TỔNG QUAN VỀ PLC

PHẦN II

CHƯƠNG 1

LỊCH SỬ PHÁT TRIỂN

PHẦN II. TỔNG QUAN VỀ PLC

CHƯƠNG 1. LỊCH SỬ PHÁT TRIỂN

I. GIỚI THIỆU

Dạng lập trình phổ biến nhất của PLC là vẽ sơ đồ hình thang (LAD-ladder), từ dạng LAD có thể chuyển sang dạng liệt kê lệnh (STL), sơ đồ khối (FBD). Những chương trình có thể nhập từ thiết bị lập trình chuyên dùng đến PLC được nối với nhau qua cổng lập trình (Programming Port). Các thiết bị lập trình như máy lập trình chuyên dụng PG 720, PG 740, PG 760 được nối với nhau qua cổng giao tiếp đa năng

(MPI-Multipoint Interface) hoặc dùng máy tính cá nhân PC được nối qua cổng COM.

Ngôn ngữ dùng để lập trình cho PLC S7-300 là Simatic Step 7. Trong chươmg trình Step 7 có những ngôn ngữ lập trình có thể dùng tuỳ vào sở thích và sự hiểu biết đó là STL, FBD, LAD.

Từ dạng STL có thể chuyển sang sơ đồ dạng FBD và LAD, nhưng từ LAD và FBD thì chỉ có thể hoặc không thể chuyển sang dạng STL.

II. LỊCH SỬ PHÁT TRIỂN PLC

Bộ điều khiển lập trình PLC (programmable logic controller) được tạo ra từ ý tưởng của một nhóm kỹ sư thuộc hãng general motors, Mỹ vào năm 1968 nhằm để thay thế những mạch điều khiển bằng Rơle và thiết bị rời rạc, cồng kềnh. Bộ điều khiển lập trình được sản xuất thương mại đầu tiên trên thế giới do hãng Modicon sản xuất có tên là Modicon 084. Các kỹ sư hãng General Motors đã đề ra các chỉ tiêu cho bộ điều khiển lập trình như sau:

- Dễ lập trình và thay đổi chương trình điều khiển, sử dụng thích hợp trong các nhà máy.
- Cấu trúc dạng module dễ bảo trì và sửa chữa.
- Độ tin cậy lớn trong môi trường sản xuất của nhà máy công nghiệp.
- Dùng linh kiện bán dẫn nên có kích thước nhỏ hơn mạch Rơle, với chức năng tương đương.
- Giá thành rẻ, có khả năng cạnh tranh.

Vào giữa thập niên 70, công nghệ PLC nổi bậc nhất là điều khiển tuần tự theo chu kỳ và theo bit dựa trên nền tảng của CPU. Thiết bị AMD 2901 và AMD 2903 trở nên phổ biến mà ngày nay một vài loại PLC vẫn còn dựa trên nền tảng của AMD 2903. Lúc này phần cứng cũng phát triển: Bộ nhớ lớn hơn, số lượng ngõ vào\ra nhiều hơn, nhiều module chuyên dùng hơn. Vào năm 1976, PLC có khả năng truyền thông, dùng điều khiển các ngõ vào/ra ở xa với khoảng cách khoảng 200 met.

Thập niên 80, bằng nổ lực tiêu chuẩn hoá hệ giao tiếp với giao diện tự động hoá của hãng General Motor (MAP-Manufactoring Automation Protocal), kích thước của PLC giảm, có thể lập trình bằng biểu tượng trên máy tính cá nhân thay vì thiết bị lập trình đầu cuối chuyên dụng hoặc lập trình bằng tay.

Thập niên 90, những giao diện phần mềm mới có cấu trúc lệnh giảm và cấu trúc

của những giao diện được cung cấp từ thập niên 80 đã được đổi mới. Hiện nay những loại PLC có thể lập trình bằng ngôn ngữ sơ đồ khối (FBD), cấu trúc lệnh (STL), sơ đồ hình thang (LAD).

Có một số thuật ngữ dùng để mô tả bộ điều khiển lập trình, tuỳ vào những nước và khu vực mà có những tên gọi khác nhau:

- PC-Programable Controller (Anh).
- PLC- Programable logic Controller (Mỹ).
- PBS-Programable Binary System (Thuỵ Điển).

Hai thuật ngữ PLC và PBS đều thể hiện bộ điều khiển lập trình làm việc với tín hiệu nhị phân. Thực tế, hầu hết các bộ điều khiển lập trình đều có khả năng xử lý tín hiệu liên tục (Analog) nên nó không nói lên hết khả năng của bộ điều khiển lập trình. Vì PC được viết tắc từ Programable Controller (bộ điều khiển lập trình) nhưng máy tính cá nhân cũng được viết tắc là PC (Personal Computer). Do vậy, để tránh nhầm lẫn người ta thường sử dụng PLC để chỉ bộ điều khiển lập trình.

Có nhiều hãng sản xuất PLC như Siemens, Allen-Bradley, Omrom, General Motor, GE Fanuc, Modicon, Tele Monanique...

Những loại PLC của Siemens gồm có các họ: Simatic S5, Simatic S7, Simatic 500/505. Mỗi họ PLC của Siemens có nhiều phiên bản khác nhau. Ví dụ, họ Simatic S7 có S7-200, S7-300, S7-400, S7-500. Trong đó, mỗi loại S7 có nhiều loại CPU khác nhau như S7-300 có CPU 312, CPU 313, CPU 314, CPU 315, CPU 316, CPU 388-4, CPU 614...

CẤU TRÚC CỦA S7-300

CHƯƠNG 2

CHƯƠNG 2 CẤU TRÚC CỦA S7-300

I. CẤU TRÚC CHƯƠNG TRÌNH S7-300

1.1. GIỚI THIỆU

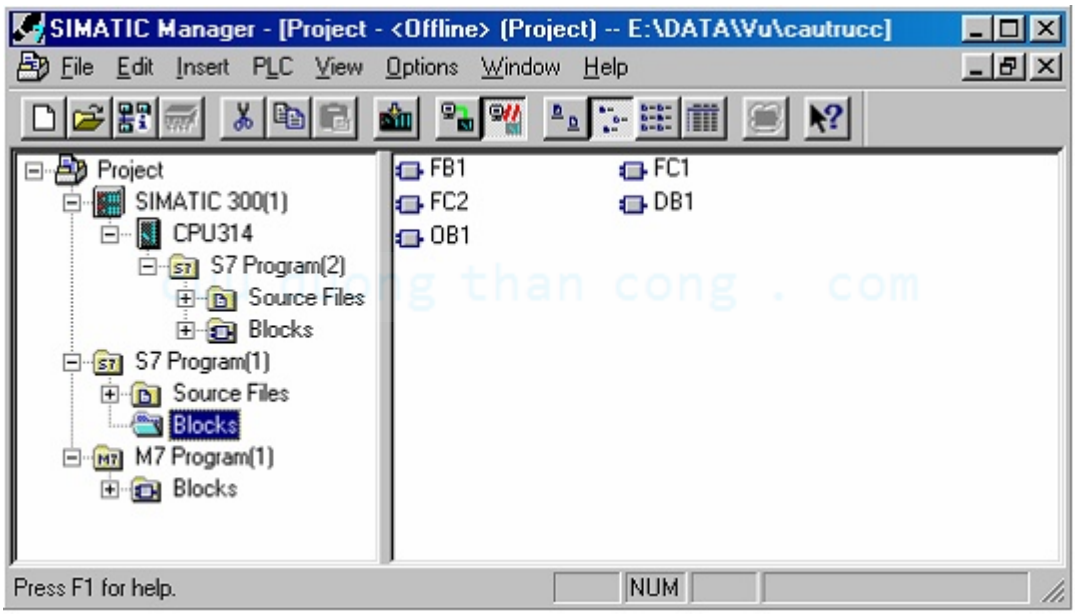
Phần mềm STEP 7 dùng để lập trình cho các họ PLC Simatic S7 (S7-300, S7-400), kết hợp với máy tính PC hoặc thiết bị lập trình chuyên dụng PG 720, PG 740, PG 760.

Để soạn thảo chương trình, sau khi cài đặt chương trình Simatic, kích hoạt Simatic Manager ở màn hình nền hoặc vào Menu Start/Progrm/ Simatic/ Simatic Manager.

Hệ quản lý Simatic dùng để quản lý các đề án và chương trình người dùng của STEP
7. Nó là chương trình chính và hiện lên màn hình nền của thiết bị lập trình.

1.2. CẤU TRÚC CHƯƠNG TRÌNH S7-300

Họ Simatic S7 có cấu trúc chương trình giống nhau. Chương trình trong đề án được sắp xếp theo cấu trúc hình cây giống như cấu trúc hình cây ở trong Window nhưng biểu tượng của đối tượng thì khác.



Cấu trúc chương trình được sắp xếp theo cấp bậc:

Cấp 1: Chứa biểu tượng của dự án (Project). Mỗi dự án tượng trưng cho một cơ sở dữ liệu, nơi lưu trữ các dữ liệu liên quan đến chương trình.

Cấp 2: Chứa các trạm (Station), các chương trình (Program), các mạng cấp dưới (Subnet).

- Các trạm là nơi lưu trữ dữ liệu các về thông tin về cấu hình phần cứng và thông số chỉ định của các khối. Đây là điểm khởi đầu cho cấu hình phần cứng.

- Các chương trình S7/M7 Progam, là điểm khởi đầu để viết chương trình. Tất cả các chương trình và thông số chỉ định về khối của S7 được lưu trữ trong thư mục chương trình S7. Trong thư mục này chứa các thư mục khác dành cho các khối và các tập tin nguồn của chương trình.

- Mạng cấp dưới gồm có MPI (Muti-point Interface), Profibus, mạng Ethernet công nghiệp. Đây là thành phần dùng để nối mạng.

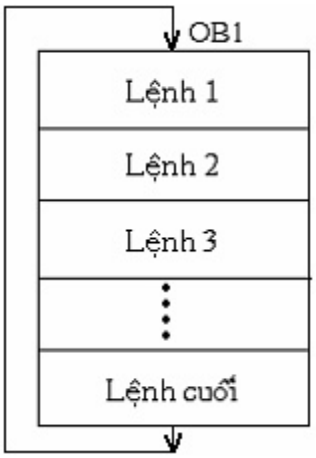
Cấp 3 và các cấp khác: Tuỳ thuộc vào kiểu đối tượng của cấp 2 mà cấp 3 và các cấp khác sẽ có thành phần khác nhau. Cấp 3 thường chứa các source file (tập tin nguồn), Blocks (các khối), cấu hình CPU...

II. CÁCH VIẾT CHƯƠNG TRÌNH TRONG STEP7

Có thể viết chương trình ở hai dạng: Lập trình tuyến tính và lập trình theo cấu trúc.

2.1. LẬP TRÌNH TUYẾN TÍNH

Lập trình tuuyến tính là toàn bộ chương trình đều nằm trong một khối, khối này là OB1. Kiểu lập trình này giống như dạng điều khiển dùng Rơle nhưng được thay thế bằng một bộ điều khiển lập trình PLC. Chương trình trong khối OB1 sẽ được hệ điều hành quét thường xuyên theo chu kỳ từ lệnh đầu tiên cho tới lệnh cuối cùng và sau đó chương trình được lặp lại từ đầu. Loại chương trình này phù hợp với những bài toán tự động điều khiển nhỏ, chương trình không phức tạp.



2.2. LẬP TRÌNH THEO CẤU TRÚC

Chương trình được chia thành nhiều khối, mỗi khối có một nhiệm vụ riêng. Loại lập trình này phù hợp với loại bài toán có nhiều nhiệm vụ, điều khiển phức tạp. Khối tổ chức OB1 chứa những lệnh để gọi những khối khác theo một trình tự đã được xác định trước.

Trong S7-300 có những loại khối cơ bản sau:

Khối OB (Organization Block): Là khối tổ chức và quản lý chương trình, có nhiều khối OB với các dữ liệu khác nhau và chúng chỉ được gọi bởi hệ điều hành. Khối được CPU xử lý thường xuyên và theo chu kỳ là khối OB1, chương trình người dùng sẽ được chứa trong khối này. Còn các khối OB khác thì làm các nhiệm vụ khác như: Ngắt thời điểm, ngắt thời gian trễ, ngắt chu kỳ, ngắt phần cứng, ngắt lỗi không đồng bộ, ngắt lỗi đồng bộ, khởi động.

Khối FC (Function Block): Khối chức năng FC giống như một chương trình con hoặc là một hàm (có thể truyền tham số). Có 128 khối FC trong CPU 314. Chương trình viết trong khối này không được vượt quá 8Kbyte. Muốn chương trình có hiệu lực thì nó phải được gọi vào khối OB mà cụ thể đối với chương trình người dùng thì khối FC được gọi vào trong khối OB1.

Khối FB (Function Block): Khối FB là khối FC đặt biệt, có thể trao đổi một lượng dữ liệu lớn với các khối chương trình khác. Dữ liệu được đặt trong một chương trình khác có tên khối dữ liệu tức thời (Instance Data Block). Có 128 khối FB trong CPU 314. Khi gọi một khối FB thì phải xác định số lượng khối DB được mở ra tự động.

Khối DB (Data Block) là khối dữ liệu lưu trữ dữ liệu người dùng. Dữ liệu trong mỗi khối DB không được vượt quá 8Kbyte. Có 128 khối FB trong CPU 314. Dữ liệu trong khối DB không bị mất đi khi khối được gọi. Có hai dạng khối dữ liệu là khối dữ liệu toàn cục (Global Data) và khối dữ liệu cục bộ (Instance Data).

• Khối dữ liệu toàn cục chứa các thông tin có thể truy cập từ tất cả các khối Logic có trong chương trình.

• Khối dữ liệu cục bộ được dùng bởi một FB. Dữ liệu trong mỗi khối DB chỉ dùng cho một FB. Tuy nhiên một khối FB có thể các khối DB khác nhau ở mỗi lần gọi. Nếu sửa đổi FB thì phải tạo mới DB một lần nữa.

CPU có hai thanh ghi khối dữ liệu DB và DI. Vì vậy, có thể mở hai khối DB cùng một lúc.

Các khối được liên kết với nhau bởi các lệnh gọi khối, chuyển khối và để có thể làm việc được thì phải được gọi vào trong khối OB1.

Khối SFC (System Functon): Chức năng hệ thống là một chức năng đặt biệt được tích hợp trong hệ điều hành của CPU S7 mà có thể được gọi giống như một chức năng FC vào trong chương trình người sử dụng khi cần thiết.

Khối SFB (System Functon Block): Khối chức năng hệ thống là một khối chức năng đặt biệt được tích hợp trong hệ điều hành của CPU S7 mà có thể được gọi giống như một khối chức năng FB vào trong chương trình người sử dụng khi cần thiết.

III. KẾT NỐI PHẦN CỨNG S7-300

3.1. ĐẶT ĐIỂM CỦA PLC

PLC là loại thiết bị cho phép thực hiện linh hoạt các thuật toán điều khiển thông qua ngôn ngữ lập trình thay cho việc phải dùng mạch số hoặc mạch dùng Rơle. Do vậy, PLC trở nên nhỏ gọn, dễ thay đổi thuật toán và dễ trao đổi thông tin với môi trường xung quanh bằng cách nối mạng PLC. Toàn bộ chương trình điều khiển được lưu trong bộ nhớ dưới dạng khối chương trình (OB, FC, FB) và thực hiện lặp lại theo chu kỳ quét.

3.2. CẤU TRÚC CHUNG CỦA PLC

PLC gồm các phần chính như ngõ vào, ngõ ra, phần điều khiển (CPU).

Khối vào	Khối điều khiển (CPU)		Khối ra
Chuyển đổi tín hiệu ngõ vào sao cho phù hợp với tín hiệu để CPU xử lý		Mạch khuếch đại	Cơ cấu chấp hành

3.3. KHỐI VÀO

Có hai loại ngõ vào là ngõ vào số (DI_Digital Input) và ngõ vào tương tự (AI_ Analog Input).

Ngõ vào số chỉ được kết nối với các bộ chuyển đổi tạo ra tín hiệu nhị phân như nút nhấn, công tắt, thermotat (cảm biến nhiệt), cảm biến nhị phân...

Ngõ vào tương tự chỉ nối với các bộ chuyển đổi tạo ra tín hiệu tương tự như các loại cảm biến vật lý. Bộ chuyển đổi tín hiệu tương tự nào thì dùng với khối Analog đó.

3.4. KHỐI RA

Có hai loại ngõ ra là ngõ ra số (DO_ Digital Output) và ngõ ra tương tự (AO_Analog Output).

Ngõ ra số chỉ được kết nối với các cơ cấu chấp hành nhận tín hiệu nhị phân như contactor, van từ... Có 3 loại ngõ ra số là ngõ ra transistor dùng nguồn 24VDC, ngõ ra rơle dùng nguồn AC và DC, ngõ ra triac dùng nguồn AC.

Ngõ ra tương tự chỉ nối với các cơ cấu chấp hành nhận tín hiệu tương tự như biến tần. Có hai loại ngõ ra tương tự là ngõ ra dạng dòng điện và ngõ ra dạng điện áp.

3.5. KHỐI CPU

Khối CPU dùng để xử lý, lưu trữ theo chương trình có sẵn trong bộ nhớ chương trình và xuất ra ngõ ra đến khối ra. Ngoài ra PLC còn có các khối chức năng đặt biệt khác như bộ đếm (Counter), bộ thời gian (Timer)... và những khối hàm chuyên dụng.

IV. PHẦN CỨNG CỦA S7-300

4.1. TÍNH NĂNG CỦA S7-300

Những đối tượng điều khiển có số tín hiệu đầu vào, đầu ra cũng như chủng loại tín hiệu vào/ra khác nhau. Các bộ điều khiển lập trình PLC được thiết kế linh hoạt, không bị cứng hoá về cấu hình làm cho chương trình ứng dụng được mềm dẻo. S7-300 có những tính năng sau:

- Hệ thống điều khiển kiểu modul nhỏ gọn dùng cho các ứng dụng trong phạm vi trung bình.

- Có nhiều loại CPU khác nhau.
- Có nhiều modul mở rộng.
- Có thể mở rộng đến 32 modul.
- Có bus nối tích hợp phía sau các modul.
- Có thể nối mạng:

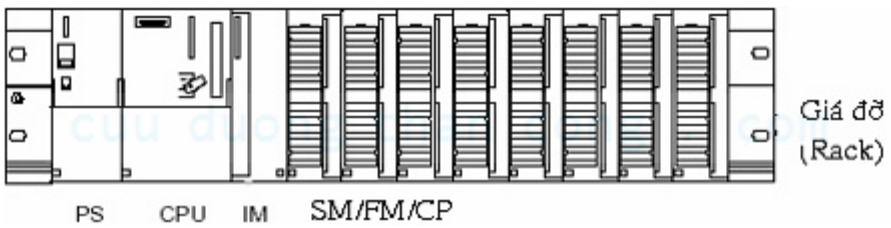
. Multi-Point Interface.
. Profibus.

Industrical Ethernet. • Thiết bị lập trình PG trung tâm có thể truy cập đến các modul. • Không hạn chế số rãnh. • Cài đặt cấu hình và thông số với công cụ trợ giúp HW-CONFIG (Hardware Configeration).

4.2. CÁC MODUL CỦA PLC S7-300

Để chương trình được mềm dẻo thì S7-300 chia thành các modul. Số các modul được sử dụng nhiều hay ít tuỳ theo từng ứng dụng nhưng tối thiểu bao giờ cũng phải có một modul chính là modul CPU, các modul còn lại là những modul nhận/truyền tín hiệu với đối tượng điều khiển như động cơ, các đèn báo, các rơle, các van từ... Chúng được gọi chung là các modul mở rộng. Tất cả các modul được gắn trên những giá đỡ (Rack).

Cấu hình của một giá đỡ S7-300 như sau:



4.2.1. MODUL CPU

Modul CPU chứa bộ vi xử lý, hệ điều hành, bộ nhớ, các bộ thời gian, bộ đếm, cổng truyền thông và có thể còn có các cổng vào/ra số gọi là cổng và/ra Onboard...

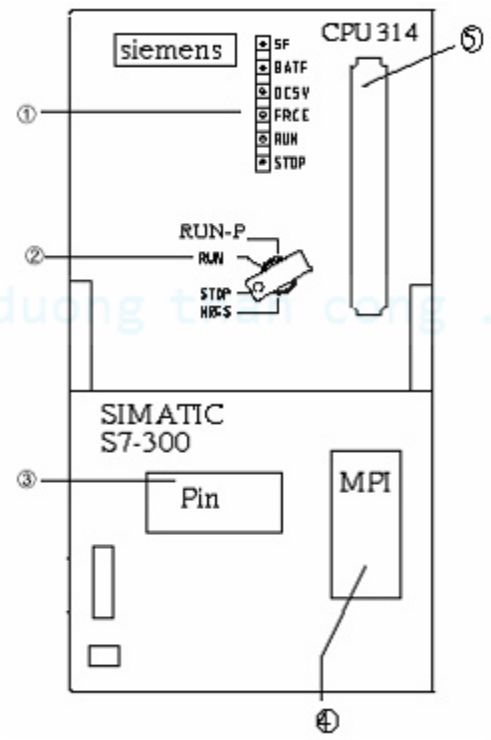
Họ PLC S7-300 có nhiều loại modul CPU khác nhau, chúng được đặt tên theo bộ

vi xử lý như CPU 312, CPU 314, CPU 315...

Những modul cùng sử dụng bộ vi xử lý nhưng khác nhau về cổng vào/ra Onboard cũng như các hàm tích hợp sẵn trong thư viện của hệ điều hành thì sẽ được phân biệt bằng cụm chữ IFM (Intergrated Function Modul). Ví dụ: CPU 312, CPU 312IFM, CPU 314, CPU 314IFM...

Những CPU có hai cổng truyền thông, với cổng thứ 2 phục vụ cho việc nối mạng phân tán thì sẽ phân biệt bằng cụm từ DP (Distributed Port). Ví dụ CPU 312, CPU 312DP, CPU 314, CPU 314DP, CPU 315DP...

Ví dụ hình dạng của CPU 314 có định dạng ở mặt trước như sau:



1) Các led báo trạng thái: • SF: Báo lỗi trong nhóm. Lỗi trong CPU hay trong modul có khả năng chuẩn

đoán. • BATF: Báo lỗi pin. Pin hết điện hay không có pin. • DC5V: Báo có 5VDC. • FRCE: Báo có ít nhất một ngõ vào/ra đang bị cưỡng bức. • RUN: Nhấp nháy khi CPU khởi động, ổn định khi CPU làm việc. • STOP: Đèn sáng

khi dừng, chớp chậm khi có yêu cầu Reset bộ nhớ, chớp

nhanh khi đang Reset bộ nhớ. Chớp chậm khi có yêu cầu

Reset bộ nhớ là cần thiết vì card nhớ được cắm vào. 2)

Nút chọn chế độ hoạt động: • RUN-P: Chế độ xử lý

chương trình, có thể đọc và ghi được từ PG. • RUN: Chế

độ xử lý chương trình, không thể đọc và ghi được. •

STOP: Chế độ dừng, chương trình không được xử lý.

• MRES (Modul Reset Function): Chức năng Reset hệ thống. 3) Ngăn để pin: Ngăn để pin nằm dưới nắp, pin dùng để cung cấp năng lượng lưu trữ nội dung RAM trong trường hợp mất điện.

4) Đầu nối MPI: Đầu nối dành cho thiết bị lập trình hay các thiết bị cần giao tiếp

qua cổng MPI.

5) Card nhớ: Dùng để lưu nội dung chương trình mà không cần dùng pin trong trường hợp mất điện.

4.2.2. CÁC MODUL MỞ RỘNG

Có 5 loại modul mở rộng chính là:

PS (Power Supply) là modul nguồn nuôi. Modul này có tác dụng chuyển đổi điện áp từ 120VACđến 230VAC thành điện áp 24VDC phù hợp với điện áp làm việc của S7-300. Có nhiều kiểu nguồn như nguồn loại 2A, 5A và 10A. Nguồn cung cấp là mạch cách ly có bảo vệ ngắn mạch, điện áp ổn định.

SM (Signal Modul) là modul tín hiệu dùng để mở rộng cổng tín hiệu vào/ra, làm thích nghi với nhiều mức xử lý của S7-300. Có bộ nối bus điều khiển cho mỗi khối và các vòng nối các bus dữ liệu phía sau, tín hiệu xử lý ở bộ nối phía trước. Modul SM bao gồm:

- DI (Digital Input) là modul mở rộng các cổng vào số. Số các cổng vào số

mở rộng có thể là 8, 16 hoặc 32 tuỳ thuộc vào từng loại modul.

- DO (Digital Output) là modul mở rộng các cổng ra số. Số các cổng ra số mở rộng có thể là 8, 16 hoặc 32 tuỳ thuộc vào từng loại modul.

- DI/DO (Digital Input/Digital Output) là modul mở rộng các cổng vào/ra số. Số các cổng vào/ra số mở rộng có thể là 8 vào/8 ra hoặc 16 vào/16 ra tuỳ thuộc vào từng loại modul.

- AI (Analog Input) là modul mở rộng các cổng vào tương tự. AI chính là những bộ chuyển đổi tương tự/số 12 bit (AD), tức là mỗi tín hiệu tương tự được chuyển thành tín hiệu số nguyên có độ dài 12 bit. Số các cổng vào tương tự có thể là 2, 4 hoặc 8 tuỳ từng loại modul.

- AO (Analog Output) là modul mở rộng các cổng ra tương tự. AO chính là những bộ chuyển đổi số/tương tự 12 bit (DAC). Số các cổng vào tương tự có thể là 2 hoặc 4 tuỳ từng loại modul.

- AI/AO (Analog Input/Analog Output) là modul mở rộng các cổng vào/ra tương tự. Số các cổng vào tương tự có thể là 4 vào/2 ra hoặc 4 vào/4 ra tuỳ từng loại modul.

IM (Interface Modul) là modul giao tiếp. Gồm các loại modul IM 360, IM 361 và IM 365 dùng để kết nối nhiều cấu hình với nhau, kết nối các bus giữa các giá trong cấu hình đa tầng. Chúng được quản lý chung bởi một modul CPU.

FM (Function Modul) là modul chức năng có chức năng điều khiển riêng. Những khối chức năng FM thay thế các khối IP. Có các chức năng đặt biệt như đếm, định vị, điều khiển hồi tiếp, điều khiển động cơ bước, động cơ servo, modul PID...

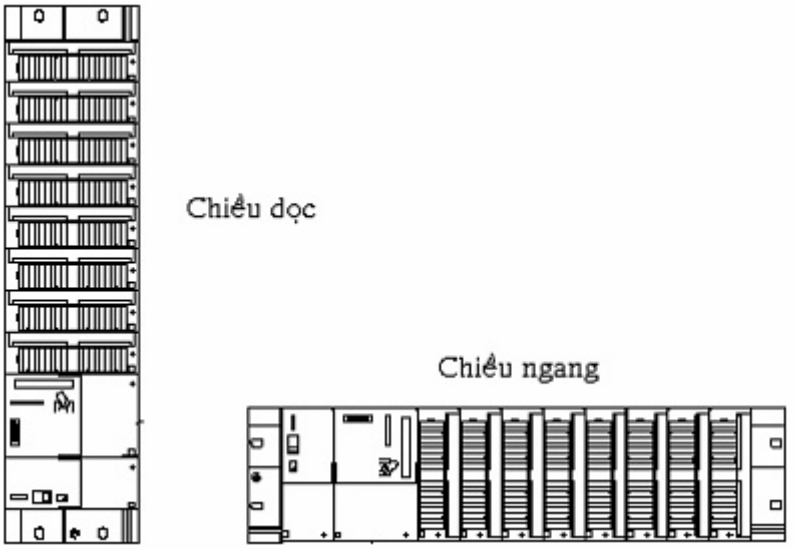
CP (Communication Process) là modul xử lý truyền thông. Modul này dùng để truyền thông trong mạng giữa các PLC với nhau hoặc giữa PLC với máy tính. Gồm có các loại: Nối điểm-điểm (PPI_ Point To Point Interface), mạng Profibus, mạng Ethernet công nghiệp.

Ngoài ra còn có modul giả lập DM (Dummy Modul) dùng để dự phòng cho các modul tín hiệu chưa được chỉ định, như giành chỗ cho các modul sẽ lắp đặt trong tương lai.

4.2.3. LẮP RÁP CÁC MODUL CỦA S7-300

Các modul S7-300 có thể lắp ráp theo chiều dọc hoặc chiều ngang. Kiểu ráp theo

chiều ngang thì nguồn và CPU phải nằm ở phía bên trái. Kiểu ráp theo chiều dọc thì nguồn và CPU phải nằm ở phía dưới cùng.



Khoảng cách hở tối thiểu cần có là

- 20mm bên phải và trái của giá.
- 40mm bên trên và dưới cho chồng đơn, ít nhất 80mm giữa hai giá.

Khối giao tiếp IM (nếu có) luôn nằm bên cạnh CPU.

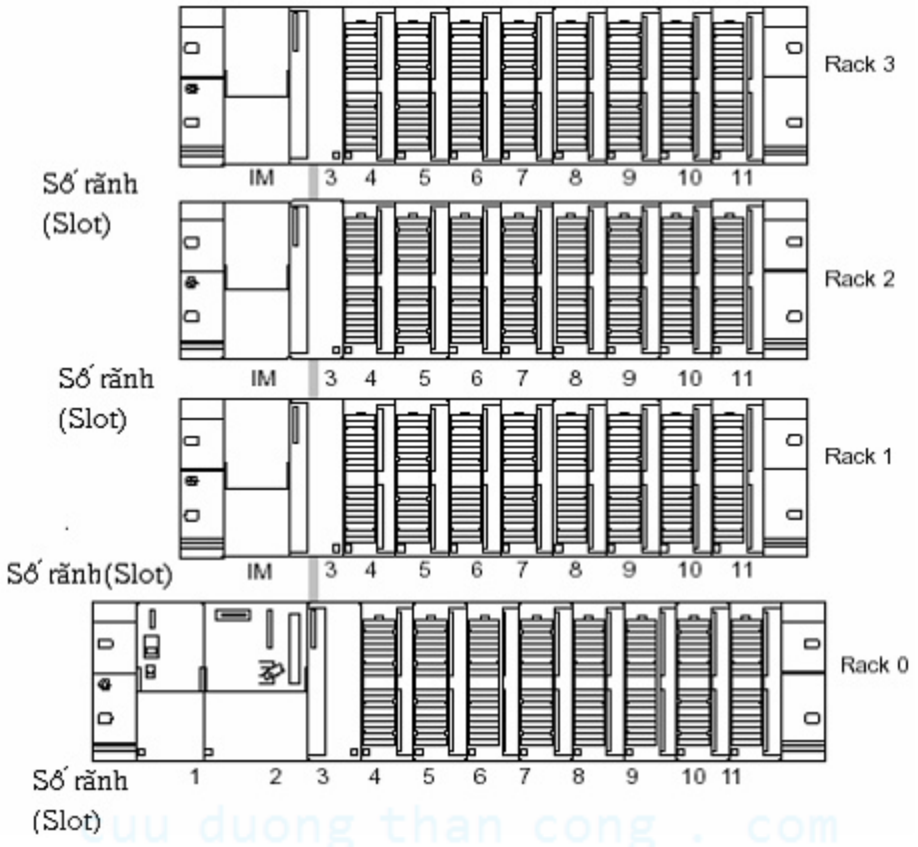
Có tối đa 8 khối vào/ra (khối tín hiệu (SM), khối chức năng (FM), khối xử lý truyền
thông (CP)) được lắp đặt trên mỗi giá.

Cách lắp chồng nhiều tầng chỉ có đối với CPU 314, CPU 315, CPU 316.

Phải đảm bảo điện trở kết nối giữa các đường trượt phải thấp như ở các vòng đệm, mối nối...

4.3. MỞ RỘNG CÁC MODUL CỦA S7-300

S7-300 sử dụng các loại CPU 314, CPU 315 có thể mở rộng đến 32 khối vào/ra. Mở rộng lên đến 4 giá đỡ (Rack), mỗi giá đỡ có tối đa 8 modul vào/ra. Không có qui luật về số rãnh đăng ký cho các khối tín hiệu, các khối chức năng và các bộ xử lý truyền thông. Nghĩa là chúng có thể đặt tại bất cứ vị trí nào.



Rack 0 chứa các khối nguồn, CPU, khối IM và các modul mở rộng. Các rack còn lại chứa khối IM, các modul mở rộng và chúng có thể có khối nguồn hoặc không có khối nguồn tuỳ thuộc vào loại IM. Với loại IM 360/361 thì phải dùng nguồn cung cấp gắn thêm ở các giá mở rộng. Loại giao tiếp IM 365 là khối giao tiếp không cần dùng thêm nguồn cung cấp và không nối với khối CP. Modul giao tiếp IMS là modul gởi và modul IMR là modul nhận.

