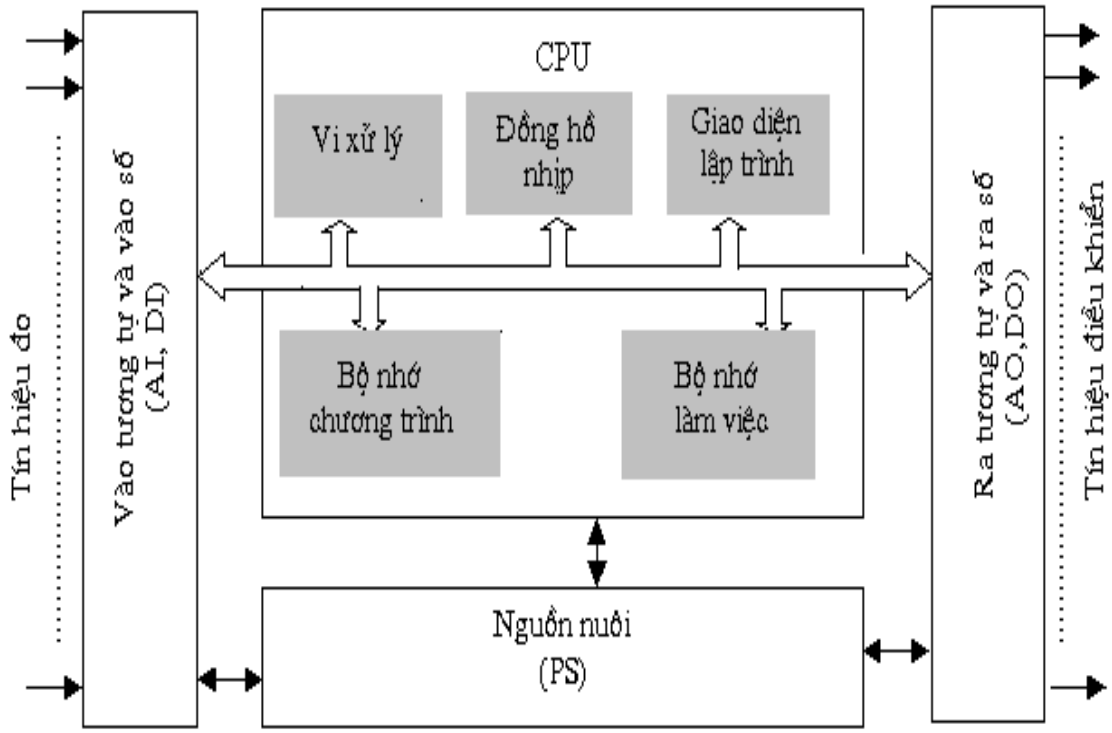
Hình 2-1 : Sơ đồ khối của hệ thống điều khiển lập trình

Khối xử lý trung tâm (CPU) gồm ba phần : Bộ xử lý, Hệ thống bộ nhớ và Hệ thống nguồn cung cấp. Hình 2-2 mô tả 3 thành phần của một CPU.



Hình 2-2 : Sơ đồ khối tổng quát của CPU

2.2. Các khối của PLC



Hình 2-3 : Các thành phần chức năng chính của một **PLC**

2.2.1. Đơn Vị Xử Lý Trung Tâm (CPU - Central Processing Unit)

CPU điều khiển các hoạt động bên trong PLC. Bộ xử lý sẽ đọc và kiểm tra chương trình được chứa trong bộ nhớ, sau đó sẽ thực hiện thứ tự từng lệnh trong chương trình, sẽ đóng hay ngắt các đầu ra. Các trạng thái ngõ ra ấy được phát tới các thiết bị liên kết để thực thi. Và toàn bộ các hoạt động thực thi đó đều phụ thuộc vào chương trình điều khiển được lưu giữ trong bộ nhớ.

2.2.2. Hệ Thống Bus

Hệ thống Bus là tuyến dùng để truyền tín hiệu, hệ thống gồm nhiều đường tín hiệu song song

Address Bus: Bus địa chỉ dùng để truyền địa chỉ đến các Modul khác nhau.

Data Bus: Bus dùng để truyền dữ liệu.

Control Bus: Bus điều khiển dùng để truyền các tín hiệu định thì và điều khiển đồng bộ các hoạt động trong PLC.

Trong PLC các số liệu được trao đổi giữa bộ vi xử lý và các modul vào ra thông qua Data Bus. Address Bus và Data Bus gồm 8 đường, ở cùng thời điểm cho phép truyền 8 bit của 1 byte một cách đồng thời hay song song.

Nếu một modul đầu vào nhận được địa chỉ của nó trên Address Bus, nó sẽ chuyển tất cả trạng thái đầu vào của nó vào Data Bus. Nếu một địa chỉ byte của 8 đầu ra xuất hiện trên Address Bus, modul đầu ra tương ứng sẽ nhận được dữ liệu từ Data bus. Control Bus sẽ chuyển các tín hiệu điều khiển vào theo dõi chu trình hoạt động của PLC.

Các địa chỉ và số liệu được chuyển lên các Bus tương ứng trong một thời gian hạn chế.

Hệ thống Bus sẽ làm nhiệm vụ trao đổi thông tin giữa CPU, bộ nhớ và I/O . Bên cạch đó, CPU được cung cấp một xung Clock có tần số từ 1÷8 MHZ. Xung này quyết định tốc độ hoạt động của PLC và cung cấp các yếu tố về định thời, đồng hồ của hệ thống.

2.2.3. Bộ Nhớ

PLC thường yêu cầu bộ nhớ trong các trường hợp:

- Làm bộ định thời cho các kênh trạng thái I/O.
- Làm bộ đệm trạng thái các chức năng trong PLC như định thời, đếm, ghi các Relay.

Mỗi lệnh của chương trình có một vị trí riêng trong bộ nhớ, tất cả mọi vị trí trong bộ nhớ đều được đánh số, những số này chính là địa chỉ trong bộ nhớ .

Địa chỉ của từng ô nhớ sẽ được trỏ đến bởi một bộ đếm địa chỉ ở bên trong bộ vi xử lý. Bộ vi xử lý sẽ giá trị trong bộ đếm này lên một trước khi xử lý lệnh tiếp theo. Với một địa chỉ mới, nội dung của ô nhớ tương ứng sẽ xuất hiện ở đầu ra, quá trình này được gọi là quá trình đọc .

Bộ nhớ bên trong PLC được tạo bởi các vi mạch bán dẫn, mỗi vi mạch này có khả năng chứa 2000 ÷ 16000 dòng lệnh, tuỳ theo loại vi mạch. Trong PLC các bộ nhớ như RAM, EPROM đều được sử dụng.

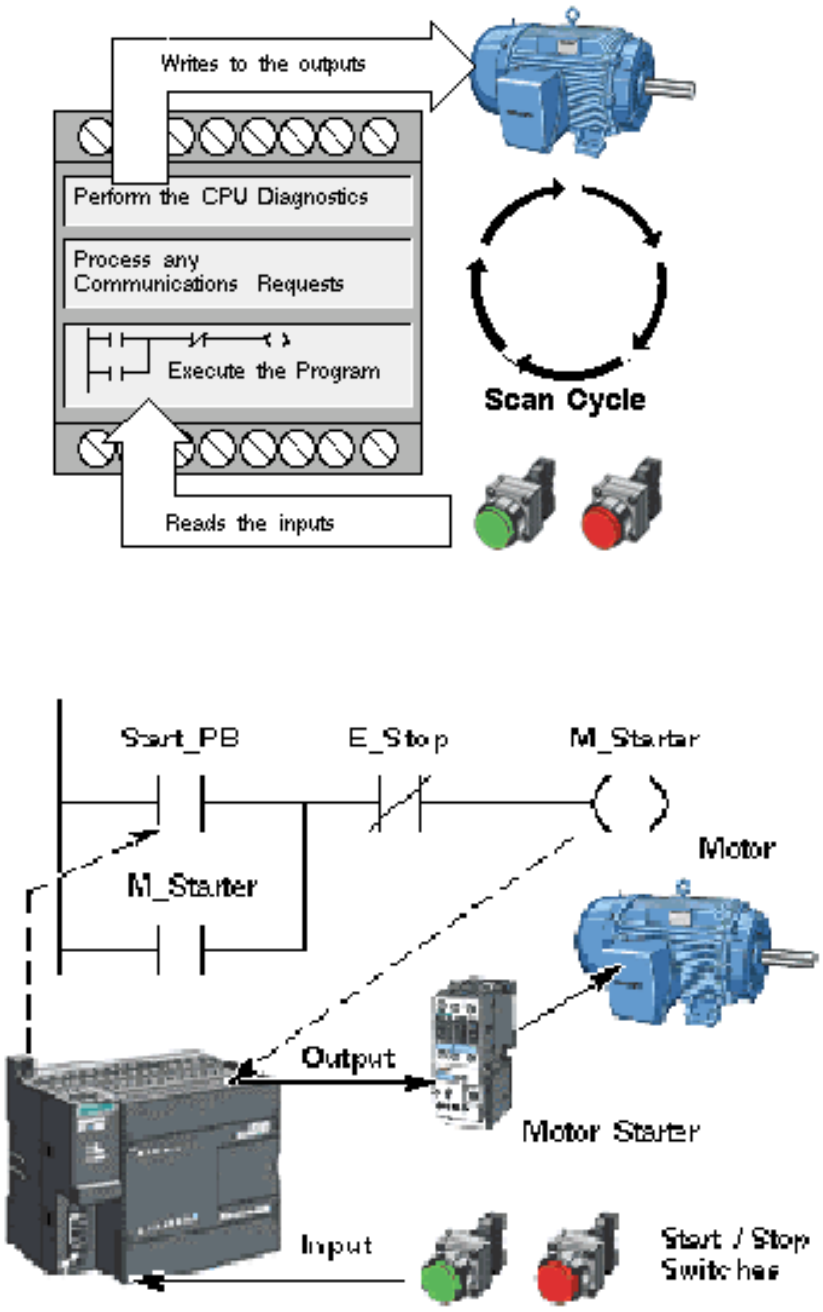
- RAM (Random Access Memory) có thể nạp chương trình, thay đổi hay xoá bỏ nội dung bất kỳ lúc nào. Nội dung của RAM sẽ bị mất nếu nguồn điện nuôi bị mất . Để tránh tình trạng này các PLC đều được trang bị một pin khô, có khả năng cung cấp năng lượng dự trữ cho RAM từ vài tháng đến vài năm. Trong thực tế RAM được dùng để khởi tạo và kiểm tra chương trình. Khuynh hướng hiện nay dùng CMOSRAM nhờ khả năng tiêu thụ năng lượng thấp và tuổi thọ lớn.
- EPROM (Electrically Programmable Read Only Memory) là bộ nhớ mà người sử dụng bình thường chỉ có thể đọc chứ không ghi nội dung vào được. Nội dung của EPROM không bị mất khi mất nguồn, nó được gắn sẵn trong máy, đã được nhà sản xuất nạp và chứa hệ điều hành sẵn. Nếu người sử dụng không muốn mở rộng bộ nhớ thì chỉ dùng EPROM gắn bên trong PLC. Trên PG (Programmer) có sẵn chỗ ghi và xoá EPROM.

Môi trường ghi dữ liệu thứ ba là đĩa cứng hoặc đĩa mềm, được sử dụng trong máy lập trình. Đĩa cứng hoặc đĩa mềm có dung lượng lớn nên thường được dùng để lưu những chương trình lớn trong một thời gian dài.

Kích thước bộ nhớ:

- Các PLC loại nhỏ có thể chứa từ 300 ÷1000 dòng lệnh tuỳ vào công nghệ chế tạo.
- Các PLC loại lớn có kích thước từ 1K ÷ 16K, có khả năng chứa từ 2000 ÷16000 dòng lệnh.

Ngoài ra còn cho phép gắn thêm bộ nhớ mở rộng như RAM, EPROM.



Hình 2-4: Minh họa hoạt động của PLC khi ghép nối với thiết bị ngoại vi.

2.2.4. Các ngõ vào ra I/O:

Các đường tín hiệu từ bộ cảm biến được nối vào các module vào (các đầu vào của PLC), các cơ cấu chấp hành được nối với các module ra (các đầu ra của PLC).

Hầu hết các PLC có điện áp hoạt động bên trong là 5V, tín hiệu xử lý là 12/24VDC hoặc 100/240VAC.

Mỗi đơn vị I/O có duy nhất một địa chỉ, các hiển thị trạng thái của các kênh I / O được cung cấp bởi các đèn LED trên PLC, điều này làm cho việc kiểm tra hoạt động nhập xuất trở nên dễ dàng và đơn giản.

Bộ xử lý đọc và xác định các trạng thái đầu vào (ON, OFF) để thực hiện việc đóng hay ngắt mạch ở đầu ra.

2.2.5. Bộ cung cấp nguồn (Power Supply, PS)

PS có vai trò biến đổi và ổn định nguồn nuôi (thông thường 5V cho CPU) và các thành phần chức năng khác từ một nguồn xoay chiều (110V, 220V,...) hoặc một chiều (12V, 24V,...).

Bên cạnh các thành phần chính nêu trên, một hệ thống PLC có thể có các thành phần chức năng khác như ghép nối mở rộng, điều khiển chuyên dụng và xử lý truyền thông.

2.3. Các ngõ vào ra và cách kết nối

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 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.

2.3.1. 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:

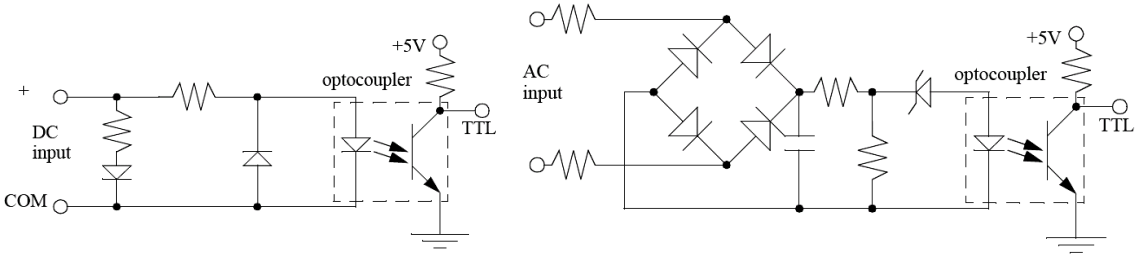
Xoay chiều: 20...29 VAC , f = 47...63 Hz;
85...264 VAC, f = 47...63 Hz
Một chiều: 20,4 ... 28,8 VDC

2.3.2. 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 4mA
79...135 VAC, f = 47...63 Hz; dòng cần thiết nhỏ nhất 4mA
Một chiều: 15 ... 30 VDC; dòng cần thiết nhỏ nhất 4mA

Sơ đồ mạch điện bên trong của các ngõ vào được cho như hình sau



a) b)

Hình 2-5: 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:

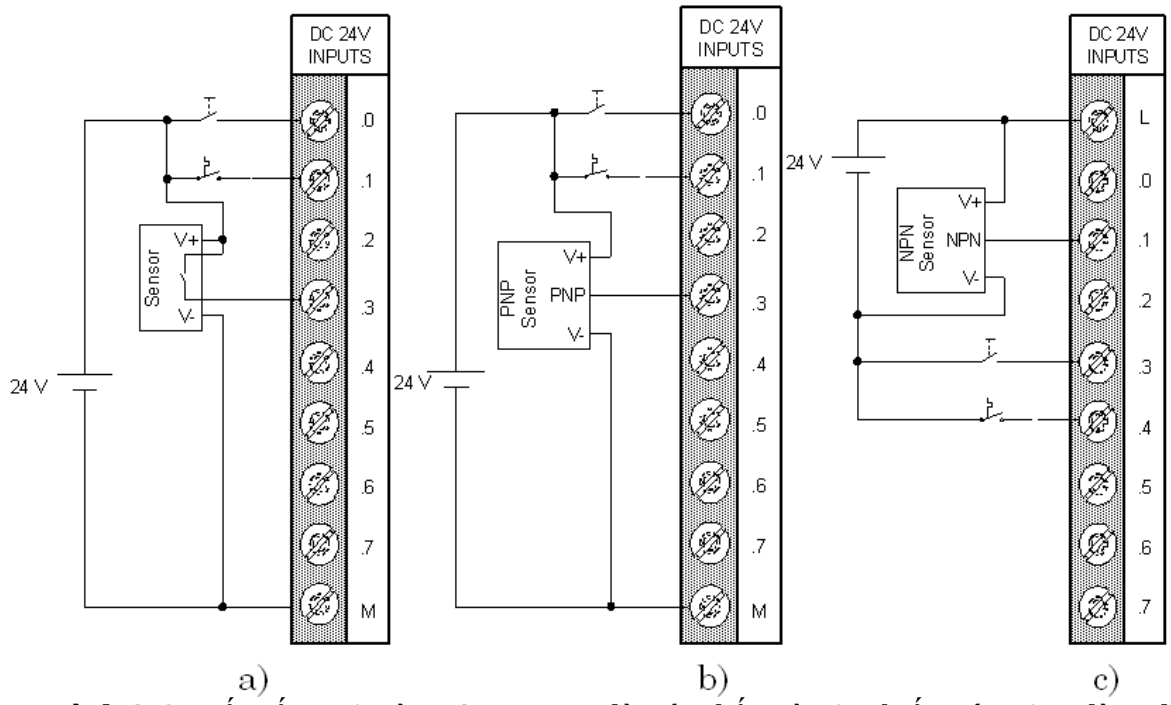
- Điện áp DC thường thấp do đó an toàn hơn.
- Đáp ứng ngõ vào DC rất nhanh.
- Điện áp DC có thể được kết nối với nhiều phần tử hệ thống điện.

+ 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.

Trong ví dụ hình 2.6 a có 3 ngõ vào, một là nút nhấn thường hở, hai là tiếp điểm của rơ le nhiệt, và ba là cảm biến tiếp cận với ngõ ra là rơle. 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 2-6: Kết nối ngõ vào với ngoại vi là nút nhấn và cảm biến có ngõ ra là rơ le, PNP và NPN

2.3.3. Kết nối các ngõ ra số với ngoại vi

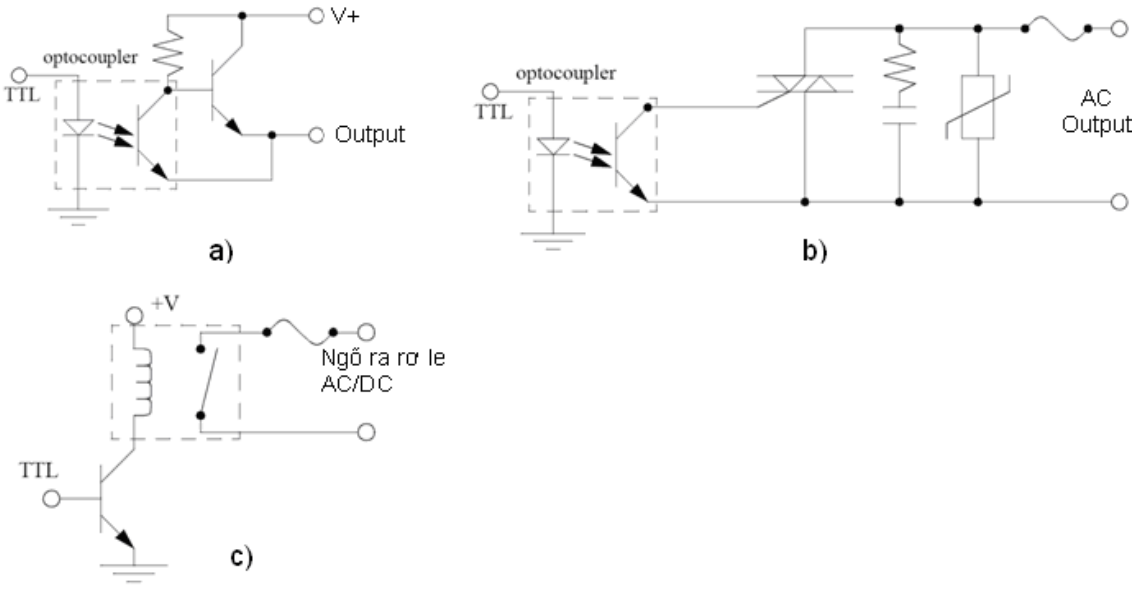
Các ngõ ra của PLC có thể được chế tạo là một khối riêng, hoặc kết hợp với các ngõ ra chung trong một khối hoặc được tích hợp trên khối CPU. Trong trường hợp nào cũng vậy, các ngõ ra cũng phải được cung cấp nguồn riêng với cấp điện áp tùy thuộc vào loại ngõ ra. Cần lưu ý trong một khối ra cũng như các 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 rơ le, transistor hoặc triac. Rơ le là ngõ ra linh hoạt nhất. Chúng có thể là ngõ ra AC và DC. Tuy nhiên đáp ứng của ngõ ra rơ le 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.