Các rãnh từ 1 đến 3 là các rãnh được chỉ định thường trực, tức là rãnh nào quy định gắn với modul nào. Nếu không có modul đó thì rãnh đó để trống. Rãnh 1 để gắn nguồn (nếu có), rãnh 2 để gắn CPU (nếu có), rãnh 3 để gắn modul giao tiếp IM (nếu có). Từ rãnh 4 đến rãnh 11 được dùng để gắn các khối vào/ra (SM, FM, CP), các khối này có thể gắn tự do ở bất kỳ rãnh nào.

4.4. KÝ HIỆU ĐỊA CHỈ VÀO/RA CỦA PLC S7-300

Các ngõ vào/ra được mã hoá theo bit, byte, word và word kép.

4.4.1. ĐỊA CHỈ THEO BIT

4.4.1.1. NGÕ VÀO BIT

Ký hiệu: I x.y

I là ký hiệu ngõ vào bit.

x là số thứ tự byte.

y: Chỉ số thứ tự bit của byte x. y là số nguyên có giá trị từ 0 đến 7.

Ví dụ: I 0.0 là địa chỉ ngõ vào bit 0 của byte 0.

Với CPU 314 thì có 128 Byte từ I0.0 đến I127.7.

4.4.1.2. NGÕ RA BIT

Ký hiệu: Q x.y

Q là ký hiệu ngõ ra bit.

x là số thứ tự byte.

y: Chỉ số thứ tự bit của byte x.

y là số nguyên có giá trị từ 0 đến 7.

Ví dụ: Q 4.0 là địa chỉ ngõ ra bit 0 của byte 4.

Với CPU 314 thì có 128 Byte từ Q0.0 đến Q127.7.

4.4.2. ĐỊA CHỈ THEO BYTE

4.4.2.1. NGÕ VÀO BYTE

Ký hiệu: IB x

IB là ký hiệu ngõ vào byte.

x là số thứ tự byte.

Ví dụ: IB 1 là địa chỉ ngõ vào byte 1.

Với CPU 314 thì có 128 Byte từ IB 0 đến IB 127.

4.4.2.2. NGÕ RA BYTE

Ký hiệu: QB x

QB là ký hiệu ngõ ra byte.

x là số thứ tự byte.

Ví dụ: QB 4 là địa chỉ ngõ ra byte 4.

Với CPU 314 thì có 128 Byte từ QB 0 đến QB 127.

4.4.3. ĐỊA CHỈ THEO WORD

4.4.3.1. NGÕ VÀO WORD

Ký hiệu: IW x

IW là ký hiệu ngõ vào word.

x là số thứ tự byte thấp của word.

Ví dụ: IW 2 là địa chỉ ngõ vào word 2. IW 2 = IB2 + IB3.

4.4.3.2. NGÕ RA WORD

Ký hiệu: QW x

QW là ký hiệu ngõ ra word.

x là số thứ tự byte thấp của word.

Ví dụ: QW 4 là địa chỉ ngõ ra word 4. QW4= QB4+QB5.

4.4.4. ĐỊA CHỈ THEO WORD KÉP

4.4.4.1. NGÕ VÀO WORD KÉP

Ký hiệu: ID x

ID là ký hiệu ngõ vào word kép.

x là số thứ tự byte thấp của word kép.

Ví dụ: ID 0 là địa chỉ ngõ vào của word kép 0. ID 0= IB0+IB1+IB2+IB3.

4.4.4.2. NGÕ RA WORD KÉP

Ký hiệu: QD x

QD là ký hiệu ngõ ra của word kép.

x là số thứ tự byte thấp của word kép.

Ví dụ: QD 4 là địa chỉ ngõ ra của word kép 4. QD4= QB4+QB5+QB6+QB7.

CHƯƠNG 3: TẬP LỆNH CƠ BẢN CỦA PLC S7-300 BÀI 1. CÁC LỆNH DẠNG BIT

1. TIẾP ĐIỂM THƯỜNG MỞ (NO_NORMALLY OPEN CONTACT)

Ký hiệu:

LAD	FBD	STL
n x.y	n x.y — &	A n x.y

n: Biểu diễn các toán hạng địa chỉ ngõ vào như sau: I, Q, M, L, T, C, D.

Lệnh này dùng để đặt một công tắc logic thường mở vào chương trình, nó để kiểm tra trạng thái tín hiệu mức logic 1. Khi đó kết quả của phép toán logic RLO=1 (Result Of Logic Operation). Tiếp điểm này sẽ đóng tức là lên 1 khi n x.y=1. nếu n x.y=0 thì tiếp điểm sẽ về mức Logic 0 (RLO=0).

Khi sử dụng tiếp điểm NO mắc nối tiếp thì kết quả giống như cổng Logic AND. Khi sử dụng tiếp điểm NO mắc song song thì kết quả giống như cổng

Logic OR.

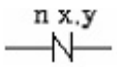
BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	X	X	X	1

Lệnh này tác động lên thanh ghi trạng thái như sau (ký hiệu"-" chỉ nội dung bit không bị thay đổi, "x" chỉ nội dung bit bị thay đổi theo trạng thái:

2. TIẾP ĐIỂM THƯỜNG ĐÓNG (NC_NORMALLY CLOSE CONTACT)

Ký hiệu:

LAD FBD STL AN n x.y



n: Biểu diễn các toán hạng địa chỉ ngõ vào như sau: I, Q, M, L, T, C, D.

Lệnh này dùng để đặt một công tắc logic thường đóng vào chương trình, nó để kiểm tra trạng thái tín hiệu mức logic 0. Khi đó kết quả của phép toán logic RLO=0 (Result Of Logic Operation). Tiếp điểm này sẽ mở tức là xuống 0 khi n x.y=1. Nếu n x.y=0 thì tiếp điểm sẽ lên mức Logic 1 (RLO=1).

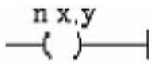
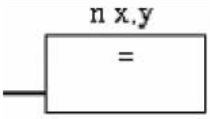
Khi sử dụng tiếp điểm NO mắc nối tiếp thì kết quả giống như cổng Logic AND. Khi sử dụng tiếp điểm NO mắc song song thì kết quả giống như cổng Logic OR.

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

Lệnh này tác động lên thanh ghi trạng thái như sau (ký hiệu"-" chỉ nội dung bit không bị thay đổi, "x" chỉ nội dung bit bị thay đổi theo trạng thái:

3. CUỘN DÂY NGÕ RA (OUTPUT COIL)

Ký hiệu:

LAD	FBD	STL
		= n x.y

n: Biểu diễn các toán hạng địa chỉ ngõ vào như sau: I, Q, M, L, D.

Chức năng của lệnh này giống như cuộn dây ở mạch Rơle. Lệnh này không duy trì

trạng thái mà có trạng thái giống như ở ngõ vào của lệnh. Nghĩa là:

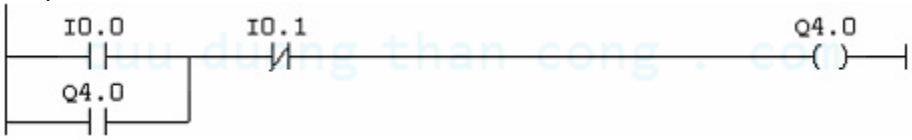
- Nếu có tín hiệu chạy qua cuộn dây thì n x.y=1.
- Nếu không có tín hiệu chạy qua cuộn dây thì n x.y=0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	x	-	0

Ví dụ: Mạch điều khiển động cơ hoạt động theo chu trình đóng ngắt. Khi nhấn một nút thì động cơ chạy. Nhấn nút khác thì động cơ ngừng.

Sơ đồ mạch:



Hoạt động của mạch: Khi nhấn I0.0 thì ngõ ra Q4.0 có điện điều khiển động cơ hoạt động, đồng thời tiếp điểm thường hở Q4.0 mắt song song với I0.0 giúp cho mạch duy trì hoạt động. Khi nhấn nút nhấn I0.1 thì làm ngắt mạch ngõ ra, động cơ ngừng hoạt động.

4. LỆNH NOT (LỆNH ĐẢO TÍN HIỆU_INVERT POWER FLOW)

Ký hiệu:

LAD

FBD

STL Not

Lệnh NOT không có toán hạng là địa chỉ. Nó có chức năng làm đảo trạng thái của kết quả phép toán Logic (RLO).

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	-	1	x	-

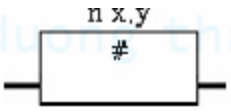
Ví dụ: Mạch đảo tín hiệu có sơ đồ như sau:

Hoạt động: Tín hiệu ngõ ra Q4.0=0 khi I0.0=1. Tín hiệu ngõ ra Q4.0=1 khi I0.0=0.

5. LỆNH KẾT NỐI (CONNECTOR)

Ký hiệu:

LAD FBD STL



= n x.y A n x.y

n: Biểu diễn các toán hạng địa chỉ ngõ vào như sau: I, Q, M, L, D. Địa chỉ L chỉ được sử dụng chỉ khi nó được khai báo trong bảng khai báo của các khối Logic (OB, FC, FB).

Kết nối là phần tử trung gian dùng để lưu trữ RLO hiện hành tại một địa chỉ cụ thể, khi kết nối nối với các phần tử khác lệnh kết nối được chèn giống như chèn một

tiếp điểm.

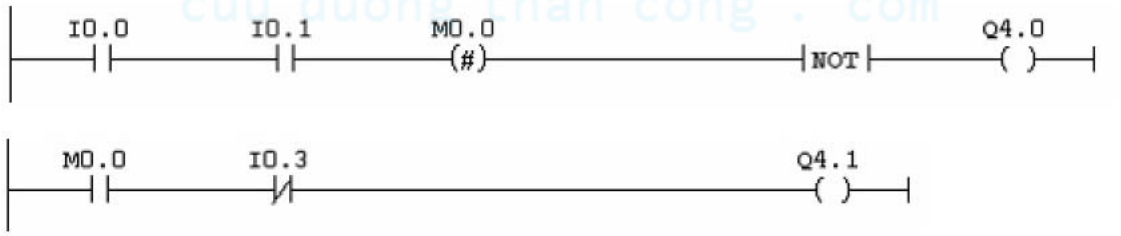
Một kết nối không bao giờ được:

- Nối vào một ray dẫn điện.
- Trực tiếp sau một sự phân nhánh.
- Sử dụng tại một điểm kết thúc một nhánh.
- Có thể tạo một kết nốiphủ định bằng phần tử NOT.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	x	-	1

Ví dụ:



M 0.0 là kết quả phép toán Logic And 

Khi mức Logic ở tiếp điểm I0.0 và I0.1 lên 1 thì M0.0 lên 1, ngõ ra Q4.0=0 (Bình thường Q4.0=1) và I0.3 là tiếp điểm thường đóng nên ngõ ra Q4.1 lên 1.

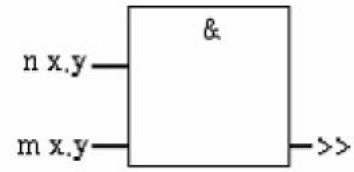
6. PHÉP LOGIC AND

Ký hiệu:

LAD

FBD

STL

		A n x.y A m x.y

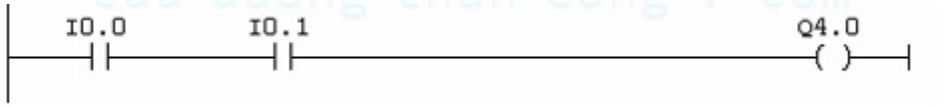
n, m: Biểu diễn các toán hạng địa chỉ ngõ vào như sau: I, Q, M, L, T, C, D.

Chúng ta có thể kiểm tra trạng thái tín hiệu của hai hoặc nhiều tín hiệu ngõ vào của cổng AND.

Kết quả RLO=1 khi tất cả các ngõ vào cổng AND lên 1. Ngược lại, RLO=0 khi có một hoặc tất cả các trạng thái ngõ vào là 0. Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

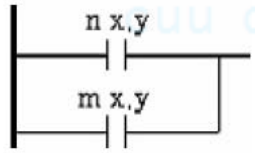
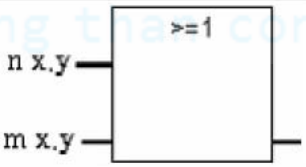
Ví dụ:



Ngõ ra Q4.0=1 chỉ khi cả hai ngõ vào I0.0 và I0.1 lên 1.

7. PHÉP LOGIC OR

Ký hiệu:

LAD	FBD	STL
		O n x.y O m x.y

n, m: Biểu diễn các toán hạng địa chỉ ngõ vào như sau: I, Q, M, L, T, C, D.

Chúng ta có thể kiểm tra trạng thái tín hiệu của hai hoặc nhiều tín hiệu ngõ vào của cổng OR.

Kết quả RLO=1 khi có một hoặc tất cả các ngõ vào cổng OR lên 1. Ngược lại, RLO=0 khi tất cả các trạng thái ngõ vào là 0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

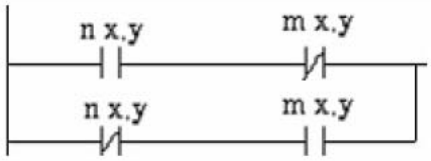
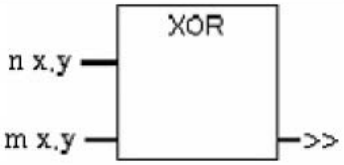
BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

Ví dụ:

Ngõ ra Q4.0=1 khi I0.0=1 hoặc I0.1=1 hoặc cả hai ngõ vào I0.0 và I0.1 lên 1. Q4.0=0 khi cả I0.0 và I0.1 bằng 0.

8. PHÉP LOGIC XOR (PHÉP CỘNG LOẠI TRỪ_EXCLUSIVE OR)

Ký hiệu:

LAD	FBD	STL
		X n x.y X m x.y

n, m: Biểu diễn các toán hạng địa chỉ ngõ vào như sau: I, Q, M, L, T, C, D.

Lệnh XOR dùng để kiểm tra trạng thái của tín hiệu ngõ vào theo bảng sự thật của XOR sau:

A	B	Y
0	0	1
0	1	0
1	0	1

1	1	0

Kết quả RLO=1 chỉ khi trạng thái của tín hiệu của một và chỉ một trong hai ngõ vào là 1.

Nếu có nhiều toán tử XOR liên kết nhau thì từ lệnh XOR thứ 3 trở đi, kết quả là liên kết XOR giữa RLO cũ với ngõ vào mới. Không thể tổng quát quy tắc một và chỉ một trong n.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

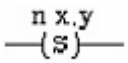
Ví dụ:

Ngõ ra Q4.0=1 chỉ khi I0.0=0 và I0.1=1 hoặc I0.0=1 và I0.1=0. Nếu I0.0=I0.1=0 hoặc I0.0=I0.1=1 thì Q4.0=0.

9. LỆNH SET (S)

Ký hiệu:

LAD FBD STL S n x.y



n: Biểu diễn các toán hạng địa chỉ ngõ vào như sau: I, Q, M, L, D.

Lệnh SET sẽ đặt kết quả địa chỉ n x.y lên 1 mãi mãi khi kết quả của phép toán Logic RLO=1. khi RLO=0 thì địa chỉ n x.y cũng vẫn bằng 1. n x.y chỉ bằng 1 khi có một lệnh Reset địa chỉ này.

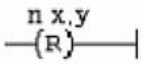
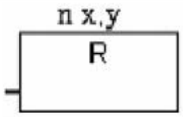
Lệnh này tác động lên thanh ghi trạng thái như sau: Ví dụ:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	x	-	0

Nếu nhấn đồng thời I0.0 và I0.1 thì Q4.0 sẽ lên 1 mãi mãi mặt dù khi thả tay tiếp điểm I0.0 và I0.1 bị hở.

10. LỆNH RESET (R)

Ký hiệu:

LAD	FBD	STL
		R n x.y

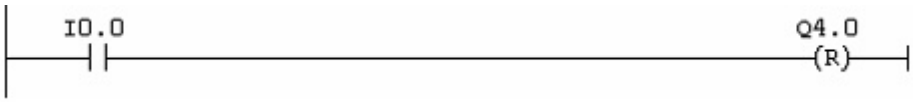
n: Biểu diễn các toán hạng địa chỉ ngõ vào như sau: I, Q, M, L, D.

Lệnh RESET sẽ xoá kết quả địa chỉ n x.y về 0 mãi mãi khi kết quả của phép toán Logic RLO=1. khi RLO=0 thì địa chỉ nx.y cũng vẫn bằng 0. n x.y chỉ bằng 1 khi có một lệnh Set địa chỉ này.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	x	-	0

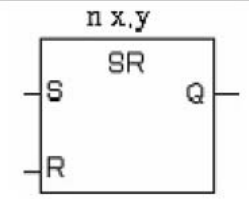
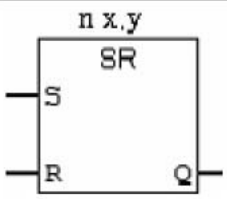
Ví dụ:



Nếu nhấn I0.0 thì Q4.0 sẽ về 0 mãi mãi mặt dù tiếp điểm I0.0 đã bị hở.

11. FLIPFLOP ƯU TIÊN RESET (SR)

Ký hiệu:

LAD	FBD	STL
		A <Bit> S n x.y A <Bit> R n x.y

n, S, R, Q: Biểu diễn các toán hạng địa chỉ ngõ vào/ra như sau: I, Q, M, L, D.

S: Lệnh cho phép Set.

R: Lệnh cho phép Reset.

Q: Biểu diễn trạng thái tín hiệu ngõ ra là 0 hay 1.

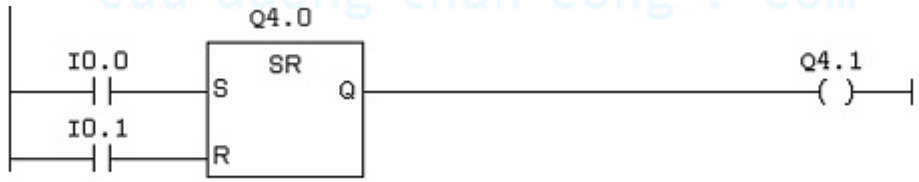
Kết quả của FlipFlop SR được đặt lên 1 nếu tín hiệu ở ngõ vào S là 1 và tín hiệu ở ngõ vào R là 0. Nếu ngõ vào R là 1 thì bất chấp tín hiệu ở ngõ vào S là 0 hay 1 thì ngõ ra Q cũng bằng 0. Nếu tín hiệu ngõ ra được đặt lên 1 và ngõ R=0 thì ngõ ra được duy trì trạng thái đó.

Khi khởi động lại hoàn toàn CPU thì ngõ ra sẽ bị Reset. Nhưng nếu n x.y là một bit nhớ cố định thì nó sẽ giữ lại trạng thái Set sau khi CPU khởi động lại và ngõ ra Q sẽ đặt lại một lần nữa.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

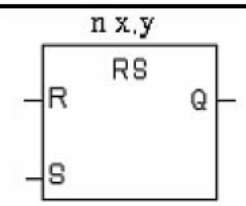
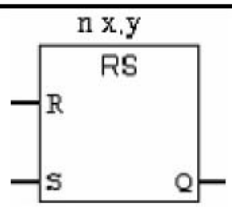
Ví dụ: Hệ thống điều khiển động cơ chỉ sử dụng 2 nút nhấn. Một nút mở máy và một nút tắt máy.



Khi nhấn I0.0 thì động cơ sẽ chạy và khi nhấn I0.1 thì động cơ sẽ ngừng.

12. FLIPFLOP ƯU TIÊN SET (RS)

Ký hiệu:

LAD	FBD	STL
		A <Bit> R n x.y A <Bit> S n x.y

n, S, R, Q: Biểu diễn các toán hạng địa chỉ ngõ vào/ra như sau: I, Q, M, L, D.

S: Lệnh cho phép Set.

R: Lệnh cho phép Reset.

Q: Biểu diễn trạng thái tín hiệu ngõ ra là 0 hay 1.

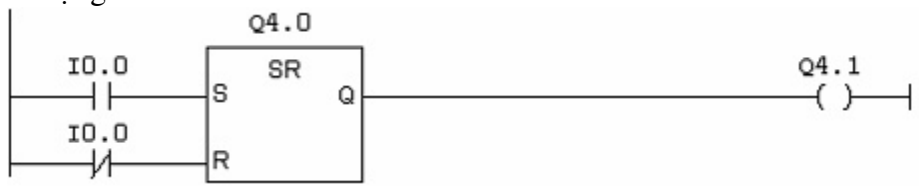
Kết quả của FlipFlop RS được Reset về 0 nếu tín hiệu ở ngõ vào S là 0 và tín hiệu ở ngõ vào R là 1. Nếu ngõ vào S là 1 thì bất chấp tín hiệu ở ngõ vào R là 0 hay 1 thì ngõ ra Q cũng bằng 1. Nếu tín hiệu ngõ ra được đặt lên 1, ngõ vào S không tác động và ngõ R=0 thì ngõ ra được duy trì trạng thái đó.

Khi khởi động lại hoàn toàn CPU thì ngõ ra sẽ bị Reset. Nhưng nếu n x.y là một bit nhớ cố định thì nó sẽ giữ lại trạng thái Set sau khi CPU khởi động lại và ngõ ra Q sẽ đặt lại một lần nữa.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

Ví dụ: Hệ thống xác định trạng thái của tín hiệu. Khi có tín hiệu tác động thì sẽ có một mạch nhận biết như chuông, đèn cảnh báo, còn nếu không có tín hiệu thì ngõ ra không tác động.

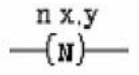
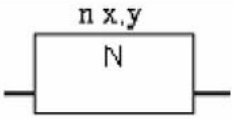


Khi I0.0=1 thì ngõ ra Q4.0=1 có tác dụng cảnh báo. Khi I0.0=0 thì sẽ Reset ngõ ra

về 0 tín hiệu cảnh báo dừng.

13. LỆNH NHẬN BIẾT RLO CẠNH XUỐNG (N)

Ký hiệu:

LAD	FBD	STL
		FN n x.y

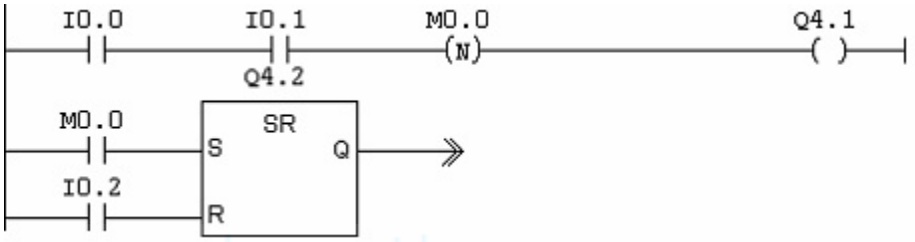
n: Biểu diễn các toán hạng địa chỉ ngõ vào/ra như sau: I, Q, M, L, D.

Lệnh N dùng để nhận biết sự thay đổi của kết quả phép toán Logic (RLO), khi RLO thay đổi từ 1 xuống 0 thì trạng thái của tín hiệu ngõ ra sẽ lên 1 trong một chu kỳ OB1. Trạng thái thay đổi này được lưu trữ trong một bit nhớ hoặc bit dữ liệu nx.y.

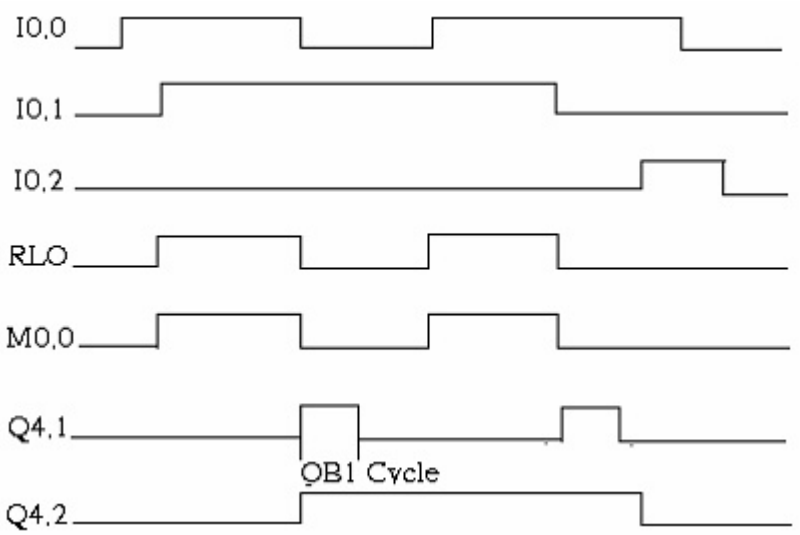
Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	x	x	1

Ví dụ: Mạch nhận biết cạnh tín hiệu thay đổi từ 1 về 0 được hiển thị ở ngõ ra Q4.1 trong một chu kỳ OB1 và hiển thị ở ngõ ra Q4.2 mãi cho đến khi có lệnh Reset.



Quá trình hoạt động được biểu diễn theo sơ đồ sau:



14. LỆNH NHẬN BIẾT RLO CẠNH LÊN (P)

Ký hiệu:

LAD	FBD	STL
n x.y —(P)—	n x.y P	FP n x.y

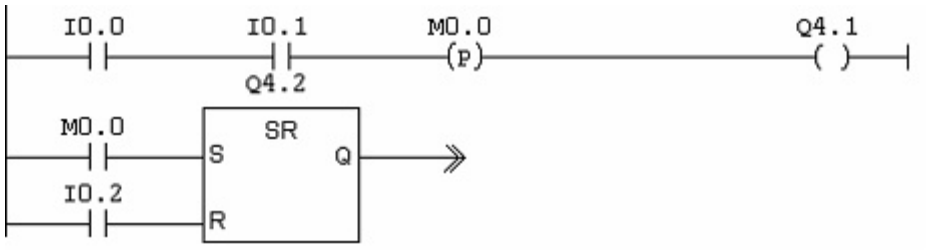
n: Biểu diễn các toán hạng địa chỉ ngõ vào/ra như sau: I, Q, M, L, D.

Lệnh P dùng để nhận biết sự thay đổi của kết quả phép toán Logic (RLO), khi RLO thay đổi từ 0 lên 1 thì trạng thái của tín hiệu ngõ ra sẽ lên 1 trong một chu kỳ OB1. Trạng thái thay đổi này được lưu trữ trong một bit nhớ hoặc bit dữ liệu nx.y.

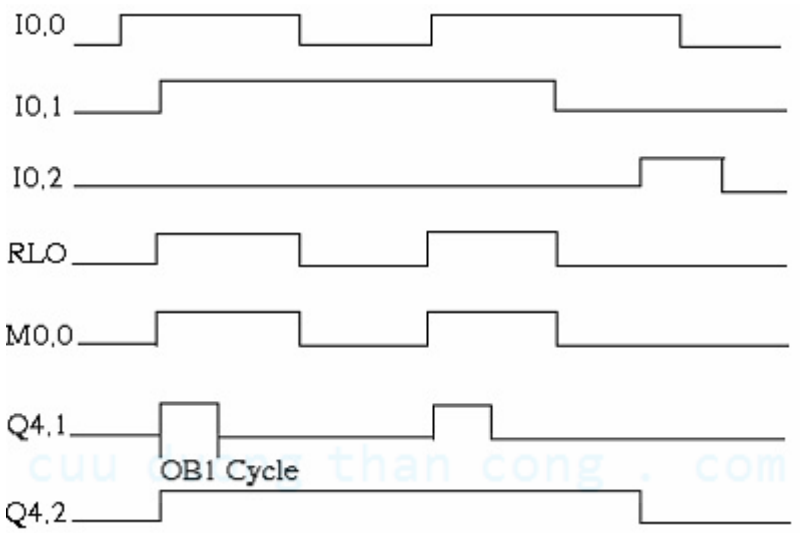
Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	x	x	1

Ví dụ: Mạch nhận biết cạnh tín hiệu thay đổi từ 0 lên 1 được hiển thị ở ngõ ra Q4.1 trong một chu kỳ OB1 và hiển thị ở ngõ ra Q4.2 mãi cho đến khi có lệnh Reset.

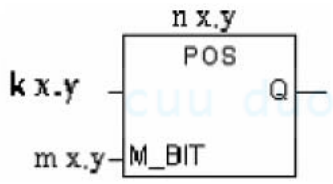
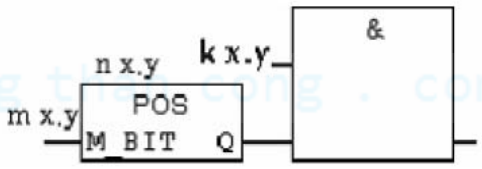


Quá trình hoạt động được biểu diễn theo sơ đồ sau:



15. NHẬN BIẾT CẠNH TÍN HIỆU THAY ĐỔI THEO CẠNH LÊN (POS-ADDRESS POSSITIVE EDGE DETECTION)

Ký hiệu:

LAD	FBD	STL
		A K x.y A (A n x.y F P M x.y) = Q

n, m, k, Q: Biểu diễn các toán hạng địa chỉ ngõ vào/ra như sau: I, Q, M, L, D.

nx.y: Ngõ vào tín hiệu cần xác định mức tín hiệu cần thay đổi mức tín hiệu.

mx.y: Bit nhớ, dùng để nhớ, lưu trữ tín hiệu ở ngõ vào nx.y.

kx.y: Ngõ vào dùng để cho phép hoặc không cho phép xác định sự thay đổi cạnh tín

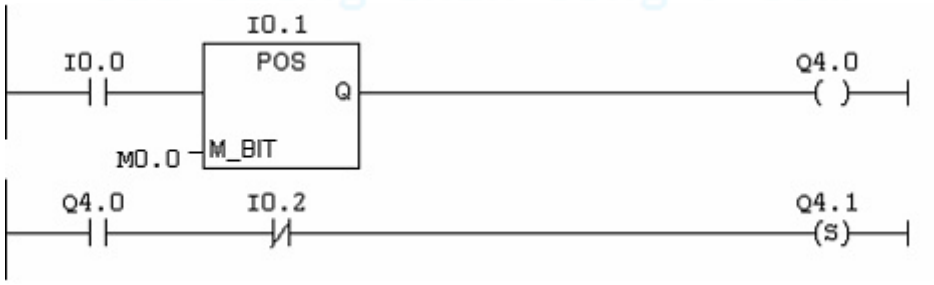
hiệu. Kx.y=1 thì cho phép, Kx.y=0 thì không cho phép.

Nếu tín hiệu ở ngõ vào nx.y thay đổi từ 0 lên 1 khi kx.y=1 thì ngõ ra lên 1 trong 1 chu kỳ máy OB1.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	-	-	-	-	x	1	x	1

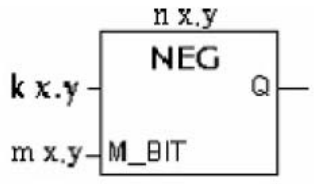
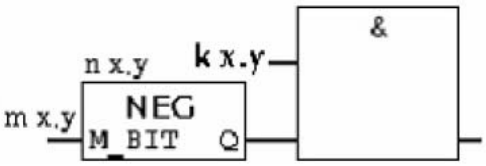
Ví dụ: Mạch nhận biết cạnh tín hiệu thay đổi từ 0 lên 1 được hiển thị ở ngõ ra Q4.0 trong một chu kỳ OB1 và hiển thị ở ngõ ra Q4.1 mãi cho đến khi có lệnh Reset.



Nếu I0.0 lên 1 trong khi I0.1 có sự thay từ 0 lên 1 thì Q4.0 tác động trong một chu kỳ, đồng thời Q4.1 tác động cho đến khi nhấn I0.2 (I0.2 là tiếp điểm thường đóng).

16. NHẬN BIẾT CẠNH TÍN HIỆU THAY ĐỔI THEO CẠNH XUỐNG (NEG-ADDRESS NEGATIVE EDGE DETECTION)

Ký hiệu:

LAD	FBD	STL
		A K x.y A (A n x.y FN M x.y) = Q

n, m, k, Q: Biểu diễn các toán hạng địa chỉ ngõ vào/ra như sau: I, Q, M, L, D. nx.y:

Ngõ vào tín hiệu cần xác định mức tín hiệu cần thay đổi mức tín hiệu. mx.y: Bit nhớ,

dùng để nhớ , lưu trữ tín hiệu ở ngõ vào nx.y.