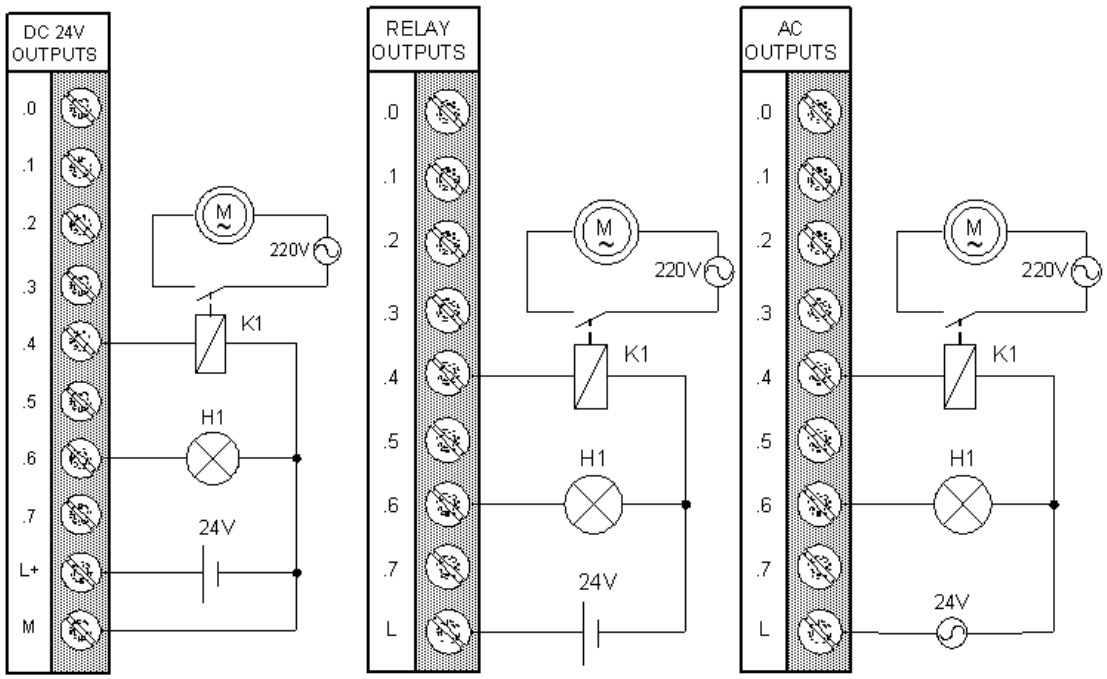
Sơ đồ mạch điện bên trong của các ngõ ra được cho như hình 2.7



Hình 2.7 : *Mạch điện bên trong của các loại ngõ ra khác nhau.*
a) *Ngõ ra Transistor ; b) Ngõ ra triac ; c) Ngõ ra rơ le*

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.

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 rơ le, chuông báo . . .. Hình 2.8 minh họa cách kết nối dây các ngõ ra PLC với các cơ cấu chấp hành.



Hình 2- 8: Cách kết nối dây các ngõ ra PLC với các cơ cấu chấp hành.

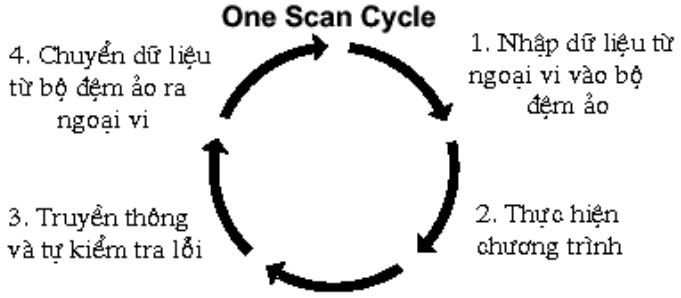
Hình 2.8a là một ví dụ cho các khối ra sử dụng 24VDC với mass chung. Tiêu biểu cho loại này là ngõ ra transistor. Trong ví dụ này các ngõ ra được kết nối với tải công suất nhỏ là đèn báo và cuộn dây rơ le. Quan sát mạch kết nối này, đèn báo sử dụng nguồn cung cấp là 24VDC. Nếu ngõ ra .6 ở mức logic "1" (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 rơ le có điện, làm tiếp điểm của nó đóng lại cung cấp điện 220 VAC cho động cơ.

Hình 2.8b là một ví dụ ngõ ra rơ le sử dụng nguồn cấp là 24 VDC, và hình 2.8c là ví dụ ngõ ra triac sử dụng nguồn xoay chiều 24 Vac.

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.

2.4. Xử lý chương trình

PLC thực hiện xử lý chương trình theo chu trình lặp. Mỗi vòng lặp được gọi là vòng quét (scan). Mỗi vòng quét được bắt đầu bằng giai đoạn đọc dữ liệu từ các cổng vào vùng bộ đệm ảo, tiếp theo là giai đoạn thực hiện chương trình. Trong từng vòng quét, chương trình được thực hiện bằng lệnh đầu tiên và kết thúc bằng lệnh kết thúc (MEND). Sau giai đoạn thực hiện chương trình là giai đoạn truyền thông nội bộ và kiểm lỗi. Vòng quét được kết thúc bằng giai đoạn chuyển các nội dung của bộ đệm ảo tới các cổng ra



2.4.1. Nhập dữ liệu vào

Ngay tại đầu vòng quét, các dữ liệu tại cổng vào số đã được CPU chuyển tới bộ đệm vào số (process image input register). Như vậy tại thời điểm thực hiện lệnh vào thông thường lệnh không làm việc trực tiếp với cổng vào số mà chỉ thông qua bộ đệm ảo của cổng trong vùng nhớ tham số.

CPU không thể tự động truy nhập dữ liệu tại các cổng vào tương tự, mà truy nhập trực tiếp bằng lệnh vào của chương trình.

2.4.2. Thực hiện chương trình

Trong mỗi vòng quét, chương trình được thực hiện bằng lệnh đầu tiên và kết thúc bằng lệnh cuối cùng. Khi gặp lệnh vào/ra ngay lập tức thì hệ thống sẽ cho dừng mọi công việc khác ngay cả chương trình xử lý ngắt, để thực hiện lệnh này một cách trực tiếp với cổng vào ra.

Nếu sử dụng các chế độ ngắt, chương trình con tương ứng với từng tín hiệu ngắt được soạn thảo và cài đặt như một bộ phận của chương trình. Chương trình xử lý ngắt chỉ được thực hiện trong vòng quét khi xuất hiện tín hiệu báo ngắt và có thể xảy ra ở bất cứ thời điểm nào trong vòng quét.

2.4.3. Truyền thông và kiểm tra lỗi

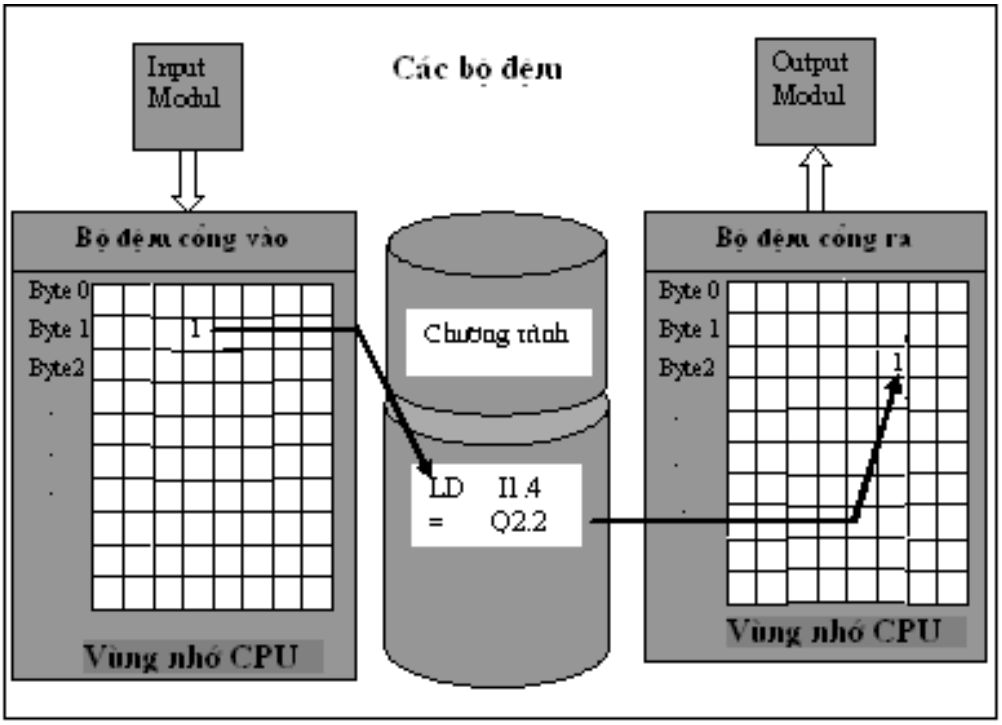
Việc truyền thông giữa bộ đệm ảo với ngoại vi trong các giai đoạn 1 và 4 do CPU quản lý.

Trong suốt giai đoạn này của mỗi vòng quét, CPU sẽ kiểm tra chương trình, bộ nhớ chương trình cũng như trạng thái của các mô đun vào ra

2.4.4. Chuyển dữ liệu ra ngoài

Cuối mỗi vòng quét nội dung của bộ đệm ra số (process image output register) lại được CPU chuyển tới cổng số. Tương tự CPU không làm việc trực tiếp với cổng ra số mà cũng chỉ thông qua bộ đệm ảo, nhưng việc truy nhập cổng ra tương tự lại được CPU thực hiện trực tiếp.

Ví dụ: Minh hoạ chu kỳ quét của CPU



CPU sẽ kiểm tra tình trạng cổng vào và cổng ra trong mỗi chu kỳ. Những dữ liệu nhị phân của mô đun vào/ra được lưu trữ vào các cùng nhớ riêng biệt của các bộ đệm ảo của cổng vào/ra.

Bộ đệm cổng vào: nằm trong vùng nhớ của CPU. Nó lưu trữ tình trạng tín hiệu của tất cả các cổng vào.

Bộ đệm cổng ra: chứa tất cả các giá trị của cổng ra là kết quả của quá trình xử lý chương trình. Chúng được chuyển tới cổng ra vào cuối chu trình.

Đầu tiên CPU sẽ kiểm tra tình trạng những tín hiệu ở đầu vào và cập nhật bộ đệm cổng vào. Sau đó thực hiện chương trình theo từng lệnh. Và cuối cùng là ghi các giá trị từ bộ đệm cổng ra đến các mô đun cổng ra.

2.5. Phương pháp lập trình PLC S7-200

PLC S7-200 biểu diễn một mạch logic cứng bằng một dãy các lệnh lập trình. Chương trình bao gồm một dãy các tập lệnh. PLC S7-200 thực hiện chương trình bắt đầu từ lệnh lập trình đầu tiên và kết thúc ở lập trình cuối trong một vòng quét (scan).

Một vòng quét (scan cycle) được bắt đầu bằng việc đọc trạng thái của đầu vào, 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, PLC S7-200 thực thi các nhiệm vụ bên trong và nhiệm vụ truyền thông. Chu trình thực hiện chương trình là chu trình lặp.

Đối với thiết bị điều khiển lập trình PLC S7 - 200, ta không thể lập trình trực tiếp ngay trên nó được mà phải lập trình gián tiếp bằng cách sử dụng một trong những phần mềm sau đây :

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

Những phần mềm này đều có thể cài đặt được trên các máy lập trình họ PG7xx hoặc các máy tính cá nhân (PC). Công việc lập trình là ta sử dụng máy tính để tiến hành lắp ghép các lệnh cơ bản lại với nhau nhằm thỏa mãn những yêu cầu đề ra của quy trình công nghệ rồi sau đó mới chuyển vào PLC để điều khiển. Đối với các thiết bị lập trình của Siemens nói chung và thiết bị PLC S7 – 200 nói riêng thì có 3 ngôn phương pháp(ngôn ngữ) lập trình cơ bản thích hợp với những người có thói quen lập trình khác nhau, đó là:

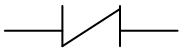
- Ngôn hình thang LAD (*Ladder Logic*)
- Ngôn ngữ liệt kê lệnh STL (*Statement List*)
- Ngôn ngữ khối hàm FBD (*Function Block Diagram*)

2.5.1. Phương pháp 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 bằng 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 rơ le.

  Tiếp điểm thường mở 

  Tiếp điểm thương đóng 

- Cuộn dây (coil): —( )— là biểu tượng mô tả rơle, được mắc theo chiều dòng điện cung cấp cho rơ le.
- Hộp (Box): là biểu tượng mô tả các hàm khác nhau, nó làm việc khi có dòng điện chạy đến hộp. Những dạng hàm thường được biểu diễn bằng hộp là các bộ thời gian (Timer), bộ đếm (counter) và các hàm toán học. Cuộn dây và các hộp phải mắc đúng chiều dòng điện.
- **Mạng LAD**: Là đường nối các phần tử thành một mạch hoàn thiện, đi từ đường nguồn bên trái sang đường nguồn bên phải. Đường nguồn bên trái là dây pha, đường nguồn bên phải là dây trung tính và cũng là đường trở về nguồn cung cấp (thường không được thể hiện khi dùng chương trình STEP 7 MICRO / DOS hoặc STEP 7 – MICRO/WIN. Dòng điện chạy từ trái qua tiếp điểm đến đóng các cuộn dây hoặc các hộp trở về bên phải nguồn.

2.5.2. Phương pháp Liệt kê lệnh (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 biểu diễn một chức năng của PLC. Phương pháp lập trình LAD phù hợp cho những người lập trình quen suy luận về kỹ thuật, còn STL phù hợp cho người lập trình quen suy luận về tin học.

2.5.3. Phương pháp khối hàm (FBD)

Đây cũng là một ngôn ngữ đồ họa dành cho người có thói quen thiết kế mạch điều khiển số. Tuy nhiên, do tính chất đặc thù của ngôn ngữ là bắt đầu thiết kế từ đầu ra sau đó đi ngược trở lại để tìm đầu vào nên khó cho bài toán có nhiều đầu ra. Do đó ngôn ngữ này ít được dùng so với 2 ngôn ngữ trên.

Nhận xét: STL là ngôn ngữ mạch nhất trong 3 loại ngôn ngữ trên. Một chương trình viết trên LAD hoặc FBD có thể chuyển sang được STL, nhưng ngược lại thì có thể không. Trong STL có nhiều lệnh không có trong LAD hoặc FBD.

Chương 3
NGÔN NGỮ LẬP TRÌNH

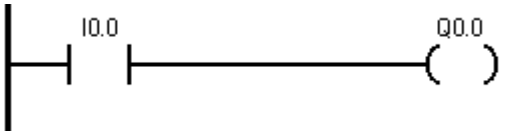
3.1. Các lệnh cơ bản

3.1.1. Lệnh vào/ra

- **LOAD (LD) :**

Lệnh LD nạp giá trị logic của một tiếp điểm vào trong bit đầu tiên của ngăn xếp, các giá trị còn lại trong ngăn xếp bị đẩy lùi xuống một bit.
Toán hạng gồm I, Q, M, SM, V, C, T.

- Dạng LAD : Tiếp điểm thường mở sẽ đóng nếu I0.0 =1

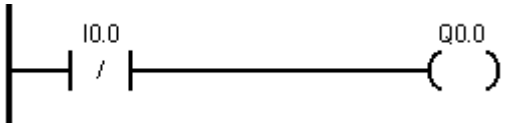


- Dạng STL : LD I0.0
  = Q0.0

- **LOAD NOT (LDN) :**

Lệnh LDN nạp giá trị logic của một tiếp điểm vào trong bit đầu tiên của ngăn xếp, các giá trị còn lại trong ngăn xếp bị đẩy lùi xuống một bit.
Toán hạng gồm : I, Q, M, SM, V, C, T.

- Dạng LAD : Tiếp điểm thường đóng sẽ mở khi I0.0 =1

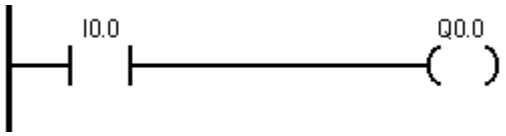


- Dạng STL : LDN I0.0
  = Q0.0

- **OUTPUT (=) :**

Lệnh sao chép nội dung của bit đầu tiên trong ngăn xếp vào bit được chỉ định trong lệnh. Nội dung ngăn xếp không bị thay đổi.
Toán hạng bao gồm : I,Q,M,SM,T,C (bit)

- Mô tả lệnh OUTPUT bằng LAD như sau :
  Nếu I0.0 = 1 thì Q0.0 sẽ lên 1 (cuộn dây nối với ngõ ra Q0.0 có điện)



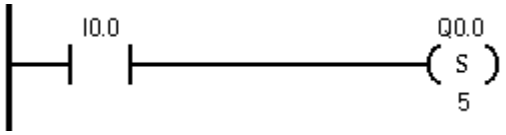
- Dạng STL : Giá trị logic I0.0 được đưa vào bit đầu tiên của ngăn xếp, và bit này được sao chép vào bit ngõ ra Q0.0 .
  LD I0.0
  = Q0.0

3.1.2. Các lệnh ghi/xóa giá trị cho tiếp điểm

- **SET (S) :**

Lệnh dùng để đóng các điểm gián đoạn đã được thiết kế. Trong LAD, logic điều khiển dòng điện đóng các cuộn dây đầu ra. Khi dòng điều khiển đến các cuộn dây thì các cuộn dây đóng các tiếp điểm. Trong STL, lệnh truyền trạng thái bit đầu tiên của ngăn xếp đến các điểm thiết kế. Nếu bit này có giá trị bằng 1, các lệnh S sẽ đóng 1 tiếp điểm hoặc một dãy các tiếp điểm (giới hạn từ 1 đến 255). Nội dung của ngăn xếp không bị thay đổi bởi các lệnh này.

- Dạng LAD : đóng một mảng gồm n các tiếp điểm kể từ địa chỉ S-bit, Toán hạng bao gồm I, Q, M, SM,T, C,V (bit)



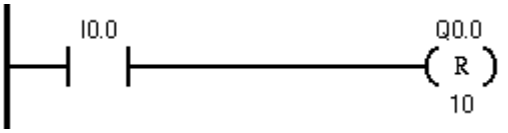
- Dạng STL : Ghi giá trị logic vào một mảng gồm n bit kể từ địa chỉ S-bit

```
LD    I0.0
S     Q0.0,   5
```

- **RESET (R) :**

Lệnh dùng để ngắt các điểm gián đoạn đã được thiết kế. Trong LAD, logic điều khiển dòng điện ngắt các cuộn dây đầu ra. Khi dòng điều khiển đến các cuộn dây thì các cuôn dây mở các tiếp điểm. Trong STL, lệnh truyền trạng thái bit đầu tiên của ngăn xếp đến các điểm thiết kế. Nếu bit này có giá trị bằng 1, các lệnh R sẽ ngắt 1 tiếp điểm hoặc một dãy các tiếp điểm (giới hạn từ 1 đến 255). Nội dung của ngăn xếp không bị thay đổi bởi các lệnh này.

- Dạng LAD : ngắt một mảng gồm n các tiếp điểm kể từ S-bit. Nếu S-bit lại chỉ vào Timer hoặc Counter thì lệnh sẽ xoá bit đầu ra của Timer/ Counter đó... .Toán hạng bao gồm I, Q, M, SM,T, C,V (bit)



- Dạng STL : xóa một mảng gồm n bit kể từ địa chỉ S-bit. Nếu S-bit lại chỉ vào Timer hoặc Counter thì lệnh sẽ xoá bit đầu ra của Timer/Counter đó.

```
LD    I0.0
R     Q0.0,   10
```

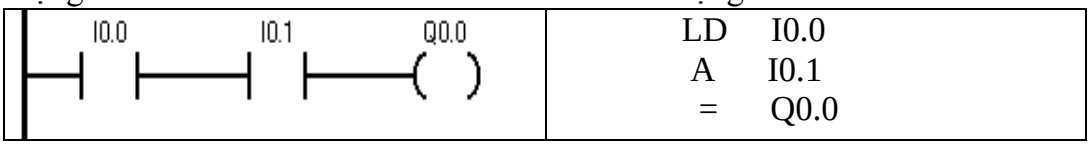
3.1.3. Các lệnh logic đại số Boolean

Các lệnh tiếp điểm đại số Boolean cho phép tạo lập các mạch logic (không có nhớ). Trong LAD các lệnh này được biểu diễn thông qua cấu trúc mạch, mắc nối tiếp hay song song các tiếp điểm thường đóng hay các tiếp điểm thường mở. Trong STL có thể sử dụng lệnh A (AND) và O (OR) cho các hàm hở hoặc các lệnh AN (AND NOT), ON (OR NOT) cho các hàm kín.

Giá trị của ngăn xếp thay đổi phụ thuộc vào từng lệnh.