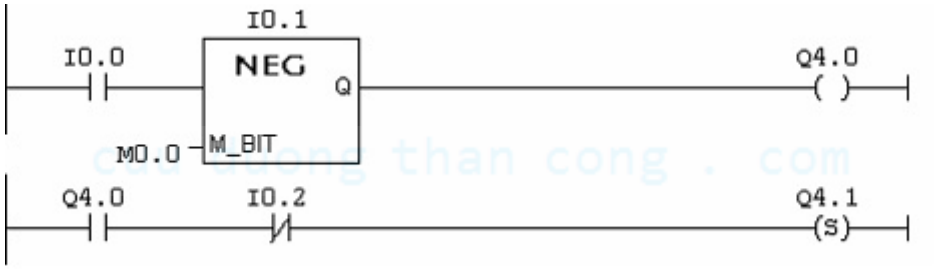
kx.y: Ngõ vào dùng để cho phép hoặc không cho phép xác định sự thay đổi cạnh tín hiệu. Kx.y=1 thì cho phép, Kx.y=0 thì không cho phép.

Nếu tín hiệu ở ngõ vào nx.y thay đổi từ 1 xuống 0 khi kx.y=1 thì ngõ ra lên 1 trong 1 chu kỳ máy OB1.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	-	-	-	-	x	1	x	1

Ví dụ: Mạch nhận biết cạnh tín hiệu thay đổi từ 1 xuống 0 được hiển thị ở ngõ ra Q4.0 trong một chu kỳ OB1 và hiển thị ở nhõ ra Q4.1 mãi cho đến khi có lệnh Reset.



Nếu I0.0 lên 1 trong khi I0.1 có sự thay từ 1 xuống 0 thì Q4.0 tác động trong một chu kỳ, đồng thời Q4.1 tác động cho đến khi nhấn I0.2 (I0.2 là tiếp điểm thường đóng).

17. LỆNH CLR (CLEAR)

Lệnh này chỉ có ở cấu trúc lệnh STR. Lệnh CLR dùng để xoá kết quả phép toán Logic (RLO) thành 0.

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	0	0	0

Ví dụ:

CLR

= M0.0

= Q4.0

Kết quả là bit nhớ M0.0 và ngõ ra Q4.0 xuống 0.

18. LỆNH SET

Lệnh này chỉ có ở cấu trúc lệnh STR. Lệnh SET dùng để đặt kết quả phép toán Logic (RLO) thành 1.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	1	1	0

Ví dụ: SET = M0.0 = Q4.0

Kết quả là bit nhớ M0.0 và ngõ ra Q4.0 lên 1.

19. LỆNH SAVE

Ký hiệu:

LAD

FBD STL SAVE

Lệnh SAVE không có toán hạng. Lệnh SAVE dùng để lưu kết quả phép toán Logic (RLO) vào bit nhớ BR trong thanh ghi từ trạng thái (Status Word).

Vì có nhiều lệnh khi thực hiện có ảnh hưởng đến bit nhớ BR, do đó khi dùng lệnh SAVE chúng ta sẽ không biết chính xác kết quả mà chúng ta lưu bằng lệnh SAVE hay bằng một lệnh nào khác có ảnh hưởng đến bit nhớ BR. Do vậy lệnh SAVE ít được dùng, nó chỉ dùng cho những loại lập trình PLC chỉ đơn thuần các lệnh contac Logic thường mở và thường đóng (NO, NC...), không nên dùng trong những loại PLC có cấu trúc lệnh

khối như lệnh cộng (ADD-R), lúc đó ngõ ra ENO sẽ lưu vào bit nhớ

BR. Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	-	-	-	-	-	-	-	-

Ví dụ: Lưu kết quả RLO vào bit nhớ BR.

Khi trạng thái I1.6 thay đổi 0 hay 1 thì kết quả này sẽ lưu vào bit nhớ BR.

20. LỆNH LIÊN QUAN ĐẾN TẠNG THÁI BIT

Những lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

20.1. LỆNH NHẬN BIẾT LỖI

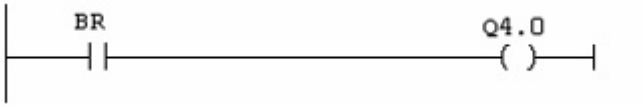
20.1.1. LỆNH BR (BINARY RESULT)

Ký hiệu:

LAD		FBD	STL	
			A BR	AN BR

Lệnh BR dùng để kiểm tra lại việc lưu kết quả phép toán Logic (RLO) hay là dùng để kiểm tra trạng thái bit nhớ BR trong Word trạng thái. Có thể mắt nối tiếp hoặc song song để tạo thành mạch Logic AND hoặc OR.

Ví dụ: Kiểm tra lại trạng thái Bit nhớ BR trong Word trạng thái.



Q4.0 sẽ ở mức Logic 0 hay 1 là tuỳ thuộc vào mức Logic của bit nhớ BR. Q4.0=0 khi BR=0, Q4.0=1 khi BR=1.

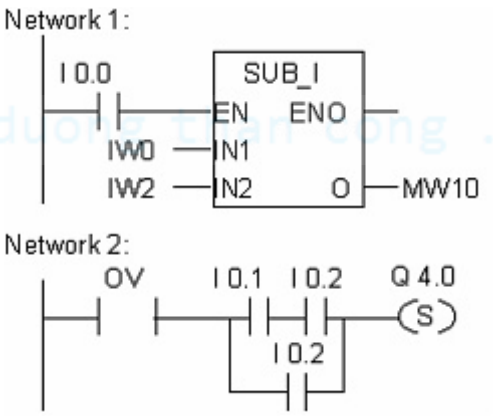
20.1.2. LỆNH OV (OVERFLOW)

Ký hiệu:

LAD		FBD	STL	
			A OV	AN OV

Lệnh OV dùng để nhận biết một lệnh toán học khi thực thi bị tràn ở vùng âm hoặc dương. Có thể mắt nối tiếp hoặc song song để tạo thành mạch Logic AND hoặc OR.

Ví dụ:



Khi I0.0=1 thì lệnh SUB-I hoạt động, kết quả ngõ ra MW10 là IW0-IW2. Nếu kết

quả này vượt ra ngoài vùng hoạt động của số nguyên (Interger) thì bit OV được đặt, khi OV=1 mà trạng thái tín hiệu của I0.2=1 hoặc cả I0.1 và I0.2 là 1 thì ngõ ra Q4.0=1.

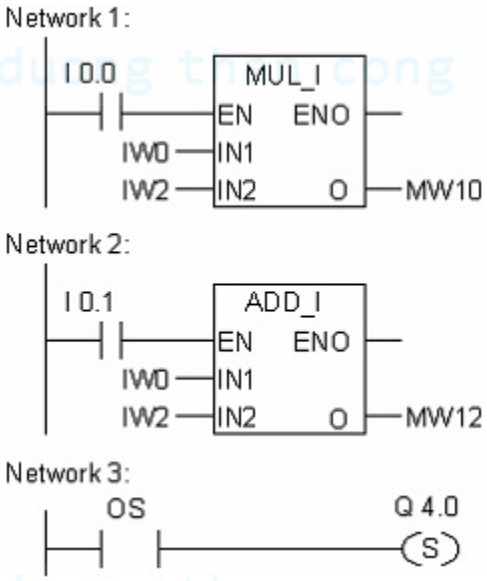
20.1.3. LỆNH OS (OVERFLOW STORED)

Ký hiệu:

LAD		FBD	STL	
			A OS	AN OS

Lệnh OS dùng để nhận biết và lưu trữ trạng thái của một lệnh toán học khi thực thi bị tràn ở vùng âm hoặc dương, khi đó bit OS trong thanh ghi trạng thái được đặt lên 1. Bit OS duy trì trạng thái của tín hiệu đặt cho tới khi chương trình không hoạt động. Có thể mắt nối tiếp hoặc song song để tạo thành mạch Logic AND hoặc OR.

Ví dụ:



Lệnh MUL-I hoạt động khi I0.0=1, lệnh ADD-I hoạt động khi I0.1=1. Nếu một trong hai lệnh có kết quả nằm ngoài vùng cho phép của số nguyên thì bit OS được đặt, khi đó Q4.0=1.

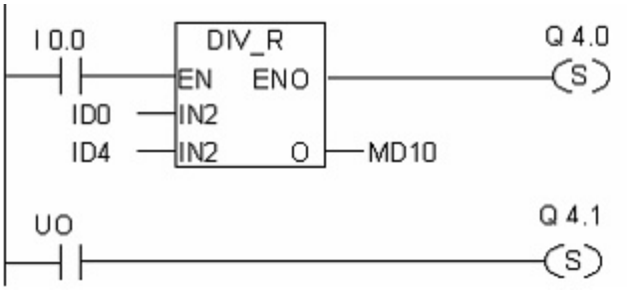
20.1.4. LỆNH UO (EXCEPTION BIT UNORDERED)

Ký hiệu:

LAD		FBD	STL	
			A UO	AN UO

Lệnh UO dùng để nhận biết một lệnh toán học khi thực thi mà có một trong những giá trị không hợp lệ. Nếu lệnh vẫn được thực thi với giá trị không hợp lệ thì bit UO=1. Nếu trạng thái CC0 và CC1 trong thanh ghi trạng thái diễn tả giá trị hợp lệ thì kết quả UO=0. Có thể mắt nối tiếp hoặc song song để tạo thành mạch Logic AND hoặc OR.

Ví dụ:



Khi I0.0=1 thì lệnh DIV-R được thực thi, nếu một trong hai giá trị ID0 hoặc ID4 không hợp lệ thì lệnh này không hợp lệ. Ngõ ra Q4.0=0 và Q4.1=1 khi lệnh DIV-R thực hiện nhưng có giá trị không hợp lệ ở ngõ vào.

20.2. CÁC LỆNH SO SÁNH KẾT QUẢ CỦA PHÉP TOÁN SỐ HỌC VỚI 0

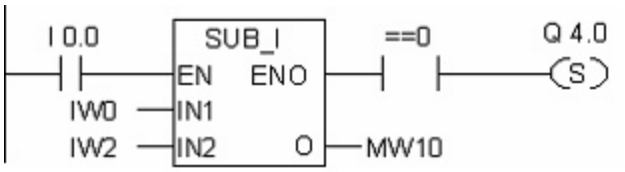
20.2.1. LỆNH SO SÁNH BẰNG 0 (= =0 _ RESULT BIT EQUAL 0)

Ký hiệu:

LAD		FBD	STL	
			A = =0	AN = =0

Lệnh so sánh bằng 0 dùng để nhận biết kết quả của lệnh toán học nào đó có bằng 0 hay không. Có thể mắt nối tiếp hoặc song song để tạo thành mạch Logic AND hoặc OR.

Ví dụ:



Khi I0.0=1 thì lệnh SUB-I thực hiện, nếu giá trị IW0=IW2 thì kết quả bằng 0. Q4.0 được đặt lên 1 nếu kết quả của lệnh SUB-I là bằng 0. Cũng với mạch trên nhưng sử dụng lệnh so sánh đảo thì Q4.0 sẽ bằng 1 khi kết quả khác 0.

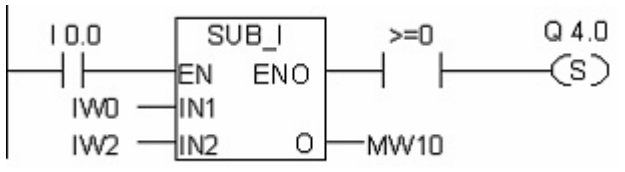
20.2.2. LỆNH SO SÁNH LỚN HƠN HOẶC BẰNG 0 (>=0_RESULT BIT GREATER EQUAL 0)

Ký hiệu:

LAD		FBD	STL	
			A >=0	AN >=0

Lệnh so sánh lớn hơn hoặc bằng 0 dùng để nhận biết kết quả của lệnh toán học nào đó có lớn hơn hay bằng 0. Có thể mắt nối tiếp hoặc song song để tạo thành mạch Logic AND hoặc OR.

Ví dụ:



Khi I0.0=1 thì lệnh SUB-I thực hiện, nếu giá trị IW0 lớn hơn hoặc bằng IW2 thì kết quả sẽ lớn hơn hoặc bằng 0. Q4.0 được đặt lên 1 nếu kết quả của lệnh SUB-I là lớn hơn hoặc bằng 0. Cũng với mạch trên nhưng sử dụng lệnh so sánh đảo thì Q4.0 sẽ bằng 1 khi kết quả không lớn hơn hoặc bằng 0.

20.2.3. LỆNH SO SÁNH NHỎ HƠN HOẶC BẰNG 0 (<=0_RESULT BIT LESS EQUAL 0)

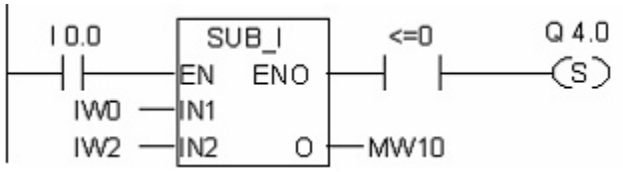
Ký hiệu:

LAD	FBD	STL

			A <=0	AN <=0

Lệnh so sánh nhỏ hơn hoặc bằng 0 dùng để nhận biết kết quả của lệnh toán học nào đó có nhỏ hơn hoặc bằng 0. Có thể mắt nối tiếp hoặc song song để tạo thành mạch Logic AND hoặc OR.

Ví dụ:



Khi I0.0=1 thì lệnh SUB-I thực hiện, nếu giá trị IW0 nhỏ hơn hoặc bằng IW2 thì kết quả sẽ nhỏ hơn hoặc bằng 0. Q4.0 được đặt lên 1 nếu kết quả của lệnh SUB-I là nhỏ hơn hoặc bằng 0. Cũng với mạch trên nhưng sử dụng lệnh so sánh đảo thì Q4.0 sẽ bằng 1 khi kết quả không nhỏ hơn hoặc bằng 0.

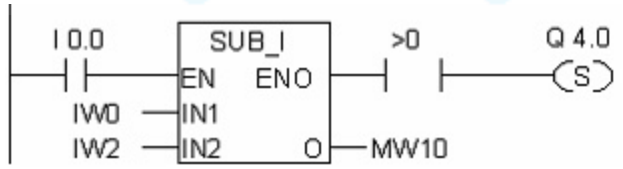
20.2.4. LỆNH SO SÁNH LỚN HƠN 0 (>0_RESULT BIT GREATER THAN 0)

Ký hiệu:

LAD		FBD	STL	
			A >0	AN >0

Lệnh so sánh lớn hơn 0 dùng để nhận biết kết quả của lệnh toán học nào đó có lớn hơn 0. Có thể mắt nối tiếp hoặc song song để tạo thành mạch Logic AND hoặc OR.

Ví dụ:



Khi I0.0=1 thì lệnh SUB-I thực hiện, nếu giá trị IW0 lớn hơn IW2 thì kết quả sẽ lớn hơn 0. Q4.0 được đặt lên 1 nếu kết quả của lệnh SUB-I là lớn hơn 0. Cũng với mạch trên nhưng sử dụng lệnh so sánh đảo thì Q4.0 sẽ bằng 1 khi kết quả không lớn hơn 0.

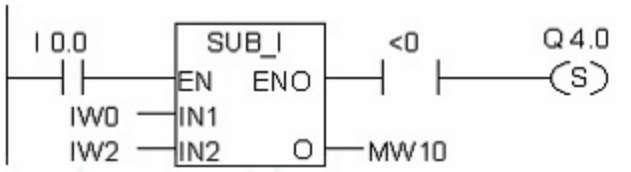
20.2.5. LỆNH SO SÁNH NHỎ HƠN 0 (<0_RESULT BIT LESS THAN 0)

Ký hiệu:

LAD		FBD	STL	
			A <0	AN <0

Lệnh so sánh nhỏ hơn 0 dùng để nhận biết kết quả của lệnh toán học nào đó có nhỏ hơn 0 hay không. Có thể mắt nối tiếp hoặc song song để tạo thành mạch Logic AND hoặc OR.

Ví dụ:



Khi I0.0=1 thì lệnh SUB-I thực hiện, nếu giá trị IW0 nhỏ hơn IW2 thì kết quả sẽ nhỏ hơn 0. Q4.0 được đặt lên 1 nếu kết quả của lệnh SUB-I là nhỏ hơn 0. Cũng với mạch trên nhưng sử dụng lệnh so sánh đảo thì Q4.0 sẽ bằng 1 khi kết quả không nhỏ hơn 0.

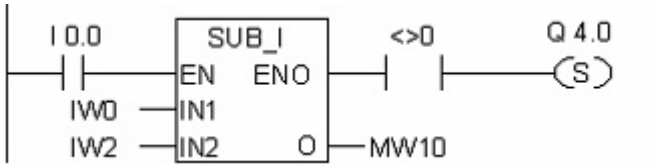
20.2.6. LỆNH SO SÁNH KHÔNG BẰNG 0 (<>0_RESULT BIT NOT EQUAL 0)

Ký hiệu:

LAD		FBD	STL	
			A <>0	AN <>0

Lệnh so sánh không bằng dùng để nhận biết kết quả của lệnh toán học nào đó không bằng 0. Có thể mắt nối tiếp hoặc song song để tạo thành mạch Logic AND hoặc OR.

Ví dụ:



Khi I0.0=1 thì lệnh SUB-I thực hiện, nếu giá trị IW0 không bằng IW2 thì kết quả sẽ không bằng 0. Q4.0 được đặt lên 1 nếu kết quả của lệnh SUB-I là không bằng 0. Cũng với mạch trên nhưng sử dụng lệnh so sánh đảo thì Q4.0 sẽ bằng 1 khi kết quả bằng 0.

21. CHỨC NĂNG MASTER CONTROL RELAY(MCR)

MCR là chuyển mạch chính dùng để đóng hay ngắt mạch. Một đường mạch bị ngắt cho biết giá trị 0 là kết quả phép tính hoặc sẽ không làm thay đổi giá trị hiện hành trong bộ nhớ. Nếu điều kiện MCR không thoả thì:

- Mức Logic 0 sẽ được chuyển đến các cuộn dây ngõ ra.
- Lệnh Set và Reset cuộn dây không làm thay đổi giá trị hiện hành.
- Lệnh Move chuyển giá trị 0 đến một địa chỉ cụ thể.

21.1. LỆNH MCRA (MASTER CONTROL RELAY ACTIVATE)

Ký hiệu:

LAD

FBD STL MCRA

Lệnh này dùng để kích hoạt chức năng MCR. Sau lệnh MCRA, có thể lập trình trong vùng MCR với hai lệnh MCR< và MCR>.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	-	-	-	-

21.2. LỆNH MCRD (MASTER CONTROL RELAY DEACTIVATE)

Ký hiệu:

LAD	FBD	STL
-(MCRD)-	MCRD	MCRD

Lệnh này làm vô hiệu hoá chức năng MCR. Sau lệnh này thì không thể làm việc trong vùng MCR cho tới khi có lệnh MCRA kích hoạt.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	-	-	-	-

21.3. LỆNH MCR< (MASTER CONTROL RELAY ON)

Ký hiệu:

LAD	FBD	STL
-(MCR<)-	MCR<	MCR<

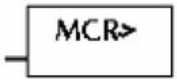
Lệnh này có chức năng mở một vùng MCR, nó lưu kết quả RLO vào trong ngăn xếp MCR. Ngăn xếp MCR là loại ngăn xếp vào sau, ra trước (LIFO- Last In, First Out), nó có 8 cổng vào/ra. Nghĩa là có 8 vùng điều khiển riêng có thể lồng vào giữa lệnh MCRA và MCRD.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	1	-	0

21.4. LỆNH MCR> (MASTER CONTROL RELAY OFF)

Ký hiệu:

LAD	FBD	STL
-(MCR>)-		MCR>

Lệng MCR> dùng để kết thúc vùng MCR. Lệnh này lấy kết quả RLO ra khỏi con trỏ MCR. Nếu con trỏ này trống, lệnh MCR> tạo ra một con trỏ lỗi MCR là MCRF (MCR stack fault).

Những lệnh sau bị ảnh hưởng bởi trạng thái RLO mà được lưu vào con trỏ MCR khi mở vùng MCR:

--(#) Midline Output (Lệnh kết nối).

--( ) Coil Output (Lệnh cuộn dây ngõ ra).

--(S) Set Output (Lệnh đặt ngõ ra).

--(R) Reset Output (Lệnh xoá ngõ ra).

SR Reset Flip Flop (Lệnh Flip Flop ưu tiên Reset).

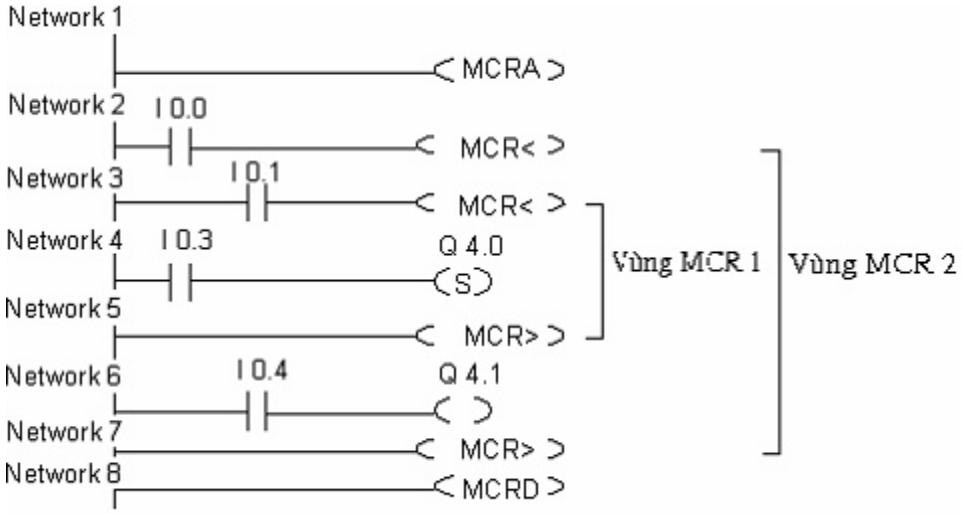
RS Set Flip Flop Lệnh Flip Flop ưu tiên Set).

MOVE Assign a Value (Lệnh di chuyển dữ liệu).

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	1	-	0

21.5. VÍ DỤ VỀ SỬ DỤNG CHỨC NĂNG MCR



Hoạt động:

Chức năng MCR được kích hoạt bởi lệnh MCRA. Giữa hai lệnh MCRA và MCRD có thể tạo ra 8 vùng MCR. Ở ví dụ trên có 2 vùng MCR, hoạt động như sau:

- Khi I0.0=1 vùng MCR 1 được kích hoạt, trạng thái ngõ ra Q4.1 thay đổi theo trạng thái của I0.4. Khi I0.0=0 thì vùng MCR 1 không còn tác động, mọi thay đổi của I0.4 không làm cho Q4.1 thay đổi.

- Khi I0.1=1 vùng MCR 2 được kích hoạt, ngõ ra Q4.0 được đặt lên 1, Q4.0=0 chỉ khi I0.1=0 nghĩa là vùng MCR 2 không còn kích hoạt.

Lệnh MCRD đánh dấu việc thoát khỏi vùng MCR.

22. CÁC LỆNH NHẢY

Lệnh nhảy có thể thực hiện được cả hai hướng lên và xuống. Cả lệnh nhảy và điểm nhảy tới phải ở trong một khối. Độ dài lớn nhất của lệnh nhảy là 6Kbyte. Giới hạn của lệnh nhảy là:

- Đích nhảy đến chỉ xuất hiện một lần trong khối.

- Lệnh nhảy có thể sử dụng trong các khối OB, FC, FB.

- Bất cứ lệnh nào hay Netword nào nằm giữa lệnh nhảy và nhãn nhảy sẽ

khôмg được thực hiện. Có hai loại nhảy là

nhảy không điều kiện và nhảy có điều kiện: • Nhảy

không điều kiện là lệnh đó làm cho việc xử lý

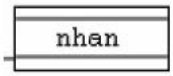
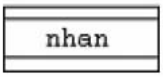
chương trình nhảy đến nhãn nhảy không phụ thuộc

vào RLO. • Nhảy có điều kiện là lệnh nhảy được

thực hiện phụ thuộc vào RLO.

22.1. LỆNH LABLE (NHÃN)

Ký hiệu:

LAD	FBD	STL
		Tên nhãn: <lệnh>

Nhãn có tối đa là 4 ký tự. Ký tự đầu phải là chữ cái hoặc ký tự "-", các ký tự sau có thể là chữ cái hoặc ký tự số. Nhãn đánh dấu điểm tiếp tục làm việc của chương trình. Một nhãn nhảy phải tồn tại cùng với lệnh nhảy JMP hoặc JMPN.

22.2. LỆNH JMP (LỆNH NHẢY- JUMP)

Ký hiệu:

LAD	FBD	STL	
		Không ĐK	Có ĐK
		JU nhan	JC nhan

Lệnh JMP dùng để nhảy đến một nhãn, nhãn có tối đa là 4 ký tự. Ký tự đầu phải là chữ cái hoặc ký tự "-", các ký tự sau có thể là chữ cái hoặc ký tự số. Có lệnh JMP có điều kiện và JMP không điều kiện.

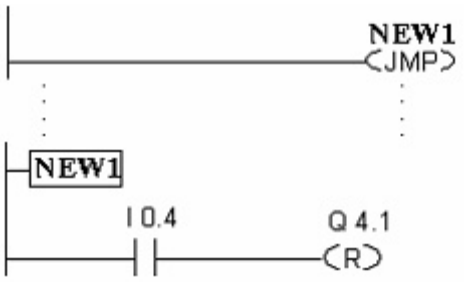
22.2.1. LỆNH NHẢY KHÔNG ĐIỀU KIỆN (JU <NHÃN>)

Đối với dạng LAD/FBD thì lệnh nhảy không điều kiện nối trực tiếp lên nguồn. Đối với dạng STL thì lệnh nhảy không điều kiện được ký hiệu là JU <nhãn>. Lệnh JMP không điều kiện nhảy trực tiếp tới nhãn mà không phụ thuộc vào RLO.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	-	-	-	-

Ví dụ:



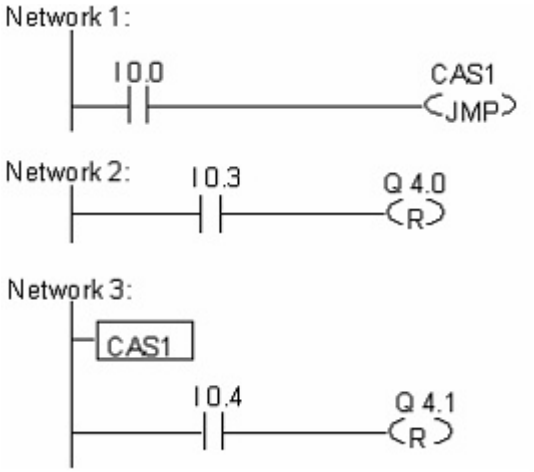
Khi chương trình thực hiện tới lệnh JMP thì sẽ nhảy tới nhãn NEW1 và thực hiện các lệnh kế tiếp, các lệnh nằm ở giữa lệnh JMP và nhãn NEW1 không thực hiện.

22.2.2. LỆNH NHẢY CÓ ĐIỀU KIỆN (JC <NHÃN)

Đối với dạng LAD/FBD thì lệnh nhảy có điều kiện JC được nối lên nguồn qua các mạch Logic. Đối với dạng STL thì lệnh nhảy có điều kiện được ký hiệu là JC<nhãn>. Lệnh nhảy có điều kiện nhảy tới nhãn phụ thuộc vào RLO. Nếu RLO=1 thì lệnh nhảy được thực hiện, khi đó tất cả các lệnh nằm giữa hai lệnh JMP và nhãn không được thực hiện. Nếu RLO=0 thì lệnh nhảy không được thực hiện mà chương trình thực hiện những lệnh kế tiếp.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	1	1	0



Ví dụ:

Nếu I0.0=1 thì lệnh nhảy thực hiện nhảy tới nhãn CAS1 và thực hiện các lệnh kế tiếp, các lệnh trong Netword 2 không được thực hiện.

22.3. LỆNH NHẢY CÓ ĐIỀU KIỆN (JMPN/JCN- JUMP-IF-NOT)

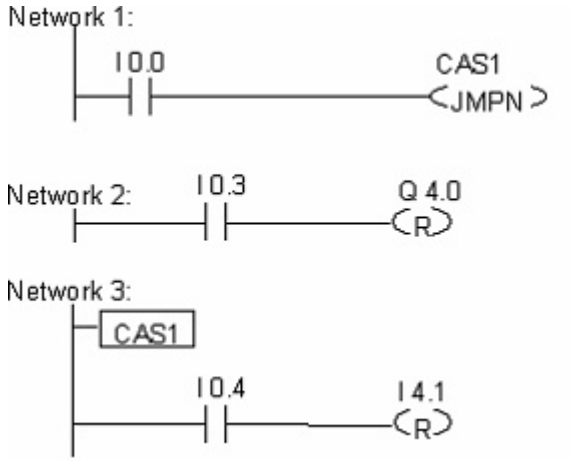
Ký hiệu:

LAD	FBD	STL
nhan —(JMPN)—	nhan JMPN	JCN <nhan>

Đối với dạng LAD/FBD thì lệnh nhảy có điều kiện JCN được nối lên nguồn qua các mạch Logic. Đối với dạng STL thì lệnh nhảy có điều kiện được ký hiệu là JCN <nhãn>. Lệnh nhảy có điều kiện JCN nhảy tới nhãn phụ thuộc vào RLO. Nếu RLO=0 thì lệnh nhảy được thực hiện, khi đó tất cả các lệnh nằm giữa hai lệnh JMPN và nhãn không được thực hiện. Nếu RLO=1 thì lệnh nhảy không được thực hiện mà chương trình thực hiện những lệnh kế tiếp.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	1	1	0



Ví dụ:

Nếu I0.0=0 thì lệnh nhảy JMPN thực hiện nhảy tới nhãn CAS1 và thực hiện các lệnh kế tiếp, các lệnh trong Netword 2 không được thực hiện.

23. LỆNH KẾT THÚC CHƯƠNG TRÌNH (RET_RETURN)

Ký hiệu: BEU

LAD	FBD	STL	
---(RET)		Không điều kiện	Có điều kiện

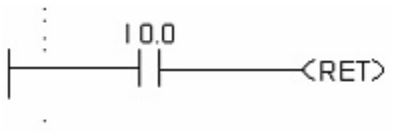
BEC

Lệnh này không có toán hạng và thực hiện kết thúc chương trình trong khối có điều liện hoặc vô điều kiện

Lệnh này tác động lên thanh ghi trạng thái như sau:

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
Không điều kiện	-	-	-	-	0	0	1	-	0
Có điều kiện	-	-	-	-	x	0	1	1	0

Ví dụ:



Một khối sẽ thoát khỏi chương trình nếu ngõ I0.0=1.

BÀI 2. CÁC LỆNH DẠNG SỐ

I. LỆNH MOVE

Ký hiệu:

LAD	FBD	STL
MOVE EN ENO IN OUT	MOVE EN OUT IN ENO	L <IN> T <OUT>

Các ngõ vào/ra EN, ENO, IN, OUT biểu diễn các toán hạng là địa chỉ ngõ vào/ra như sau: I, Q, M, L, D.

EN là ngõ vào cho phép lệnh MOVE hoạt động (Enable Input). Loại dữ liệu dạng BOOL.

ENO là ngõ ra cho phép các lệnh sau lệnh MOVE mà có nối với ngõ ENO hoạt động (Enable Output). Ngõ ENO có cùng trạng thái với ngõ vào EN. Loại dữ liệu dạng BOOL.

IN là ngõ vào biểu diễn địa chỉ nguồn (Source Address). Loại dữ liệu có chiều dài 8, 16, 32 Bit.

OUT là ngõ ra biểu diễn địa chỉ đích (Destination Address). Loại dữ liệu có chiều dài 8, 16, 32 Bit.

Hoạt động của lệnh MOVE:

Ở dạng LAD/FBD thì khi ngõ vào EN được kích hoạt thì giá trị ngõ vào IN được sao chép tới địa chỉ tại ngõ ra OUT. Ngõ ENO có cùng trạng thái với ngõ EN.

Ở dạng STL thì lệnh nạp và truyền dữ liệu không phụ thuộc vào kết quả RLO. Dữ liệu được trao đổi nhờ bộ tích luỹ (ACCU).

Lệnh L (Load) ghi giá trị từ địa chỉ nguồn bên phải vào bộ tích lũy ACCU1, nội

dung của ACCU1 được chuyển vào bộ tích luỹ ACCU2. Trường hợp giá trị chuyển vào bộ tích luỹ có kích thước nhỏ hơn 16 Bit (Double Word) thì chúng sẽ ghi vào bộ tích luỹ theo thứ tự Byte thấp của Word thấp đến Byte cao của Word thấp đến Byte thấp của Word cao đến Byte cao của Word cao. Những Bit còn trống trong ACCU được ghi vào giá trị 0.

Lệnh T (Transfer) sao chép một phần hoặt tất cả nội dung của bộ tích lũy ACCU1 đến địa chỉ cụ thể. Lệnh T không ảnh hưởng đến bộ tích luỹ ACCU2.

Sơ lược về bộ tích luỹ (ACCU):

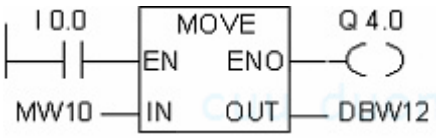
- Bộ tích luỹ là bộ nhớ phụ trong CPU dùng để trao đổi dữ liệu giữa những địa chỉ khác nhau và các phép toán so sánh, toán học. S7-300 có 2 bộ tích luỹ là ACCU1 và ACCU2, mỗi bộ có 32 Bit.

- ACCU1 là thanh ghi quan trọng nhất trong CPU. Khi một lệnh nạp (Load) được thi hành thì giá trị nạp được ghi vào ACCU1. Khi lệnh truyền (transfer) được thi hành thì đọc giá trị trong ACCU1, kết quả của các phép tính số học, các lệnh thay thế (Shift) và quay (Rotate)... cũng được ghi vào trong ACCU1.

- ACCU2: Khi lệnh nạp được thi hành, những nội dung cũ của ACCU1 trước tiên di chuyển sang ACCU2 và ACCU1 được xoá trước khi giá trị mới được chuyển vào ACCU1. ACCU2 cũng được sử dụng cho các lệnh so sánh, các phép tính số, số học, lệnh dịch...

Lệnh MOVE tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
1	-	-	-	-	0	1	1	1



Ví dụ:

Khi I0.0=1 thì lệnh MOVE được thực hiện. Khi đó giá trị chứa trong Word nhớ MW10 được sao chép đến Word của khối dữ liệu DB12.

II. CÁC LỆNH ĐỊNH THỜI (TIMER)

S7-300 có 5 loại định thời khác nhau: Timer đóng mạch chậm không nhớ (SD),

Timer đóng mạch chậm có nhớ (SS), Timer mở mạch chậm (SF), Timer xung (SP), Timer giữ độ rộng xung (SE).

Bốn loại Timer SD, SS, SP, SE bắt đầu hoạt động tại thời điểm có sườn lên của tín hiệu đầu vào. Còn Timer SF thì hoạt động tại thời điểm có sườn xuống của tín hiệu đầu vào.

Timer là bộ tạo thời gian trễ giữa tín hiệu đầu vào với tín hiệu đầu ra. Có một vùng đặt biệt của bộ nhớ được giành riêng cho timer trong CPU, vùng này chứa một Word 16 Bit cho mỗi địa chỉ thời gian. Mỗi loại Timer được đánh số từ 0 đến 255 tuỳ thuộc vào từng loại CPU.

Ký hiệu một vùng khai báo cho Timer (T-Word):

Số thứ tự bit

1514131211109876 543210 Giá trị thời gian trễ đặt trước cần khai báo cho timer

- 2 Bit 14 và 15 không sử dụng.

- 2 Bit 12 và 13 dùng để khai báo độ phân giải dưới dạng mã nhị phân.

. Độ phân giải được xác định là khoảng thời gian mà tại giá trị thời gian đó thì giá trị thời gian giảm đi 1 đơn vị.
. Khi thời gian đặt trước qua một hằng số S5T#... (S5Time#...) thì độ phân giải được phân chia tự động bằng hệ thống.
. Khi thời gian đặt trước qua bộ điều chỉnh số hoặc qua giao tiếp dữ liệu thì ngườii sử dụng phải đưa ra độ phân giải.

- 12 Bit còn lại từ 0 đến 11 chứa giá trị thời gian đặt trước (PV- Preset Value) là một số nguyên mã BCD trong khoảng từ 0 đến 999.

Thời gian trễ trong Timer được tính theo công thức:

τ =Độ phân giải*PV

Hoạt động của Timer:

Tại thời điểm kích Timer, giá trị PV được chuyển vào thanh ghi 16 Bit của Timer là thanh ghi T-Word, nội dung của T-Word gọi là giá trị tức thời được ký hiệu là CV (Current Value). Timer sẽ ghi nhớ khoảng thời gian trôi qua kể từ khi được kích bằng cách giảm dần một cách tương ứng với nội dung thanh ghi CV. Nếu nội dung thanh ghi trở về bằng 0 thì thời gian trễ mong muốn của Timer được báo ra ngoài bằng cách

thông báo. Trạng thái tín hiệu đầu ra tuỳ thuộc vào từng loại Timer nào được sử dụng.

Ký hiệu chung của các loại Timer:

LAD

FBD

STL

Tx Loại T S Q TV BI R BCD	Tx Loại T S BI TV BCD R Q	A <Set timer-S> -Cho phép T hoạt động L <Giá trị đặt trước-TV> -Đặt thời gian trễ SD/SS/SF/SP/SE <Tên TIMER> -Loại Timer A <Reset timer-R> -Tín hiệu xoá Timer R <Tên timer> -Lệnh xoá Timer L <Tên timer> -Nạp giá trị của Timer T <Địa chỉ BI> -Ngõ ra số nguyên LC <Tên timer> -Nạp giá trị BCD vào ACCU1 T <Địa chỉ BCD> -Ngõ ra số BCD A <Tên timer> -Trạng thái tín hiệu ra = <Địa chỉ ngõ ra–Q> -Nhận biết Timer hoạt động

T: Biểu diễn toán hạng là tên của Timer T. Loại dữ liệu dạng Timer.

S, TV, R, Q, BI, BCD: Biểu diễn các toán hạng là địa chỉ ngõ vào/ra như sau: I, Q, M, L, D.

S, R, Q: Biểu diễn các toán hạng là dữ liệu dạng BOOL.

TV: Biểu diễn toán hạng là dữ liệu dạng S5TIMER (dũ liệu dạng thời gian). Khai báo giá trị đặt trước ở dạng S5TIMER như sau: S5T#-xh-ym-xs-xxms. Trong đó: x, y, z, xx: Biểu diễn giá trị thời gian đặt trước.

h: Biến đơn vị chỉ giờ (hours).

m: Biến đơn vị chỉ phút (minutes).

s: Biến đơn vị chỉ giây (seconds).

ms: Biến đơn vị chỉ mili giây (milliseconds).

BI, BCD: Biểu diễn các toán hạng là dữ liệu dạng Word. BI là ngõ ra biểu diễn giá

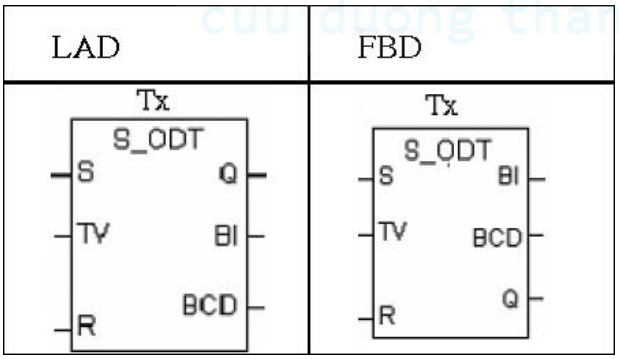
trị thời gian ở dạng số nhị phân (không có độ phân giải). BCD là ngõ ra chứa giá trị thời gian như một số BCD 12 Bit và độ phân giải (Bit 12 và 13). Bảng liệt kê độ phân giải ứng với những giá trị thời gian khai báo dạng S5TIMER:

Giá trị thời gian	Độ phân giải
10ms đến 9s990ms	0.01s
100ms đến 1m39s900ms	0.1s
1s đến 16m39s	1s
10s đến 2h46m30s	10s

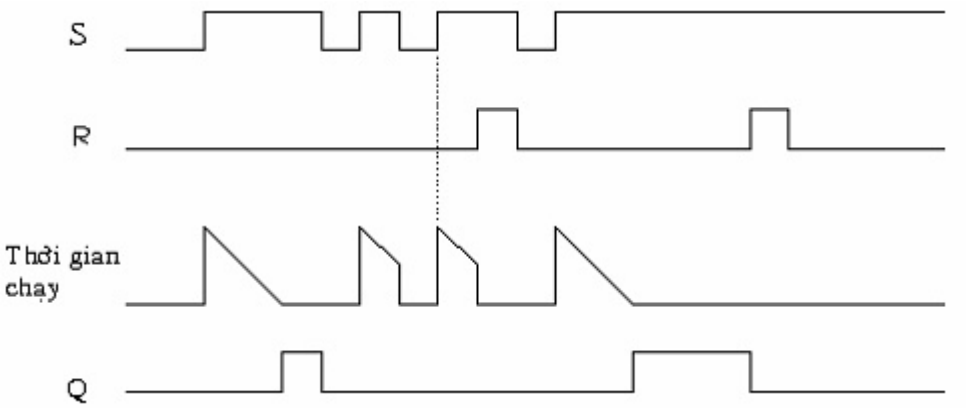
2.1. TIMER ĐÓNG MẠCH CHẬM KHÔNG NHỚ

(S-ODT/SD_ON-DELLAY TIMER)

Ký hiệu:



Giản đồ thời gian:



Hoạt động của Timer đóng mạch chậm không nhớ như sau:

Timer hoạt động khi kết quả RLO tại ngõ vào S thay đổi từ 0 lên 1. Khi đó timer bắt đầu hoạt động với giá trị thời gian đặt trước tại ngõ vào TV.

Trạng thái tín hiệu tại ngõ ra Q=1 khi timer hoạt động song, không có lỗi và ngõ vào S vẫn bằng 1. Nếu ngõ vào S thay đổi từ 1 về 0 trong khi Timer đang hoạt động thì ngõ ra Q trở về 0.

Ngõ ra BI và BCD dùng để quan sát giá trị hiện hành của Timer, ngõ ra BI để quan sát dạng số nhị phân, ngõ ra BCD để quan sát dạng số BCD. Giá trị thời gian hiện hành là giá trị ban đầu của TV trừ đi giá trị thời gian đã hoạt động của Timer, tính từ khi Timer hoạt động.

Khi kết quả RLO tại ngõ vào R là 1 thì giá trị thời gian hiện hành và độ phân giải bị xoá, ngõ ra Q ở trạng thái reset.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR

CC1 CC0

OV

OS

OR

STA

RLO

FC

--

-

-

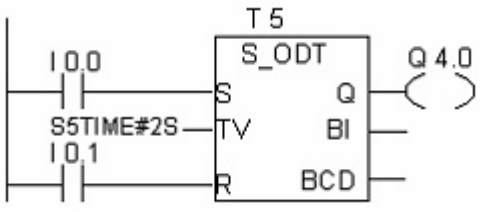
-

x

x

x

1

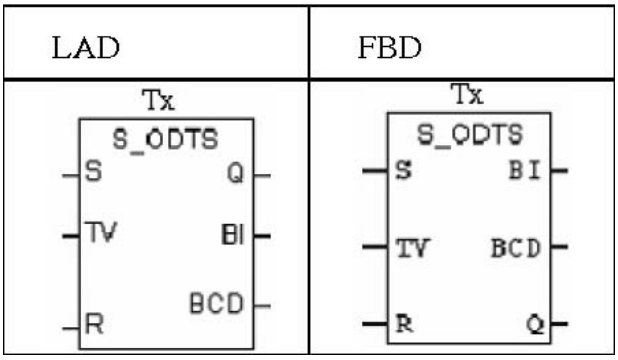


Ví dụ:

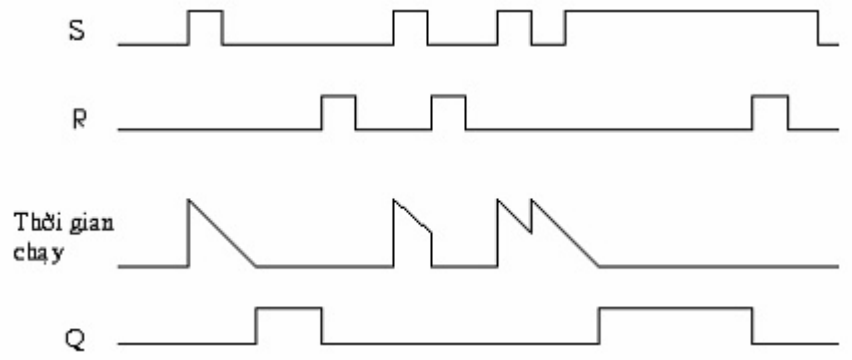
Nếu trạng thái tín hiệu tại I0.0 thay đổi từ 0 lên 1 thì T5 hoạt động, timer hoạt động đến 2s nếu ngõ vào I0.0 vẫn bằng 1. Nếu Timer hoạt động trong thời gian nhỏ hơn 2s mà I0.0 trở về 0 thì Timer ngừng hoạt động. Nếu tín hiệu tại ngõ vào I0.1 thay đổi từ 0 lên 1 thì Timer sẽ bị Reset. Trong thời gian Timer hoạt động, nếu thời gian I0.0 ở mức 1 lớn hơn 2s thì ngõ ra Q4.0 sẽ lên mức 1 mãi cho đến khi I0.0 về 0.

2.2. TIMER ĐÓNG MẠCH CHẬM CÓ NHỚ (S-ODTS/SS_RETENTIVE ON-DELLAY TIMER)

Ký hiệu:



Giản đồ thời gian:



Hoạt động của Timer đóng mạch chậm có nhớ như sau:

Timer hoạt động khi kết quả RLO tại ngõ vào S thay đổi từ 0 lên 1. Khi đó timer bắt đầu hoạt động với giá trị thời gian đặt trước tại ngõ vào TV và tiếp tục hoạt động cho dù ngõ vào S thay đổi về 0 trong suốt thời gian đó. Nếu tín hiệu tại ngõ vào S thay đổi từ 0 lên 1 trong khi Timer đang hoạt động thì Timer sẽ khởi động mới lại.

Trạng thái tín hiệu tại ngõ ra Q=1 khi timer hoạt động song, không có lỗi thì không cần chú ý đến trạng thái tín hiệu ngõ vào S là 0 hay 1. Ngõ ra Q chỉ về 0 khi có lệnh Reset.

Ngõ ra BI và BCD dùng để quan sát giá trị hiện hành của Timer, ngõ ra BI để quan sát dạng số nhị phân, ngõ ra BCD để quan sát dạng số BCD. Giá trị thời gian hiện hành là giá trị ban đầu của TV trừ đi giá trị thời gian đã hoạt động của Timer, tính từ khi Timer hoạt động.

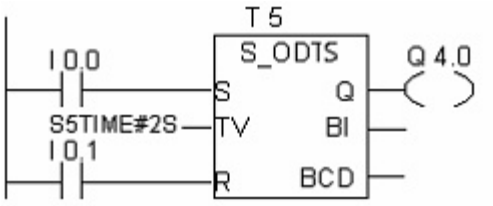
Khi kết quả RLO tại ngõ vào R là 1 thì giá trị thời gian hiện hành và độ phân giải bị xoá, ngõ ra Q ở trạng thái reset.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC

-	-	-	-	-	x	x	x	1

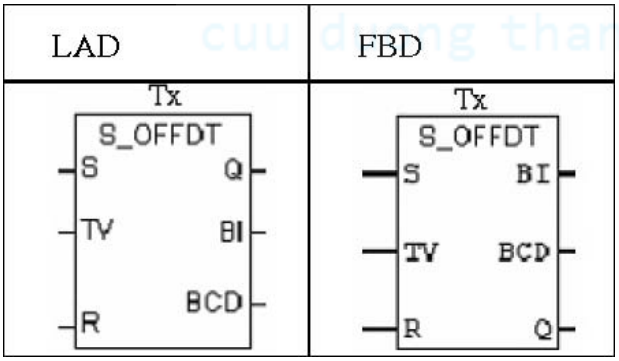
Ví dụ:



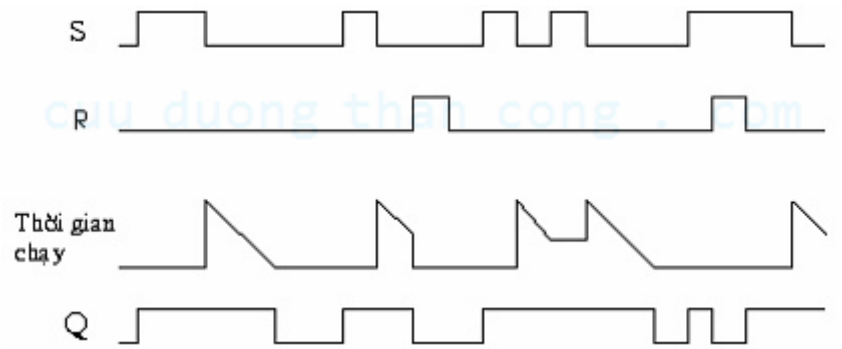
Nếu trạng thái tín hiệu tại I0.0 thay đổi từ 0 lên 1 thì T5 hoạt động, timer vẫn hoạt động cho dù ngõ vào S ở mức 0 hay 1. Nếu ngõ vào S thay đổi từ 0 lên 1 trong khi Timer đang hoạt động thì Timer sẽ khởi động mới trở lại. Nếu tín hiệu tại ngõ vào I0.1 thay đổi từ 0 lên 1 thì Timer sẽ bị Reset. Ngõ ra Q4.0 sẽ lên mức 1 mãi khi Timer hoạt động song, Q4.0 chỉ Reset khi I0.1=1.

2.3. TIMER MỞ MẠCH CHẬM (S-OFFDT/SF_OFF DELLAY TIMER)

Ký hiệu:



Giản đồ thời gian:



Hoạt động của Timer mở mạch chậm như sau:

Timer hoạt động khi kết quả RLO tại ngõ vào S thay đổi từ 1 về 0. Khi đó Timer hoạt động trong khoảng thời gian đặt trước tại ngõ vào TV. Nếu tín hiệu tại ngõ vào S thay đổi từ 0 lên 1 trong khi Timer đang hoạt động thì Timer sẽ dừng và thời gian kế tiếp trạng thái tín hiệu của S thay đổi từ 1 về 0 thì Timer sẽ bắt đầu hoạt động lại từ đầu.

Trạng thái tín hiệu tại ngõ ra Q=1 khi ngõ vào S thay đổi từ 0 lên 1, nếu trạng thái của S trở về 0 thì Q cũng vẫn bằng 1, sau khi Timer hoạt động song ngõ ra Q mới trở về
0.

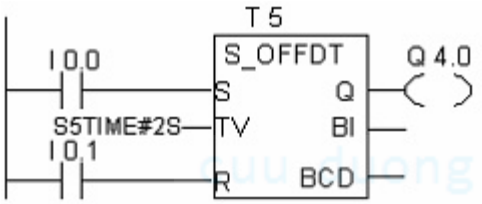
Ngõ ra BI và BCD dùng để quan sát giá trị hiện hành của Timer, ngõ ra BI để quan sát dạng số nhị phân, ngõ ra BCD để quan sát dạng số BCD. Giá trị thời gian hiện hành là giá trị ban đầu của TV trừ đi giá trị thời gian đã hoạt động của Timer, tính từ khi Timer hoạt động.

Khi kết quả RLO tại ngõ vào R là 1 thì giá trị thời gian hiện hành và độ phân giải bị xoá, ngõ ra Q ở trạng thái reset. Nếu trạng thái tín hiệu tại hai ngõ vào R và S đều bằng 1 thì ngõ ra Q không được Set cho đến khi ngõ R=0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

Ví dụ:



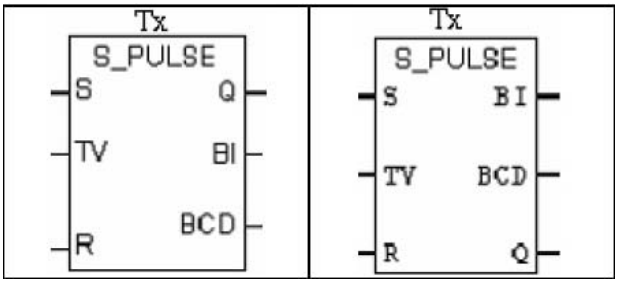
Nếu trạng thái tín hiệu tại I0.0 thay đổi từ 0 lên 1 thì Q4.0=1, T5 chưa hoạt động. Khi S thay đổi từ 1 về 0 thì Timer hoạt động, Q4.0 vẫn bằng 1 đến khi Timer hoạt động song thì Q4.0 trở về 0. Nếu ngõ vào S thay đổi từ 0 lên 1 trong khi Timer đang hoạt động thì Timer sẽ dừng lại đến khi S thay đổi từ 1 về 0. Nếu tín hiệu tại ngõ vào I0.1 thay đổi từ 0 lên 1 thì Timer sẽ bị Reset.

2.4. TIMER XUNG (S-PULSE/SP_PULSE TIMER)

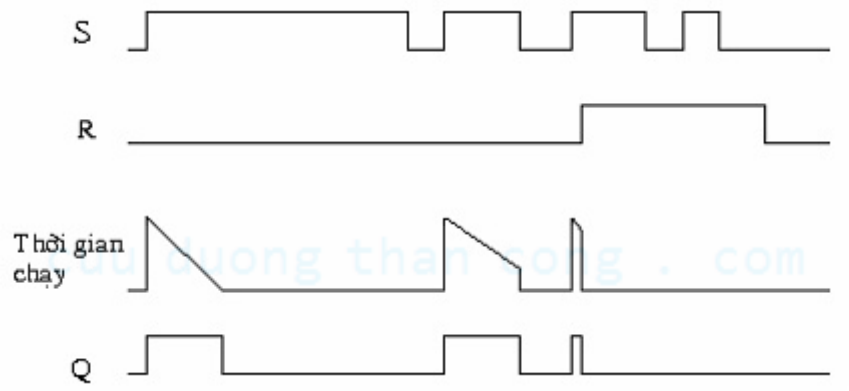
Ký hiệu:

LAD

FBD



Giản đồ thời gian:



Hoạt động của Timer xung như sau:

Timer hoạt động khi kết quả RLO tại ngõ vào S thay đổi từ 0 lên 1. Khi đó Timer hoạt động trong khoảng thời gian đặt trước tại ngõ vào TV nếu thời gian ở mức 1 của ngõ vào S lớn hơn thời gian đặt trước, còn nếu thời gian ở mức 1 của ngõ vào S nhỏ hơn thời gian đặt trước TV thì Timer sẽ ngưng hoạt động.

Trạng thái tín hiệu tại ngõ ra Q=1 khi Timer hoạt động, ngược lại Q=0 khi Timer ngưng hoạt động.

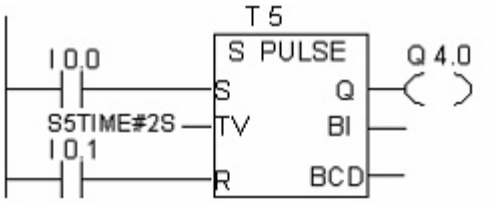
Ngõ ra BI và BCD dùng để quan sát giá trị hiện hành của Timer, ngõ ra BI để quan sát dạng số nhị phân, ngõ ra BCD để quan sát dạng số BCD. Giá trị thời gian hiện hành là giá trị ban đầu của TV trừ đi giá trị thời gian đã hoạt động của Timer, tính từ khi Timer hoạt động.

Khi kết quả RLO tại ngõ vào R là 1 thì giá trị thời gian hiện hành và độ phân giải bị xoá, ngõ ra Q ở trạng thái reset. Khi Timer hoạt động song hoặc tín hiệu tại S chuyển từ 1 về 0 thì ngõ ra Q cũng bị Reset. Nếu trạng thái tín hiệu tại hai ngõ vào R

và S đều bằng 1 thì ngõ ra Q không được Set cho đến khi ngõ R=0.

Lệnh này tác động lên thanh ghi trạng thái như sau: Ví dụ:

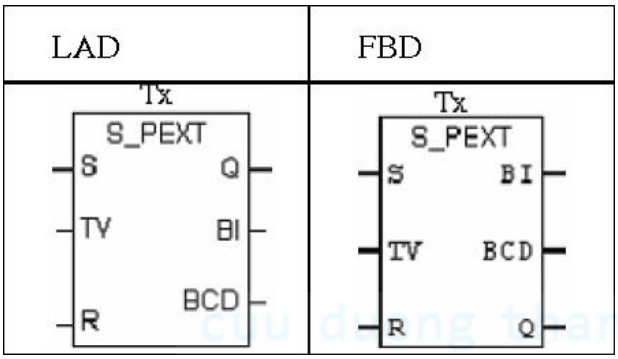
BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1



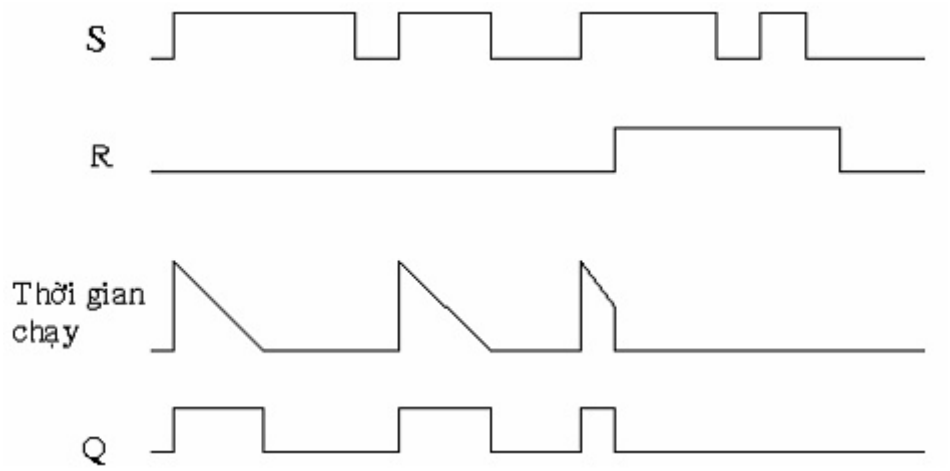
Nếu trạng thái tín hiệu tại I0.0 thay đổi từ 0 lên 1 thì Q4.0=1, T5 hoạt động. Timer hoạt động đến 2s nếu I0.0=1 trong 2s. Nếu I0.0=1 trong thời gian nhỏ hơn 2s thì Timer ngưng hoạt động. Nếu I0.1 thay đổi từ 0 lên 1 trong thời gian Timer đang hoạt động thì Timer sẽ bị Reset. Ngõ ra Q4.0=1 trong thời gian Timer hoạt động, Q4.0=0 khi Timer Reset.

2.5. TIMER GIỮ ĐỘ RỘNG XUNG (S-PEXT/SE_EXTENDED PULSE TIMER)

Ký hiệu:



Giản đồ thời gian:



Hoạt động của Timer giữ độ rộng xung như sau:

Timer hoạt động khi kết quả RLO tại ngõ vào S thay đổi từ 0 lên 1. Khi đó Timer hoạt động trong khoảng thời gian đặt trước tại ngõ vào TV. Nếu thời gian ở mức 1 của ngõ vào S lớn hơn hay nhỏ hơn thời gian đặt trước TV thì Timer cũng chỉ hoạt động trong khoảng thời gian đặt trước TV. Nếu tín hiệu tại ngõ vào S thay đổi từ 0 lên 1 trong khi Timer đang hoạt động thì Timer sẽ khởi động mới lại.

Trạng thái tín hiệu tại ngõ ra Q=1 khi Timer hoạt động, ngược lại Q=0 khi Timer ngưng hoạt động.

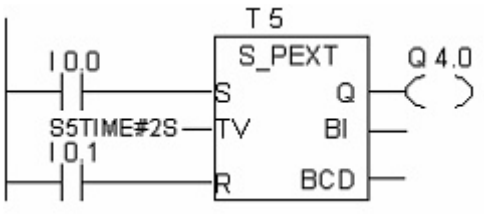
Ngõ ra BI và BCD dùng để quan sát giá trị hiện hành của Timer, ngõ ra BI để quan sát dạng số nhị phân, ngõ ra BCD để quan sát dạng số BCD. Giá trị thời gian hiện hành là giá trị ban đầu của TV trừ đi giá trị thời gian đã hoạt động của Timer, tính từ khi Timer hoạt động.

Khi kết quả RLO tại ngõ vào R là 1 thì giá trị thời gian hiện hành và độ phân giải bị xoá, ngõ ra Q ở trạng thái reset hoặc khi Timer hoạt động thì ngõ ra Q cũng bị Reset. Nếu trạng thái tín hiệu tại hai ngõ vào R và S đều bằng 1 thì ngõ ra Q không được Set cho đến khi ngõ R=0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

Ví dụ:



Nếu trạng thái tín hiệu tại I0.0 thay đổi từ 0 lên 1 thì Q4.0=1, T5 hoạt động. Timer hoạt động đến 2s cho dù thời gian I0.0=1 lớn hơn hay nhỏ hơn 2s. Nếu I0.1 thay đổi từ 0 lên 1 trong thời gian Timer đang hoạt động thì Timer sẽ bị Reset. Ngõ ra Q4.0=1 trong thời gian Timer hoạt động, Q4.0=0 khi Timer Reset.

2.6. CÁC TIMER DẠNG BIT

Tất cả các chức năng của Timer có thể được khởi động bằng những lệnh bit đơn giản.

Tất cả các Timer dạng bit này tác động lên thanh ghi trang thái như nhau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	-	-	0

2.6.1. TIMER XUNG (SP_PULSE TIMER COIL)

Ký hiệu:

LAD	FBD	STL
n x.y Tx (SP) TV	n x.y Tx SP TV	A n x.y L <TV> SP <T x>

n: Biểu diễn toán hạng là dữ liệu dạng nhị phân, toán hạng là địa chỉ lần lượt là: I, Q, M, L, D. nx.y dùng để cho phép Timer hoạt động.

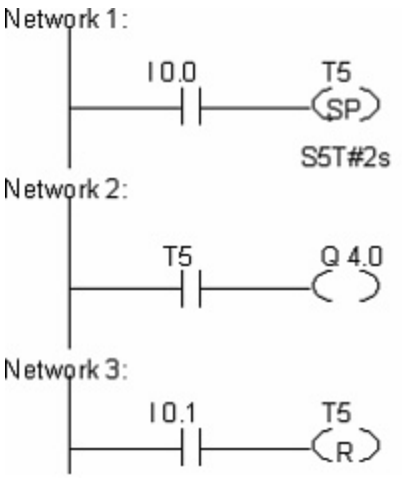
T: Biểu diễn toán hạng là dữ liệu dạng Timer, toán hạng là địa chỉ dạng T, diễn tả tên của Timer có số lượng Timer tuỳ thuộc vào từng loại CPU.

TV: Biểu diễn toán hạng là dữ liệu dạng S5Timer (dạng giá trị thời gian), toán hạng là địa chỉ lần lượt là: I, Q, M, L, D. TV dùng để đặt trước giá trị thời gian.

Hoạt động của Timer SP:

Timer hoạt động khi kết quả RLO tại ngõ vào n x.y thay đổi từ 0 lên 1. Khi đó Timer hoạt động trong khoảng thời gian đặt trước tại ngõ vào TV nếu thời gian ở mức 1 của ngõ vào n x.y lớn hơn thời gian đặt trước, còn nếu thời gian ở mức 1 của ngõ vào nx.y nhỏ hơn thời gian đặt trước TV thì Timer sẽ ngưng hoạt động.

Ví dụ:



Tín hiệu ngõ vào I0.0 thay đổi từ 0 lên 1 thì T5 hoạt động. T5 tiếp tục hoạt động trong thời gian 2s nếu I0.0 ở mức 1 trong thời gian lớn hơn hoặc bằng 2s. Nếu I0.0 ở mức 1 trong thời gian nhỏ hơn 2s thì Timer ngưng hoạt động. Trong thời gian T5 hoạt động thì trạng thái ngõ ra Q4.0=1. Nếu trạng thái I0.1 =1 thì T5 sẽ bị Reset.

2.6.2. TIMER GIỮ ĐỘ RỘNG XUNG (SE_EXTENDED PULSE TIMER COIL)

Ký hiệu:

LAD	FBD	STL
n x.y Tx (SE) TV	n x.y — Tx SE — TV	A n x.y L <TV> SE <T x>

n: Biểu diễn toán hạng là dữ liệu dạng nhị phân, toán hạng là địa chỉ lần lượt là: I, Q, M, L, D. nx.y dùng để cho phép Timer hoạt động.

T: Biểu diễn toán hạng là dữ liệu dạng Timer, toán hạng là địa chỉ dạng T, diễn tả tên của Timer có số lượng Timer tuỳ thuộc vào từng loại CPU.