- **AND (A) :**

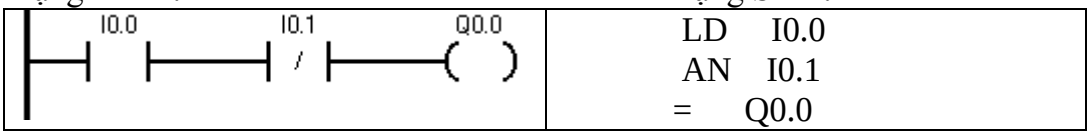
Dạng LAD : Dạng STL :




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

- **AND NOT(AN) :**

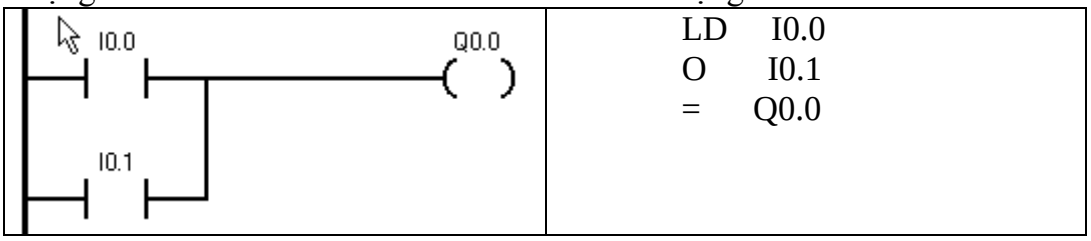
Dạng LAD : Dạng STL :




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

- **OR (O):**

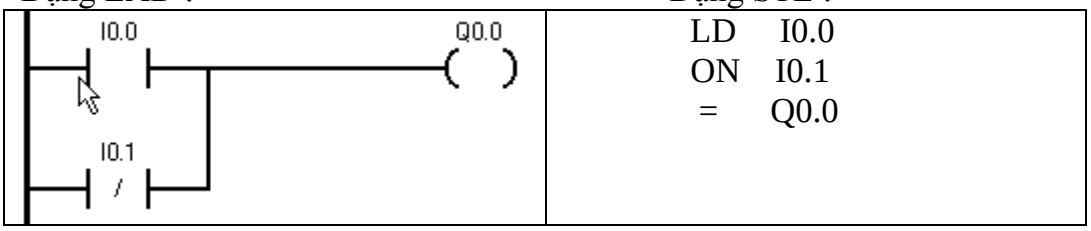
Dạng LAD : Dạng STL :




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

- **OR NOT (ON):**

Dạng LAD : Dạng STL :

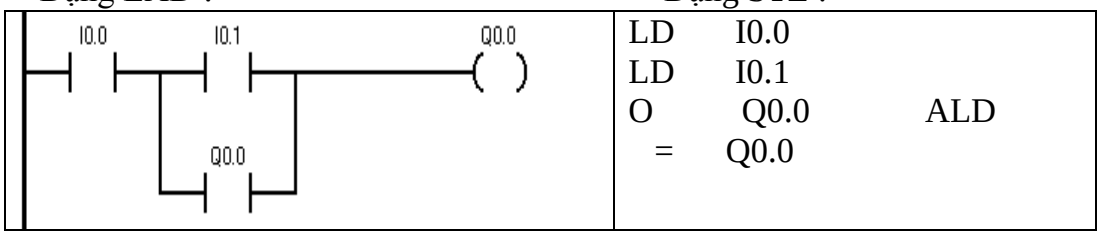



```
LD   I0.0
ON   I0.1
=    Q0.0
```

Ngoài những lệnh làm việc trực tiếp với tiếp điểm, S7-200 còn có 5 lệnh đặc biệt biểu diễn các phép tính của đại số Boolean cho các bit trong ngăn xếp, được gọi là 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 Stack logic. STL sử dụng các lệnh stack logic để thực hiện phương trình tổng thể có nhiều biểu thức con và được biểu diễn như sau:

- **AND LOAD (ALD) :**

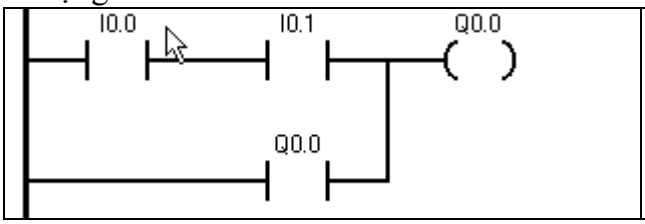
Dạng LAD : Dạng STL :




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

- **OR LOAD (OLD) :**

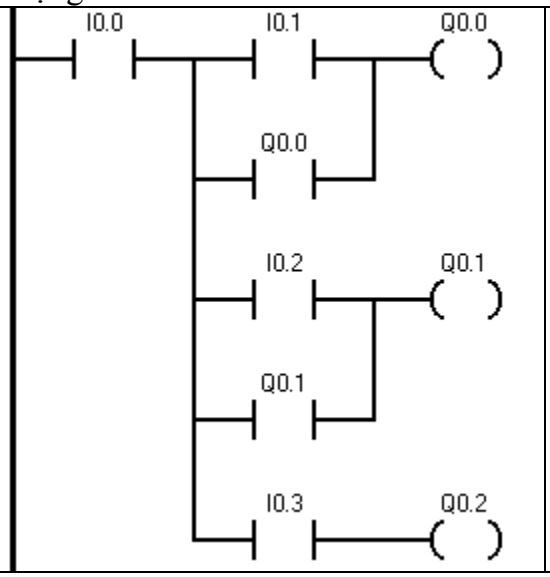
Dạng LAD : Dạng STL :




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

- **LOGIC PUSH (LPS), LOGIC READ (LRD) , LOGIC POP (LPP) :**

Dạng LAD: Dạng STL:



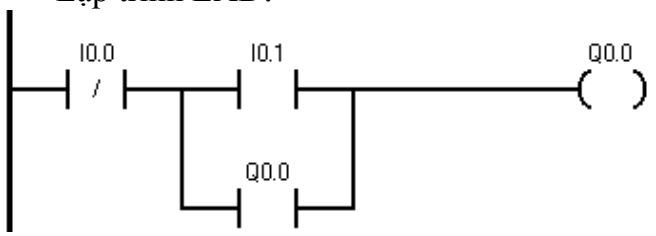

```
LD    I0.0
LPS
LD    I0.1
O           Q0.0
ALD
=           Q0.0
LRD
LD    I0.2
O           Q0.1
ALD
=           Q0.1
LPP
A           I0.3
=           Q0.2
```

Ý nghĩa của các lệnh :

Lệnh	Mô tả	Toán hạng
ALD (And load)	Lệnh tổ hợp giá trị của bit đầu tiên và thứ hai của ngăn xếp bằng phép tính logic AND. Kết quả ghi lại vào bit đầu tiên. Giá trị còn lại của ngăn xếp được kéo lên một bit.	Không có
OLD (Or load)	Lệnh tổ hợp giá trị của bit đầu tiên và thứ hai của ngăn xếp bằng phép tính logic OR. Kết quả ghi lại vào bit đầu tiên. Giá trị còn lại của ngăn xếp được kéo lên một bit.	Không có
LPS (Logic Push)	Lệnh Logic Push (LPS) sao chụp giá trị của bit đầu tiên vào bit thứ hai trong ngăn xếp. Giá trị còn lại bị đẩy xuống một bit. Bit cuối cùng bị đẩy ra khỏi ngăn xếp.	Không có
LRD (Logic read)	Lệnh sao chép giá trị của bit thứ hai vào bit đầu tiên trong ngăn xếp. Các giá trị còn lại của ngăn xếp giữ nguyên vị trí	Không có

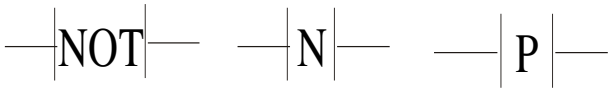
Ví dụ :

Viết chương trình điều khiển động cơ bằng PLC.

Lập trình LAD:	Ghi chú :
	I0.0 : Nút nhấn dừng I0.1 : Nút nhấn mở Q0.0 : Cuộn dây KĐT Q0.0 : Tiếp điểm duy trì

3.1.4. Các lệnh tiếp điểm đặc biệt

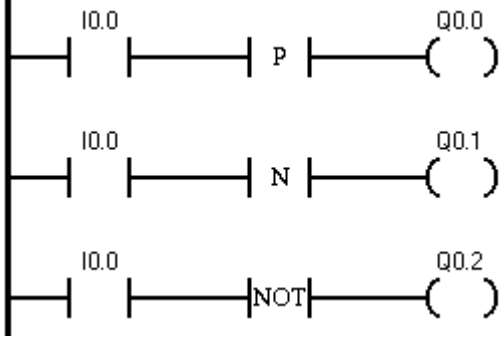
- **Tiếp điểm đảo, tác động cạnh xuống, tác động cạnh lên :**



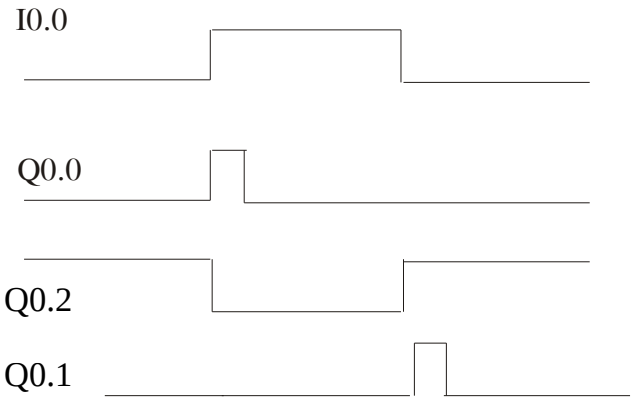
Có thể dùng các lệnh tiếp điểm đặc biệt để phát hiện sự chuyển tiếp trạng thái của xung (sườn xung) và đảo lại trạng thái của dòng cung cấp (giá trị đỉnh của ngăn xếp). LAD sử dụng các tiếp điểm đặc biệt này để tác động vào dòng cung cấp. Các tiếp điểm đặc biệt không có toán hạng riêng của chính chúng vì thế phải đặt chúng phía trước cuộn dây hoặc hộp đầu ra. Tiếp điểm chuyển tiếp dương/âm (các lệnh sườn trước và sườn sau) có nhu cầu về bộ nhớ bởi vậy đối với CPU 214 có thể sử dụng nhiều nhất là 256 lệnh.

Ví dụ:

Dạng LAD : Dạng STL :

	LD I0.0 EU = Q0.0 LD I0.0 ED = Q0.1 LD I0.0 NOT = Q0.2

Biểu đồ thời gian :



- **Tiếp điểm trong vùng nhớ đặc biệt :**
    - **SM0.1 :** Vòng quét đầu tiên tiếp điểm này đóng, kể từ vòng quét thứ hai thì mở ra và giữ nguyên trong suốt quá trình hoạt động.
    - **SM0.0 :** Ngược lại với SM0.1, vòng quét đầu tiên thì mở nhưng từ vòng quét thứ hai trở đi thì đóng.
    - **SM0.4 :** Tiếp điểm tạo xung với nhịp xung với chu kỳ là 1 phút.
    - **SM0.5 :** Tiếp điểm tạo xung với nhịp xung với chu kỳ là 1s

3.1.5. Các lệnh can thiệp vào thời gian vòng quét MEND, END, STOP, NOP, WDR

Các lệnh này được dùng để kết thúc chương trình đang thực hiện, và kéo dài một khoảng thời gian của một vòng quét.

Trong LAD và STL chương trình phải được kết thúc bằng lệnh kết thúc không điều kiện MEND. Có thể sử dụng lệnh kết thúc có điều kiện END trước lệnh kết thúc không điều kiện.

Lệnh STOP kết thúc chương trình, nó chuyển điều khiển chương trình đến chế độ STOP. Nếu gặp lệnh STOP trong chương trình chính, hoặc trong chương trình con thì chương trình đang thực hiện sẽ kết thúc ngay lập tức.

Lệnh rỗng NOP không có tác dụng gì trong việc thực hiện chương trình.

Lệnh WDR sẽ khởi động lại đồng hồ quan sát (*watchdog timer*), và chương trình tiếp tục được thực hiện trong vòng quét ở chế độ quan sát, cẩn thận khi sử dụng lệnh WDR.

3.2. Các lệnh thời gian (timer) và lệnh đếm (counter)

3.2.1. Các lệnh điều khiển thời gian Timer

Timer là bộ tạo thời gian trễ giữa tín hiệu vào và tín hiệu ra nên trong điều khiển vẫn thường được gọi là khâu trễ. Nếu ký hiệu tín hiệu (logic) vào là x(t) và thời gian trễ tạo ra bằng Timer là τ thì tín hiệu đầu ra của Timer đó sẽ là $x(t - \tau)$

S7-200 có 64 bộ Timer (với CPU 212) hoặc 128 Timer (với CPU 214) được chia làm hai loại khác nhau:

- Timer tạo thời gian trễ không có nhớ (On-Delay Timer), ký hiệu là TON.
- Timer tạo thời gian trễ có nhớ (Retentive On-Delay Timer), ký hiệu TONR.

Hai kiểu Timer của S7-200 (TON và TONR) phân biệt với nhau ở phản ứng của nó đối với trạng thái đầu vào.

Cả hai Timer kiểu 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 gian Timer được kích, và không tính khoảng thời gian khi đầu vào có giá trị logic 0 vào thời gian trễ tín hiệu đặt trước.

Khi đầu vào có giá trị logic bằng 0, TON tự động reset còn TONR thì không. Timer TON được dùng để tạo thời 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 ba 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 được chọn và giá trị đặt trước cho Timer. Ví dụ Timer có độ phân giải 10ms và giá trị đặt trước 50 thì thời gian trễ là 500ms.

Độ phân giải các loại Timer của S7-200, loại CPU 214, được trình bày trong bảng bên dưới.

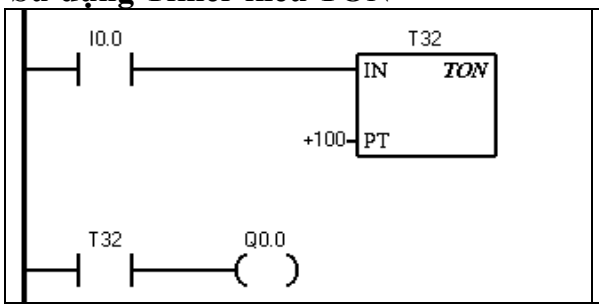
Lệnh	Độ phân giải	Giá trị cực đại	CPU 214
TON	1 ms	32,767 s	T32 và T96
	10 ms	327,67 s	T33 ÷ T36, T97 ÷ T100
	100 ms	3276,7 s	T37 ÷ T63, T101 ÷ T127
TONR	1 ms	32,767 s	T0 và T64
	10 ms	327,67 s	T1 ÷ T4, T65 ÷ T68
	100 ms	3276,7 s	T5 ÷ T31, T69 ÷ T95

Cú pháp khai báo sử dụng Timer như sau :

LAD	Mô tả	Toán hạng
TON-Txx IN PT	Khai báo Timer số hiệu xx 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.	Txx (*word*) CPU214:32÷63 96÷127 PT: VW, T, (*word*) C, IW, QW, MW, SMW, C, IW, hằng số
TONR-Txx IN PT	Khai báo Timer số hiệu xx kiểu TONR 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. Chỉ có thể reset Timer kiểu TON bằng lệnh R cho T-bit.	Txx (*word*) CPU214: 0÷31 64 ÷95 PT: VW, T, (*word*) C,IW,QW, MW, SMW, AC, AIW, hằng số

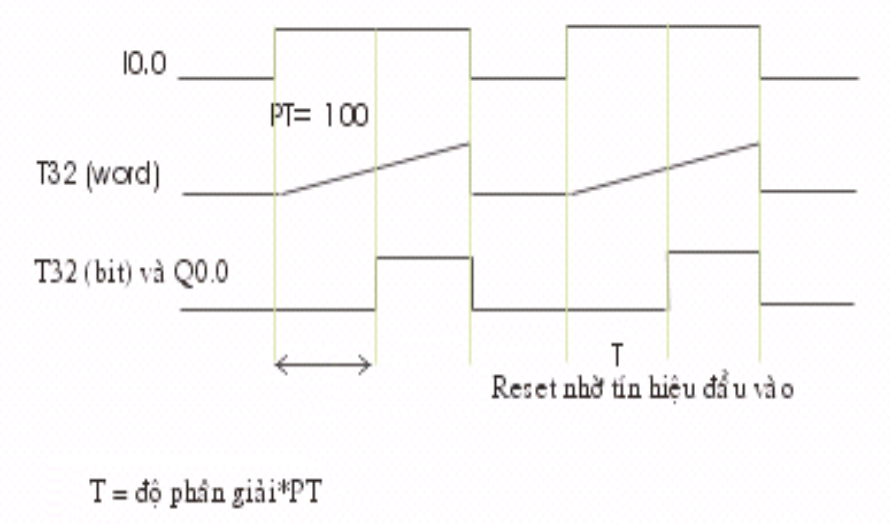
Khi sử dụng Timer TONR, giá trị đếm tức thời được lưu lại và không bị thay đổi trong khoảng thời gian khi tín hiệu đầu vào có logic 0. Giá trị của T-bit không được nhớ mà hoàn toàn phụ thuộc vào kết quả so sánh giữa giá trị đếm tức thời và giá trị đặt trước.
Khi Reset một bộ Timer, T-word và T-bit của nó đồng thời được xóa và có giá trị bằng 0, như vậy giá trị đếm tức thời được đặt về 0 và tín hiệu đầu ra cũng có trạng thái logic bằng 0.

Ví dụ: Sử dụng Timer kiểu TON

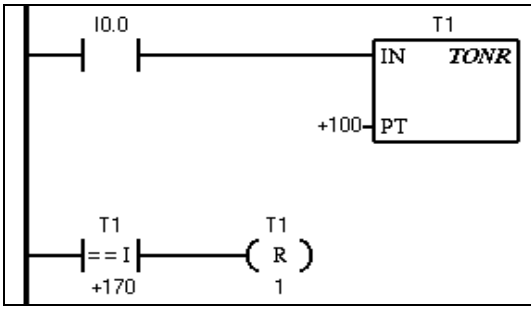



```
LD    I0.0
TON   T32, +100
LD    T32
=     Q0.0
```

Giản đồ thời gian :

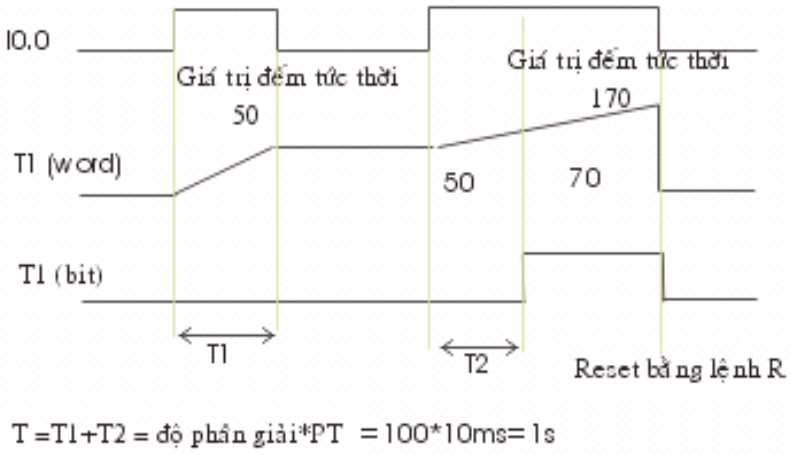


Ví dụ: Sử dụng timer kiểu TONR




```
LD    I0.0
TONR  T1, +100
LDW=  T1, +170
R     T1, 1
```

Giản đồ thời gian:



3.2.2. Các lệnh Đếm Counter

Counter là bộ đếm thực hiện chức năng đếm sườn xung, trong S7-200 các bộ đếm được chia 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 logic từ 0 lên 1 của tín hiệu. Số xung đếm được ghi vào thanh ghi 2 byte của bộ đếm, gọi là thanh ghi C-word.

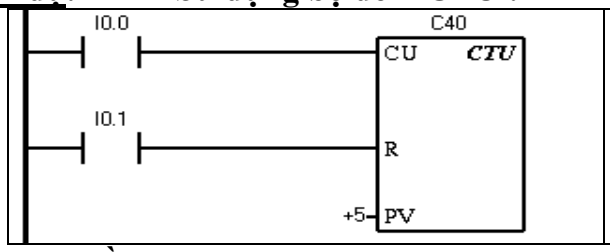
Nội dung của thanh ghi C-word, gọi là giá trị đếm tức thời của bộ đếm, luôn được so sánh với giá trị đặt trước của bộ đếm, được ký hiệu là PV. Khi giá trị đếm tức thời bằng hoặc lớn hơn giá trị đặt trước này thì bộ đếm báo ra ngoài bằng cách đặt giá trị logic 1 vào một bit đặc biệt của nó, gọi là C-bit. Trường hợp giá trị đếm tức thời nhỏ hơn giá trị đặt trước thì C-bit có giá trị logic là 0.
Khác với các bộ Timer, các bộ đếm CTU và CTUD đều có chân nối với tín hiệu điều khiển xóa để thực hiện việc đặt lại chế độ khởi phát ban đầu (reset) cho bộ đếm, được ký hiệu bằng chữ cái R trong LAD, hay được qui định là trạng thái logic của bit đầu tiên của ngăn xếp trong STL. Bộ đếm được reset khi tín hiệu xoá này có mức logic là 1 hoặc khi lệnh R (reset) được thực hiện với C-bit. Khi bộ đếm được reset, cả C-word và C-bit đều nhận giá trị 0.

LAD	Mô tả	Toán hạng
CTU – Cxx CU R PV	Khai báo bộ đếm tiến theo sườn lên của CU. Khi giá trị đếm tức thời C-word Cxx lớn hơn hoặc bằng giá trị đặt trước PV, C-bit (Cxx) có giá trị logic bằng 1. Bộ đếm được reset khi đầu vào R có giá trị logic bằng 1. Bộ đếm ngừng đếm khi C-word Cxx đạt được giá trị cực đại.	Cxx : (*word*) PV(*word*) : VW, T, C, IW, QW, MW, SMW, AC, AIW, hằng số, *VD, *AC
CTD-Cxx	Khai báo bộ đếm tiến/lùi, đếm tiến theo sườn lên của CU, đếm lùi theo sườn lên của CD. Khi giá trị đếm tức thời C-word Cxx lớn hơn hoặc bằng giá trị đặt trước PV, C-bit (Cxx) có giá trị logic bằng 1. Bộ đếm ngừng đếm tiến khi C-word Cxx đạt được giá trị cực đại 32.767 và ngừng đếm lùi khi C-word Cxx đạt được giá trị cực đại –32.768. CTUD reset khi đầu vào R có giá trị logic bằng 1.	Cxx (*word*) PV(*word*) : VW, T, C, IW, QW, MW, SMW, AC, AIW, hằng số, *VD, *AC

Bộ CTUD đếm tiến khi gặp sườn lên của xung vào cổng đếm tiến, ký hiệu là CU ứ 3 của ngăn xếp trong STL, và đếm lùi khi gặp sườn lên của xung vào cổn ý hiệu là CD trong LAD hoặc bit thứ 2 của ngăn xếp trong STL.
Bộ TU có miền giá trị đếm tức thời từ 0 đến 32.767. Bộ đếm tiến/lùi CTUD có miề n tức thời từ –32.768 đến 32.767.