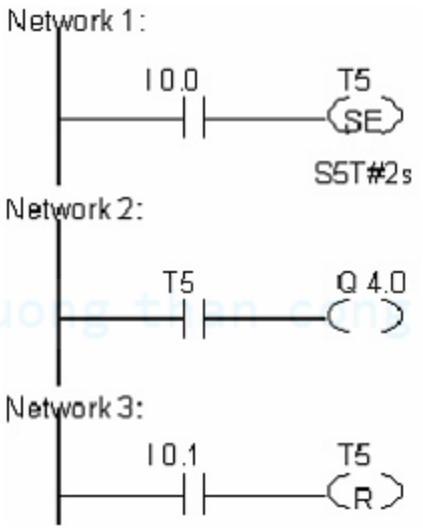
TV: Biểu diễn toán hạng là dữ liệu dạng S5Timer (dạng giá trị thời gian), toán

hạng là địa chỉ lần lượt là: I, Q, M, L, D. TV dùng để đặt trước giá trị thời gian.

Hoạt động của Timer SE:

Timer hoạt động khi kết quả RLO tại ngõ vào n x.y thay đổi từ 0 lên 1. Khi đó Timer hoạt động trong khoảng thời gian đặt trước tại ngõ vào TV. Nếu thời gian ở mức 1 của ngõ vào n x.y lớn hơn hay nhỏ hơn thời gian đặt trước TV thì Timer cũng chỉ hoạt động trong khoảng thời gian đặt trước TV. Nếu tín hiệu tại ngõ vào n x.y thay đổi từ 0 lên 1 trong khi Timer đang hoạt động thì Timer sẽ khởi động mới lại.

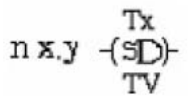
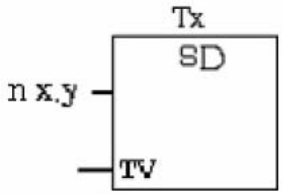
Ví dụ:



Tín hiệu ngõ vào I0.0 thay đổi từ 0 lên 1 thì T5 hoạt động. T5 tiếp tục hoạt động trong thời gian 2s cho dù thời gian của I0.0 ở mức lớn hơn hay nhỏ hơn 2s. Khi T5 hoạt động mà I0.0 thay đổi từ 0 lên 1 thì T5 hoạt động lại từ đầu. Trong thời gian T5 hoạt động thì trạng thái ngõ ra Q4.0=1. Nếu trạng thái I0.1 =1 thì T5 sẽ bị Reset.

2.6.3. **TIMER ĐÓNG MẠCH CHẬM KHÔNG NHỚ (SD_ON DELAY TIMER COIL)**

Ký hiệu:

LAD	FBD	STL
		A n x.y L <TV> SD <T x>

n: Biểu diễn toán hạng là dữ liệu dạng nhị phân, toán hạng là địa chỉ lần lượt là: I, Q, M, L, D. nx.y dùng để cho phép Timer hoạt động.

T: Biểu diễn toán hạng là dữ liệu dạng Timer, toán hạng là địa chỉ dạng T, diễn tả tên của Timer có số lượng Timer tuỳ thuộc vào từng loại CPU. TV: Biểu diễn toán hạng là dữ liệu dạng S5Timer (dạng giá trị thời gian), toán hạng là địa chỉ lần lượt là: I, Q, M, L, D. TV dùng để đặt trước giá trị thời gian.

Hoạt động của Timer SD:

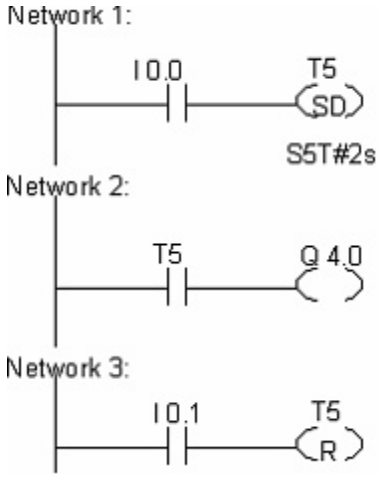
Timer hoạt động khi kết quả RLO tại ngõ vào n x.y thay đổi từ 0 lên 1. Khi đó

timer

bắt đầu hoạt động với giá trị thời gian đặt trước tại ngõ vào TV.

Trạng thái tín hiệu của Timer lên 1 khi timer hoạt động song, không có lỗi và ngõ vào n x.y vẫn bằng 1. Nếu ngõ vào n x.y thay đổi từ 1 về 0 trong khi Timer đang hoạt động thì Timer bị Reset.

Ví dụ:



Tín hiệu ngõ vào I0.0 thay đổi từ 0 lên 1 thì T5 hoạt động. T5 tiếp tục hoạt động trong thời gian 2s nếu I0.0 ở mức 1 trong thời gian lớn hơn hoặc bằng 2s. Nếu I0.0 ở mức 1 trong thời gian nhỏ hơn 2s thì Timer ngưng hoạt động. Khi T5 hoạt động song nếu I0.0 ở mức 1 trong thời gian lớn hơn 2s thì trạng thái ngõ ra Q4.0=1, cho tới khi I0.0=0. Nếu trạng thái I0.1 =1 thì T5 sẽ bị Reset.

2.6.4. TIMER ĐÓNG MẠCH CHẬM CÓ NHỚ (SS_RETENTIVE ON DELAY TIMER COIL)

Ký hiệu:

LAD	FBD	STL
Tx n x.y -(SS)- TV	Tx SS n x.y TV	A n x.y L <TV> SS <T x>

n: Biểu diễn toán hạng là dữ liệu dạng nhị phân, toán hạng là địa chỉ lần lượt là: I, Q, M, L, D. nx.y dùng để cho phép Timer hoạt động.

T: Biểu diễn toán hạng là dữ liệu dạng Timer, toán hạng là địa chỉ dạng T, diễn tả tên của Timer có số lượng Timer tuỳ thuộc vào từng loại CPU.

TV: Biểu diễn toán hạng là dữ liệu dạng S5Timer (dạng giá trị thời gian), toán hạng là địa chỉ lần lượt là: I, Q, M, L, D. TV dùng để đặt trước giá trị thời gian.

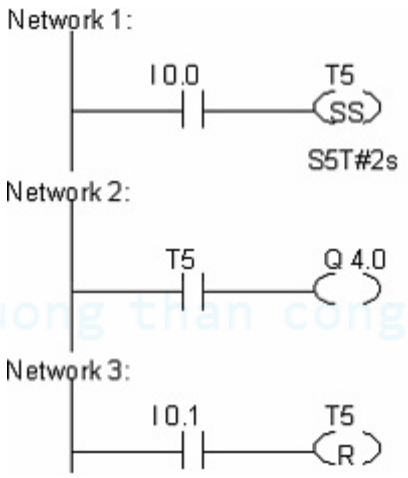
Hoạt động của Timer SD:

Timer hoạt động khi kết quả RLO tại ngõ vào n x.y thay đổi từ 0 lên 1. Khi đó

timer bắt đầu hoạt động với giá trị thời gian đặt trước tại ngõ vào TV và tiếp tục hoạt động cho dù ngõ vào n x.y thay đổi về 0 trong suốt thời gian đó. Nếu tín hiệu tại ngõ vào nx.y thay đổi từ 0 lên 1 trong khi Timer đang hoạt động thì Timer sẽ khởi động mới lại.

Trạng thái tín hiệu của Timer là 1 khi timer hoạt động song, không có lỗi thì không cần chú ý đến trạng thái tín hiệu ngõ vào n x.y là 0 hay 1. Timer chỉ về 0 khi có lệnh Reset.

Ví dụ:



Tín hiệu ngõ vào I0.0 thay đổi từ 0 lên 1 thì T5 hoạt động. T5 tiếp tục hoạt động trong thời gian 2s cho dù I0.0 ở mức 1 trong thời gian lớn hơn hay nhỏ hơn 2s. Khi T5 hoạt động song thì trạng thái ngõ ra Q4.0=1, cho tới khi có lệnh Reset ở ngõ vào I0.1.

2.6.5. **TIMER MỞ MẠCH CHẬM (SF_OFF DELAY TIMER COIL)**

Ký hiệu:

LAD	FBD	STL
n x.y Tx (SF) TV	n x.y — Tx SF — TV	A n x.y L <TV> SF <T x>

n: Biểu diễn toán hạng là dữ liệu dạng nhị phân, toán hạng là địa chỉ lần lượt là: I, Q, M, L, D. nx.y dùng để cho phép Timer hoạt động.

T: Biểu diễn toán hạng là dữ liệu dạng Timer, toán hạng là địa chỉ dạng T, diễn tả tên của Timer có số lượng Timer tuỳ thuộc vào từng loại CPU.

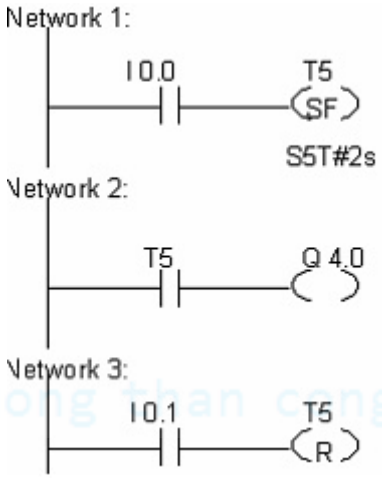
TV: Biểu diễn toán hạng là dữ liệu dạng S5Timer (dạng giá trị thời gian), toán hạng là địa chỉ lần lượt là: I, Q, M, L, D. TV dùng để đặt trước giá trị thời gian.

Hoạt động của Timer SF:

Timer hoạt động khi kết quả RLO tại ngõ vào n x.y thay đổi từ 1 về 0. Khi đó Timer hoạt động trong khoảng thời gian đặt trước tại ngõ vào TV. Nếu tín hiệu tại ngõ vào nx.y thay đổi từ 0 lên 1 trong khi Timer đang hoạt động thì Timer sẽ dừng và thời gian kế tiếp trạng thái tín hiệu của n x.y thay đổi từ 1 về 0 thì Timer sẽ bắt đầu hoạt động lại từ đầu.

Trạng thái tín hiệu của Timer Tx=1 khi ngõ vào n x.y thay đổi từ 0 lên 1, nếu trạng thái của n x.y trở về 0 thì Tx cũng vẫn bằng 1, sau khi Timer hoạt động song thì Tx mới trở về 0.

Ví dụ:

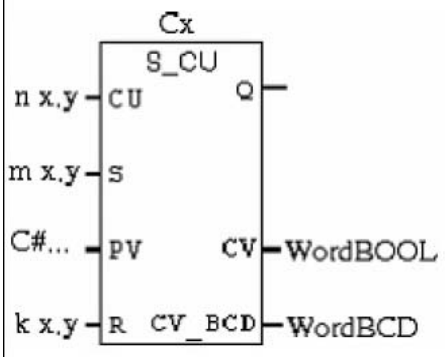
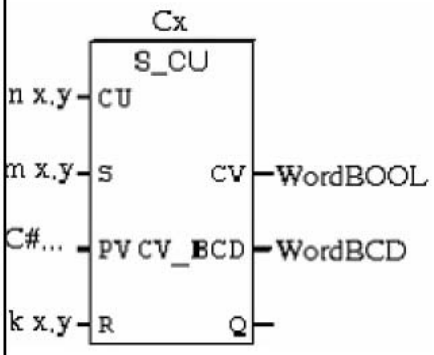


Tín hiệu ngõ vào I0.0 thay đổi từ 1 về 0 thì T5 hoạt động. T5 tiếp tục hoạt động trong thời gian 2s nếu I0.0 vẫn ở mức 0. Nếu trong thời gian Timer hoạt động mà I0.0 lên 1 thì Timer ngưng hoạt động, khi I0.0 xuống mức 0 thì Timer hoạt động trở lại. Trạng thái ngõ ra Q4.0=1 khi I0.0 chuyển từ 0 lên 1 hoặc khi Timer đang hoạt động. Khi Q4.0=1 mà I0.1=1 thì Q4.0=0 và Timer bị Reset.

III. CÁC LỆNH ĐẾM(COUNTER)

3.1. BỘ ĐẾM LÊN (S-CU_UP COUNTER)

Ký hiệu:

LAD	FBD	STL
		A n x.y CU <Cx> L C#... A m x.y S <C x> A k x.y R <Cx> L <Cx> T WordBOOL LC <Cx> T WordBCD A <Cx> = Q x.y

C: Biểu diễn toán hạng là tên của bộ đếm C. Loại dữ liệu dạng Counter, diễn tả tên của bộ đếm, số bộ đếm tuỳ thuộc vào loại CPU.

S, PV, R, Q, CV, CV-BCD: Biểu diễn các toán hạng là địa chỉ ngõ vào/ra như sau: I, Q, M, L, D.

S, R, Q: Biểu diễn các toán hạng là dữ liệu dạng BOOL.

PV: Biểu diễn toán hạng là dữ liệu dạng Word. Khai báo giá trị đặt trước số đếm.

CV, CV- BCD: Biểu diễn các toán hạng là dữ liệu dạng Word. CV là ngõ ra biểu diễn giá trị đếm hiện hành, ở dạng số thập lục phân (Hexadeximal). BCD là ngõ ra chứa giá trị hiện hành, ở dạng số BCD.

Q: Biểu diễn trạng thái giá trị đếm. Giá trị đếm bằng 0 thì Q=0, giá trị đếm bằng 1 thì Q=1.

CU: Biểu diễn toán hạng là dữ liệu dạng BOOL, địa chỉ dạng I, Q, M, L, D. Dùng để cho phép bộ đếm đếm lên khi kết quả tại ngõ CU=1.

Hoạt động của bộ đếm:

Khi trạng thái ngõ vào CU thay đổi từ 0 lên 1 thì bộ đếm sẽ đếm tăng lên 1 và giá trị bộ đếm nhỏ hơn 999. Bộ đếm sẽ đếm lên với giá trị đặt trước tại ngõ PV khi trạng thái tín hiệu ngõ vào S thay đổi từ 0 lên 1 (Bộ đếm đếm tối đa từ 0 đếm 999 trong mã nhị phân). Bộ đếm sẽ bị Reset về 0 nếu trạng thái tín hiệu tại ngõ vào R=1. Q=1 nếu giá trị bộ đếm khác 0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR

CC1 CC0

OV

OS

OR

STA

RLO

FC

--

-

-

-

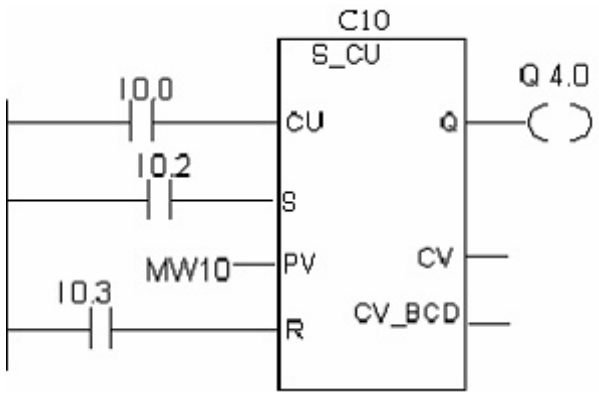
x

x

x

1

Ví dụ:



Khi I0.2 thay đổi từ 0 lên 1 thì bộ đếm được đặt với giá trị tại ngõ vào PV. Khi I0.0 thay đổi từ 0 lên 1 thì bộ đếm sẽ đếm lên nếu giá trị bộ đếm C10 nhỏ hơn 999. Ngõ ra Q4.0=1 nếu giá trị C10 khác 0. Nếu I0.3 =1 thì bộ đếm sẽ bị Reset về 0.

3.2. BỘ ĐẾM XUỐNG (S-CD_DOWN COUNTER)

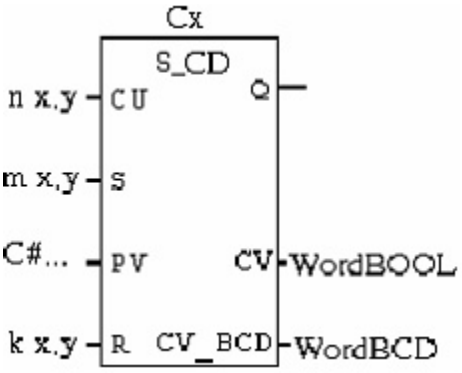
Ký hiệu:

LAD

FBD

STL

A n x.y
CD <Cx >
L C#...
A m x.y
S <C x>
A k x.y
R <Cx >
L <Cx>
T
WordBOOL

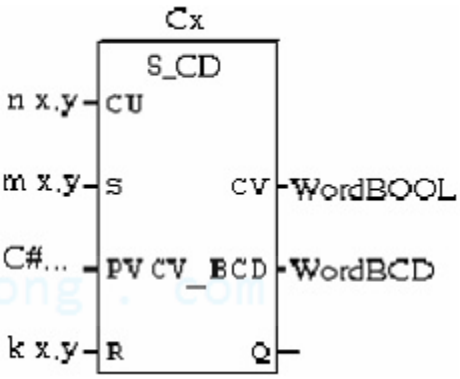


LC <Cx>

T WordBCD

A <Cx>

= Q x.y



C: Biểu diễn toán hạng là tên của bộ đếm C. Loại dữ liệu dạng Counter, diễn tả tên của bộ đếm, số bộ đếm tuỳ thuộc vào loại CPU. S, PV, R, Q, CV, CV-BCD: Biểu diễn các toán hạng là địa chỉ ngõ vào/ra như sau:

I, Q, M, L, D. S, R, Q: Biểu diễn các toán hạng là dữ liệu dạng BOOL. PV: Biểu diễn toán hạng là dữ liệu dạng Word. Khai báo giá trị đặt trước số đếm. CV, CV- BCD: Biểu diễn các toán hạng là dữ liệu dạng Word. CV là ngõ ra biểu diễn giá trị đếm hiện hành, ở dạng số thập lục phân (Hexadeximal). BCD là ngõ ra chứa giá trị hiện hành, ở dạng số BCD.

Q: Biểu diễn trạng thái giá trị đếm. Giá trị đếm bằng 0 thì Q=0, giá trị đếm bằng 1 thì Q=1.

CD: Biểu diễn toán hạng là dữ liệu dạng BOOL, địa chỉ dạng I, Q, M, L, D. Dùng để cho phép bộ đếm đếm xuống khi kết quả tại ngõ CD=1.

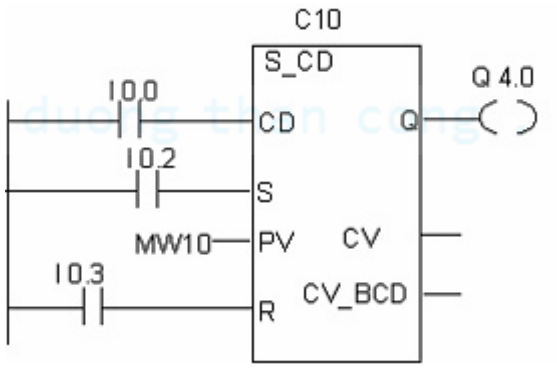
Hoạt động của bộ đếm:

Khi trạng thái ngõ vào CD thay đổi từ 0 lên 1 thì bộ đếm sẽ đếm giảm đi 1 và giá trị bộ đếm lớn hơn 0. Bộ đếm sẽ đếm xuống với giá trị đặt trước tại ngõ PV khi trạng thái tín hiệu ngõ vào S thay đổi từ 0 lên. Bộ đếm sẽ bị Reset về 0 nếu trạng thái tín hiệu tại ngõ vào R=1. Q=1 nếu giá trị bộ đếm khác 0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

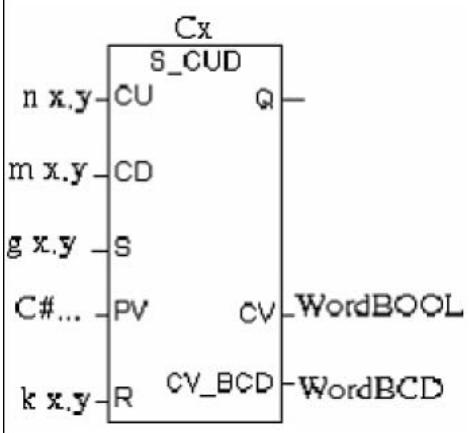
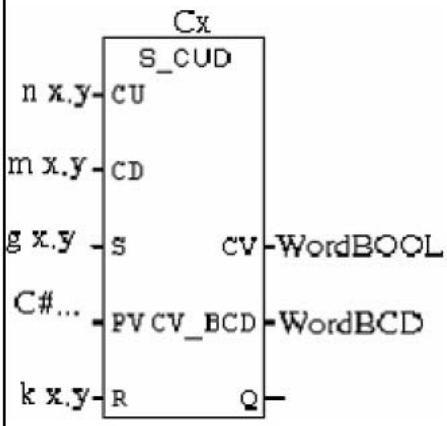
Ví dụ:



Khi I0.2 thay đổi từ 0 lên 1 thì bộ đếm được đặt với giá trị tại ngõ vào PV. Khi I0.0 thay đổi từ 0 lên 1 thì bộ đếm sẽ đếm xuống một đơn vị nếu giá trị bộ đếm C10 lớn hơn
0. Ngõ ra Q4.0=1 nếu giá trị C10 khác 0. Nếu I0.3 =1 thì bộ đếm sẽ bị Reset về 0.

3.3. BỘ ĐẾM LÊN- ĐẾM XUỐNG (S-CUD_UP/DOWN COUNTER)

Ký hiệu:

LAD	FBD	STL
		A n x.y CU <Cx> A m x.y CD <Cx> L C#... A g x.y S <C x> A k x.y R <Cx> L <Cx> T WordBOOL LC <Cx> T WordBCD A <Cx> = Q x.y

C: Biểu diễn toán hạng là tên của bộ đếm C. Loại dữ liệu dạng Counter, diễn tả tên của bộ đếm, số bộ đếm tuỳ thuộc vào loại CPU.

S, PV, R, Q, CV, CV-BCD: Biểu diễn các toán hạng là địa chỉ ngõ vào/ra như sau: I, Q, M, L, D.

S, R, Q: Biểu diễn các toán hạng là dữ liệu dạng BOOL.

PV: Biểu diễn toán hạng là dữ liệu dạng Word. Khai báo giá trị đặt trước số đếm.

CV, CV- BCD: Biểu diễn các toán hạng là dữ liệu dạng Word. CV là ngõ ra biểu diễn giá trị đếm hiện hành, ở dạng số thập lục phân (Hexadeximal). BCD là ngõ ra chứa giá trị hiện hành, ở dạng số BCD.

Q: Biểu diễn trạng thái giá trị đếm. Giá trị đếm bằng 0 thì Q=0, giá trị đếm bằng 1 thì Q=1.

CU/CD: Biểu diễn toán hạng là dữ liệu dạng BOOL, địa chỉ dạng I, Q, M, L, D. Dùng để cho phép bộ đếm đếm lên hoặc đếm xuống khi kết quả tại ngõ CD=1.

Hoạt động của bộ đếm:

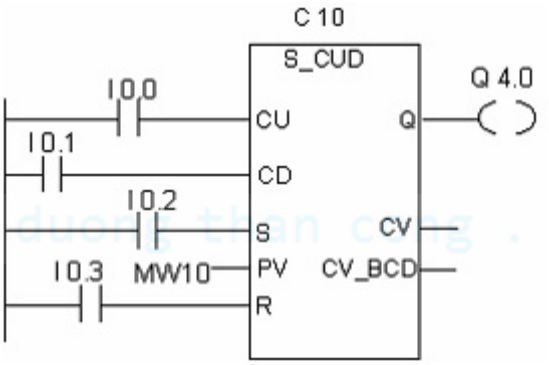
Khi trạng thái ngõ vào CU thay đổi từ 0 lên 1 thì bộ đếm sẽ đếm tăng thêm 1 và

giá trị bộ đếm nhỏ hơn 999. Khi trạng thái ngõ vào CD thay đổi từ 0 lên 1 thì bộ đếm sẽ đếm giảm đi 1 và giá trị bộ đếm lớn hơn 0. Bộ đếm sẽ đếm lên hoặc đếm xuống với giá trị đặt trước tại ngõ PV khi trạng thái tín hiệu ngõ vào S thay đổi từ 0 lên. Bộ đếm sẽ bị Reset về 0 nếu trạng thái tín hiệu tại ngõ vào R=1. Q=1 nếu giá trị bộ đếm khác 0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	x	x	x	1

Ví dụ:

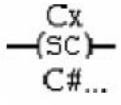
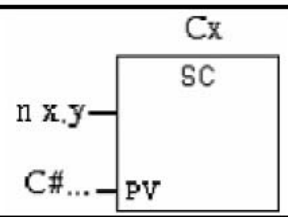


Khi I0.2 thay đổi từ 0 lên 1 thì bộ đếm được đặt với giá trị tại ngõ vào PV. Khi I0.0 thay đổi từ 0 lên 1 thì bộ đếm sẽ đếm tăng một đơn vị nếu giá trị bộ đếm C10 nhỏ hơn 999. Khi I0.1 thay đổi từ 0 lên 1 thì bộ đếm sẽ đếm giảm một đơn vị nếu giá trị bộ đếm C10 lớn hơn 0. Ngõ ra Q4.0=1 nếu giá trị C10 khác 0. Nếu I0.3 =1 thì bộ đếm sẽ bị Reset về 0.

3.4. CÁC BỘ ĐẾM DẠNG BIT

3.4.1. LỆNH ĐẶT GIÁ TRỊ ĐẾM (SC_SET COUNTER VALUE)

Ký hiệu:

LAD	FBD	STL
		A n x.y L C#... S Cx

C: Biểu diễn toán hạng là tên của bộ đếm C. Loại dữ liệu dạng Counter, diễn tả tên của bộ đếm, số bộ đếm tuỳ thuộc vào loại CPU. PV: Biểu diễn các toán hạng là địa chỉ ngõ vào như sau: I, Q, M, L, D. Biểu diễn toán hạng là dữ liệu dạng Word. Dùng để khai báo giá trị đặt trước số đếm. n x.y: Là ngõ vào dùng để cho phép lệnh SC hoạt động. Biểu diễn các toán hạng là địa chỉ ngõ vào như sau: I, Q, M, L, D. Biểu diễn toán hạng là dữ liệu dạng số nhị phân. Hoạt động: Khi trạng thái ngõ vào n x.y thay đổi từ 0 lên 1 thì lệnh đặt trước số đếm hoạt động,

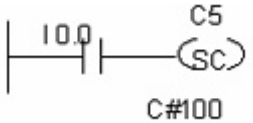
giá trị đặt trước số đếm sẽ truyền đến bộ đế Cx.

Lệnh này tác động lên thanh ghi trạng thái như

sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	x	-	0

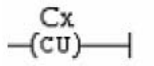
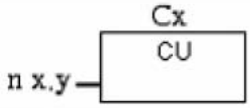
Ví dụ:



Bộ đếm C5 sẽ đặt trước giá trị là 100 khi ngõ vào I0.0 thay đổi từ 0 lên 1 và giá trị này vẫn duy trì khi I0.0=0.

3.4.2. LỆNH ĐẾM LÊN (CU_UP COUNTER)

Ký hiệu:

LAD	FBD	STL
		A n x.y CU Cx

C x: Biểu diễn toán hạng là tên của bộ đếm C. Loại dữ liệu dạng Counter, diễn tả tên của bộ đếm, số bộ đếm tuỳ thuộc vào loại CPU.

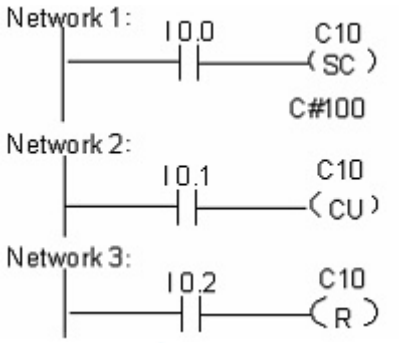
Hoạt động:

Bộ đếm sẽ đếm lên 1 đơn vị khi kết quả của phép toán Logic thay đổi từ 0 lên 1 và giá trị của bộ đếm nhỏ hơn 999. Nếu kết quả của phép toán Logic không thay đổi từ 0 lên 1 hoặc nếu giá trị của bộ đếm bằng 999 thì bộ đếm sẽ không thay đổi.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	-	-	0

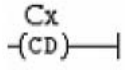
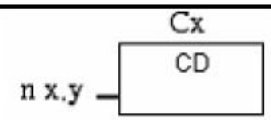
Ví dụ:



Bộ đếm C10 sẽ đặt trước giá trị là 100 khi ngõ vào I0.0 thay đổi từ 0 lên 1. Nếu trạng thái tín hiệu tại ngõ I0.1 thay đổi từ 0 lên 1 thì bộ đếm C10 sẽ tăng lên 1 khi giá trị C10 nhỏ hơn 999. Nếu trạng thái tín hiệu tại ngõ I0.1 không thay đổi từ 0 lên 1 thì C10 không thay đổi. Nếu I0.2=1 thì C10 sẽ bị Reset về 0.

3.4.3. LỆNH ĐẾM XUỐNG (CD_DOWN COUNTER)

Ký hiệu:

LAD	FBD	STL
		A n x.y CD Cx

Cx: Biểu diễn toán hạng là tên của bộ đếm C. Loại dữ liệu dạng Counter, diễn tả tên của bộ đếm, số bộ đếm tuỳ thuộc vào loại CPU.

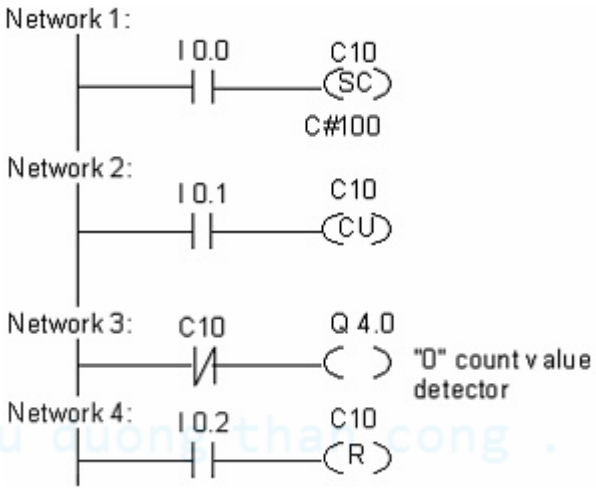
Hoạt động:

Bộ đếm sẽ đếm xuống 1 đơn vị khi kết quả của phép toán Logic thay đổi từ 0 lên 1 và giá trị của bộ đếm lớn hơn. Nếu kết quả của phép toán Logic không thay đổi từ 0 lên 1 hoặc nếu giá trị của bộ đếm bằng 0 thì bộ đếm sẽ không thay đổi.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
-	-	-	-	-	0	-	-	0

Ví dụ:



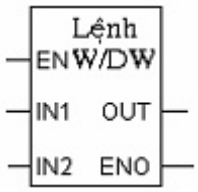
Bộ đếm C10 sẽ đặt trước giá trị là 100 khi ngõ vào I0.0 thay đổi từ 0 lên 1. Nếu trạng thái tín hiệu tại ngõ I0.1 thay đổi từ 0 lên 1 thì bộ đếm C10 sẽ tăng lên 1 khi giá trị C10 nhỏ hơn 999. Nếu trạng thái tín hiệu tại ngõ I0.1 không thay đổi từ 0 lên 1 thì C10 không thay đổi. Nếu I0.2=1 thì C10 sẽ bị Reset về 0. Nếu giá trị đếm C10=0 thì Q4.0 sẽ đặt lên 1, ngược lại Q4.0=0.

IV. CÁC PHÉP TOÁN LOGIC (WAND_W/DW, WOR_W/DW,

WXOR_W/DW)

Có 6 phép toán Logic là AND, OR, XOR theo Word và Word kép.

Các lệnh này có ký hiệu tương tự nhau như sau (ví dụ ở dạng STL):



EN là ngõ vào cho phép lệnh so sánh hoạt động (Enable Input). Loại toán hạng là dữ liệu dạng số nhị phân. Loại toán hạng là địa chỉ dạng: I, Q, L, M, D.

ENO là ngõ cho phép ngõ ra hoạt động (Enable Output). Ngõ ra ENO có cùng trạng thái với ngõ vào EN. Loại toán hạng là dữ liệu dạng số nhị phân. Loại toán hạng là địa chỉ dạng: I, Q, L, M, D.

IN1, IN2: Là giá trị ngõ vào thứ nhất và thứ hai. Loại toán hạng là dữ liệu ngõ vào dạng Word hoặc Word kép. Loại toán hạng là địa chỉ ngõ vào dạng: I, Q, L, M, D.

OUT: Giá trị ngõ ra của kết quả phép toán Logic. Biểu diễn toán hạng là dữ liệu ngõ ra dạng Word hoặc Word kép. Loại toán hạng là địa chỉ ngõ ra dạng: I, Q, L, M, D.