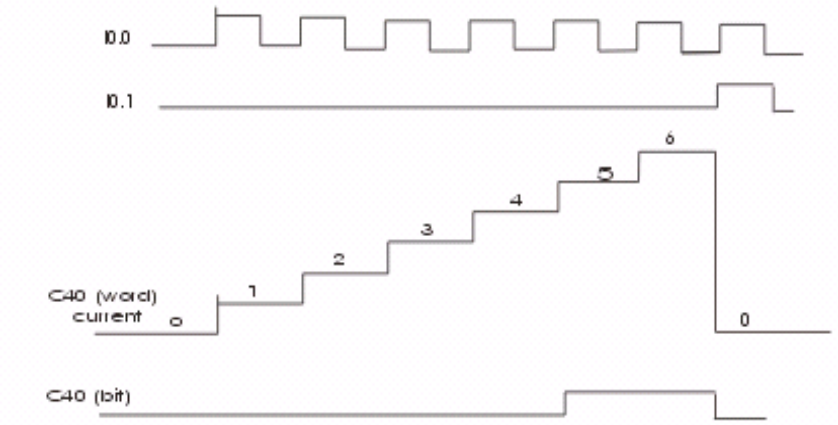
CU
CD
R
PV

Ví dụ: **Sử dụng bộ đếm CTU :**

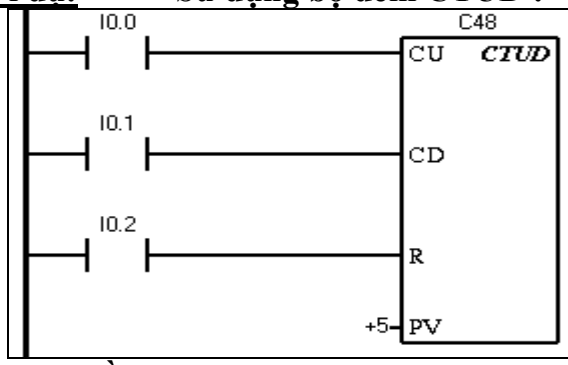



```
LD    I0.0
LD    I0.1
CTU   C40, +5
```

Giản đồ thời gian :

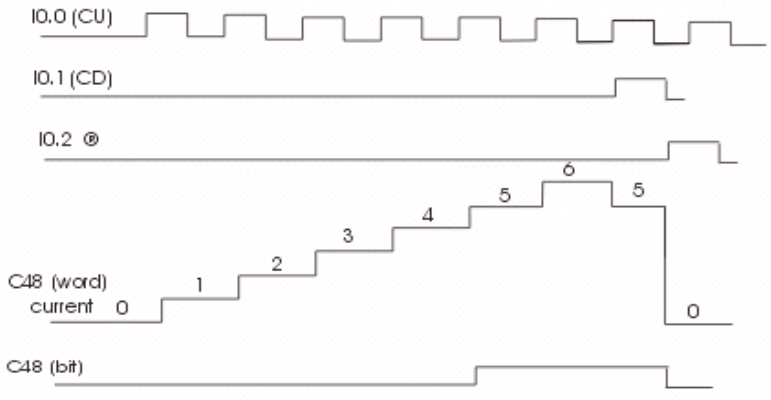


Ví dụ: **Sử dụng bộ đếm CTUD :**




```
//NETWORK COMMENTS
//
LD    I0.0
LD    I0.1
LD    I0.2
CTUD  C48, +5
```

Giản đồ thời gian :



3.3. Các lệnh so sánh

Khi lập trình, nếu các quyết định về điều khiển được thực hiện dựa trên kết quả của việc so sánh thì có thể sử dụng lệnh so sánh theo Byte, Word hay DWord của S7-200.

LAD sử dụng lệnh so sánh để so sánh các giá trị của byte, word hay DWord (giá trị thực hoặc nguyên).

Những lệnh so sánh thường là: so sánh nhỏ hơn hoặc bằng (<=); so sánh bằng (= =) và so sánh lớn hơn hoặc bằng (>=).

Khi so sánh giá trị của byte thì không cần phải để ý đến dấu của toán hạng, ngược lại khi so sánh các từ hay từ kép với nhau thì phải để ý đến dấu của toán hạng là bit cao nhất trong từ hoặc từ kép.

Ví dụ: 7FFF > 8000 và 7FFFFFFF > 80000000

LAD	Mô tả	Toán hạng
n1 n2 ==B n1 n2 ==I n1 n2 ==D n1 n2 ==R	Tiếp điểm đóng khi n1=n2 B = byte I = Integer = Word D = Double Integer R = Real	n1, n2(*byte*): VB, IB, QB, MB, SMB, AC, Const, *VD, *AC
n1 n2 >=B n1 n2 >=I n1 n2 >=D n1 n2 >=R	Tiếp điểm đóng khi n1>= n2 B = byte I = Integer = Word D = Double Integer R = Real	n1, n2 (*word*): VW, T, C, QW, MW, SMW, AC, AIW, hằng số, *VD, *AC
n1 n2 <=B n1 n2 <=I n1 n2 <=D n1 n2 <=R	Tiếp điểm đóng khi n1<= n2 B = byte I = Integer = Word D = Double Integer R = Real	n1, n2(D*word*) : VD, ID, QD, MD, SMD, AC, HC, hằng số, *VD, *AC

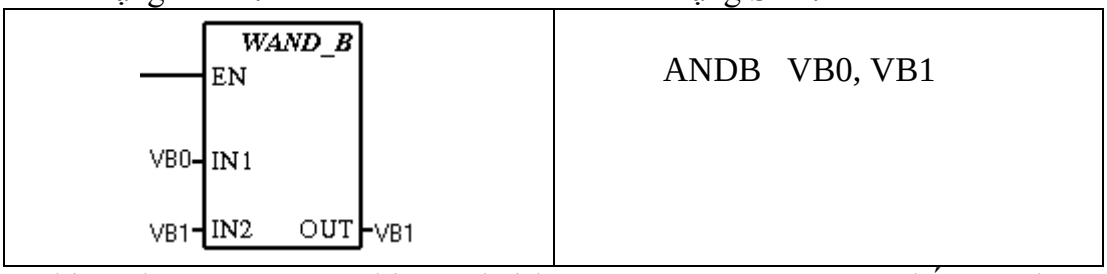
Trong STL những lệnh so sánh thực hiện phép so sánh byte, Word hay DWord. Căn cứ vào kiểu so sánh (<=, = =, >=), kết quả của phép so sánh có giá trị bằng 0 (nếu đúng) hoặc bằng 1 (nếu sai) nên nó có thể được kết hợp cùng các lệnh LD, A, O. Để tạo ra được các phép so sánh mà S7-200 không có lệnh so sánh tương ứng (như so sánh không bằng nhau <>, so sánh nhỏ hơn <, hoặc so sánh lớn hơn >) ta có thể kết hợp lệnh **NOT** với các lệnh đã có (= =, >=, <=)

3.4. Lệnh về cổng logic

Ngoài những lệnh ghép nối tiếp, song song và tổng hợp các tiếp điểm thì tập lệnh của S7-200 còn cung cấp các cổng logic AND, OR, EXOR thực hiện đối với byte (8 bit hay 8 tiếp điểm), word (16 bit hay 16 tiếp điểm) và double word (32 bit hay 32 tiếp điểm). Sau đây là chi tiết của từng cổng :

3.4.1. Lệnh AND byte

Dạng LAD : Dạng STL:



Lệnh thực hiện phép AND từng bit của hai byte ngõ vào IN1 và IN2, kết quả được ghi vào 1 byte ở ngõ ra OUT. Đặc biệt ở đây địa chỉ byte ngõ vào IN2 và byte ngõ ra OUT là giống nhau.

Toán hạng trong câu lệnh thuộc một trong các vùng địa chỉ sau :

IN1 : VB, T, C, IB, QB, SMB, AC, const

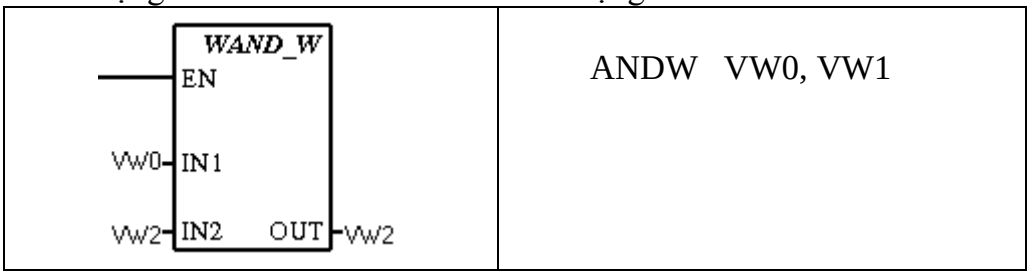
IN2 : VB, T, C, IB, QB, SMB, AC

Ví dụ:

VB10	1	0	0	0	1	1	1	0
AND								
VB20	0	0	1	1	0	1	1	1
Kết quả								
VB20	0	0	0	0	0	1	1	0

3.4.2. Lệnh AND word

Dạng LAD : Dạng STL:



Lệnh thực hiện phép AND từng bit của hai Word ngõ vào IN1 và IN2, kết quả được ghi vào 1 Word ở ngõ ra OUT. Đặc biệt ở đây địa chỉ Word ngõ vào IN2 và Word ngõ ra OUT là giống nhau.

Toán hạng trong câu lệnh thuộc một trong các vùng địa chỉ sau

IN1 : VW, T, C, IW, QW, SMW, AC, const

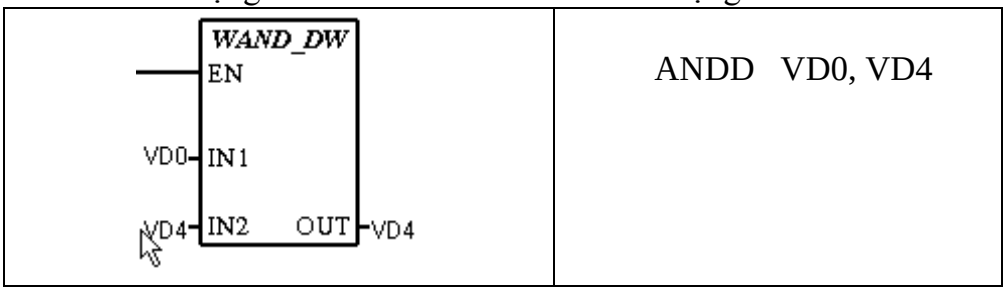
IN2 : VW, T, C, IW, QW, SMW, AC

Ví dụ :

VW10	1	0	1	0	1	1	1	0	0	1	1	1	1	1	0	1
AND																
VW12	1	0	0	0	1	0	1	0	1	1	0	1	1	1	0	1
Kết quả																
VW12	1	0	0	0	1	0	1	0	0	1	0	1	1	1	0	1

3.4.3. Lệnh AND DWord

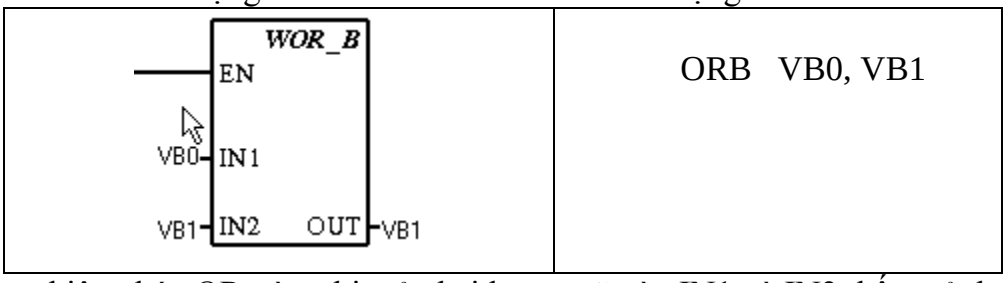
Dạng LAD : Dạng STL:




```
ANDD   VD0, VD4
```

3.4.4. Lệnh OR byte

Dạng LAD : Dạng STL:




```
ORB   VB0, VB1
```

Lệnh thực hiện phép OR từng bit của hai byte ngõ vào IN1 và IN2, kết quả được ghi vào 1 byte ở ngõ ra OUT. Đặc biệt ở đây địa chỉ byte ngõ vào IN2 và byte ngõ ra OUT là giống nhau.

Toán hạng trong câu lệnh thuộc một trong các vùng địa chỉ sau

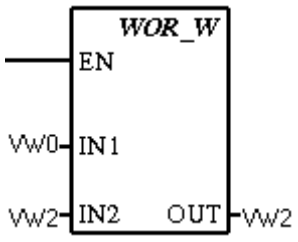
IN1 : VB, T, C, IB, QB, SMB, AC, const

IN2 : VB, T, C, IB, QB, SMB, AC

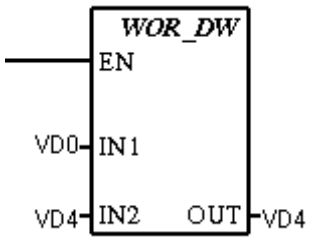
Ví dụ:

VD10	1	0	0	0	1	1	1	0
OR								
VD20	0	0	1	1	0	1	1	1
Kết quả								
VD20	1	0	1	1	1	1	1	1

3.4.5. Lệnh OR word

Dạng LAD :	Dạng STL:
	ORW VW0, VW2

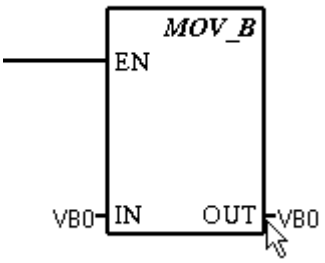
3.4.6. Lệnh OR double word

Dạng LAD :	Dạng STL:
	ORDW VD0, VD4

3.5. Các lệnh di chuyển nội dung ô nhớ

Các lệnh di chuyển thực hiện việc di chuyển hoặc sao chép số liệu từ vùng này sang vùng khác trong bộ nhớ. Trong LAD và STL lệnh dịch chuyển thực hiện việc di chuyển hay sao chép nội dung một byte, một từ đơn, hoặc một từ kép từ vùng này sang vùng khác trong bộ nhớ. Lệnh trao đổi nội dung của hai byte trong một từ đơn thực hiện việc chuyển nội dung của byte thấp sang byte cao và ngược lại chuyển nội dung của byte cao sang byte thấp của từ đó. Sau đây là chi tiết của từng lệnh.

- **MOV_B :**

Dạng LAD	Dạng STL
	MOVB VB0, VB0

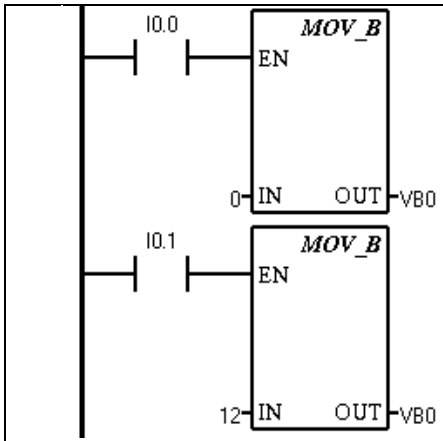
Lệnh sao chép nội dung của byte ở địa chỉ ngõ vào IN sang byte có địa chỉ ở ngõ ra OUT. Đặc biệt trong lệnh này địa chỉ của byte ngõ vào IN và địa chỉ byte ở địa chỉ ngõ ra OUT giống nhau và thường nằm trong các vùng sau:

IN : VB, IB, QB, MB, SMB, AC, const

OUT: VB, IB, QB, MB, SMB, AC,

Ví dụ :

Dạng LAD	Dạng STL

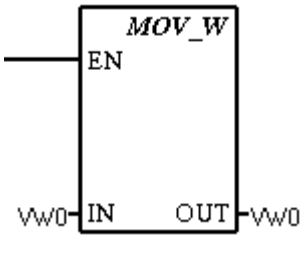
	LD I0.0
	MOVB 0, VB0
	LD I0.1
	MOVB 12, VB0

Giải thích :

Nếu tiếp điểm I0.0 đóng thì lấy giá trị 0 ghi vào byte VB0 (xóa VB0)

Tiếp theo đóng tiếp điểm I0.1 thì lấy số 12 ghi vào VB0. Kết quả địa chỉ byte VB0 có giá trị bằng 12.

- **MOV_W :**

Dạng LAD	Dạng STL
	MOVW VW0, VW0

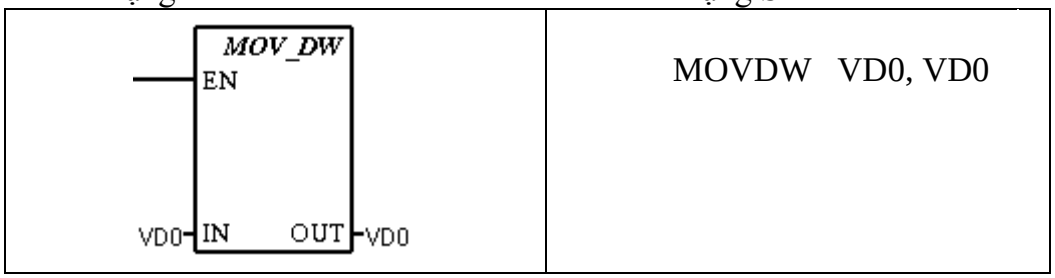
Lệnh sao chép nội dung của Word ở địa chỉ ngõ vào IN sang Word có địa chỉ ở ngõ ra OUT. Đặc biệt trong lệnh này địa chỉ của Word ngõ vào IN và địa chỉ Word ở địa chỉ ngõ ra OUT giống nhau và thường nằm trong các vùng sau:

IN: VW, IW, QW, MW, SMW, AC, const

OUT: VW, IW, QW, MW, SMW, AC

- **MOV_DW :**

Dạng LAD Dạng STL



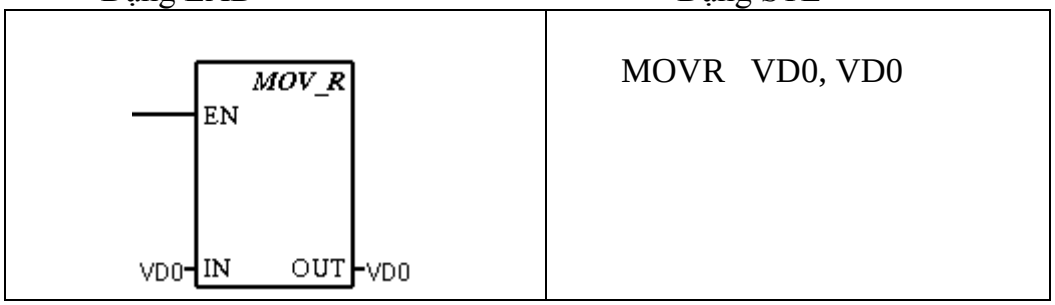
Lệnh sao chép nội dung của DWord ở địa chỉ ngõ vào IN sang DWord có địa chỉ ở ngõ ra OUT. Đặc biệt trong lệnh này địa chỉ của DWord ngõ vào IN và địa chỉ DWord ở địa chỉ ngõ ra OUT giống nhau và thường nằm trong các vùng sau:

IN: VD, ID, QD, MDW, SMD, AC, const

OUT: VD, ID, QD, MD, SMD, AC

- **MOV_R : (dịch chuyển số thực)**

Dạng LAD Dạng STL



Lệnh sao chép nội dung của số thực chứa trong double word có địa chỉ ở ngõ vào IN sang double word có địa chỉ ở ngõ vào OUT. Đặc biệt trong lệnh này địa chỉ của double word ở ngõ vào IN và double word ở ngõ ra OUT giống nhau và thường nằm trong các vùng sau:

IN: VD, ID, QD, MD, SMD, AC, const

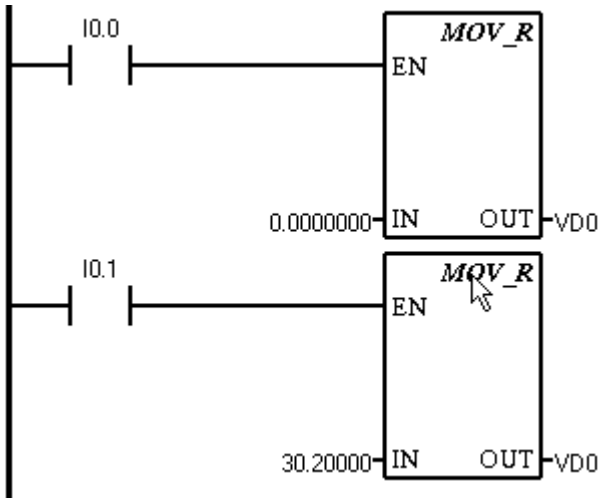
OUT: VD, ID, QD, MD, SMD, AC

Khi dữ liệu ghi vào trong các địa chỉ này theo nguyên tắc sau :

Phần nguyên ghi vào word thấp

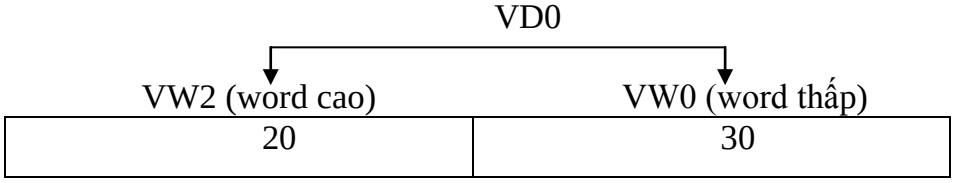
Phần thập phân ghi vào word cao

Ví dụ :



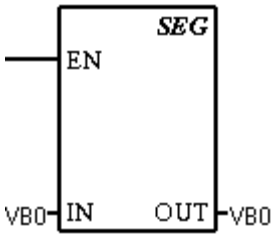
Giải Thích :

Tiếp điểm I0.0 đóng thì xóa double word 0 (VD0), tiếp điểm I0.1 đóng thì ghi số thực 30,2 vào double word VD0, kết quả như sau :



3.6. Lệnh chuyển đổi dữ liệu

3.6.1. Lệnh chuyển đổi số nguyên hệ thập lục phân sang led 7 đoạn

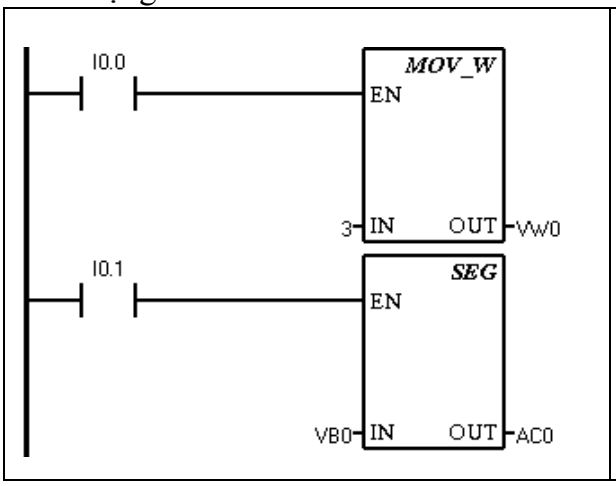
Dạng LAD :	Dạng STL :
	SEG VB0, VB0

Lệnh này có tác dụng chuyển đổi các số trong hệ thập lục phân từ 0 đến F chứa trong 4 Bit thấp của byte có địa chỉ ở ngõ vào IN thành giá trị BIT chứa trong 8 bit của byte có địa chỉ ở ngõ ra OUT tương ứng với thanh led 7 đoạn. Trong lệnh này byte có địa chỉ ở ngõ vào IN và byte có địa chỉ ở ngõ ra OUT có thể cùng địa chỉ và nằm trong những vùng sau:

IN: VB, IB, QB, MB, SMB,AC, const
OUT: VB, IB,AB,MB,SMB,AC

Ví dụ :

Dạng LAD: Dạng STL:




```
LD    I0.0
MOVW  +3, VW0
LD    I0.1
SEG   VB0, AC0
```

Giải thích :

Khi tiếp điểm I0.0 đóng thì số 7 được ghi vào VW0, sau đó tiếp điểm I0.1 đóng thì giá trị chứa trong 4 bit thấp của byte VB0 chuyển thành 8 bit chứa trong thanh ghi AC0. Ta có thể minh họa theo bit như sau :

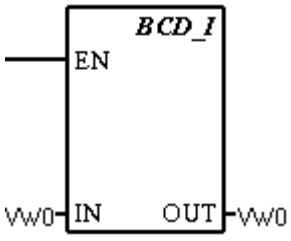
	VB1							VB0							
VW0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	VB1							VB0							
								0	0	0	0	0	0	1	1

SEG		.	g	f	e	d	c	b	a
AC0	0	0	1	0	0	1	1	1	1

3.6.2. Lệnh chuyển đổi số mã BCD sang số nguyên

Dạng LAD : Dạng STL :




```
BCDI   VW0
```

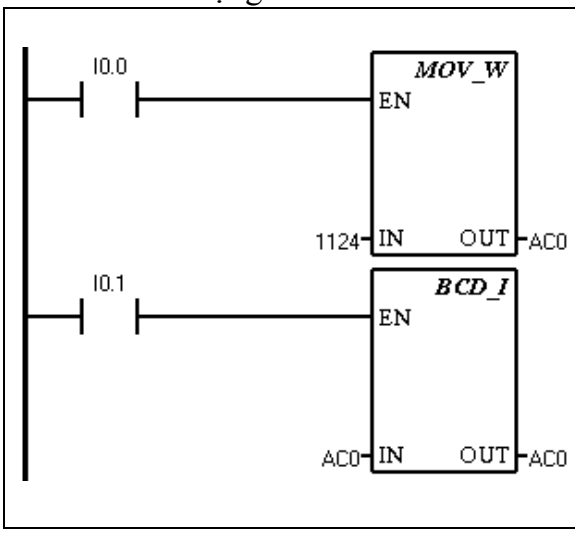
Lệnh này thực hiện phép biến đổi một số nhị thập phân 16 bit chứa trong word có địa chỉ ở ngõ vào IN sang số nguyên 16 bit chứa trong word có địa chỉ ở ngõ ra OUT. Đặc biệt ở đây word có địa chỉ ở ngõ vào IN và word có địa chỉ ở ngõ ra OUT có thể cùng một địa chỉ. Địa chỉ này thường nằm trong các vùng sau :

IN: VW, T, C, IW, QW, MW, SMW, AC, AIW, const

OUT: VW, T, C, IW, QW, MW, SMW, AC.

Ví dụ :

Dạng LAD



Dạng STL

```
LD    I0.0
MOVW  +1124, AC0
LD    I0.1
BCDI  AC0
```

Giải thích:

Khi I0.0 đóng , giá trị 1124 theo mã BCD là 0001 0001 0010 0100 được ghi vào địa chỉ AC0. Tiếp điểm I0.1 đóng thì giá trị BCD đó được chuyển sang số nguyên và lưu vào AC0. Ta biểu diễn theo bit như sau :

MOV AC0

Mã BCD	0	0	0	1		0	0	0	1		0	0	1	0		0	1	0	0

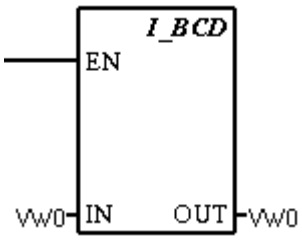
BCD-I AC0

Số nguyên	0	0	0	0		0	1	0	0		0	1	1	0		0	1	0	0

AC0= 1024 + 64 + 32 + 4 = 1124

3.6.3. Lệnh chuyển đổi số nguyên sang mã BCD

Dạng LAD :



Dạng STL :

```
I BCD   VW0
```

Lệnh này thực hiện phép biến đổi một số nguyên 16 bit chứa trong word có địa chỉ ở ngõ vào IN sang số nhị thập phân 16 bit chứa trong word có địa chỉ ở ngõ ra OUT. Đặc biệt ở đây word có địa chỉ ở ngõ vào IN và word có địa chỉ ở ngõ ra OUT có thể cùng một địa chỉ.

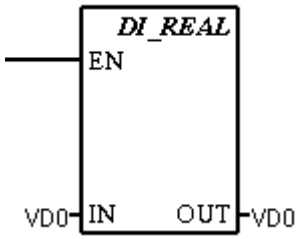
Địa chỉ này thường nằm trong các vùng sau :

IN : VW, T, C, IW, QW, MW, SMW, AC, AIW, const

OUT : VW, T, C, IW, QW, MW, SMW, AC.

3.6.4. Lệnh chuyển đổi số nguyên sang số thực

Dạng LAD : Dạng STL :




```
DTR    VD0, VD0
```

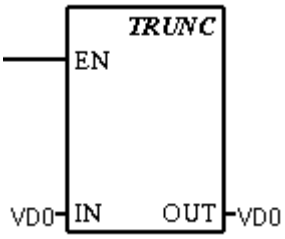
Lệnh này thực hiện phép biến đổi một số nguyên 32 bit có địa chỉ ở ngõ vào IN thành số thực 32 bit rồi ghi vào Dword có địa chỉ ở ngõ ra OUT. Trong đó toán hạng IN và OUT có thể cùng địa chỉ và thuộc một trong các vùng sau :

IN : VD, ID, QD, MD, SMD, AC, HC, const

OUT : VD, ID, QD, MD, SMD, AC

3.6.5. Lệnh chuyển đổi số thực sang số nguyên

Dạng LAD : Dạng STL :




```
TRUNC    VD0, VD0
```

Lệnh này thực hiện phép biến đổi một số thực 32 bit chứa trong Dword có địa chỉ ở ngõ vào IN thành số nguyên 32 bit rồi ghi vào Dword có địa chỉ ở ngõ ra OUT. Trong đó toán hạng IN và OUT có thể cùng địa chỉ và thuộc một trong các vùng sau :

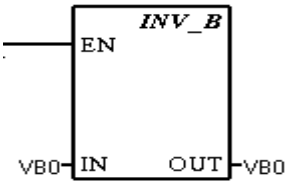
IN : VD, ID, QD, MD, SMD, AC, HC, const

OUT : VD, ID, QD, MD, SMD, AC.

3.6.6. Lệnh lấy giá trị nghịch đảo

Dạng LAD : Dạng STL :

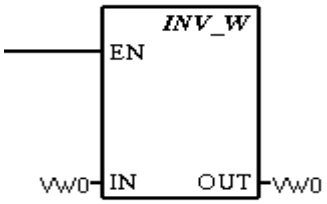
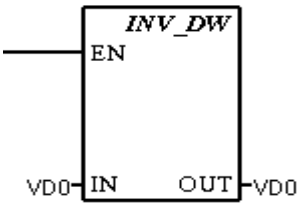
Giá trị Byte




```
INVB   VB0
```

Giá trị Word

```
INVW   VW0
```

 ***Giá trị Double Word*** 	INVD VD0

Lệnh có tác dụng đảo giá trị từng Bit của toán hạng của DW có địa chỉ ở ngõ vào IN rồi ghi kết quả vào Dword có địa chỉ ở ngõ ra OUT. Lệnh này còn được gọi là lệnh lấy giá trị bù của một số.

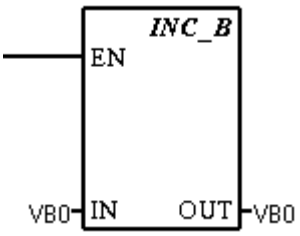
Ví dụ :

	VB10	0	0	0	0	0	1	1	1
INVB									
	VB10	1	1	1	1	1	0	0	0

3.7. Lệnh tăng giảm một đơn vị

3.7.1. Lệnh cộng số nguyên 1 vào nội dung byte

Dạng LAD : Dạng STL :

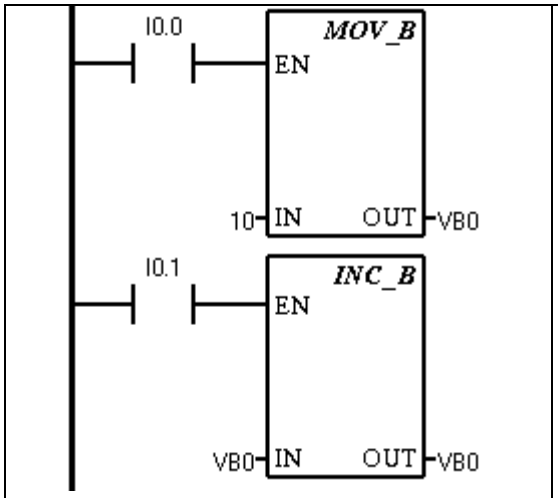
	INCB VB0

Lệnh này có tác dụng cộng số nguyên 1 đơn vị với nội dung của byte có địa chỉ ở ngõ vào IN, kết quả được ghi vào byte có địa chỉ ở ngõ ra OUT. Byte IN và byte OUT có thể cùng địa chỉ và ở lệnh này có sử dụng các bit nhớ đặc biệt SM1.0, SM1.1, SM1.2 để báo trạng thái kết quả phép tính theo nguyên tắc như sau :

Kết quả tính	SM1.0	SM1.1	SM1.2
= 0	1		
Số âm			1
> byte		1	

Ví dụ :

Dạng LAD : Dạng STL :



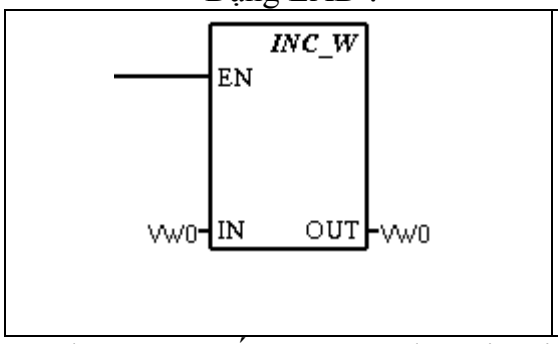

```
MOVB   10,  VB0
INCB   VB0
```

Giải thích :

Tiếp điểm I0.0 đóng thì số 10 được ghi vào VB0, tiếp điểm I0.1 đóng thì nội dung của VB0 tăng lên 1 đơn vị và kết quả được lưu lại VB0. Lúc này VB0 = 11.

3.7.2. Lệnh cộng số nguyên 1 vào nội dung word

Dạng LAD : Dạng STL :



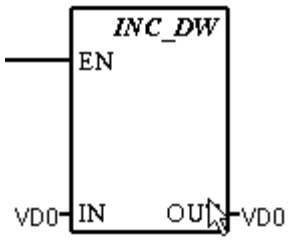

```
INCW      VW0
```

Lệnh này có tác dụng cộng số nguyên 1 đơn vị với nội dung của word có địa chỉ ở ngõ vào IN, kết quả được ghi vào word có địa chỉ ở ngõ ra OUT , word IN và word OUT có thể cùng địa chỉ và ở lệnh này có sử dụng các bit nhớ đặc biệt SM1.0, SM1.1, SM1.2 để báo trạng thái kết quả phép tính theo nguyên tắc như sau :

Kết quả tính	SM1.0	SM1.1	SM1.2
= 0	1		
Số âm			1
> byte		1	

3.7.3. Lệnh cộng số nguyên 1 vào nội dung double word

Dạng LAD : Dạng STL :



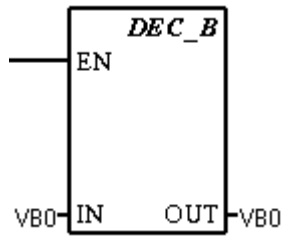
INCD VD0

Lệnh này có tác dụng cộng số nguyên 1 đơn vị với nội dung của double word có địa chỉ ở ngõ vào IN, kết quả được ghi vào doub word có địa chỉ ở ngõ ra OUT , double word IN và double word OUT có thể cùng địa chỉ và ở lệnh này cũng sử dụng các bit nhớ đặc biệt SM1.0, SM1.1, SM1.2 để báo trạng thái kết quả phép tính theo nguyên tắc như sau :

Kết quả tính	SM1.0	SM1.1	SM1.2
= 0	1		
Số âm			1
> byte		1	

3.7.4. Lệnh trừ nội dung của byte đi 1 đơn vị

Dạng LAD : Dạng STL :



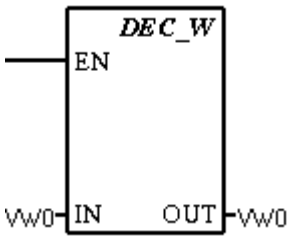
DECB VB0

Lệnh này có tác dụng lấy nội dung của byte có địa chỉ ở ngõ vào IN trừ đi 1 đơn vị , kết quả được ghi vào byte có địa chỉ ở ngõ ra OUT , byte IN và byte OUT có thể cùng địa chỉ và ở lệnh này cũng sử dụng các bit nhớ đặc biệt SM1.0, SM1.1, SM1.2 để báo trạng thái kết quả phép tính theo nguyên tắc như sau :

Kết quả tính	SM1.0	SM1.1	SM1.2
= 0	1		
Số âm			1
> byte		1	

3.7.5. Lệnh trừ nội dung của word đi 1 đơn vị

Dạng LAD : Dạng STL :



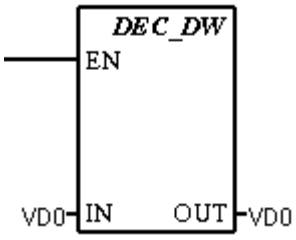
DECW VW0

Lệnh này có tác dụng lấy nội dung của word có địa chỉ ở ngõ vào IN trừ đi 1 đơn vị , kết quả được ghi vào word có địa chỉ ở ngõ ra OUT, trong lệnh word IN và word OUT có thể cùng địa chỉ và ở lệnh này cũng sử dụng các bit nhớ đặc biệt SM1.0, SM1.1, SM1.2 để báo trạng thái kết quả phép tính theo nguyên tắc như sau :

Kết quả tính	SM1.0	SM1.1	SM1.2
= 0	1		
Số âm			1
> byte		1	

3.7.6. Lệnh trừ nội dung của double word đi 1 đơn vị

Dạng LAD : Dạng STL :



DECD VD0

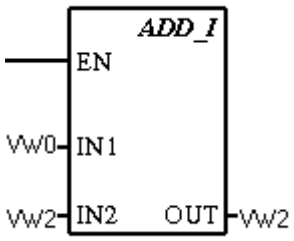
Lệnh này có tác dụng lấy nội dung của double word có địa chỉ ở ngõ vào IN trừ đi 1 đơn vị , kết quả được ghi vào double word có địa chỉ ở ngõ ra OUT, trong lệnh double word IN và double word OUT có thể cùng địa chỉ và ở lệnh này cũng sử dụng các bit nhớ đặc biệt SM1.0, SM1.1, SM1.2 để báo trạng thái kết quả phép tính theo nguyên tắc như sau :

Kết quả tính	SM1.0	SM1.1	SM1.2
= 0	1		
Số âm			1
> byte		1	

3.8. Các lệnh số học

3.8.1. Lệnh cộng số nguyên 16 bit

Dạng LAD : Dạng STL :



+I VW0, VW2

Lệnh thực hiện cộng các số nguyên 16 bit IN1 và IN2 , kết quả là một số nguyên 16-bit được ghi vào OUT , tức là : IN1 + IN2 = OUT

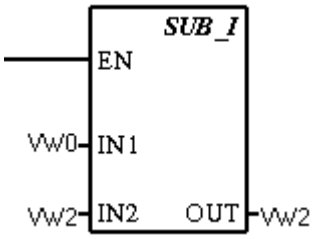
Trong đó IN2 và OUT có thể cùng địa chỉ và thuộc các vùng nhớ sau :

IN1, IN2 : VW,T ,C, IW, QW, MW, SMW, AC, AIW, const.

OUT : VW,T ,C, IW, QW, MW, SMW, AC, AIW

3.8.2. Lệnh trừ số nguyên 16 bit

Dạng LAD : Dạng STL :



+I VW0, VW2

Lệnh được thực hiện phép trừ các số nguyên 16 –bit IN1và IN2, kết quả là một số nguyên 16- bit và được ghi vào OUT , tức là : IN1 – IN2 = OUT

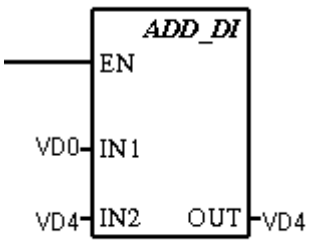
Tương tự lệnh cộng số nguyên 16 bit lệnh này cũng có IN2 và OUT cùng địa chỉ và thuộc các vùng nhớ sau :

IN1, IN2 : VW,T ,C, IW, QW, MW, SMW, AC, AIW, const.

OUT : VW,T ,C, IW, QW, MW, SMW, AC, AIW.

3.8.3. Cộng số nguyên 32 bit

Dạng LAD Dạng STL :



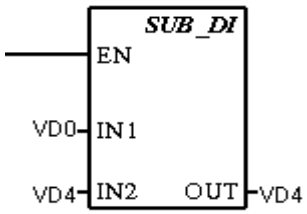
+D VD0, VD4

Lệnh thực hiện phép cộng các số nguyên 32 bit IN1 và IN2, kết qủa là một số nguyên 32 bit được ghi vào OUT, tức là IN1 + IN2 = OUT .
Trong lệnh IN2 và OUT có thể cùng địa chỉ và thuộc các vùng nhớ sau :

IN1, IN2 : VD,T ,C, ID, QD, MD, SMD, AC, HC, const.
OUT : VD,T ,C, ID, QD, MD, SMD, AC.

3.8.4 Trừ số nguyên 32 bit

Dạng LAD : Dạng STL :



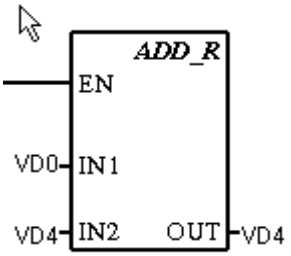

```
INVD   VD4
INCD   VD4
+D     VD0, VD4
```

Lệnh thay vì thực hiện phép trừ các số nguyên 32 bit IN1 và IN2 thì ở đây thực hiện bằng cách lấy nghịch đảo của số thực VD4 sau đó tăng lên 1 đơn vị rồi thực hiện cộng với VD0 kết quả là một số nguyên 32 bit được ghi vào OUT . Trong đó IN2 và OUT có thể cùng địa chỉ và thuộc các vùng nhớ sau :

IN1, IN2 : VD,T ,C, ID, QD, MD, SMD, AC, HC, const.
OUT : VD,T ,C, ID, QD, MD, SMD, AC.