Các lệnh này tác động lên thanh ghi trạng thái như sau:

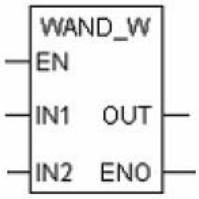
BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
1	x	0	0	-	x	1	1	1

4.1. LỆNH WAND-W (AND WORD)

Ký hiệu:

LAD/FBD

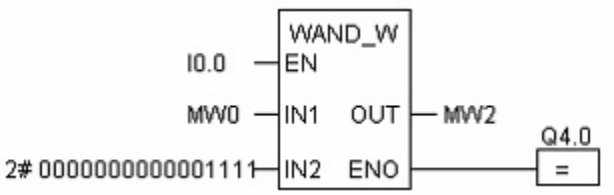
STL

	A <EN> L <IN1> L <IN2> A W T <OUT>

Các ngõ vào/ra IN1, IN2, OUT: Biểu diễn toán hạng là dữ liệu ngõ ra dạng Word.

Hoạt động: Lệnh WAND-W hoạt động khi trạng thái tín hiệu tại ngõ vào EN là 1. Khi đó hai dữ liệu dạng Word IN1 và IN2 sẽ And với nhau theo bảng sự thật của lệnh And, kết quả gửi ra ngõ OUT. ENO có cùng trạng thái với ngõ EN.

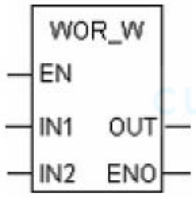
Ví dụ:



Lệnh thực hiện khi I0.0=1. Khi đó, ở ngõ ra chỉ có 4 Bit từ Bit 0 đếm Bit 3 có khả năng lên 1, các bit còn lại đều bằng 0. Ví dụ khi giá trị MW0=00110101 01100110 thì giá trị ở ngõ ra là 00000000 00000101. Q4.0=1 khi lệnh thực hiện.

4.2. LỆNH WOR-W (OR WORD)

Ký hiệu:

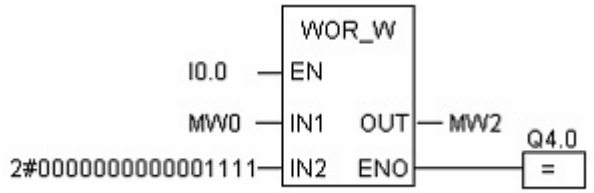
LAD/ FBD	STL
	A <EN> L <IN1> L <IN2> O W T <OUT>

Các ngõ vào/ra IN1, IN2, OUT: Biểu diễn toán hạng là dữ liệu ngõ ra dạng Word.

Hoạt động:

Lệnh WOR-W hoạt động khi trạng thái tín hiệu tại ngõ vào EN là 1. Khi đó hai dữ

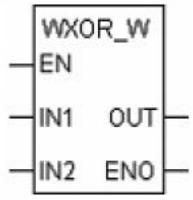
liệu dạng Word IN1 và IN2 sẽ OR với nhau theo bảng sự thật của lệnh OR, kết quả gửi ra ngõ OUT. ENO có cùng trạng thái với ngõ EN. Ví dụ:



Lệnh thực hiện khi I0.0=1. Khi đó, ở ngõ ra có 4 bit thấp luôn bằng 1, các bit còn lại có thể là 0 hay 1 tuỳ thuộc vào giá trị của MW0. Ví dụ khi giá trị MW0=00110101 01100110 thì giá trị ở ngõ ra là 00110101 01101111. Q4.0=1 khi lệnh thực hiện.

4.3. LỆNH WXOR-W (EXCLUSIVE OR WORD)

Ký hiệu:

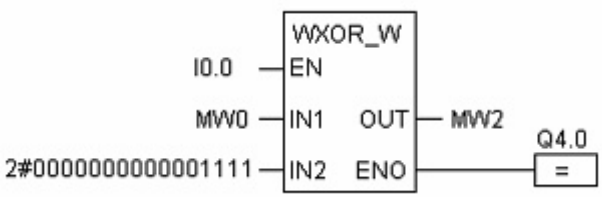
LAD/ FBD	STL
	A <EN> L <IN1> L <IN2> XOW T <OUT>

Các ngõ vào/ra IN1, IN2, OUT: Biểu diễn toán hạng là dữ liệu ngõ ra dạng Word.

Hoạt động:

Lệnh WXOR-W hoạt động khi trạng thái tín hiệu tại ngõ vào EN là 1. Khi đó hai dữ

liệu dạng Word IN1 và IN2 sẽ Xor với nhau theo bảng sự thật của lệnh Xor, kết quả gửi ra ngõ OUT. ENO có cùng trạng thái với ngõ EN. Ví dụ:

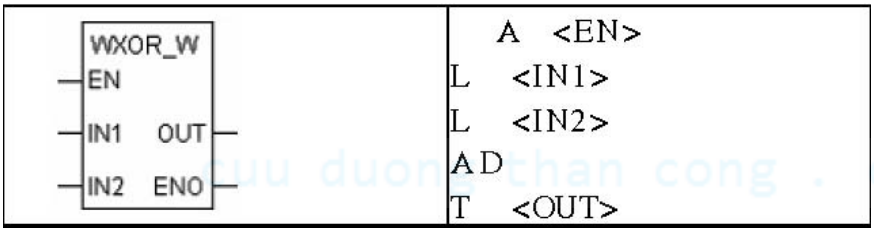


Lệnh thực hiện khi I0.0=1. Nếu giá trị MW0=00110101 01100110 thì giá trị ở ngõ ra là 00110101 01101001. Q4.0=1 khi lệnh thực hiện.

4.4. LỆNH WAND-DW (AND DOUBLE WORD)

Ký hiệu:

LAD/ FBD
STL

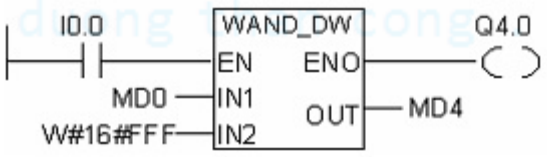


Các ngõ vào/ra IN1, IN2, OUT: Biểu diễn toán hạng là dữ liệu ngõ ra dạng Word kép.

Hoạt động:

Lệnh WAND-DW hoạt động khi trạng thái tín hiệu tại ngõ vào EN là 1. Khi đó hai dữ liệu dạng Word kép IN1 và IN2 sẽ and với nhau theo bảng sự thật của lệnh And, kết quả gửi ra ngõ OUT. ENO có cùng trạng thái với ngõ EN.

Ví dụ:



Lệnh thực hiện khi I0.0=1. Khi đó, ở ngõ ra chỉ có 12 bit thấp có khả năng lên 1, các bit còn lại đều bằng 0. Ví dụ khi giá trị MD0=00110101 01100110 00010010 00011101 thì giá trị ở ngõ ra là 00000000 00000000 00000010 00011101. Q4.0=1 khi lệnh thực hiện.

4.5. LỆNH WOR-DW (OR DOUBLE WORD)

Ký hiệu:

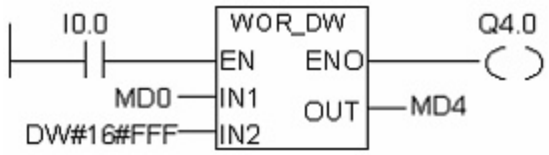
LAD/ FBD	STL
WOR_DW EN IN1 OUT IN2 ENO	A <EN> L <IN1> L <IN2> OD T <OUT>

Các ngõ vào/ra IN1, IN2, OUT: Biểu diễn toán hạng là dữ liệu ngõ ra dạng Word kép.

Hoạt động:

Lệnh WOR-DW hoạt động khi trạng thái tín hiệu tại ngõ vào EN là 1. Khi đó hai dữ liệu dạng Word kép IN1 và IN2 sẽ or với nhau theo bảng sự thật của lệnh or, kết quả gửi ra ngõ OUT. ENO có cùng trạng thái với ngõ EN.

Ví dụ:



Lệnh thực hiện khi I0.0=1. Khi đó, ở ngõ ra có 12 bit thấp luôn bằng 1, các bit còn lại có thể là 0 hay 1 tuỳ thuộc vào giá trị của MW0. Ví dụ khi giá trị MW0= 01010101 01010101 01010101 01010101 thì giá trị ở ngõ ra là 01010101 01010101 01011111 11111111. Q4.0=1 khi lệnh thực hiện.

4.6. Lệnh WXOR-DW (Exclusive OR doubleword)

Ký hiệu:

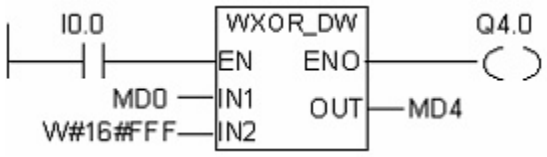
LAD/ FBD	STL
WXOR_DW EN IN1 OUT IN2 ENO	A <EN> L <IN1> L <IN2> XOD T <OUT>

Các ngõ vào/ra IN1, IN2, OUT: Biểu diễn toán hạng là dữ liệu ngõ ra dạng Word kép.

Hoạt động:

Lệnh WXOR-W hoạt động khi trạng thái tín hiệu tại ngõ vào EN là 1. Khi đó hai dữ liệu dạng Word kép IN1 và IN2 sẽ xor với nhau theo bảng sự thật của lệnh xor, kết quả gửi ra ngõ OUT. ENO có cùng trạng thái với ngõ EN.

Ví dụ:



Lệnh thực hiện khi I0.0=1. Nếu giá trị MW0= 01010101 01010101 01010101 01010101 thì giá trị ở ngõ ra là 01010101 01010101 01011010 10101010. Q4.0=1 khi lệnh thực hiện.

v. CÁC LỆNH SO SÁNH

Có 6 lệnh so sánh là <, >, = =, <>, <=, >=. Có thể dùng lệnh so sánh để so sánh các cặp giá trị dạng số nguyên (Interger), số nguyên kép (Double Interger), số thực (Real).
Nếu kết quả của sự so sánh là đúng thì kết quả của phép toán Logic RLO là 1. Ngược lại RLO=0.

Ký hiệu chung:

CPM xx I/DI/R: Tuỳ thuộc vào lệnh so sánh mà xx có ký hiệu là <, >, = =, <>, <=, >=. I, DI, R dùng để chỉ dạng so sánh số nguyên, số nguyên kép, số thực.

in: Ngõ vào cho phép lệnh so sánh hoạt động. Biểu diễn toán hạng là địa chỉ ngõ vào như sau: I, Q, M, L, D. Biểu diễn toán hạng là dữ liệu dạng số nhị phân.

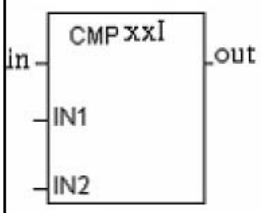
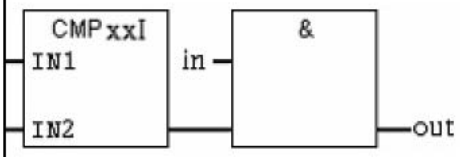
out: Ngõ ra, diễn tả kết quả của lệnh so sánh. Nếu kết quả so sánh là đúng thì kết quả RLO=1, ngược lại RLO=0. Biểu diễn toán hạng là địa chỉ ngõ ra như sau: I, Q, M,

L, D. Biểu diễn toán hạng là dữ liệu dạng số nhị phân.

IN1, IN2: Là giá trị so sánh thứ nhất và thứ hai. Biểu diễn các toán hạng là địa chỉ ngõ vào như sau: I, Q, M, L, D hoặc là hằng số. Biểu diễn toán hạng là dữ liệu dạng số Word, Word kép, số thực.

5.1. LỆNH SO SÁNH SỐ NGUYÊN (COMPARE INTEGER-<, >, = =, <>, <=, >=)

Ký hiệu:

LAD	FBD	STL
		A <in> A(L <IN1> L <IN2> xxI) = out

Lệnh so sánh số nguyên CPMxxI dùng để so sánh hai số nguyên với nhau. xx là các dạng so sánh <, >, = =, <>, <=, >=.

Hoạt động:

Lệnh CPMxxI có thể sử dụng như một tiếp điểm bình thường. Nó có thể đặt tại bất cứ vị trí nào mà tiếp điểm có thể đặt. IN1, IN2 được so sánh theo lệnh so sánh mà chúng ta đặt như so sánh lớn hơn (>), nhỏ hơn (<), bằng (= =), không bằng (<>), lớn hơn hoặc bằng (>=), nhỏ hơn hoặc bằng (<=). Nếu lệnh so sánh là đúng thì kết quả phép toán Logic RLO=1. Lệnh so sánh có thể mắt nối tiếp hoặc song song để tạo thành cổng AND hoặc cổng OR.

Lệnh này tác động lên thanh ghi trạng thái như sau:

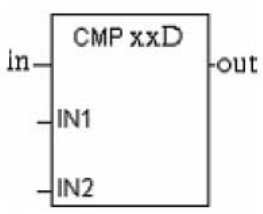
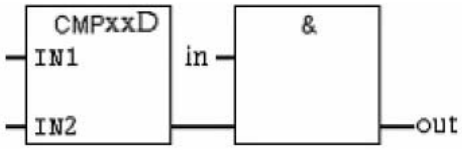
BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	x	x	0	-	0	x	x	1

Ví dụ:

Ngõ ra Q4.0 sẽ đặt lên 1 khi I0.0 và I0.1 bằng 1, giá trị MW0>=MW2.

5.2. LỆNH SO SÁNH SỐ NGUYÊN KÉP (COMPARE DOUBLE INTEGER-<, >, = =, <>, <=, >=)

Ký hiệu:

LAD	FBD	STL
		A <in> A(L <IN1> L <IN2> xxD) = out

Lệnh so sánh số nguyên CPMxxD dùng để so sánh hai số nguyên kép với nhau. xx là các dạng so sánh <, >, = =, <>, <=, >=.

Hoạt động:

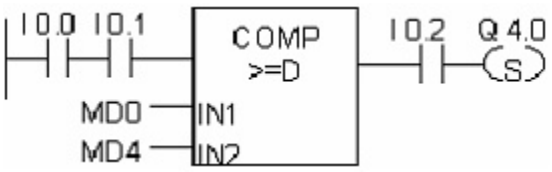
Lệnh CPMxxD có thể sử dụng như một tiếp điểm bình thường. Nó có thể đặt tại bất cứ vị trí nào mà tiếp điểm có thể đặt. IN1, IN2 được so sánh theo lệnh so sánh mà chúng ta đặt như so sánh lớn hơn (>), nhỏ hơn (<), bằng (= =), không bằng (<>), lớn hơn hoặc bằng (>=), nhỏ hơn hoặc bằng (<=). Nếu lệnh so sánh là đúng thì kết quả phép toán Logic RLO=1. Lệnh so sánh có thể mắt nối tiếp hoặc song song để tạo thành cổng

AND hoặc cổng OR.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	x	x	0	-	0	x	x	1

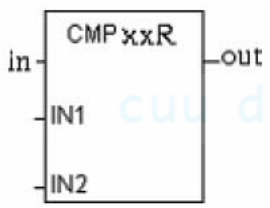
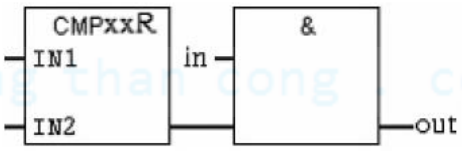
Ví dụ:



Ngõ ra Q4.0 sẽ đặt lên 1 khi I0.0 và I0.1 bằng 1, giá trị MD0>=MD4 và I0.2=1.

5.3. LỆNH SO SÁNH SỐ THỰC (COMPARE REAL-<, >, = =, <>, <=, >=)

Ký hiệu:

LAD	FBD	STL
		A <in> A(L <IN1> L <IN2> xxR) = out

Lệnh so sánh số nguyên CPMxxI dùng để so sánh hai số thực với nhau. xx là các dạng so sánh <, >, = =, <>, <=, >=.

Hoạt động:

Lệnh CPMxxR có thể sử dụng như một tiếp điểm bình thường. Nó có thể đặt tại bất cứ vị trí nào mà tiếp điểm có thể đặt. IN1, IN2 được so sánh theo lệnh so sánh mà chúng ta đặt như so sánh lớn hơn (>), nhỏ hơn (<), bằng (= =), không bằng (<>), lớn hơn hoặc bằng (>=), nhỏ hơn hoặc bằng (<=). Nếu lệnh so sánh là đúng thì kết quả phép toán Logic RLO=1. lệnh so sánh có thể mắt nối tiếp hoặc song song để tạo thành cổng AND hoặc cổng OR.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	x	x	x	x	0	x	x	1

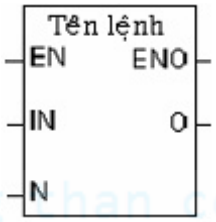
Ví dụ:

Ngõ ra Q4.0 sẽ đặt lên 1 khi I0.0 và I0.1 bằng 1, giá trị MD0>=MD4 và I0.2=1.

VI. CÁC LỆNH DỊCH/XOAY

Hầu hết các lệnh dịch và lệnh xoay đều có ký hiệu tương tự nhau. Các lệnh này tác động lện thanh ghi trạg thái như nhau.

Ký hiệu chung là (dạng LAD):



EN là ngõ vào cho phép lệnh dịch/xoay hoạt động (Enable Input). Biểu diễn toán hạng là dữ liệu dạng số nhị phân. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

ENO là ngõ cho phép ngõ ra hoạt động (Enable Output). Ngõ ra ENO có cùng trạng thái với ngõ vào EN. Loại toán hạng là dữ liệu dạng số nhị phân. Loại toán hạng là địa chỉ dạng: I, Q, L, M, D.

IN: là ngõ vào của dữ liệu cần dịch/xoay. Biểu diễn toán hạng là dữ liệu dạng số nguyên, số nguyên kép, Word, Word kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

N: Biểu diễn số bit cần dịch/xoay. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

O/OUT: Là kết quả của lệnh dịch/xoay. Biểu diễn toán hạng là dữ liệu dạng số nguyên, số nguyên kép, Word, Word kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

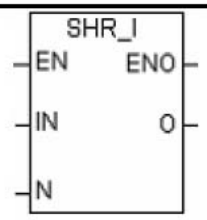
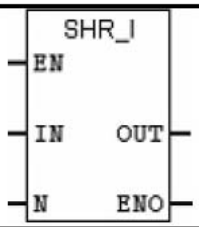
Khi thựchiện lệnh dịch/xoay thì giá trị cần dịch/xoay được đưa vào bộ tích luỹ ACCU1 và thực hiện lệnh dịch/xoay, sau đó giá trị được chuyển đến ngõ ra OUT.

Các lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	x	x	x	-	x	x	x	1

6.1. LỆNH DỊCH PHẢI SỐ NGUYÊN (SHR-I _ SHIFT RIGHT INTEGER)

Ký hiệu:

LAD	FBD	STL
		A <EN> L <IN> L <N> SSI T <OUT>

IN: là ngõ vào của dữ liệu cần dịch. Biểu diễn toán hạng là dữ liệu dạng số nguyên. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

N: Biểu diễn số bit cần dịch. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D. Nếu N>=16 thì lệnh vẫn được thực hiện như là N=16.

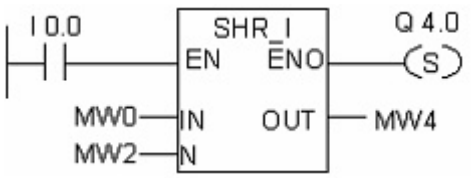
O/OUT: Là kết quả của lệnh dịch. Biểu diễn toán hạng là dữ liệu dạng số nguyên. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

Hoạt động:

Lệnh SHR-I hoạt động khi ngõ vào EN=1. Lệnh SHR-I thực hiện dịch 16 bit sang bên phải, 16 bit cao còn lại không sử dụng. Ngõ vào N đặt số bit cần dịch. Kết quả dịch được lưu trữ vào địa chỉ ngõ ra OUT. Khi lệnh thực hiện thì ngõ ra ENO cho biết trạng thái bit cuối cùng của số bit bị dịch. Các lệnh phụ thuộc vào ENO khác sẽ không thực

hiện nếu trạng thái của bit cuối cùng của số bit bị dịch là

0. Ví dụ:



Lệnh SHR-I hoạt động khi ngõ vào I0.0=1. MW0 bị dịch phải bởi số bit đặt trước tại ngõ vào N (MW2), kết quả được ghi ra MW4. Q4.0 được đặt lên 1.

6.2. LỆNH DỊCH PHẢI SỐ NGUYÊN KÉP (SHR-DI_ SHIFT RIGHT DOUBLE INTEGER)

Ký hiệu:

LAD	FBD	STL
SHR_DI EN ENO IN O N	SHR_DI EN IN OUT N ENO	A <EN> L <IN> L <N> SSD T <OUT>

IN: Là ngõ vào của dữ liệu cần dịch. Biểu diễn toán hạng là dữ liệu dạng số nguyên kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

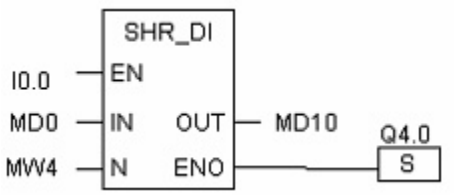
N: Biểu diễn số bit cần dịch. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

O/OUT: Là kết quả của lệnh dịch. Biểu diễn toán hạng là dữ liệu dạng số nguyên kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

Hoạt động:

Lệnh SHR-DI hoạt động khi ngõ vào EN=1. Lệnh SHR-DI thực hiện dịch 32 bit sang bên phải. Ngõ vào N đặt số bit cần dịch. Kết quả dịch được lưu trữ vào địa chỉ ngõ ra OUT. Khi lệnh thực hiện thì ngõ ra ENO cho biết trạng thái bit cuối cùng của số bit bị dịch. Các lệnh phụ thuộc vào ENO khác sẽ không thực hiện nếu trạng thái của bit cuối cùng của số bit bị dịch là 0.

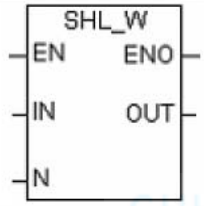
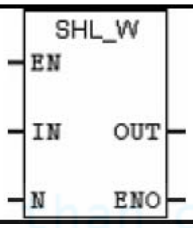
Ví dụ:



Lệnh SHR-DI hoạt động khi ngõ vào I0.0=1. MD0 bị dịch phải bởi số bit đặt trước tại ngõ vào N (MW4), kết quả được ghi ra MD10. Q4.0 được đặt lên 1.

6.3. LỆNH DỊCH TRÁI WORD (SHL-W _SHIFT LEFT WORD)

Ký hiệu:

LAD	FBD	STL
		A <EN> L <IN> L <N> SLW T <OUT>

IN: Là ngõ vào của dữ liệu cần dịch. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

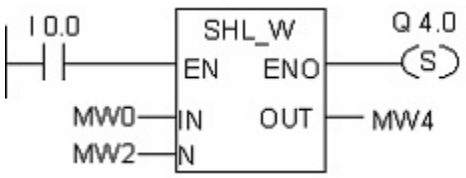
N: Biểu diễn số bit cần dịch. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D. Nếu N>=16 thì ngõ ra OUT=0.

O/OUT: Là kết quả của lệnh dịch. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

Hoạt động:

Lệnh SHL-W hoạt động khi ngõ vào EN=1. Lệnh SHL-W thực hiện dịch 16 bit thấp của bộ tích luỹ ACCU1 sang bên trái với số vị trí được đặt tại ngõ vào N. 16 bit cao còn lại không sử dụng, các bit ở bên phải được điền giá trị 0. Kết quả dịch được lưu trữ vào địa chỉ ngõ ra OUT. Khi lệnh thực hiện thì ngõ ra ENO cho biết trạng thái bit cuối cùng của số bit bị dịch. Các lệnh phụ thuộc vào ENO khác sẽ không thực hiện nếu trạng thái của bit cuối cùng của số bit bị dịch là 0.

Ví dụ:



Lệnh SHL-W hoạt động khi ngõ vào I0.0=1. MW0 bị dịch trái bởi số bit đặt trước tại ngõ vào N (MW2), kết quả được ghi ra MW4. Q4.0 được đặt lên 1.

6.4. LỆNH DỊCH TRÁI WORD KÉP (SHL-DW_SHIFT LEFT DOUBLE WORD)

Ký hiệu:

LAD	FBD	STL
SHL_DW EN ENO IN O N	SHL_DW EN IN OUT N ENO	A <EN> L <IN> L <N> SLD T <OUT>

IN: Là ngõ vào của dữ liệu cần dịch. Biểu diễn toán hạng là dữ liệu dạng Word kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

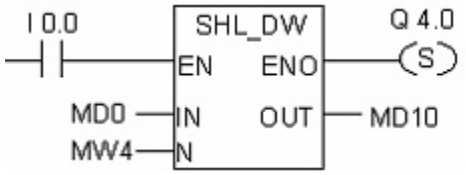
N: Biểu diễn số bit cần dịch. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D. Nếu N>=32 thì ngõ ra OUT=0.

O/OUT: Là kết quả của lệnh dịch. Biểu diễn toán hạng là dữ liệu dạng Word kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

Hoạt động:

Lệnh SHL-DW hoạt động khi ngõ vào EN=1. Lệnh SHL-DW thực hiện dịch 32 bit của bộ tích luỹ ACCU1 sang bên trái với số vị trí được đặt tại ngõ vào N. Kết quả dịch được lưu trữ vào địa chỉ ngõ ra OUT. Khi lệnh thực hiện thì ngõ ra ENO cho biết trạng thái bit cuối cùng của số bit bị dịch. Các lệnh phụ thuộc vào ENO khác sẽ không thực hiện nếu trạng thái của bit cuối cùng của số bit bị dịch là 0.

Ví dụ:



Lệnh SHL-DW hoạt động khi ngõ vào I0.0=1. MD0 bị dịch trái bởi số bit đặt trước tại ngõ vào N (MW4), kết quả được ghi ra MD10. Q4.0 được đặt lên 1.

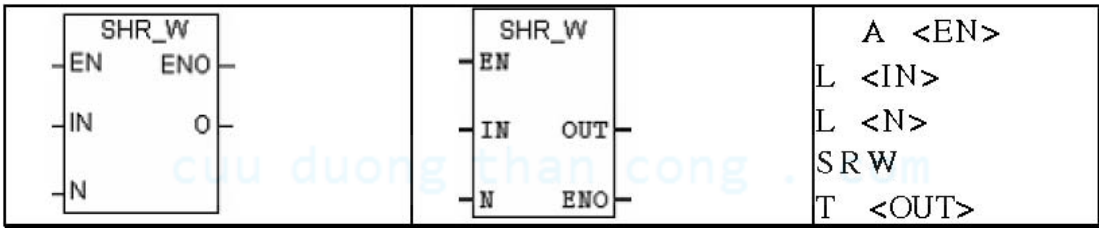
6.5. LỆNH DỊCH PHẢI WORD (SHR-W _ SHIFT RIGHT WORD)

Ký hiệu:

LAD

FBD

STL



IN: Là ngõ vào của dữ liệu cần dịch. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

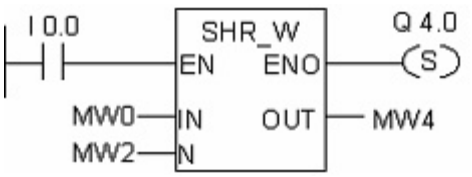
N: Biểu diễn số bit cần dịch. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D. Nếu N>=16 thì ngõ ra OUT=0.

O/OUT: Là kết quả của lệnh dịch. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

Hoạt động:

Lệnh SHR-W hoạt động khi ngõ vào EN=1. Lệnh SHR-W thực hiện dịch 16 bit thấp của bộ tích luỹ ACCU1 sang bên phải với số vị trí được đặt tại ngõ vào N. 16 bit cao còn lại không sử dụng, các bit ở bên trái được điền giá trị 0. Kết quả dịch được lưu trữ vào địa chỉ ngõ ra OUT. Khi lệnh thực hiện thì ngõ ra ENO cho biết trạng thái bit cuối cùng của số bit bị dịch. Các lệnh phụ thuộc vào ENO khác sẽ không thực hiện nếu trạng thái của bit cuối cùng của số bit bị dịch là 0.

Ví dụ:



Lệnh SHR-W hoạt động khi ngõ vào I0.0=1. MW0 bị dịch phải bởi số bit đặt trước tại ngõ vào N (MW2), kết quả được ghi ra MW4. Q4.0 được đặt lên 1.

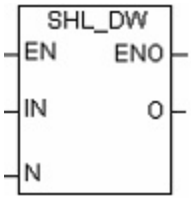
6.6. LỆNH DỊCH PHẢI WORD KÉP (SHR-DW_SHIFT RIGHT DOUBLE WORD)

Ký hiệu:

LAD

FBD

STL



A <EN>
L <IN>
L <N>
SRD
T <OUT>

IN: Là ngõ vào của dữ liệu cần dịch. Biểu diễn toán hạng là dữ liệu dạng Word kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

N: Biểu diễn số bit cần dịch. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D. Nếu N>=32 thì ngõ ra OUT=0.

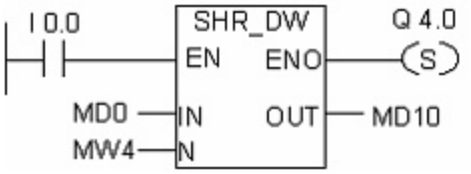
O/OUT: Là kết quả của lệnh dịch. Biểu diễn toán hạng là dữ liệu dạng Word kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

Hoạt động:

Lệnh SHR-DW hoạt động khi ngõ vào EN=1. Lệnh SHR-DW thực hiện dịch 32 bit của bộ tích luỹ ACCU1 sang bên phải với số vị trí được đặt tại ngõ vào N. Kết quả dịch được lưu trữ vào địa chỉ ngõ ra OUT. Khi lệnh thực hiện thì ngõ ra ENO cho biết

trạng thái bit cuối cùng của số bit bị dịch. Các lệnh phụ thuộc vào ENO khác sẽ không thực hiện nếu trạng thái của bit cuối cùng của số bit bị dịch là 0.

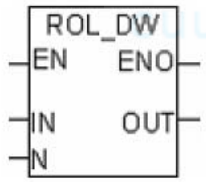
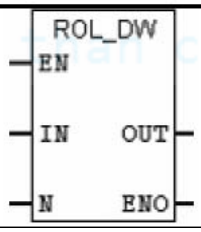
Ví dụ:



Lệnh SHR-DW hoạt động khi ngõ vào I0.0=1. MD0 bị dịch phải bởi số bit đặt trước tại ngõ vào N (MW4), kết quả được ghi ra MD10. Q4.0 được đặt lên 1.

6.7. LỆNH XOAY TRÁI WORD KÉP (ROL-DW_ROTATE LEFT DOUBLE WORD)

Ký hiệu:

LAD	FBD	STL
		A <EN> L <IN> L <N> RLD T <OUT>

IN: Là ngõ vào của dữ liệu cần xoay. Biểu diễn toán hạng là dữ liệu dạng Word kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

N: Biểu diễn số bit cần xoay. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

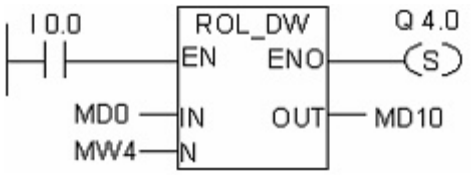
O/OUT: Là kết quả của lệnh xoay. Biểu diễn toán hạng là dữ liệu dạng Word kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

Hoạt động:

Lệnh ROL-DW hoạt động khi ngõ vào EN=1. Lệnh ROL-DW thực hiện xoay toàn bộ 32 bit của bộ tích luỹ ACCU1 sang bên trái với số bit được xoay được đặt tại ngõ vào
N. Kết quả xoay được lưu trữ vào địa chỉ ngõ ra OUT. Những bit trống được làm đầy bằng trạng thái của những bit bị đẩy ra. Bit được xoay cuối cùng được nạp vào bit 1 của Word trạng thái và cũng được lưu trữ tại ngõ ra ENO. Nghĩa là các lệnh phụ thuộc vào ENO khác sẽ không thực hiện nếu trạng thái của bit cuối cùng của số bit bị dịch

là 0.

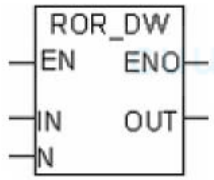
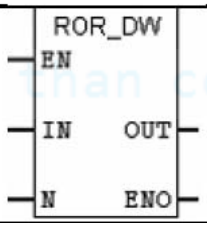
Ví dụ:



Lệnh ROL-DW hoạt động khi ngõ vào I0.0=1. MD0 bị xoay trái với số bit đặt trước tại ngõ vào N (MW4), kết quả được ghi ra MD10. Q4.0 được đặt lên 1.