3.8.5. Cộng số thực

Dạng LAD : Dạng STL :



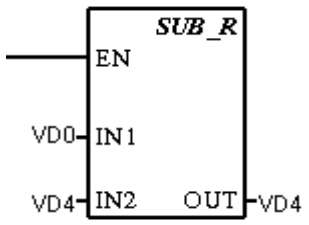

```
+R     VD0, VD4
```

Lệnh thực hiện phép cộng các số thực 32 bit IN1 và IN2, kết quả là một số thực 32 bit được ghi vào OUT, tức là : IN1 + IN2 = OUT
Trong đó toán hạng IN2 và OUT có thể cùng địa chỉ, thường nằm trong các vùng sau :

IN1, IN2 : VD, ID, QD, MD, SMD, AC, HC, const
OUT : VD, ID, QD, MD, SMD, AC

3.8.6. Trừ số thực

Dạng LAD : Dạng STL :

	*R -1.000000, VD4 (nhân –1) +R VD0, VD4

Lệnh thay vì thực hiện phép trừ các số thực 32 bit IN1 và IN2 thì ở đây thực hiện bằng cách nhân số thực IN2 với –1 rồi sau đó lấy IN1 cộng với IN2, kết quả là một số thực 32 bit được ghi vào OUT, tức là: IN1 + (IN2)(-1) = OUT.

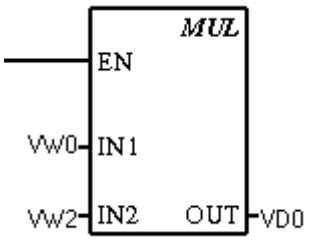
Trong đó toán hạng IN2 và OUT có thể cùng địa chỉ, thường nằm trong các vùng sau :

IN1, IN2 : VD, ID, QD, MD, SMD, AC, HC, const

OUT : VD, ID, QD, MD, SMD, AC

3.8.7. Nhân số nguyên 16 bit

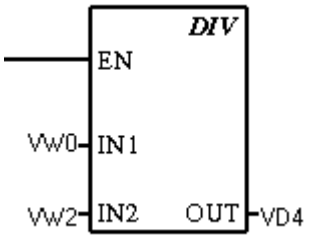
Dạng LAD : Dạng STL :

	MUL VW0, VD0

Lệnh thực hiện phép nhân 2 số nguyên 16bit IN1 và IN2. Kết quả 32 bit chứa trong từ kép OUT (4 byte).

3.8.8. Chia số nguyên 16 bit

Dạng LAD : Dạng STL :

	MOVW VW0, VW6 DIV VW2, VD4

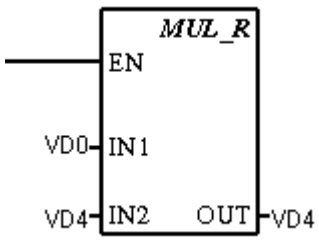
Lệnh thực hiện phép chia số nguyên 16 bit IN1 cho số nguyên 16 bit IN2. Kết quả 32 bit chứa trong từ kép OUT (4 byte) gồm thương số ghi trong mảng 16 bit từ bit 0 đến bit 15 (từ thấp) và phần dư cũng 16 bit ghi trong mảng từ bit 16 đến bit 31 (từ cao). Trong lệnh này có sử dụng các bit nhớ đặc biệt sau để báo trạng thái.

Kết quả tính	SM1.0	SM1.1	SM1.2	SM1.3
= 0	1			
Báo tràn		1		

Số âm			1	
Mẫu = 0				1

3.8.9. Nhân số thực 32 bit

Dạng LAD : Dạng STL :

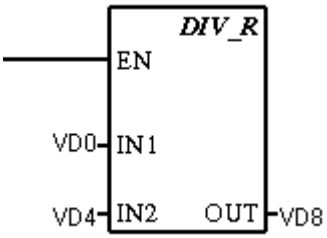



```
*R     VD0, VD4
```

Lệnh thực hiện phép nhân hai số thực 32bit IN1 và IN2 cho ra kết quả 32 bit chứa trong từ kép OUT (4 byte).

3.8.10 Chia số thực 32 bit :

Dạng LAD : Dạng STL :



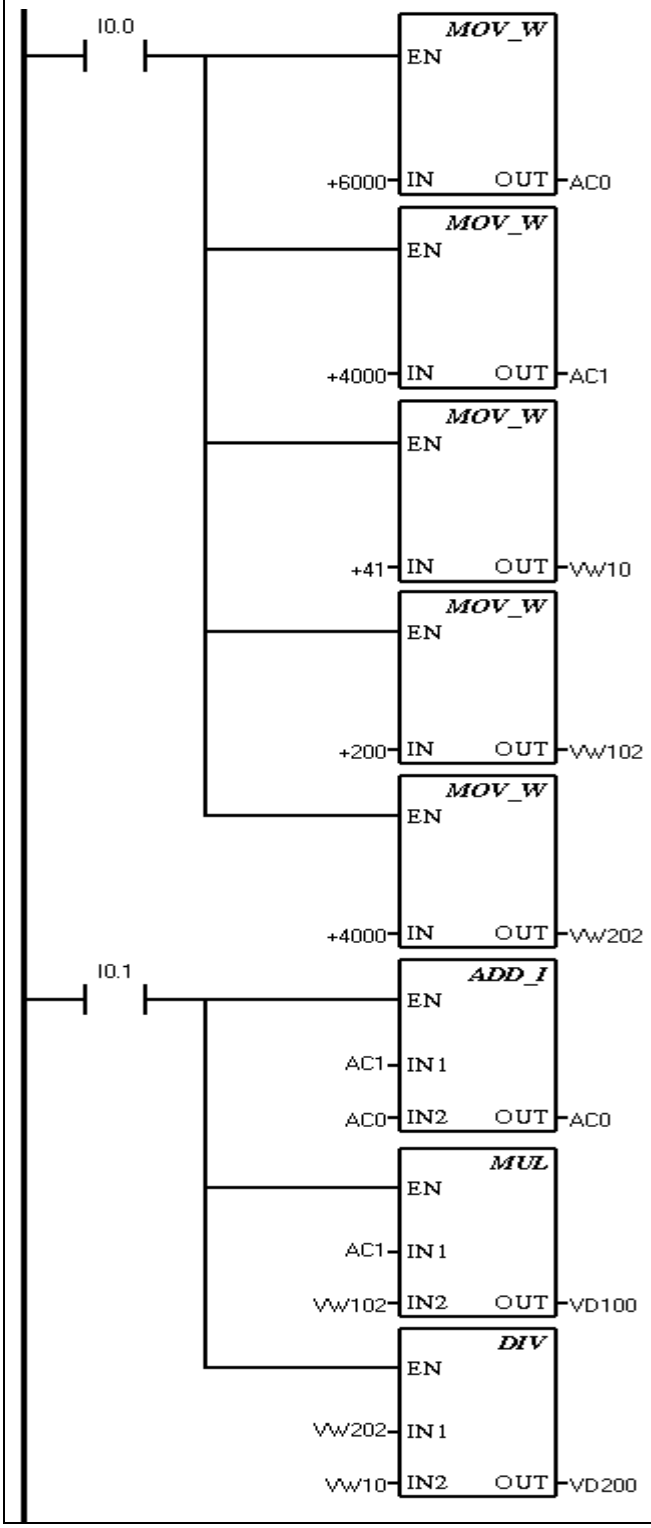

```
MOVR   VD0, VD8
/R     VD4, VD8
```

Lệnh thực hiện phép chia số thực 32 bit IN1 cho số thực 32 bit IN2. Kết qủa 32 bit chứa trong từ kép OUT (4 byte). Trong lệnh này cũng sử dụng các bit nhớ đặc biệt sau để báo trạng thái:

Kết quả tính	SM1.0	SM1.1	SM1.2	SM1.3
= 0	1			
Báo tràn		1		
Số âm			1	
Mẫu = 0				1

Ví dụ:

Dạng LAD:	**Dạng STL:**

	LD I0.0
	MOVW +6000, AC0
	MOVW +4000, AC1
	MOVW +41, VW10
	MOVW +200, VW102
	MOVW+4000, W202
	
	LD I0.1
	+I AC1, AC0
	MUL AC1, VD100
	DIV VW10, VD200

Kết quả các phép tính trên:

Phép cộng:			Phép nhân:
AC0=10000			VD100=800000
Phép chia:	VW202	VW200	
VD200=	23	97	
	Phần dư	Phần nguyên	

3.9. Lệnh nhảy và lệnh gọi chương trình con

Thông thường hoạt động của chương trình là thực hiện các lệnh theo thứ tự từ trên xuống dưới trong một vòng quét, bên cạnh đó chương trình cũng cho phép thay đổi và chuyển thứ tự thực hiện lệnh tùy theo yêu cầu của người lập trình sao cho việc lập trình đơn giản, vòng quét chương trình ngắn gọn và chương trình xử lý tốt các chức năng điều khiển.

Để làm được điều đó ta phải sử dụng tới nhóm lệnh điều khiển chương trình gồm : ***lệnh nhảy, lệnh gọi chương trình con***.

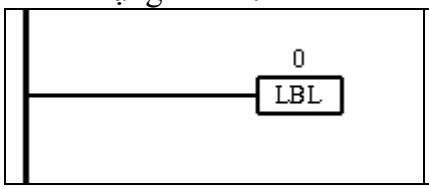
Muốn nhảy đến xử lý ở đoạn nào trong chương trình thì ta phải đánh dấu trước đoạn đó bằng một ký hiệu gọi là nhãn, tên của chương trình con hoặc tên của ngắt xử lý.

Việc đặt nhãn cho *lệnh nhảy* phải nằm trong chương trình. Nhãn của chương trình con hoặc của chương trình xử lý ngắt phải khai báo ở đầu chương trình.

Chương trình sẽ không hiểu khi dùng *lệnh nhảy* **JMP** để chuyển điều khiển từ chương trình chính vào một nhãn bất kỳ trong chương trình con hoặc trong chương trình xử lý ngắt. Tương tự như vậy cũng không thể từ một chương trình con hay chương trình xử lý ngắt nhảy vào bất cứ một nhãn nào nằm ngoài các chương trình đó.

3.9.1. Lệnh đặt nhãn

Dạng LAD : Dạng STL :

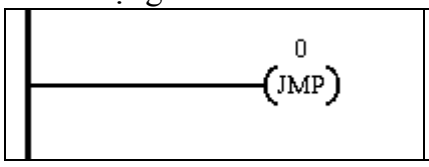


LBL 0

Lệnh này có thể đặt ở trong chương trình chính và cũng có thể đặt ở trong chương trình con. Ta có thể hiểu đây là một địa chỉ để chương trình thực hiện bắt đầu từ đó khi có lệnh nhãy đến. Địa chỉ nhãn này được đặt tên theo thứ tự từ 0, 1, 2, 3.....

3.9.2. Lệnh nhảy đến nhãn

Dạng LAD : Dạng STL :

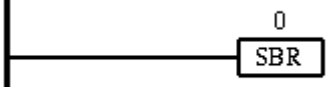


JMP 0

Lệnh này có thể đặt ở trong chương trình chính và cũng có thể đặt ở trong chương trình con nhưng với điều kiện là ở trong đó có địa chỉ nhãn đó. Khi chương trình thực hiện gặp lệnh này thì chương trình lập tức nhảy đến địa chỉ nhãn đó.

3.9.3. Lệnh gán nhãn cho chương trình con

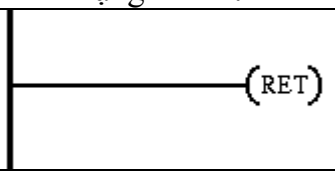
Dạng LAD : Dạng STL :

	SBR 0

Lệnh này thường đặt đầu chương trình con và có địa chỉ theo thứ tự từ 0, 1, 2, 3, 4.... Khi trong chương trình chính gọi đến địa chỉ của chương trình con nào thì chương trình con đó bắt đầu thực hiện từ đây.

3.9.4. Lệnh kết thúc chương trình con

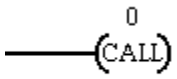
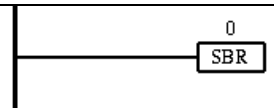
Dạng LAD : Dạng STL :

	RET

Lệnh này đặt ở cuối chương trình con. Khi gặp lệnh này thì chương trình sẽ kết thúc chương trình con và trở về (Return) thi hành lệnh kề sau lệnh đã gọi nó.

3.9.5. Lệnh gọi chương trình con

Dạng LAD : Dạng STL :

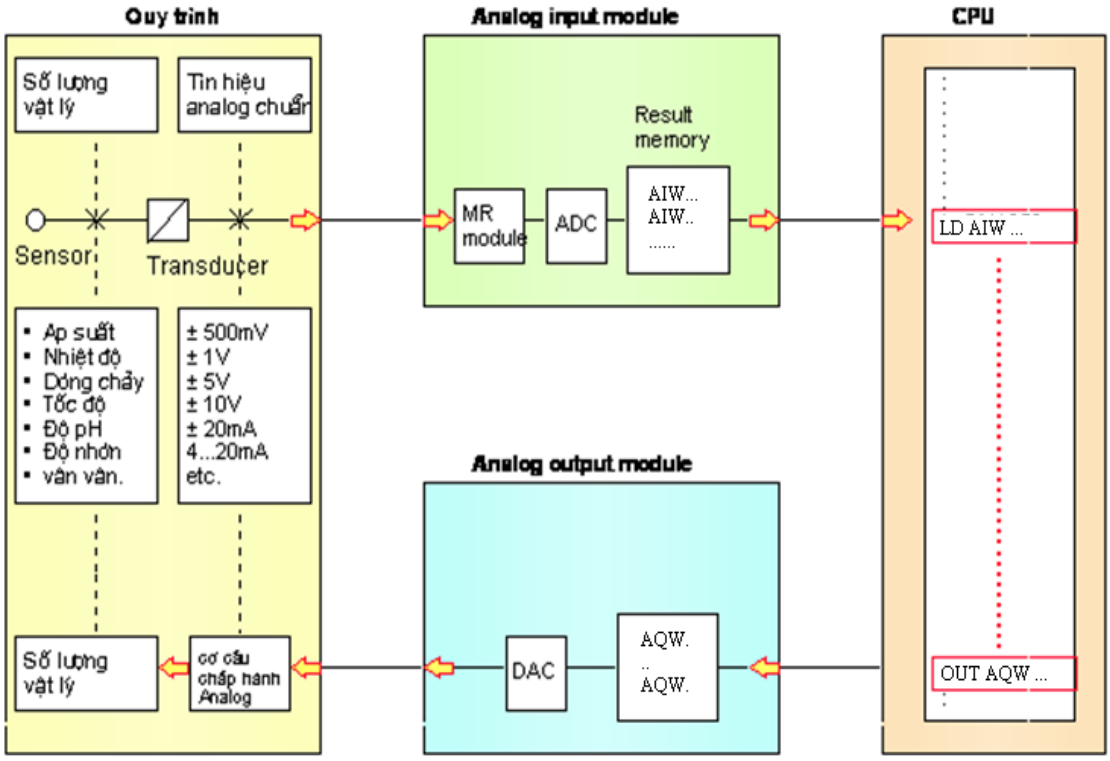
CPU21x	CPU22x	CPU21x	CPU22x
		CALL 0	Call sbr 0

Lệnh gọi chương trình con là lệnh chuyển điều khiển đến chương trình con được gọi. Khi kết thúc chương trình con thì việc điều khiển lại được chuyển trở về lệnh tiếp theo trong chương trình chính ngay sau lệnh gọi chương trình con. Từ một chương trình con có thể gọi được một chương trình con khác trong nó, có thể gọi như vậy nhiều nhất là 8 lần đối với S7-200.

Chương 4
XỬ LÝ TÍN HIỆU ANALOG

4.1. Tín hiệu Analog

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

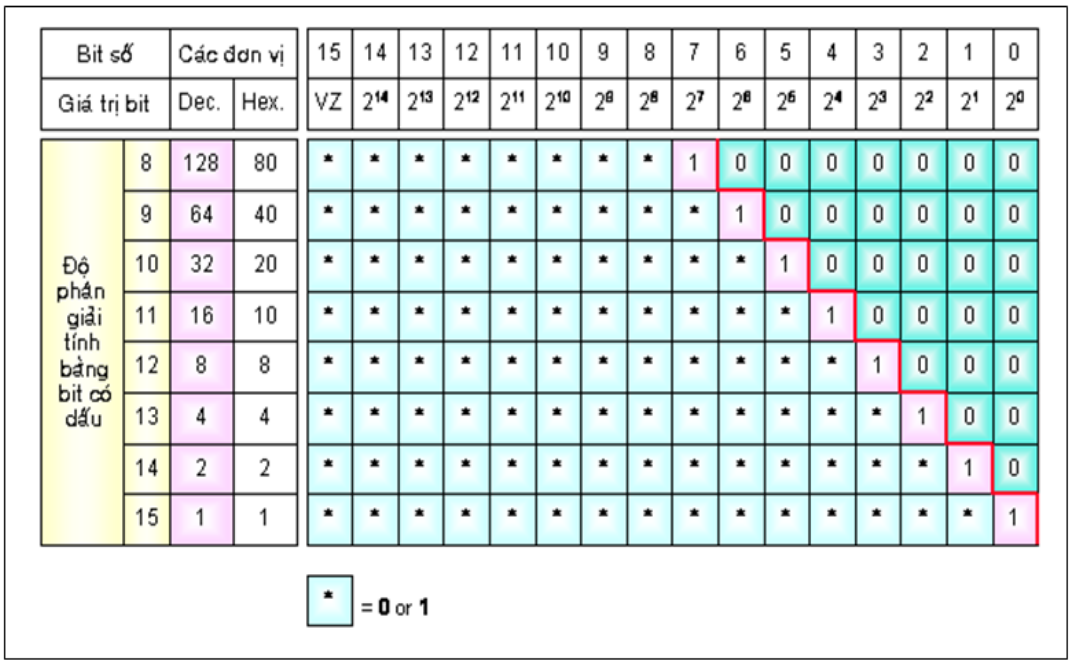


Hình 4-1

4.2. Biểu diễn các giá trị Analog

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

bằng số 0. (chú ý Bit thứ 15 là Bit dấu : = 0 khi giá trị chuyển đổi là số nguyên dương và = 1 khi giá trị chuyển đổi là số nguyên âm).



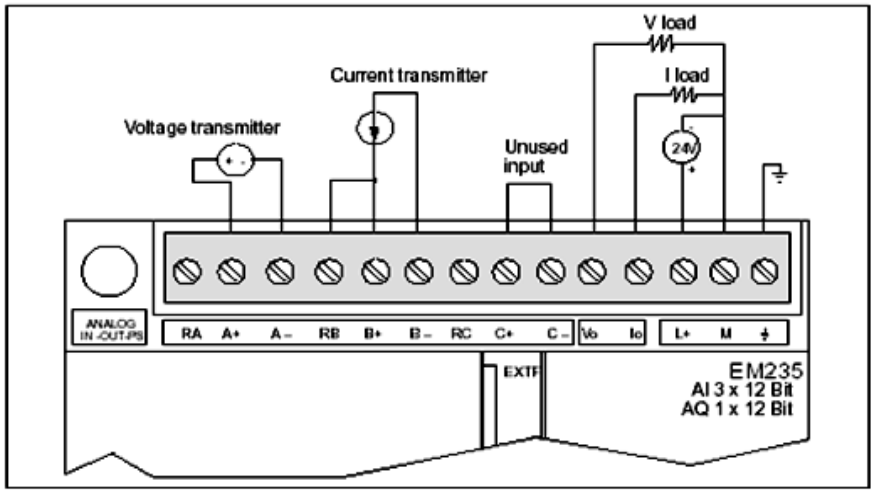
Hình 4-2

4.3. Kết nối ngõ vào-ra Analog

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

+ Đảm bảo rằng điện áp 24 VDC cấp nguồn cho Sensor không bị ảnh hưởng bởi nhiễu và ổn định .
+ Định tỷ lệ cho mô đun (được mô tả bên dưới).
+ Dây nối cho Sensor cần để ngắn nhất tới mức có thể.
+ Sử dụng cáp đôi dây xoắn cho sensor.
+ Tất cả các ngõ vào không sử dụng phải được nối tắt.
+ Tránh bẻ cong dây dẫn thành những góc nhọn.
+ Sử dụng máng đi dây hay các ống đi dây cho tuyến dây.
+ Tránh đặt các đường dây tín hiệu Analog gần với các đường dây có điện áp cao, nếu 2 đường dây này cắt nhau phải đặt chúng vuông góc với nhau.

Ví dụ về kết nối tín hiệu AI và AO vào Modul analog



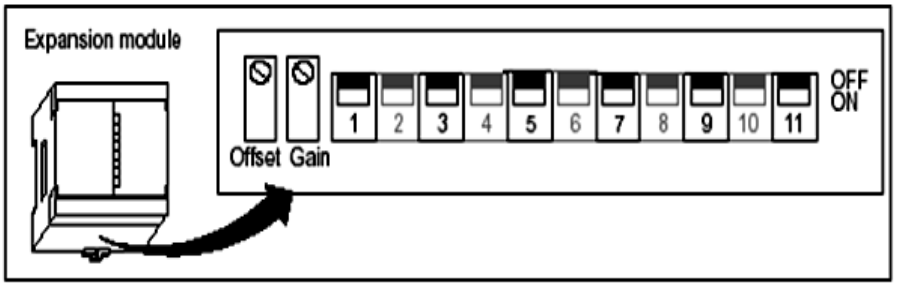
Hình 4.3

Phương pháp định tỷ lệ ngõ vào Analog (Input calibration)

Việc định tỷ lệ ngõ vào analog có ảnh hưởng đến tất cả các ngõ vào của modul EM có AI. Để định tỷ lệ ngõ vào một cách chính xác, cần sử dụng một chương trình được thiết kế để tính trung bình các giá trị đọc được từ Modul. Có thể sử dụng Analog Input Filtering wizard trong STEP7-MicroWIN để tạo ra chương trình này. Nên sử dụng 64 giá trị lấy mẫu hoặc hơn để tính giá trị trung bình của tín hiệu Analog.

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

+ *Tắt nguồn cung cấp cho mô đun, chọn phạm vi ngõ vào mong muốn*
+ *Cấp nguồn lại cho CPU và mô đun có AI.*
+ *Sử dụng một Transmiter, một nguồn áp, hay một nguồn dòng và đặt vào giá trị 0 cho một trong các ngõ vào.*
+ *Đọc giá trị mà CPU nhận được tại ngõ vào tương ứng đó.*
+ *Điều chỉnh biến trở đặt giá trị OFFSET cho tới khi giá trị đọc được là 0.*
+ *Điều chỉnh để tăng giá trị đặt vào tới định mức và xem giá trị mà CPU nhận được.*
+ *Điều chỉnh biến trở GAIN cho tới khi giá trị nhận được là 32000 hoặc tới 1 giá trị số mong muốn.*
+ *Lặp lại các bước trên nếu cần.*



Hình 4-4: Điều chỉnh các Switch và biến trở chỉnh GAIN

Việc chỉnh định các công tắc (Switch) trên modul Analog EM sẽ thay đổi các phạm vi đo lường định mức và độ phân giải của Modul. Các phạm vi và độ phân giải được cho ở bảng dưới đây :

Configuration Switch						Full-Scale Input	Resolution
1[1]	3	5	7	9	11		
ON	ON	OFF	ON	OFF	OFF	0 to 50 mV	12.5 µV
ON	ON	OFF	OFF	ON	OFF	0 to 100 mV	25 µV
ON	OFF	ON	ON	OFF	OFF	0 to 500 mV	125 µV
ON	OFF	ON	OFF	ON	OFF	0 to 1 V	250 µV
ON	OFF	OFF	ON	OFF	OFF	0 to 5 V	1.25 mV
ON	OFF	OFF	ON	OFF	OFF	0 to 20 mA[2]	5 µA
ON	OFF	OFF	OFF	ON	OFF	0 to 10 V	2.5 mV
OFF	ON	OFF	ON	OFF	OFF	±25 mV	12.5 µV
OFF	ON	OFF	OFF	ON	OFF	±50 mV	25 µV
OFF	ON	OFF	OFF	OFF	ON	±100 mV	50 µV
OFF	OFF	ON	ON	OFF	OFF	±250 mV	125 µV
OFF	OFF	ON	OFF	ON	OFF	±500 mV	250 µV
OFF	OFF	ON	OFF	OFF	ON	±1 V	500 µV
OFF	OFF	OFF	ON	OFF	OFF	±2.5 V	1.25 mV
OFF	OFF	OFF	OFF	ON	OFF	±5 V	2.5 mV
OFF	OFF	OFF	OFF	OFF	ON	±10 V	5 mV

Hình 4-5

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

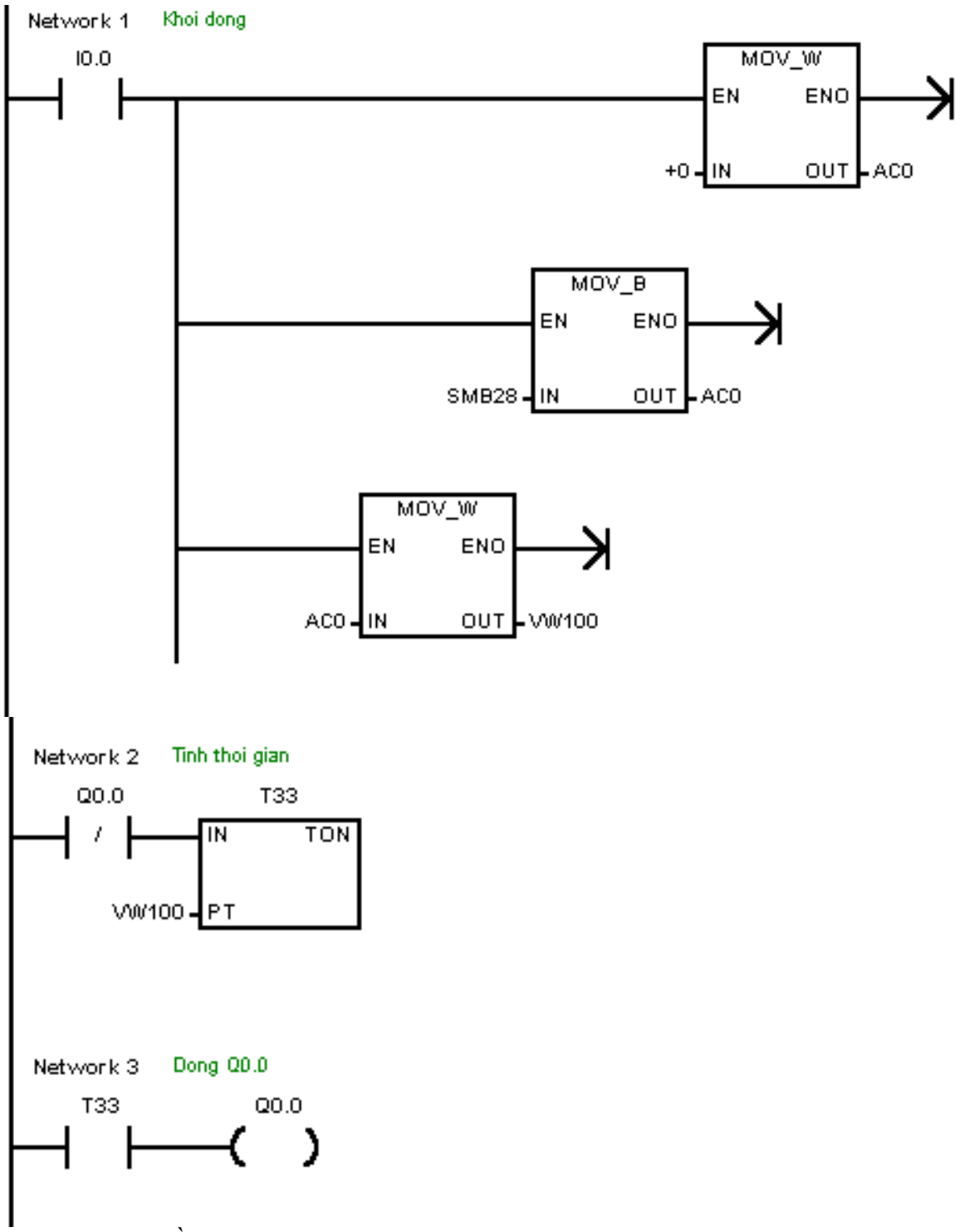
4.4. Hiệu chỉnh tín hiệu Analog

Trên CPU S7-200 có 2 biến trở (2 biến trở này nằm dưới nắp của mô đun), có thể sử dụng 2 biến trở này để tăng hoặc giảm giá trị được lưu trữ trong các Byte của vùng nhớ Special Memory (SMB 28 và SMB 29). Các giá trị chỉ đọc trong 2 Byte này có thể được sử dụng cho nhiều chức năng khác nhau. Chẳng hạn, dùng để cập nhật giá trị hiện hành cho 1 Timer, một Counter, thay đổi giá trị đặt trước, đặt các giá trị giới hạn.

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

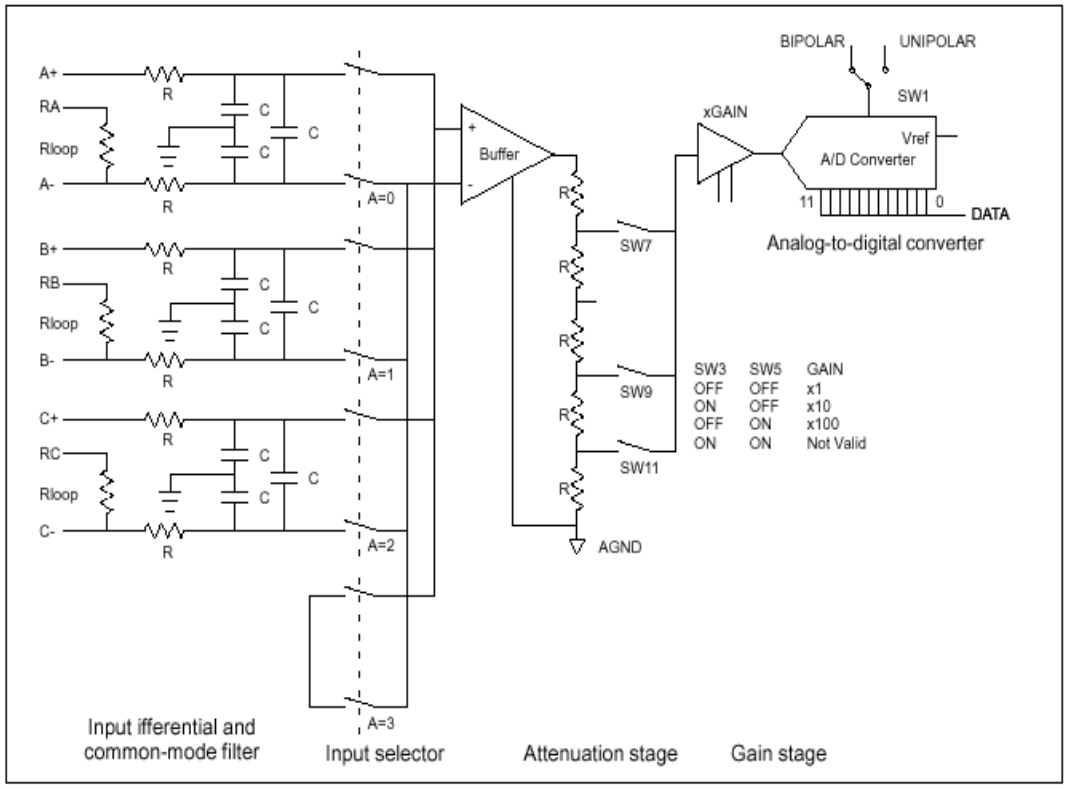
Để thực hiện điều chỉnh này, phải sử dụng một Tuộc vít nhỏ: nếu xoay biến trở sang phải là tăng giá trị, còn xoay sang trái là giảm giá trị. Dưới đây là một ví dụ ứng dụng:

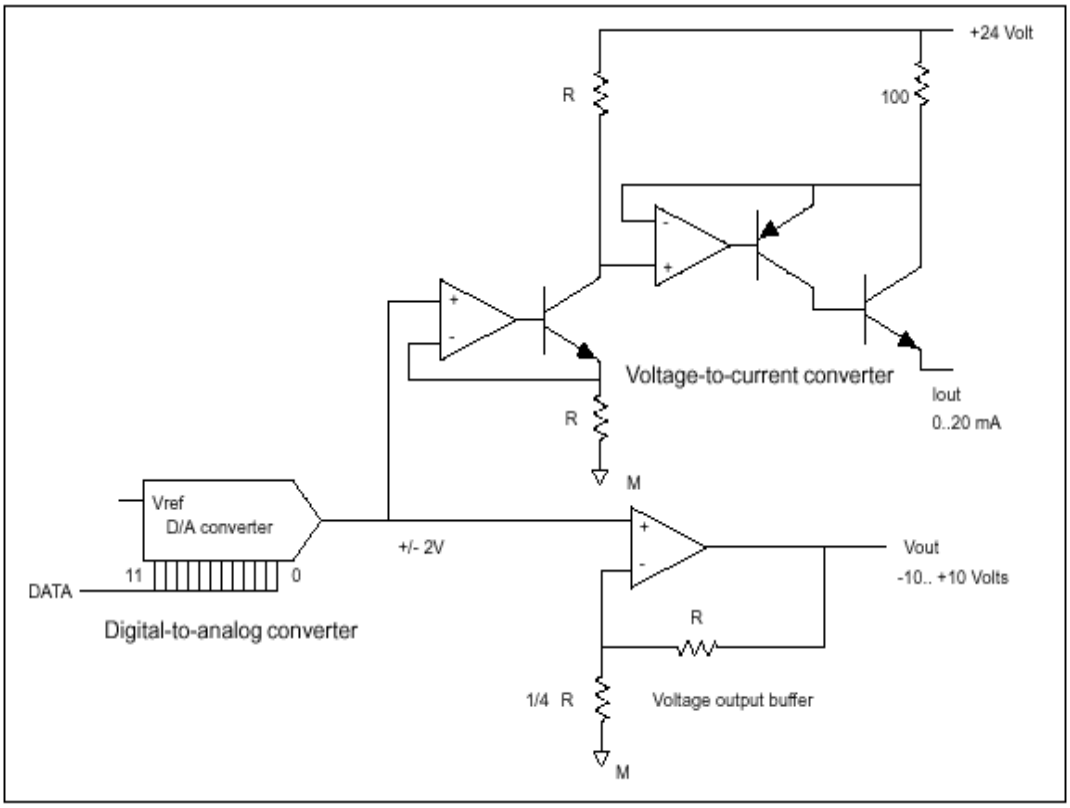
Timer T33 đóng tiếp điểm khi VW 100 đạt giá trị đặt trước



Sau đây là sơ đồ nguyên lý mạch của ***modul EM 235 3AI/ 1AO***

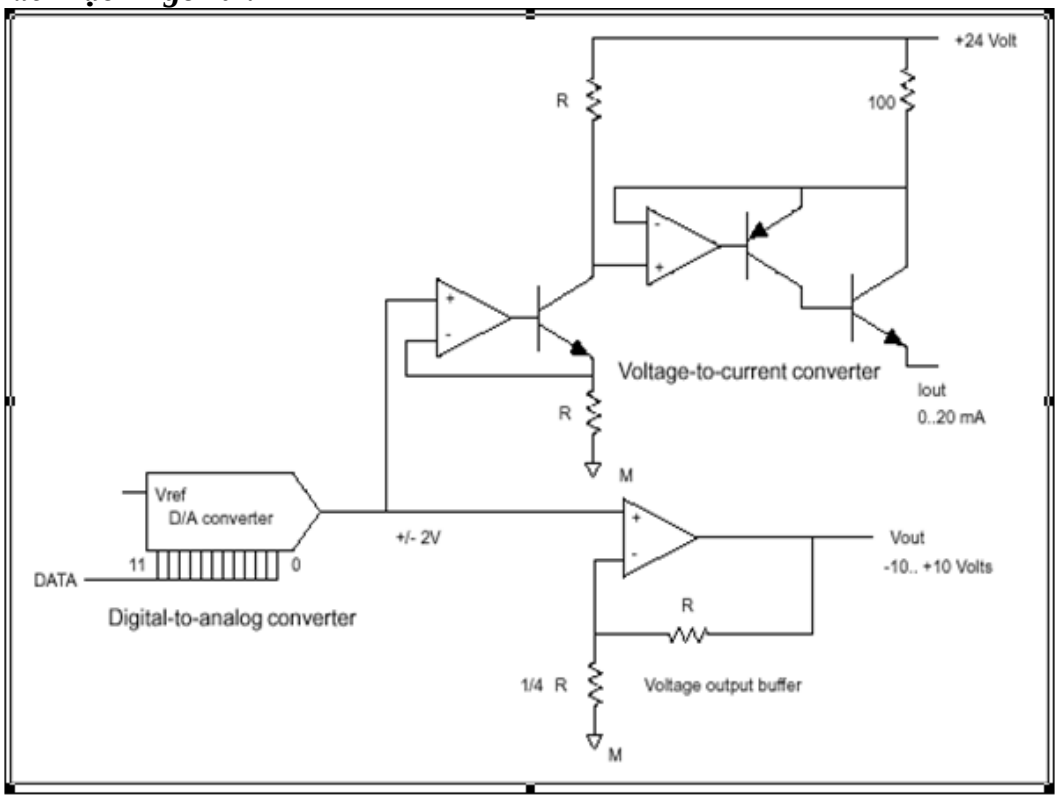
Sơ đồ mạch ngõ vào :





Hình 4-6

Sơ đồ mạch ngõ ra :



Hình 4-7

4.5. Giới thiệu về module analog PLC S7 200

PLC S7 200 có các module analog mở rộng như sau:

- EM 231: Gồm có bốn ngõ vào analog.
- EM 232: Gồm có hai ngõ vào analog.
- EM 235: Gồm có bốn ngõ vào analog và 1 ngõ ra analog.

4.5.1 Đặc tính chung

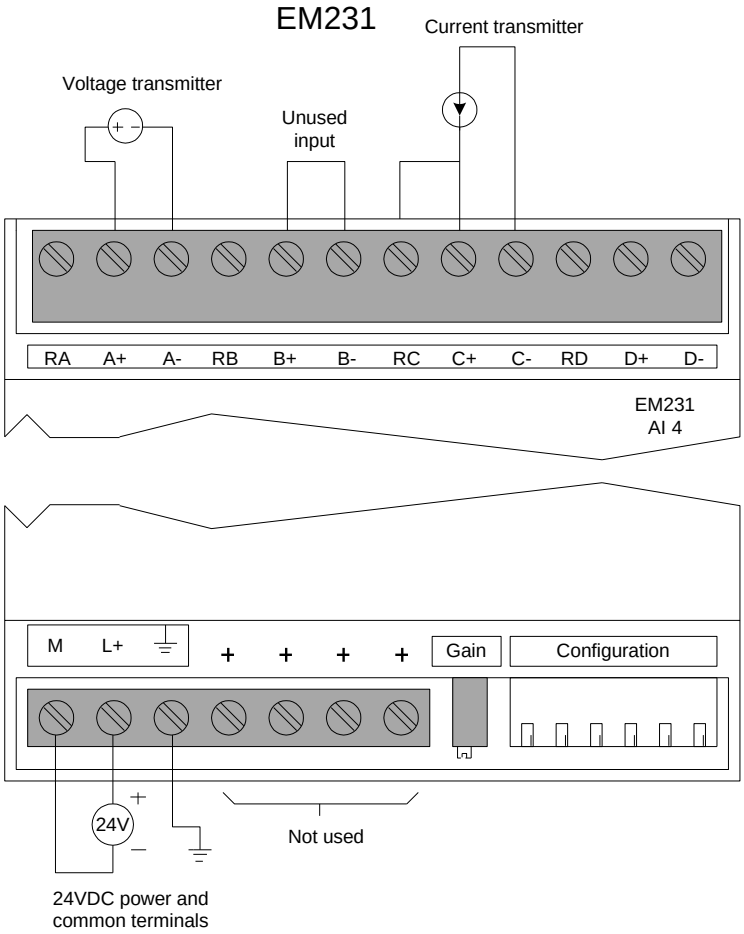
- Trở kháng vào # 10M
- Bộ lọc đầu vào -3db tại 3.1 Khz.
- Điện áp cực đại cung cấp cho mô đun: 30VDC
- Dòng điện cực đại cung cấp cho mô đun: 32mA.
- Có led báo trạng thái.
- Có núm chỉnh OFFSET và chỉnh độ lợi.

Hình 4- 8

4.5.2 Đặc tính kỹ thuật của mô đun Analog EM 231

a/ Đầu vào:

- Số đầu vào: 4, độc lập nhau.
- Chức năng bảo vệ cực tính: 0



Hình 4-9

- Phạm vi đầu vào: 0 ~ 50mV // 0 ~ 100mV // 0 ~ 500mV // 0 ~ 1V // 0 ~ 10V // 0 ~ 20mA // 25mA // 50mA // 100mA // 250mA // 500mA // 1V // 2,5V // 5V // 10V.
- Điện áp đầu vào cho phép với đầu vào điện áp, tối đa là 30V.

- Dòng điện đầu vào cho phép với đầu vào dòng điện tối đa là 32mA
- Cách ly điện: không.
- Bộ chuyển đổi: 12 bit.
- Thời gian chuyển đổi từ tương tự sang số : 250#s.
- Độ phân giải 12 bit
- Điện áp chế độ chân dung tối đa : 12V
- Triệt nhiễu: 40dB, DC đến 60 Hz (không triệt nhiễu tần số).
- Phạm vi có thể thay thế của các giá trị chuyển đổi:
- Các tín hiệu không có cực tính: 0 ~ 32.000
- Các tín hiệu có cực tính: - 32.000 ~ +32.000.
- Khả năng tuyến tính hoá đặc tính: không
- Khả năng bù nhiệt độ: không.
- Hiển thị chuẩn đoán lỗi: LED, EXTF.

b/ Đầu ra

Số đầu ra: 1

Phạm vi đầu ra:

- Đầu ra điện áp: -10V ~ +10V
- Đầu ra dòng điện 0 ~ 20mA.