6.8. LỆNH XOAY PHẢI WORD KÉP (ROR-DW_ROTATE RIGHT DOUBLE WORD)

Ký hiệu:

LAD	FBD	STL
		A <EN> L <IN> L <N> RRD T <OUT>

IN: Là ngõ vào của dữ liệu cần xoay. Biểu diễn toán hạng là dữ liệu dạng Word kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

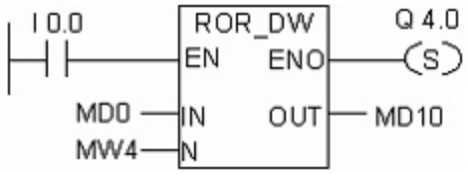
N: Biểu diễn số bit cần xoay. Biểu diễn toán hạng là dữ liệu dạng Word. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

O/OUT: Là kết quả của lệnh xoay. Biểu diễn toán hạng là dữ liệu dạng Word kép. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

Hoạt động:

Lệnh ROR-DW hoạt động khi ngõ vào EN=1. Lệnh ROR-DW thực hiện xoay toàn bộ 32 bit của bộ tích luỹ ACCU1 sang bên phải với số bit được xoay được đặt tại ngõ vào N. Kết quả xoay được lưu trữ vào địa chỉ ngõ ra OUT. Những bit trống được làm đầy bằng trạng thái của những bit bị đẩy ra. Bit được xoay cuối cùng được nạp vào bit 32 của Word trạng thái và cũng được lưu trữ tại ngõ ra ENO. Nghĩa là các lệnh phụ thuộc vào ENO khác sẽ không thực hiện nếu trạng thái của bit cuối cùng của số bit bị dịch là 0.

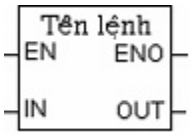
Ví dụ:



Lệnh ROR-DW hoạt động khi ngõ vào I0.0=1. MD0 bị xoay phải với số bit đặt trước tại ngõ vào N (MW4), kết quả được ghi ra MD10. Q4.0 được đặt lên 1.

VII. CÁC LỆNH CHUYỂN ĐỔI DỮ LIỆU

Có nhiều lệnh dùng để chuyển đổi dữ liệu khác nhau như: Số nguyên 16 bit, số nguyên 32 bit (số nguyên kép), số nguyên dạng BCD, số thực dấu phẩy động, số bù 1 của số nguyên, số bù 2 của số nguyên. Do việc làm với nhiều dạng dữ liệu khác nhau đặt ra cho chúng ta vấn đề phải chuyển đổi chúng. Ví dụ khi đọc tín hiệu tương tự từ cổng tương tự ta nhận được số liệu dạng số nguyên 16 bit mang giá trị tín hiệu tương tự chứ không phải bản thân giá trị đó, bởi vậy để xử lý tiếp thì cần phải chuyển số nguyên đó thành đúng giá trị thực dấu phẩy động của tín hiệu tương tự ở cổng. Tất cả các lệnh này đều có ký hiệu tương tự nhau như sau:



EN là ngõ vào cho phép lệnh chuyển đổi dữ liệu hoạt động (Enable Input). Biểu diễn toán hạng là dữ liệu dạng số nhị phân. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

ENO là ngõ cho phép ngõ ra hoạt động (Enable Output). Ngõ ra ENO có cùng trạng thái với ngõ vào EN. Loại toán hạng là dữ liệu dạng số nhị phân. Loại toán hạng là địa chỉ dạng: I, Q, L, M, D.

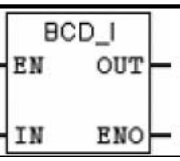
IN: Giá trị tại ngõ vào cần chuyển đổi, có toán hạng là dữ liệu tùy vào dạng chuyển đổi như Word (16 bit), Word kép (32 bit), số thực, toán hạng là địa chỉ: I, Q, L, M, D. Khi EN=1 thì giá trị cần chuyển đổi (IN) được đọc vào địa chỉ ngõ ra OUT.

OUT: Ngõ ra là kết quả của lệnh chuyển đổi. Giá trị tại ngõ vào cần chuyển đổi được đưa đến ngõ ra OUT, có toán hạng là dữ liệu tùy vào dạng chuyển đổi như Word (16 bit), Word kép (32 bit), số thực, toán hạng là địa chỉ: I, Q, L, M, D.

7.1. LỆNH CHUYỂN ĐỔI SỐ BCD THÀNH SỐ NGUYÊN VÀ NGƯỢC LẠI

7.1.1. LỆNH CHUYỂN ĐỔI SỐ BCD THÀNH SỐ NGUYÊN 16 BIT (BCD-I_ BCD TO INTERGER)

Ký hiệu:

LAD	FBD	STL
		L <IN> BTI T <OUT>

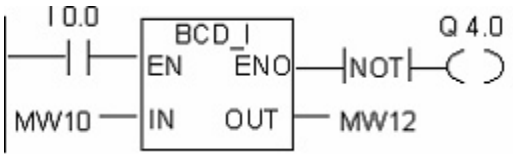
Hoạt động:

Khi ngõ vào En=1 thì số BCD 3 số (+/-999) ở ngõ vào IN được chuyển thành số nguyên 16 bit, giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
1	-	-	-	-	0	1	1	1

Ví dụ:



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MW10) được đọc như một số BCD 3 số và chuyển thành số nguyên 16 bit. Kết quả được lưu vào ngõ ra MW12. ngõ ra Q4.0=1 nếu lệnh chuyển đổi không thực hiện, khi đó EN=ENO=0.

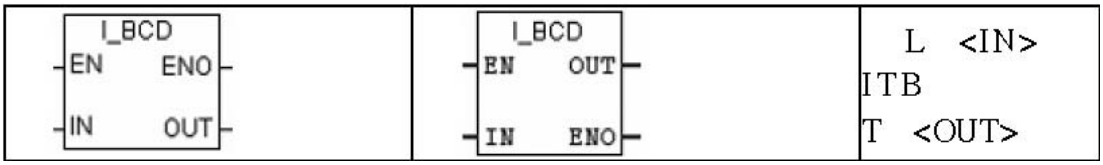
7.1.2. LỆNH CHUYỂN ĐỔI SỐ NGUYÊN 16 BIT THÀNH SỐ BCD (I-BCD_INTERGER TO BCD)

Ký hiệu:

LAD

FBD

STL



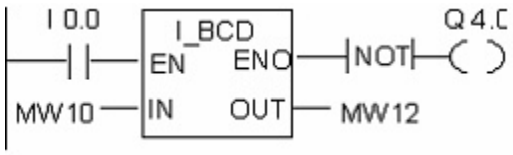
Hoạt động:

Khi ngõ vào En=1 thì số nguyên 16 bit ở ngõ vào IN được chuyển thành số BCD 3 số (+/-999), giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN. Khi có tràn thì ngõ ENO=0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	-	-	x	x	0	x	x	1

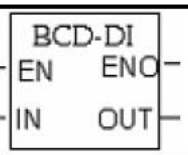
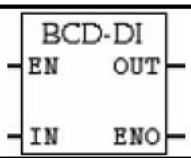
Ví dụ:



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MW10) được đọc như một số nguyên 16 bit và chuyển thành số BCD 3 số. Kết quả được lưu vào ngõ ra MW12. ngõ ra Q4.0=1 nếu lệnh chuyển đổi không thực hiện hoặc bị tràn, khi đó EN=ENO=0.

7.1.3. LỆNH CHUYỂN ĐỔI SỐ BCD THÀNH SỐ NGUYÊN 32 BIT (BCD-DI_BCD TO DOUBLEINTERGER)

Ký hiệu:

LAD	FBD	STL
		L <IN> BTD T <OUT>

Hoạt động:

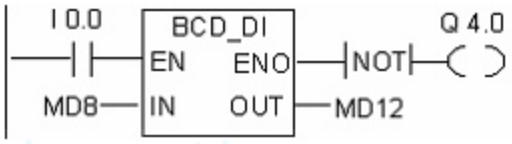
Khi ngõ vào En=1 thì số BCD 7 số (+/-9999999) ở ngõ vào IN được chuyển thành số nguyên kép 32 bit, giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu

giống với ngõ vào EN.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
1	-	-	-	-	0	1	1	1

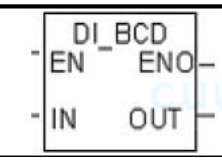
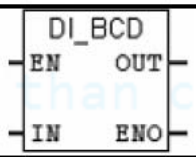
Ví dụ:



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MD8) được đọc như một số BCD 7 số và chuyển thành số nguyên kép 32 bit. Kết quả được lưu vào ngõ ra MD12. Ngõ ra Q4.0=1 nếu lệnh chuyển đổi không thực hiện, khi đó EN=ENO=0.

7.1.4. LỆNH CHUYỂN ĐỔI SỐ NGUYÊN KÉP 32 BIT THÀNH SỐ BCD (DI-BCD_DOUBLE INTERGER TO BCD)

Ký hiệu:

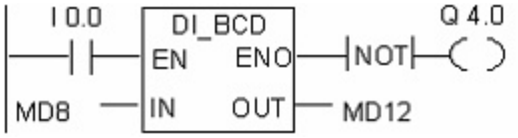
LAD	FBD	STL
		L <IN> DTB T <OUT>

Hoạt động:

Khi ngõ vào En=1 thì số nguyên kép 32 bit ở ngõ vào IN được chuyển thành số BCD 7 số (+/-9999999), giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN. Khi có tràn thì ngõ ENO=0.

Lệnh này tác động lên thanh ghi trạng thái như sau: Ví dụ:

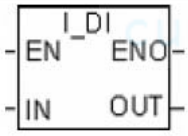
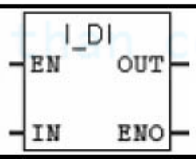
BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	-	-	x	x	0	x	x	1



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MD8) được đọc như một số nguyên kép 32 bit và chuyển thành số BCD 7 số. Kết quả được lưu vào ngõ ra MD12. Ngõ ra Q4.0=1 nếu lệnh chuyển đổi không thực hiện hoặc bị tràn, khi đó EN=ENO=0.

7.2. LỆNH CHUYỂN ĐỔI SỐ NGUYÊN 16 BIT THÀNH SỐ NGUYÊN KÉP 32 (I-DI_ INTERGER TO DOUBLE INTERGER)

Ký hiệu:

LAD	FBD	STL
		L <IN> ITD T <OUT>

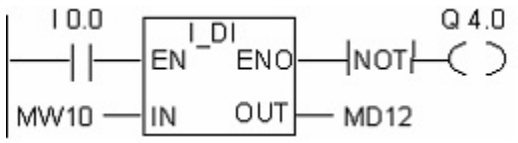
Hoạt động:

Khi ngõ vào En=1 thì số nguyên 16 bit ở ngõ vào IN được chuyển thành số nguyên kép 32 bit, giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
1	-	-	-	-	0	1	1	1

Ví dụ:

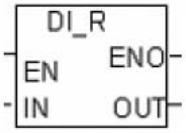
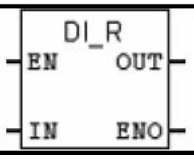


Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MW10) được đọc như một số

nguyên 16 bit và chuyển thành số nguyên kép 32 bit. Kết quả được lưu vào ngõ ra MD12. Ngõ ra Q4.0=1 nếu lệnh chuyển đổi không thực hiện, khi đó EN=ENO=0.

7.3. LỆNH CHUYỂN ĐỔI SỐ NGUYÊN KÉP 32 BIT THÀNH SỐ THỰC (DI-R_DOUBLE INTERGER TO REAL)

Ký hiệu:

LAD	FBD	STL
		L <IN> DTR T <OUT>

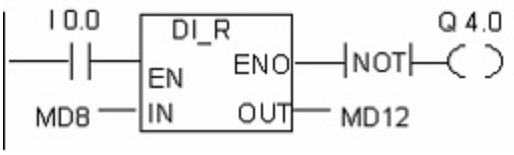
Hoạt động:

Khi ngõ vào En=1 thì số nguyên kép 32 bit ở ngõ vào IN được chuyển thành số thực 32 bit dấu phẩy động, giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
1	-	-	-	-	0	1	1	1

Ví dụ:

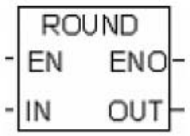
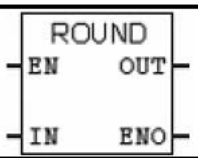


Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MD8) được đọc như một số nguyên kép 32 bit và chuyển thành số thực 32 bit. Kết quả được lưu vào ngõ ra MD12. Ngõ ra Q4.0=1 nếu lệnh chuyển đổi không thực hiện, khi đó EN=ENO=0.

7.4. LỆNH CHUYỂN ĐỔI SỐ THỰC THÀNH SỐ NGUYÊN KÉP 32 BIT

7.4.1. LỆNH CHUYỂN ĐỔI SỐ THỰC THÀNH SỐ NGUYÊN GẦN NHẤT (ROUND_ROUND TO DOUBLE INTERGER)

Ký hiệu:

LAD	FBD	STL
		L <IN> RND T <OUT>

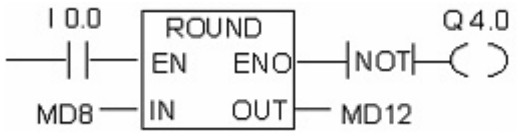
Hoạt động:

Khi ngõ vào En=1 thì số thực 32 bit dấu phẩy động ở ngõ vào IN được chuyển thành số nguyên kép 32 bit có giá trị gần nhất so với số thực đã cho, nếu số thực nằm giữa hai số nguyên thì CPU sẽ lấy số chẵn, giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN. Nếu có tràn sảy ra thì ENO=0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	-	-	x	x	0	x	x	1

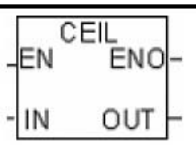
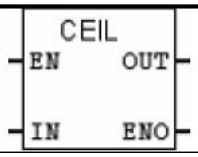
Ví dụ:



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MD8) được đọc như một số thực 32 bit dấu phẩy động và chuyển thành số nguyên kép 32 bit có giá trị gần nhất so với số thực đã cho. Kết quả được lưu vào ngõ ra MD12. Ngõ ra Q4.0=1 nếu lệnh chuyển đổi không thực hiện hoặc có tràn sảy ra, khi đó EN=ENO=0.

7.4.2. LỆNH CHUYỂN ĐỔI SỐ THỰC THÀNH SỐ NGUYÊN NHỎ NHẤT NHƯNG KHÔNG NHỎ HƠN SỐ THỰC (CEIL_CEILING)

Ký hiệu:

LAD	FBD	STL
		L <IN> RND+ T <OUT>

Hoạt động:

Khi ngõ vào En=1 thì số thực 32 bit dấu phẩy động ở ngõ vào IN được chuyển thành số nguyên kép 32 bit có giá trị nhỏ nhất nhưng không nhỏ hơn số thực đã cho, giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN. Nếu có tràn sảy ra thì ENO=0.

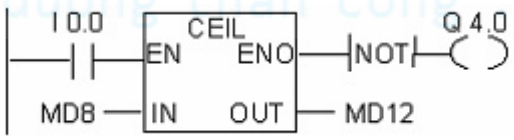
Lệnh này tác động lên thanh ghi trạng thái như sau (khi EN=1):

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	-	-	x	x	0	x	x	1

Lệnh này tác động lên thanh ghi trạng thái như sau (khi EN=0):

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
0	-	-	-	-	0	0	0	1

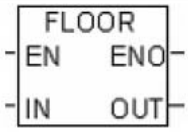
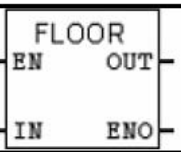
Ví dụ:



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MD8) được đọc như một số thực 32 bit dấu phẩy động và chuyển thành số nguyên kép 32 bit có giá trị nhỏ nhất nhưng không nhỏ hơn số thực đã cho. Kết quả được lưu vào ngõ ra MD12. Ngõ ra Q4.0=1 nếu lệnh chuyển đổi không thực hiện hoặc có tràn sảy ra, khi đó EN=ENO=0.

7.4.3. LỆNH CHUYỂN ĐỔI SỐ THỰC THÀNH SỐ NGUYÊN LỚN NHẤT NHƯNG KHÔNG LỚN HƠN SỐ THỰC (FLOOR_FLOOR)

Ký hiệu:

LAD	FBD	STL
		L <IN> RND- T <OUT>

Hoạt động:

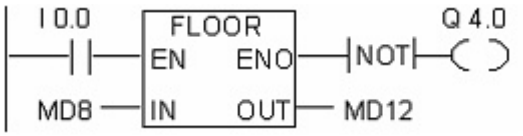
Khi ngõ vào En=1 thì số thực 32 bit dấu phẩy động ở ngõ vào IN được chuyển

thành số nguyên kép 32 bit có giá trị lớn nhất nhưng không lớn hơn số thực đã cho, giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN. Nếu có tràn sảy ra thì ENO=0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	-	-	x	x	0	x	x	1

Ví dụ:



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MD8) được đọc như một số thực 32 bit dấu phẩy động và chuyển thành số nguyên kép 32 bit có giá trị lớn nhất nhưng không lớn hơn số thực đã cho. Kết quả được lưu vào ngõ ra MD12. Ngõ ra Q4.0=1 nếu lệnh chuyển đổi không thực hiện hoặc có tràn sảy ra, khi đó EN=ENO=0.

7.4.4. LỆNH LẤY PHẦN NGUYÊN (TRUNC_TRUNCATE DOUBLE INTEGER PART)

Ký hiệu:

LAD

FBD

STL

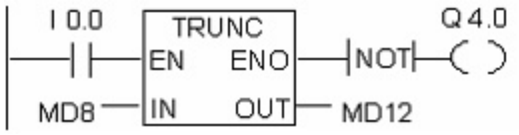
Hoạt động:

Khi ngõ vào En=1 thì số thực 32 bit dấu phẩy động ở ngõ vào IN được chuyển thành số nguyên kép 32 bit có giá trị là phần nguyên của số thực 32 bit dấu phẩy động, giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN. Nếu có tràn sảy ra thì ENO=0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	-	-	x	x	0	x	x	1

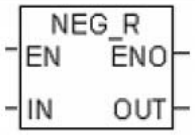
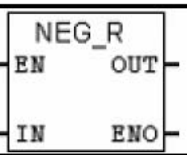
Ví dụ:



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MD8) được đọc như một số thực 32 bit dấu phẩy động và chuyển thành số nguyên kép 32 bit là phần nguyên của số thực 32 bit dấu phẩy động. Kết quả được lưu vào ngõ ra MD12. Ngõ ra Q4.0=1 nếu lệnh chuyển đổi không thực hiện hoặc có tràn sảy ra, khi đó EN=ENO=0.

7.5. LỆNH PHỦ ĐỊNH SỐ THỰC (NEG-R_NEGATE FLOATING-POINT NUMBER)

Ký hiệu:

LAD	FBD	STL
		L <IN> NEGR T <OUT>

Hoạt động:

Khi ngõ vào En=1 thì số thực 32 bit dấu phẩy động ở ngõ vào IN được nhân với (-1)

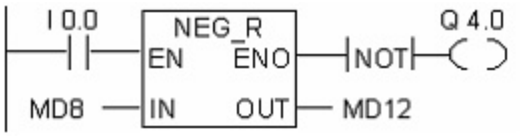
tạo thành số thực có dấu ngược với số thực ở ngõ, giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC

x	-	-	-	-	0	x	x	1

Ví dụ:

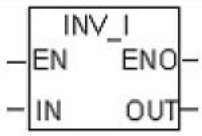
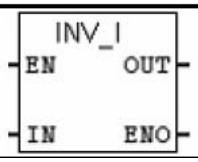


Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MD8) được đọc như một số thực 32 bit dấu phẩy động và được đảo lại thành số thực có dấu ngược với dấu của số thực ở ngõ vào. Kết quả được lưu vào ngõ ra MD12. Ngõ ra Q4.0=1 nếu lệnh đảo không thực hiện, khi đó EN=ENO=0. Chẳng hạn: MD8=+6.234*10-3 thì kết quả MD12=-6.234*10-3.

7.6. CÁC LỆNH BÙ SỐ NGUYÊN

7.6.1. LỆNH BÙ 1 SỐ NGUYÊN 16 BIT (INV-I_ONES COMPLEMENT INTEGER)

Ký hiệu:

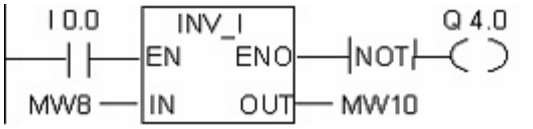
LAD	FBD	STL
		L <IN> INVI T <OUT>

Hoạt động:

Khi ngõ vào En=1 thì số nguyên 16 bit ở ngõ vào IN được thực hiện lệnh giống như lệnh XOR với số thập lục phân phân là FFFFh. Lệnh này làm thay đổi trạng thái bit đối nghịch nhau và giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN.

Lệnh này tác động lên thanh ghi trạng thái như sau: Ví dụ:

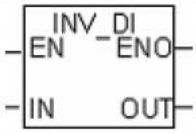
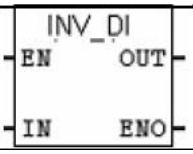
BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
1	-	-	-	-	0	1	1	1



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MW8) được đảo lại theo từng bit. Kết quả được lưu vào ngõ ra MW10. Ví dụ như: MW8=01000010 01110010 thì kết quả MW10=10111101 10001101. Ngõ ra Q4.0=1 nếu lệnh này không thực hiện hoặc có tràn sảy ra, khi đó EN=ENO=0.

7.6.2. LỆNH BÙ 1 SỐ NGUYÊN KÉP 32 BIT (INV-DI_ONES COMPLEMENT DOUBLE INTEGER)

Ký hiệu:

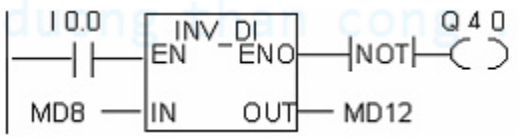
LAD	FBD	STL
		L <IN> INVD T <OUT>

Hoạt động:

Khi ngõ vào En=1 thì số nguyên kép 32 bit ở ngõ vào IN được thực hiện lệnh giống như lệnh XOR với số thập lục phân là FFFF FFFFh. Lệnh này làm thay đổi trạng thái bit đối nghịch nhau và giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN.

Lệnh này tác động lên thanh ghi trạng thái như sau: Ví dụ:

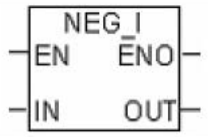
BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
1	-	-	-	-	0	1	1	1



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MD8) được đảo lại theo từng bit. Kết quả được lưu vào ngõ ra MD12. Ví dụ như: MD8=FF0F FFF0 thì kết quả MD12=00F0 000F. Ngõ ra Q4.0=1 nếu lệnh này không thực hiện hoặc có tràn sảy ra, khi đó EN=ENO=0.

7.6.3. LỆNH BÙ 2 SỐ NGUYÊN 16 BIT (NEG-I_TWOS COMPLEMENT INTEGER)

Ký hiệu:

LAD	FBD	STL
		L <IN> NEGI T <OUT>

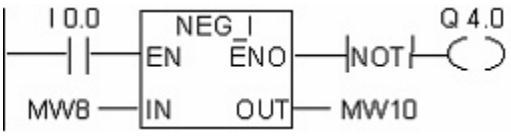
Hoạt động:

Khi ngõ vào En=1 thì số nguyên 16 bit ở ngõ vào IN được thực hiện lệnh bù 2. Lệnh bù 2 thực hiện giống như lệnh nhân với (-1) (ví dụ từ giá trị dương thành giá trị âm). Giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN ngoại trừ nếu EN=1 mà có tràn sảy ra thì ENO=0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	x	x	x	x	0	x	x	1

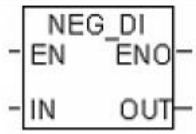
Ví dụ:



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MW8) được đảo ngược lại với giá trị tại ngõ vào. Kết quả được lưu vào ngõ ra MW10. Ví dụ như: MW8=+10 thì kết quả MW10=-10. Ngõ ra Q4.0=1 nếu lệnh này không thực hiện hoặc có tràn sảy ra, khi đó EN=ENO=0. Nếu EN=1 và có tràn sảy ra thì ngõ ra ENO=0.

7.6.4. LỆNH BÙ 2 SỐ NGUYÊN KÉP 32 BIT (NEG-DI_TWOS COMPLEMENT DOUBLE INTEGER)

Ký hiệu:

LAD	FBD	STL
		L <IN> NEGD T <OUT>

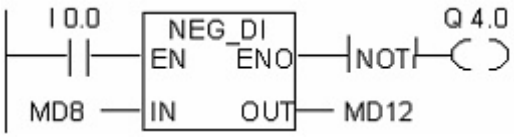
Hoạt động:

Khi ngõ vào En=1 thì số nguyên kép 32 bit ở ngõ vào IN được thực hiện lệnh bù 2. Lệnh bù 2 thực hiện giống như lệnh nhân với (-1) (ví dụ từ giá trị dương thành giá trị âm). Giá trị này xuất ra ngõ ra OUT. Ngõ ENO luôn có trạng thái tín hiệu giống với ngõ vào EN ngoại trừ nếu EN=1 mà có tràn sảy ra thì ENO=0.

Lệnh này tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	x	x	x	x	0	x	x	1

Ví dụ:



Nếu ngõ vào I0.0=1 thì giá trị tại ngõ vào IN (MD8) được đảo ngược lại với giá trị tại ngõ vào. Kết quả được lưu vào ngõ ra MD12. Ví dụ như: MD8=+1000 thì kết quả MD12=-1000. Ngõ ra Q4.0=1 nếu lệnh này không thực hiện hoặc có tràn sảy ra, khi đó EN=ENO=0. Nếu EN=1 và có tràn sảy ra thì ngõ ra ENO=0.

VIII. CÁC LỆNH TOÁN HỌC

Tất cả các lệnh toán học đều thực hiện với nội dung hai thanh ghi tích lũy ACCU1, ACCU2 và cùng tác động lên thanh ghi trạng thái như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
x	x	x	x	x	0	x	x	1

Chỉ có lệnh ABS (Lấy giá trị tuyệt đối của số thực) là tác động vào thanh ghi trạng thái khác với các lệnh còn lại như sau:

BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
1	-	-	-	-	0	1	1	1

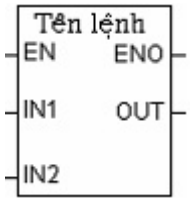
Trong đó hai bit trạng thái CC1 và CC0 được thay đổi theo qui tắc sau:

CC1	CC0	Ý nghĩa
0	0	Kết quả bằng 0 (=0)
0	1	Kết quả nhỏ hơn 0 (<0)
1	0	Kết quả lớn hơn 0 (>0)

Các lệnh toán học được chia làm hai nhóm chức năng là nhóm chức năng toán học cơ bản như các lệnh cộng, trừ, nhân, chia, lấy phần dư của phép chia và nhóm chức năng toán học cao cấp như các lệnh lấy giá trị tuyệt đối, căn bậc hai, bình phương, lấy Logarithm, thiết lập số mũ của e(=2,71828...), sin, cos, tg, arsin, arcos, artg. Nhóm chức năng toán học cơ bản làm việc với số nguyên 16 bit, số nguyên 32 bit, số thực 32 bit. Các lệnh thuộc nhóm chức năng toán học cao cấp đều làm việc với số thực.

8.1. NHÓM CHỨC NĂNG TOÁN HỌC CƠ BẢN

Các lệnh trong nhóm chức năng toán học cơ bản có ký hiệu tương tự nhau và tác động lên thanh ghi trạng thái như nhau, có ký hiệu như sau:



EN là ngõ vào cho phép lệnh toán học hoạt động (Enable Input). Biểu diễn toán hạng là dữ liệu dạng số nhị phân. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

ENO là ngõ cho phép ngõ ra hoạt động (Enable Output). Ngõ ra ENO có cùng trạng thái với ngõ vào EN. Loại toán hạng là dữ liệu dạng số nhị phân. Loại toán hạng là địa chỉ dạng: I, Q, L, M, D.

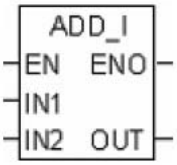
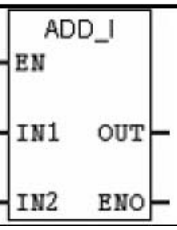
IN1, IN2: Có chức năng phụ thuộc vào từng loại lệnh. Giá trị tại ngõ vào để thực hiện lệnh toán học, có toán hạng là dữ liệu tùy vào dạng chuyển đổi như số nguyên

16 bit, số nguyên kép 32 bit, số thực, toán hạng là địa chỉ: I, Q, L, M, D hoặc là hằng số nếu là số nguyên. Khi EN=1 thì lệnh toán học thực hiện và giá trị được đọc vào địa chỉ ngõ ra OUT.

OUT: Ngõ ra là kết quả của lệnh toán học. Có toán hạng là dữ liệu tùy vào dạng chuyển đổi như số nguyên 16 bit, số nguyên kép 32 bit, số thực, toán hạng là địa chỉ: I, Q, L, M, D.

8.1.1. LỆNH CỘNG SỐ NGUYÊN 16 BIT (ADD-I _ADD INTERGER)

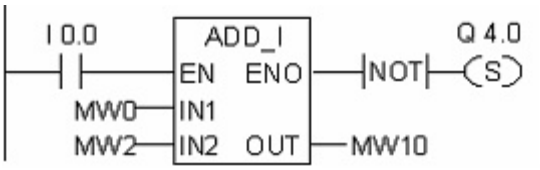
Ký hiệu:

LAD	FBD	STL
		L <IN1> L <IN2> +I T <OUT>

Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh ADD-I hoạt động, giá trị IN1 và IN2 được cộng lại và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số nguyên 16 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được

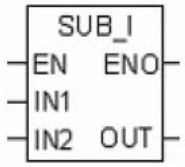
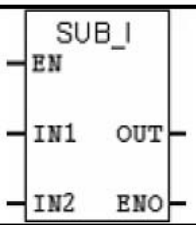
nối với ngõ ENO sẽ không hoạt động. Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh ADD-I hoạt động, kết quả của lệnh cộng MW0+MW2 được xuất ra ngõ MW10. Nếu kết quả nằm ngoài vùng hoạt động của số nguyên 16 bit thì Q4.0=1.

8.1.2. LỆNH TRỪ SỐ NGUYÊN 16 BIT (SUB-I _SUBTRACT INTERGER)

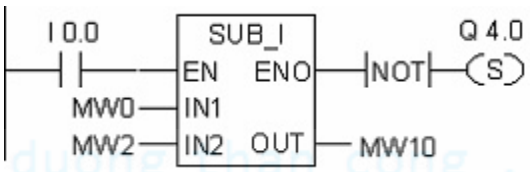
Ký hiệu:

LAD	FBD	STL
		L <IN1> L <IN2> -I T <OUT>

Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh SUB-I hoạt động, giá trị IN1 trừ giá trị IN2 và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số nguyên 16 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

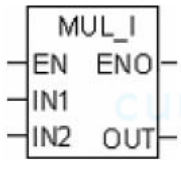
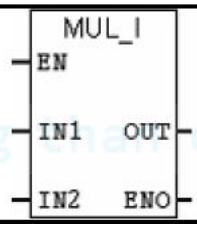
Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh SUB-I hoạt động, kết quả của lệnh trừ MW0-MW2 được xuất ra ngõ MW10. Nếu kết quả nằm ngoài vùng hoạt động của số nguyên 16 bit hoặc I0.0=0 thì Q4.0=1.

8.1.3. LỆNH NHÂN SỐ NGUYÊN 16 BIT (MUL-I _MULTIPLY INTERGER)

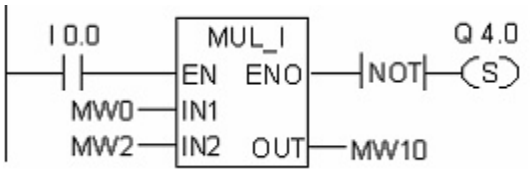
Ký hiệu:

LAD	FBD	STL
		L <IN1> L <IN2> *I T <OUT>

Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh MUL-I hoạt động, giá trị IN1 và IN2 được nhân lại và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số nguyên 16 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

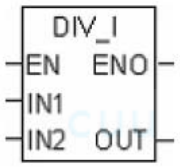
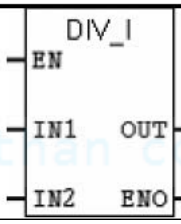
Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh MUL-I hoạt động, kết quả của lệnh nhân MW0*MW2 được xuất ra ngõ MW10. Nếu kết quả nằm ngoài vùng hoạt động của số nguyên 16 bit thì Q4.0=1.