Điện trở tải:

- Với đầu ra điện áp nhỏ nhất là: 5kΩ
- Với đầu ra dòng điện lớn nhất là: 0,5kΩ

Độ phân giải:

- Với đầu điện áp nhỏ nhất là: 12 bit
- Với đầu ra dòng điện lớn nhất là: 11 bit

Thời gian đặt:

- Với đầu ra điện áp là 100#s
- Với đầu ra là dòng điện 2ms.

Phạm vi có thể hiển thị được của giá trị chuyển đổi:

- Các tín hiệu đơn cực tính: - 32.000 ~ + 32.000.

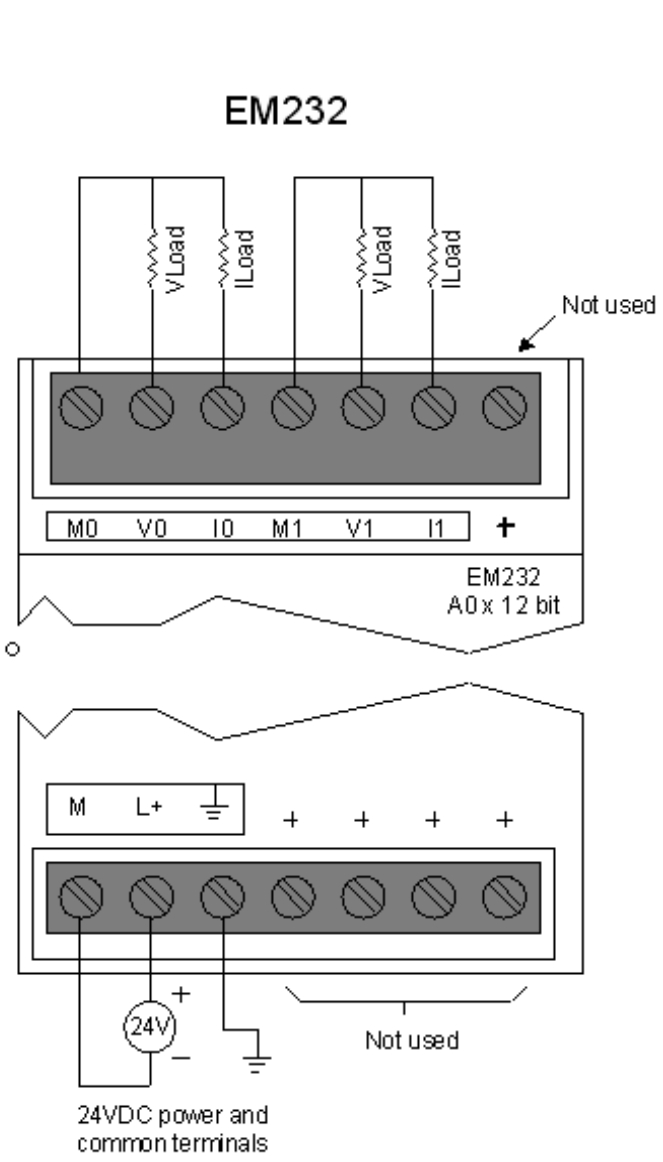
Giới hạn lỗi hoạt động ở 60^0C

- Điện áp: 2%
- Dòng điện: 2%

Giới hạn lỗi hoạt động ở 25^0 C :

- Điện áp: 0,5%
- Dòng điện: 0,5%

Tiêu thụ dòng điện:



Hình 4-10

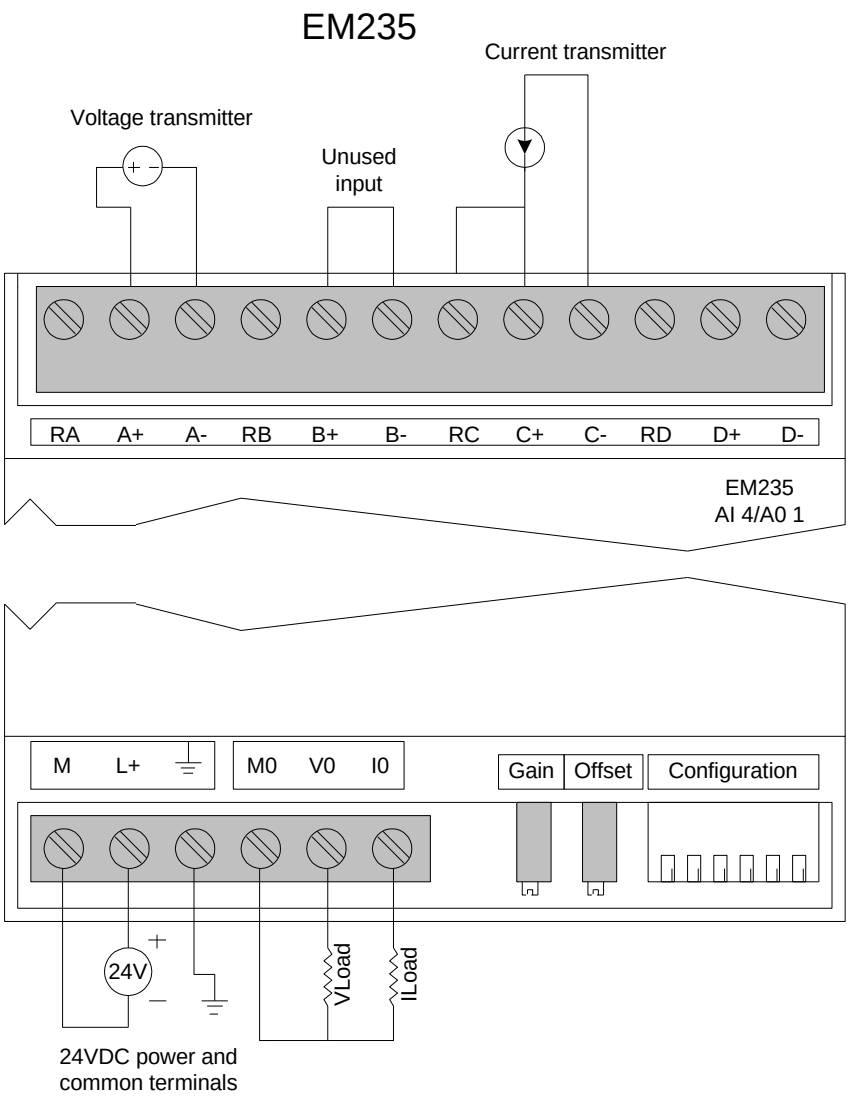
- Từ bus backplane (VDC): 30mA.
- Từ nguồn cấp sensor hoặc nguồn cấp ngoài: 60W.

Tổn thất công công suất: 2W

Kích thước (W*H*D) mm: 71,2*80*62

Trọng lượng: 186g.

Sơ đồ kết nối các thiết bị ngoại vi, sử dụng theo dạng áp và dòng. Các contact (Switch) để lựa chọn phạm vi ngõ vào (contact ở một trong hai vị trí ON và OFF). Contact 1 lựa chọn cực tính áp ngõ vào: ON đối với áp đơn cực, OFF đối với áp lưỡng cực; contact 2, 3, 4, 5, 6, chọn phạm vi điện áp.



Hình 4-11

c. Các chú ý khi cài đặt ngõ ra analog:

Chắc chắn là nguồn 24VDC cung cấp không bị nhiễu và ổn định.

Xác định được mô đun.

Dùng dây cảm biến ngắn nhất nếu có thể.

Sử dụng dây bọc giáp cho cảm biến và dây chỉ dùng cho một mình cảm biến.

Tránh đặt các dây tín hiệu song song với các dây có năng lượng cao. Nếu hai dây bắt buộc phải gặp nhau thì bắt chéo chúng về phía bên phải.

Kiểm tra các ngõ vào analog.

Với chương trình bất kỳ đang chạy trong PLC thì ta có thể thấy giá trị sau khi biến đổi A/D ở các kênh analog thông qua chức năng Debug > Chart Status với Chart Status có liệt kê AIW0 đến AIW6. Ta chỉnh các biến trở bên ngoài và khảo sát những thay đổi ở các word ATW0 đến AIW6.

Lập lại tín hiệu vào:

Bộ thí nghiệm S7 200 đang đặt cấu hình nhập và xuất cùng tỉ lệ, có nghĩa là nếu đọc vào x V và xuất lại ngõ ra thì cũng được x V. (giá trị ngõ vào cho phép là từ 0 đến 10V)

Hãy viết chương trình sử dụng các ngõ vào I0.0 đến I0.3 để chọn ngõ ra lập lại giá trị analog của kênh vào nào (I0.0 ứng với kênh 0, I0.1 ứng với kênh 1, I0.2 ứng với kênh 2 và I0.3 ứng với kênh 3; Nghĩa là tương ứng với các kênh A, B, C và D ở PLC).

Thí dụ: Nhập đoạn chương trình sau thực hiện xuất giá trị analog ra lập lại ở kênh 0.

```
LD     I0.0
MOVW   AIW0, AQW0
```

Nếu các ngõ vào I0.0 đến I0.3 sử dụng loại công tắc NO, hãy viết chương trình cho trường hợp này.

Một số xử lý đơn giản trên analog

Gọi giá trị analog là Y volt và analog vào kênh A là XA volt, kênh B là XB volt, kênh C là XC volt, kênh D là XD volt; gọi M (thí dụ sử dụng VW0) là hằng số cần thực hiện với dữ liệu analog và chú ý M là số nguyên.

Thực hiện Y = M*XA

Ta chỉ cần viết 1 đoạn chương trình như sau:

```
MOVW   AIW0, MW0        // lấy số liệu XA
*I     M, MW0
MOVW   MW0, AQW0        // xuất ra Y = XA*M
```

Thực hiện Y = XA / M

Ta chỉ cần viết 1 đoạn chương trình như sau:

```
MOVW   AIW0, MW0        // lấy số liệu XA
/I     M, MW0
MOVW   MW0, AQW0        // xuất ra Y = XA / M
```

Chú ý đây là phép chia nguyên nên trị số sẽ không chính xác

Thực hiện Y = (XA + XB + XC + XD) / 4

Ta chỉ cần viết 1 đoạn chương trình sau:

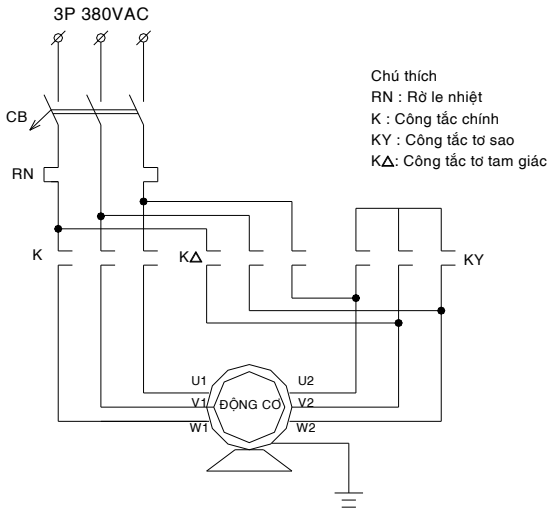
```
MOVW   AIW0, MW0        // MW0 = XA
+I     AIW2, MW0        // MW0 = XA + XB
+I     AIW4, MW0        // MW0 = XA + XB + XC
+I     AIW6, MW0        // MW0 = XA + XB + XC + XD
/I     4, MW0           // MW0 = (XA + XB + XC + XD) / 4
MOVW   MW0, AQW0        // xuất ra Y = (XA + XB + XC + XD) / 4
```

Chú ý là việc thực hiện chỉ đúng khi các trị số tính toán không bị tràn.

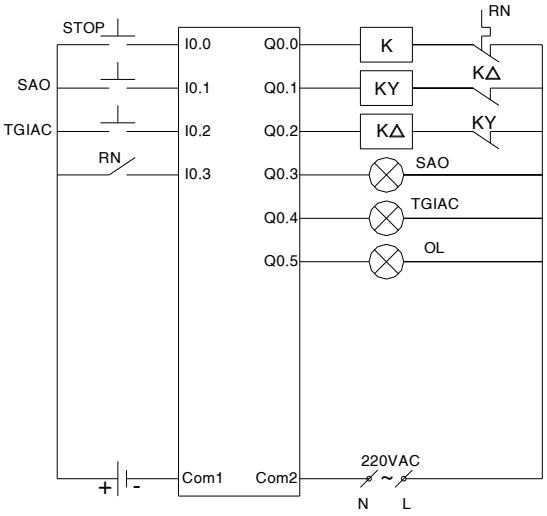
Chương 5
MỘT SỐ BÀI TOÁN ỨNG DỤNG

5.1. Khởi động động cơ Sao/Tam giác

Mở máy động cơ 3 pha bằng phương pháp đổi nối sao-tam giác dùng 3 nút nhấn SAO,TAM GIAC, OFF.



SƠ ĐỒ MẠCH ĐỘNG LỰC



SƠ ĐỒ KẾT NỐI PLC

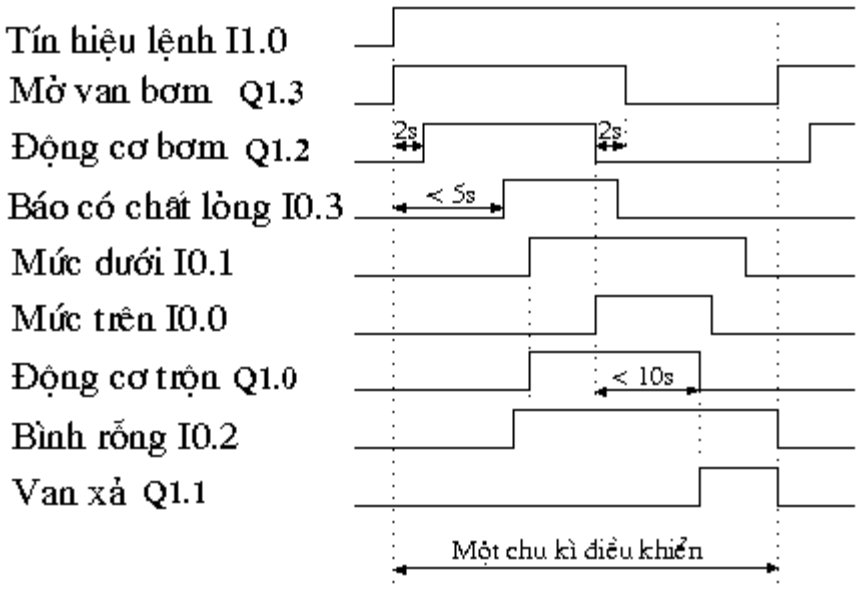
Yêu cầu: Lập trình kết nối hệ thống sao cho khi

- Nhấn nút SAO động cơ chạy theo đấu nối sao, đèn SAO sáng.Nhấn STOP động cơ dừng đèn tắt
- Nhấn nút TGIAC động cơ chạy theo đấu nối tam giác, đèn TGIAC sáng. Nhấn STOP động cơ dừng đèn tắt.
- Nếu động cơ đang chạy, muốn chuyển đổi chế độ chạy sao hay tam giác thì phải nhấn nút STOP để dừng động cơ trước, sau đó nhấn nút SAO hay TGIAC để động cơ chạy theo chế độ sao hay tam giác.
- Nếu động cơ gặp sự cố như quá tải, rờ le nhiệt RN tác động, động cơ dừng, đèn SAO và TGIAC đều tắt, đèn OL sáng nhấp nháy.

Khi sự cố được khắc phục nhấn nút RESET ở rờ le nhiệt , sau đó nhấn nút SAO hay TGIAC để động cơ chạy.

5.2. Hệ thống trộn sơn tự động

Hãy viết chương trình điều khiển hệ thống trộn sơn theo giản đồ sau



5.3. Điều khiển mô hình băng tải

Hãy viết chương trình điều khiển theo yêu cầu sau:

Ấn nút Start 1 động cơ 1 khởi động dẫn động cho băng tải 1 hoạt động.Sản phẩm được vận chuyển, khi gặp cảm biến 1, bộ cảm biến nhận biết có sản phẩm trên dây chuyền và ra lệnh cho các động cơ 2 và 3 khởi động dẫn động cho băng tải 2 và 3 hoạt động, đồng thời đếm các sản phẩm được vận chuyển trên băng tải. Khi sản phẩm qua cảm biến 2, cảm biến này sẽ xác định sản phẩm là chính phẩm hay phế phẩm để phân loại. Nếu sản phẩm là chính phẩm(sản phẩm thấp), thì bộ khí nén không tác động, sản phẩm sẽ đi thẳng vào rãnh phải, còn nếu là phế phẩm(sản phẩm cao) thì bộ khí nén tác động đẩy sản phẩm về phái trái và phế phẩm sẽ trôi xuống rãnh trái.

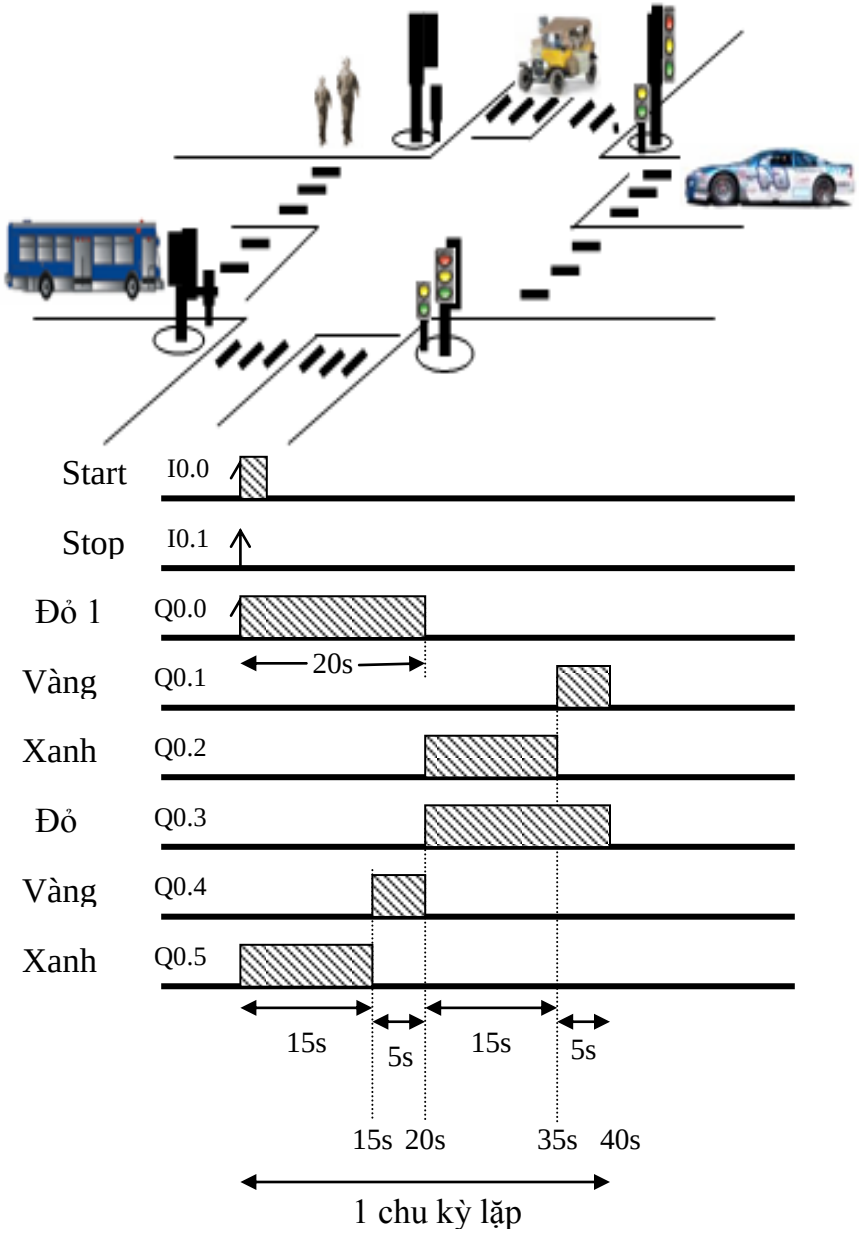
Trong khoảng thời gian hoạt động nếu sau một khoảng thời gian chờ không có sản phẩm vào băng tải(nhận biết qua cảm biến 1) thì băng tải 2 và 3 dừng lại để tiết kiệm năng lượng.

Trong trường hợp băng tải đang hoạt động bị sự cố và sản phẩm bị mắc lại trong băng tải 2 và 3, người vận hành ấn nút START 2 và START 3 cho băng tải 2 và 3 chạy cho đến khi sản phẩm ra khỏi băng tải. Hai nút này được thiết kế không duy trì nên nhả tay ra băng tải dừng lại ngay.

Khi băng tải đang chạy, muốn dừng hoặc xảy ra sự cố muốn dừng ấn nút STOP thì toàn bộ hệ thống sẽ dừng.

5.4. Điều khiển mô hinh đèn giao thông ngã tư

Hãy viết chương trình điều khiển đèn giao thông ngã tư theo giản đồ sau



TÀI LIỆU THAM KHẢO

[1]. *Giáo trình Thiết bị điện*, Lê Thành Bắc, NXB KHKT, Hà nội (2001)

[2]. *Lập trình với S7-300*, Nguyễn Xuân Công, ĐH SPKT Hưng yên

[3]. *Bài giảng Hệ thống thông tin đo lường,* Phạm Thượng Hàn, ĐHBK Hà nội

[4]. *Điều khiển Logic Lập trình*, Tăng Văn Mùi, NXB Thống Kê, TP HCM (2003)

[5] *Lập trình với S7-200,* Phan Xuân Minh- Nguyễn Doãn Phước,NXB Nông Nghiệp