8.1.4. LỆNH CHIA SỐ NGUYÊN 16 BIT (DIV-I_DIVIDE INTERGER)

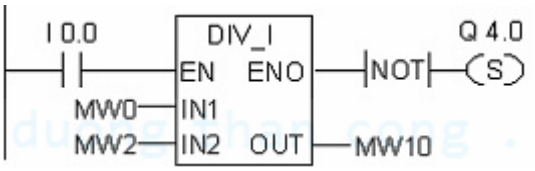
Ký hiệu:

LAD	FBD	STL
		L <IN1> L <IN2> /I T <OUT>

Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh DIV-I hoạt động, giá trị IN1 được chia bởi giá trị IN2 và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số nguyên 16 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

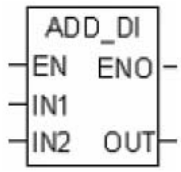
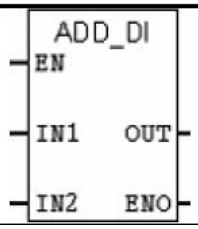
Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh DIV-I hoạt động, kết quả của lệnh chiaMW0 bởi MW2 được xuất ra ngõ MW10. Nếu kết quả nằm ngoài vùng hoạt động của số nguyên 16 bit thì Q4.0=1.

8.1.5. LỆNH CỘNG SỐ NGUYÊN KÉP 32 BIT (ADD-DI_ADD DOUBLE INTERGER)

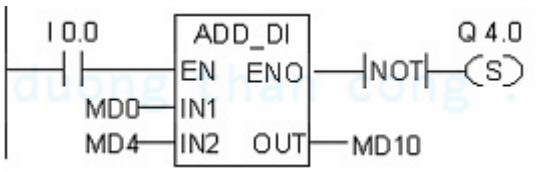
Ký hiệu:

LAD	FBD	STL
		L <IN1> L <IN2> +D T <OUT>

Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh ADD-DI hoạt động, giá trị IN1 và IN2 được cộng lại và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số nguyên kép 32 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh ADD-DI hoạt động, kết quả của lệnh cộng MD0+MD4 được xuất ra ngõ MD10. Nếu kết quả nằm ngoài vùng hoạt động của số nguyên kép 32 bit thì Q4.0=1.

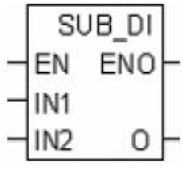
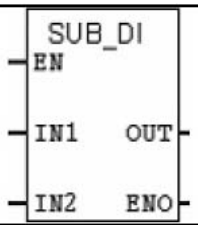
8.1.6. LỆNH TRỪ SỐ NGUYÊN KÉP 32 BIT (SUB-DI_SUBTRACT DOUBLE INTERGER)

Ký hiệu:

LAD

FBD

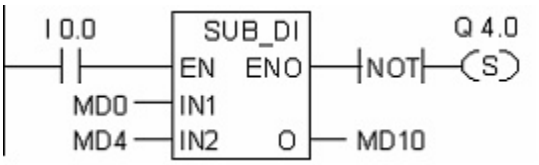
STL

		L <IN1> L <IN2> -D T <OUT>

Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh SUB-DI hoạt động, giá trị IN1 trừ giá trị IN2 và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số nguyên kép 32 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh SUB-DI hoạt động, kết quả của lệnh trừ MD0-MD4 được xuất ra ngõ MD10. Nếu kết quả nằm ngoài vùng hoạt động của số nguyên kép 32 bit hoặc I0.0=0 thì Q4.0=1.

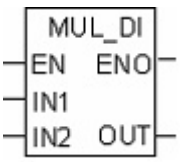
8.1.7. LỆNH NHÂN SỐ NGUYÊN KÉP 32 BIT (MUL-DI_MULTIPLY DOUBLE INTERGER)

Ký hiệu:

LAD

FBD

STL

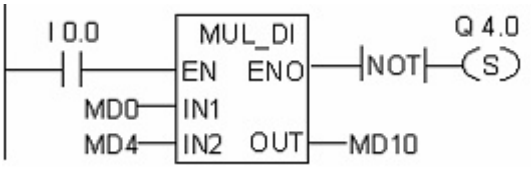



```
L <IN1>
L <IN2>
*D
T <OUT>
```


Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh MUL-DI hoạt động, giá trị IN1 và IN2 được nhân lại và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số nguyên kép 32 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh MUL-DI hoạt động, kết quả của lệnh nhân MD0*MD4 được xuất ra ngõ MD10. Nếu kết quả nằm ngoài vùng hoạt động của số nguyên kép 32 bit thì Q4.0=1.

8.1.8. LỆNH CHIA SỐ NGUYÊN KÉP 32 BIT (DIV-DI_DIVIDE DOUBLE INTERGER)

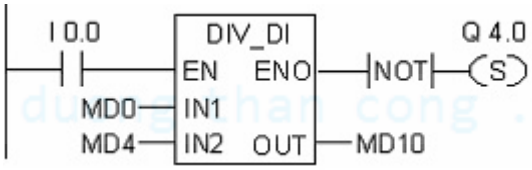
Ký hiệu:

LAD	FBD	STL
DIV_DI EN ENO IN1 IN2 OUT	DIV_DI EN IN1 OUT IN2 ENO	L <IN1> L <IN2> /D T <OUT>

Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh DIV-DI hoạt động, giá trị IN1 được chia bởi giá trị IN2 và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số nguyên kép 32 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh DIV-DI hoạt động, kết quả của lệnh chia MD0 bởi MD4 được xuất ra ngõ MD10. Nếu kết quả nằm ngoài vùng hoạt động của số nguyên kép 32 bit thì Q4.0=1.

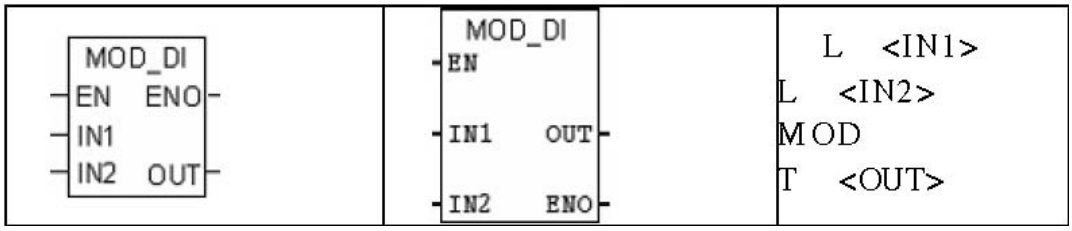
8.1.9. LỆNH LẤY PHẦN DƯ CỦA PHÉP CHIA SỐ NGUYÊN KÉP 32 BIT (MOD-DI_RETURN FRACTION DOUBLE INTEGER)

Ký hiệu:

LAD

FBD

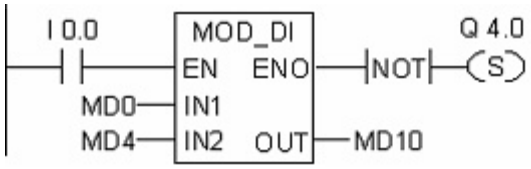
STL



Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh MOD-DI hoạt động, giá trị IN1 được chia bởi giá trị IN2 và kết quả phần dư được gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số nguyên kép 32 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh MOD-DI hoạt động, kết quả phần dư của lệnh chia MD0:MD4 được xuất ra ngõ MD10. Nếu kết quả nằm ngoài vùng hoạt động của số nguyên kép 32 bit thì Q4.0=1.

8.1.10. LỆNH CỘNG SỐ THỰC 32 BIT (ADD-R_ADD REAL)

Ký hiệu:

LAD

FBD

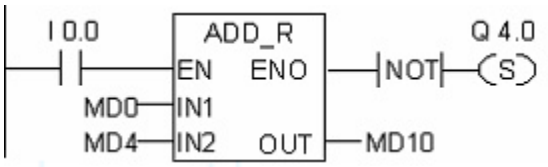
STL

L <IN1>
L <IN2>
+R
T <OUT>

Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh ADD-DI hoạt động, giá trị IN1 và IN2 được cộng lại và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số thực 32 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

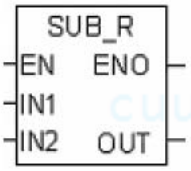
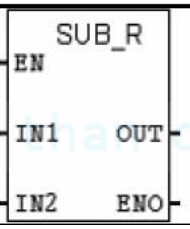
Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh ADD-R hoạt động, kết quả của lệnh cộng MD0+MD4 được xuất ra ngõ MD10. Nếu kết quả nằm ngoài vùng hoạt động của số thực 32 bit thì Q4.0=1.

8.1.11. LỆNH TRỪ SỐ THỰC 32 BIT (SUB-R_SUBTRACT REAL)

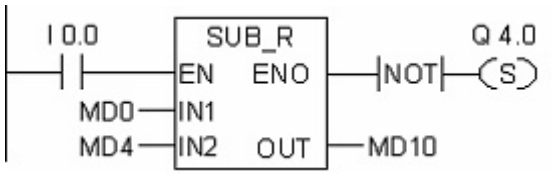
Ký hiệu:

LAD	FBD	STL
		L <IN1> L <IN2> -R T <OUT>

Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh SUB-R hoạt động, giá trị IN1 trừ giá trị IN2 và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số thực 32 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh SUB-R hoạt động, kết quả của lệnh trừ MD0-MD4 được xuất ra ngõ MD10. Nếu kết quả nằm ngoài vùng hoạt động của số thực 32 bit hoặc I0.0=0 thì Q4.0=1.

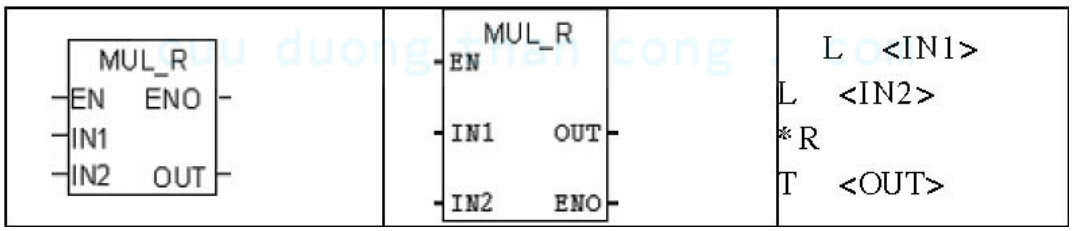
8.1.12. LỆNH NHÂN SỐ THỰC 32 BIT (MUL-R_MULTIPLY REAL)

Ký hiệu:

LAD

FBD

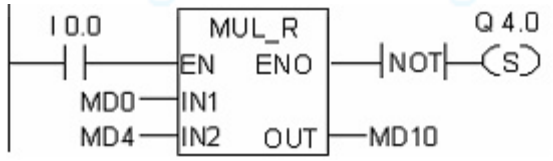
STL



Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh MUL-R hoạt động, giá trị IN1 và IN2 được nhân lại và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số thực 32 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

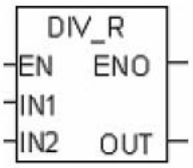
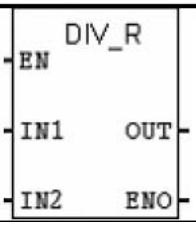
Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh MUL-R hoạt động, kết quả của lệnh nhân MD0*MD4 được xuất ra ngõ MD10. Nếu kết quả nằm ngoài vùng hoạt động của số thực 32 bit thì Q4.0=1.

8.1.13. LỆNH CHIA SỐ THỰC 32 BIT (DIV-R_DIVIDE REAL)

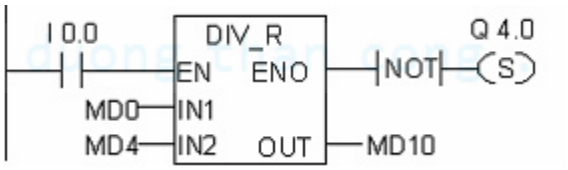
Ký hiệu:

LAD	FBD	STL
		L <IN1> L <IN2> /R T <OUT>

Hoạt động:

Khi trạng thái ngõ vào EN=1 thì lệnh DIV-R hoạt động, giá trị IN1 được chia bởi giá trị IN2 và kết quả gửi ra ngõ ra OUT. Nếu kết quả vượt quá giới hạn cho phép của số thực 32 bit thì bit OV và OS lên 1 và ENO=0, nên những chức năng khác mà được nối với ngõ ENO sẽ không hoạt động.

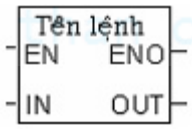
Ví dụ:



Khi ngõ vào I0.0=1 thì lệnh DIV-R hoạt động, kết quả của lệnh chia MD0 bởi MD4 được xuất ra ngõ MD10. Nếu kết quả nằm ngoài vùng hoạt động của số thực 32 bit thì Q4.0=1.

8.2. NHÓM CHỨC NĂNG TOÁN HỌC CAO CẤP

Các lệnh trong nhóm chức năng toán học cao cấp có ký hiệu tương tự, có ký hiệu như sau:



EN là ngõ vào cho phép lệnh toán học hoạt động (Enable Input). Biểu diễn toán hạng là dữ liệu dạng số nhị phân. Biểu diễn toán hạng là địa chỉ dạng: I, Q, L, M, D.

ENO là ngõ cho phép ngõ ra hoạt động (Enable Output). Ngõ ra ENO có cùng trạng thái với ngõ vào EN. Loại toán hạng là dữ liệu dạng số nhị phân. Loại toán

hạng là địa chỉ dạng: I, Q, L, M, D.

IN: Có chức năng phụ thuộc vào từng loại lệnh. Giá trị tại ngõ vào để thực hiện lệnh toán học, có toán hạng là dữ liệu dạng số thực, toán hạng là địa chỉ: I, Q, L, M, D. Khi EN=1 thì lệnh toán học thực hiện và giá trị được đọc vào địa chỉ ngõ ra OUT.

OUT: Ngõ ra là kết quả của lệnh toán học. Có toán hạng là dữ liệu dạng số thực, toán hạng là địa chỉ: I, Q, L, M, D.

8.2.1. Lệnh lấy giá trị tuyệt đối (ABS- Establish the Absolute Value of a Floating-Point Number)

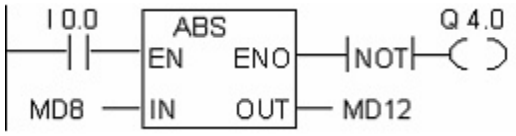
Ký hiệu:

LAD/FBD	STL
ABS EN ENO IN OUT	L <IN> ABS T <OUT>

Hoạt động:

Lệnh ABS lấy giá trị tuyệt đối của số thực khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Kết quả được xuất ra ngõ ra OUT.

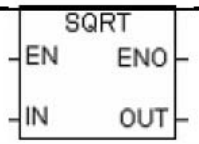
Ví dụ:



Khi I0.0=1 thì giá trị tuyệt đối của MD8 được xuất ra tại ngõ ra MD12. Chẳng hạn MD8=-12.2 thì MD12=12.2. Ngõ ra Q4.0=1 khi lệnh không thực hiện (EN=ENO=0).

8.2.2. LỆNH TÍNH CĂN BẬC HAI (SQRT- ESTABLISH THE SQUARE ROOT)

Ký hiệu:

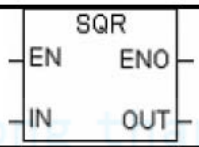
LAD/FBD	STL
	L <IN> SQRT T <OUT>

Hoạt động:

Lệnh SQRT dùng để tính căn bậc hai của số thực khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Lệnh này chỉ thực hiện với những số thực >=0. Kết quả được xuất ra ngõ ra OUT.

8.2.3. LỆNH TÍNH BÌNH PHƯƠNG (SQR- ESTABLISH THE SQUARE)

Ký hiệu:

LAD/FBD	STL
	L <IN> SQR T <OUT>

Hoạt động:

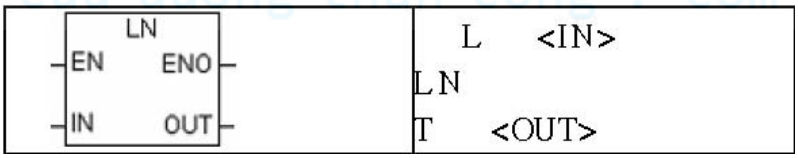
Lệnh SQR dùng để tính bình phương số thực khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Kết quả được xuất ra ngõ ra OUT.

8.2.4. LỆNH TÍNH LOGARIT CƠ SỐ E(LN- ESTABLISH THE NATURAL LOGARITHM)

Ký hiệu:

LAD/FBD

STL



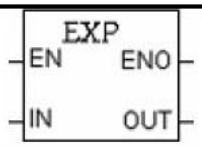
Hoạt động:

Lệnh LN dùng để tính Logarit cơ số **e** số thực khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Kết quả được xuất ra ngõ ra

OUT.

8.2.5. LỆNH TÍNH LŨY THỪA CỦA CƠ SỐ E (EXP- ESTABLISH THE EXPONENTIAL VALUE)

Ký hiệu:

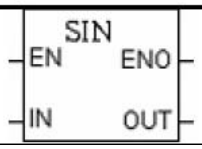
LAD/FBD	STL
	L <IN> EXP T <OUT>

Hoạt động:

Lệnh EXP dùng để tính lũy thừa cơ số e của số thực (ex) khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Kết quả được xuất ra ngõ ra OUT.

8.2.6. LỆNH SIN (SIN- ESTABLISH THE SINE VALUE)

Ký hiệu:

LAD/FBD	STL
	L <IN> SIN T <OUT>

Hoạt động:

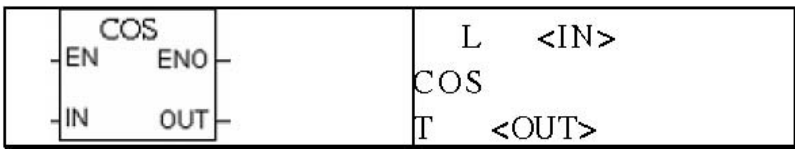
Lệnh SIN dùng để tính sin của số thực, số thực đại diện cho một góc có thứ nguyên là radian. Khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Kết quả được xuất ra ngõ ra OUT.

8.2.7. LỆNH COS (COS- ESTABLISH THE COSINE VALUE)

Ký hiệu:

LAD/FBD

STL

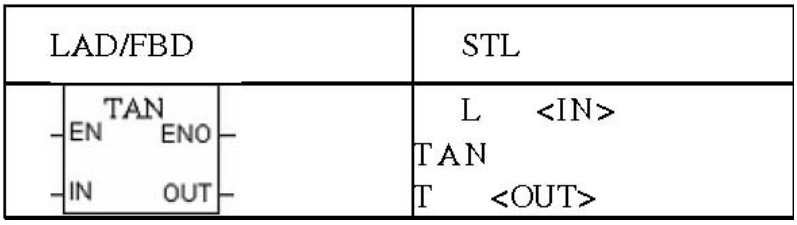


Hoạt động:

Lệnh COS dùng để tính Cos của số thực, số thực đại diện cho một góc có thứ nguyên là radian. Khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Kết quả được xuất ra ngõ ra OUT.

8.2.8. LỆNH TG (TAN- ESTABLISH THE TANGENT VALUE)

Ký hiệu:

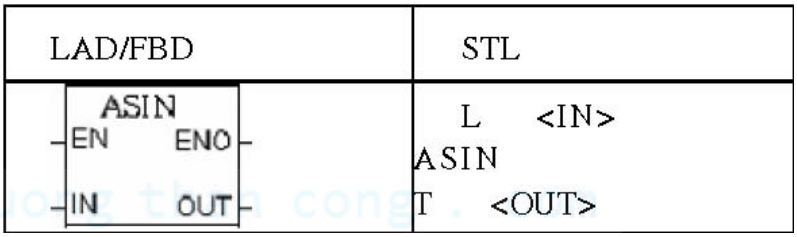


Hoạt động:

Lệnh TAN dùng để tính tg của số thực, số thực đại diện cho một góc có thứ nguyên là radian. Khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Kết quả được xuất ra ngõ ra OUT.

8.2.9. LỆNH ARSIN (ASIN- ESTABLISH THE ARC SINE VALUE)

Ký hiệu:



Hoạt động:

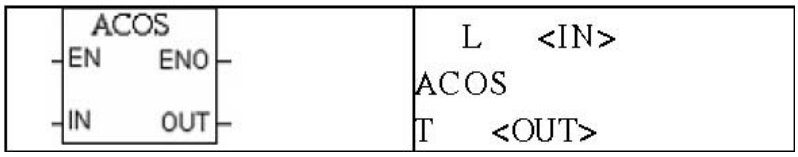
Lệnh ASIN dùng để tính arsin của số thực với giá trị ngõ vào nằm trong khoảng từ (-1<= giá trị vào<=+1), số thực đại diện cho một góc có thứ nguyên là radian. Khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Kết quả được xuất ra ngõ ra OUT.

8.2.10. LỆNH ARCOS (ACOS- ESTABLISH THE ARC COSINE VALUE)

Ký hiệu:

LAD/FBD

STL

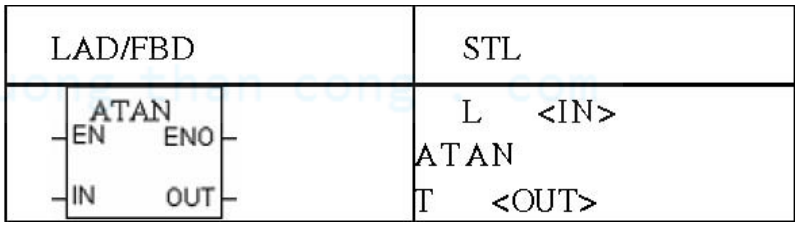


Hoạt động:

Lệnh ACOS dùng để tính arcos của số thực với giá trị ngõ vào nằm trong khoảng từ (-1<= giá trị vào<=+1), số thực đại diện cho một góc có thứ nguyên là radian. Khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Kết quả được xuất ra ngõ ra OUT.

8.2.11. LỆNH ARTG (ATAN- ESTABLISH THE ARC TANGENT VALUE)

Ký hiệu:



Hoạt động:

Lệnh ATAN dùng để tính artg của số thực, số thực đại diện cho một góc có thứ nguyên là radian. Khi trạng thái tín hiệu tại ngõ vào EN=1, ngõ ra ENO có cùng trạng thái với ngõ vào EN. Kết quả được xuất ra ngõ ra OUT.

CHƯƠNG 4

CÁC LỆNH GỌI CHỨC NĂNG ĐẶT BIỆT

CHƯƠNG 4 CÁC LỆNH GỌI CHỨC NĂNG ĐẶT BIỆT

I. LỆNH MỞ KHỐI DỮ LIỆU (OPN_OPEN DATA BLOCK)

Ký hiệu:

LAD FBD STL OPN DB <n>

DB <n>: Chỉ số dữ liệu cần mở, số khối dữ liệu phụ thuộc vào từng loại CPU. Biểu diễn toán hạng là dữ liệu dạng khối DB.

Hoạt động: Lệnh mở khối dữ liệu cho phép mở khối dữ liệu toàn cục hoặc khối dữ liệu cục bộ. CPU có hai thanh ghi khối dữ liệu là DB và DI nên có thể mở hai khối dữ liệu cùng một lúc. Lệnh OPN là lệnh mở khối dữ liệu không điều kiện. Số của khối dữ liệu được truyền đến thanh ghi DB hoặc DI. Lệnh mở khối dữ liệu truy cập tuần tự các khối dữ liệu phụ thuộc vào dung lượng của thanh ghi.

Ví dụ:

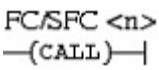
Khối dữ liệu DB1 được mở khi ngõ vào I0.0=1. Bit dữ liệu chứa trong khối dữ liệu DB1 là DBX0.0 được truyền đến ngõ ra Q4.0.

II. LỆNH GỌI CHỨC NĂNG/CHỨC NĂNG HỆ THỐNG (FC/SFC_CALL

FC/SFC FROM COIL)

Ký hiệu:

LAD FBD STL CC FC/SFC <n>



FC/SFC <n>: Chỉ số thứ tự của khối FC/SFC, phụ thuộc vào từng loại CPU. Biểu diễn toán hạng là dữ liệu dạng khối FC/SFC.

Hoạt động: Lệnh CALL dùng để gọi chức năng (FC) và chức năng hệ thống (SFC) ở dạng có điều kiện hoặc không có điều kiện. Lệnh gọi chỉ được thực thi nếu kết quả phép

toán Logic RLO=1. Nếu lệnh CALL được thực thi thì: • Địa chỉ của lệnh gọi được lưu

trữ. • Vùng dữ liệu cục bộ trước được thay thế bằng vùng dữ liệu cục

bộ hiện hành. • Bit MA (bit kích hoạt vùng MCR) được dịch vào thanh ghi con

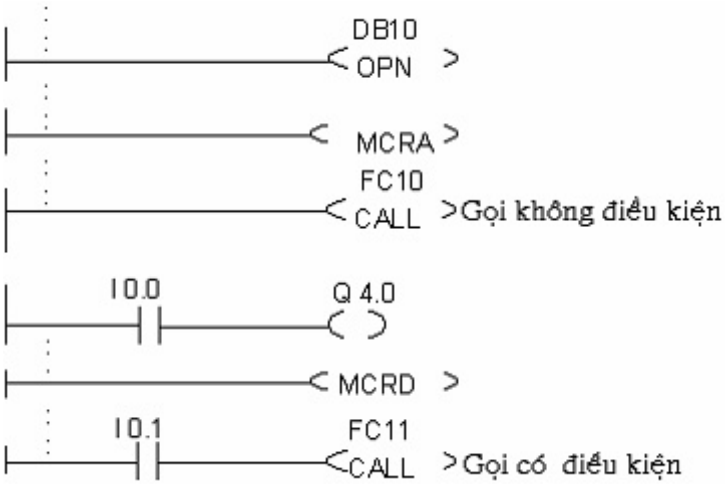
trỏ B. • Vùng dữ liệu cục bộ của chức năng gọi khối mới được tạo ra. Sau lệnh gọi

khối, chương trình tiếp tục thực hiện các lệnh tiếp theo trong khối

FC/SFC. Lệnh này tác động lên thanh ghi trạng thái như sau:

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
Không điều kiện	-	-	-	-	0	0	1	-	0
Có điều kiện	-	-	-	-	0	0	1	1	0

Ví dụ:



Chương trình trên được viết trong khối chức năng FB. Khi chương trình thực hiện đến lệnh OPN thì khối DB10 được mở, chức năng MCR được kích hoạt. Sau quá trình thực thi thì lệnh gọi không điều kiện khối FC10 được thực thi và xảy ra các quá trình như sau:

- Khối dữ liệu DB10 được đưa vào khối FB, địa chỉ gọi khối DB10 này được lưu. Bit MA (Dùng kích hoạt chức năng MCR) được dịch vào con trỏ B và khi đó đặt mức 0 cho lệnh gọi FC10. Chương trình xử lý tiếp tục trong FC10. Nếu chức năng MCR được kích hoạt bởi khối FC10 thì nó phải kích hoạt lại cùng với FC10. Khi FC10 thực hiện xong thì chương trình quay lại gọi khối FB. Bit MA được lưu trữ, DB10 và khối dữ liệu cục bộ do người dùng viết trong khối FB được mở trở lại. Chương trình tiếp tục làm những lệnh kế bằng việc đánh dấu vào trạng thái của I0.0 đến ngõ ra Q4.0. Lệnh gọi FC11 là lệnh gọi không điều kiện, nó chỉ thực hiện nếu I0.1 =1. Nếu FC11 thực thi thì quá trình xử lý chương trình và quay trở lại chương trình chính thực hiện giống như lệnh gọi khối FC10.

III. LỆNH GỌI KHỐI CHỨC NĂNG (FB_CALL FB FROM BOX)

Ký hiệu:

LAD/FBD	STL
DB<n> FB<n> EN ENO	CALL FB <n>,DB<n>

Ký hiệu trên còn phụ thuộc vào từng loại khối FB mà nó có thể có nhiều ngõ vào/ra. Một khối FB bao giờ cũng phải có ngõ EN, ENO, tên hoặc số của FB.

EN là ngõ vào cho phép lệnh gọi hoạt động (Enable Input). Loại toán hạng là dữ

liệu dạng BOOL, toán hạng là địa chỉ dạng I, Q, M, L, D.

ENO là ngõ ra cho phép các lệnh sau lệnh gọi mà có nối với ngõ ENO hoạt động (Enable Output). Ngõ ENO có cùng trạng thái với ngõ vào EN. Loại toán hạng là dữ liệu dạng BOOL, toán hạng là địa chỉ dạng I, Q, M, L, D.

FB <n>, DB <n>: Chỉ số của khối FB/DB, phụ thuộc vào từng loại CPU. Biểu diễn toán hạng là dữ liệu dạng khối FB/DB.

Hoạt động: Lệnh CALL-FB dùng để gọi khối chức năng (FB) ở dạng có điều kiện hoặc không có điều kiện. Lệnh gọi chỉ được thực thi nếu kết quả phép toán Logic RLO=1. Nếu lệnh CALL-FB được thực thi thì:

- Địa chỉ của lệnh gọi được lưu trữ.
- Dữ liệu của hai khối dữ liệu hiện hành (DB toàn cục và DB cục bộ) được lưu.
- Vùng dữ liệu cục bộ trước được thay thế bằng vùng dữ liệu cục bộ hiện hành.
- Bit MA (bit kích hoạt vùng MCR) được dịch vào thanh ghi con trỏ B.
- Vùng dữ liệu cục bộ của chức năng gọi khối mới được tạo ra.

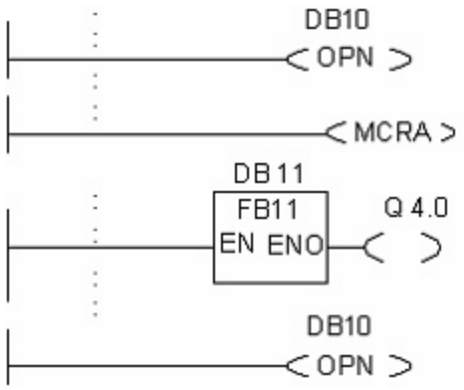
Sau lệnh gọi khối, chương trình tiếp tục xử lý trong khối chức năng đã được gọi. Bit BR được quét để mà nhận biết trạng thái ngõ ra ENO. Người sử dụng phải đánh dấu trạng thái cần thiết vào bit BR bằng lệnh lưu (SAVE) để nhận biết lỗi.

Khi mở một khối FB hoặc khối SFB thì khối DB được mở lúc trước bị mất. Do đó, phải mở khối DB lại.

Lệnh này tác động lên thanh ghi trạng thái như sau:

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
Không điều kiện	x	-	-	-	0	0	x	x	x
Có điều kiện	-	-	-	-	0	0	x	x	x

Ví dụ:



Chương trình trên được viết trong khối chức năng FB bởi người sử dụng. Khi chương trình thực hiện đến lệnh OPN thì khối DB10 được mở và khối dữ liệu cục bộ DB11 từ lệnh gọi khối FB11 được lưu, chức năng MCR được kích hoạt. Sau quá trình thực thi thì lệnh gọi không điều kiện khối FB11 được thực thi và xảy ra các quá trình như sau:

Khối dữ liệu DB10 được đưa vào khối FB, khối dữ liệu cục bộ DB11 được gọi từ lệnh gọi khối FB11 được lưu. Bit MA (Dùng kích hoạt chức năng MCR) được dịch vào con trỏ B và khi đó đặt mức 0 cho lệnh gọi FB11. Chương trình xử lý tiếp tục trong FB11. Nếu chức năng MCR được kích hoạt bởi khối FC11 thì nó phải kích hoạt lại cùng với FB11. Trạng thái của kết quả phép toán Logic RLO phải được lưu vào bit BR của thanh ghi trạng thái bằng lệnh SAVE để xác định lỗi trong quá trình gọi khối FB. Khi FB11 thực hiện xong thì chương trình quay lại gọi khối FB. Bit MA được lưu trữ và khối dữ liệu cục bộ do người dùng viết trong khối FB được mở trở lại. Nếu FB11 xử lý đúng thì ENO=1, vì vậy Q4.0=1.

IV. LỆNH GỌI CHỨC NĂNG (FC_CALL FC FROM BOX)

Ký hiệu:

LAD/FBD	STL
FC<n> EN ENO	CALL FC <n>

Ký hiệu trên còn phụ thuộc vào từng loại khối FC mà nó có thể có nhiều ngõ vào/ra. Một khối FC bao giờ cũng phải có ngõ EN, ENO, tên hoặc số của FC.

EN là ngõ vào cho phép lệnh gọi hoạt động (Enable Input). Loại toán hạng là dữ liệu dạng BOOL, toán hạng là địa chỉ dạng I, Q, M, L, D.

ENO là ngõ ra cho phép các lệnh sau lệnh gọi mà có nối với ngõ ENO hoạt động (Enable Output). Ngõ ENO có cùng trạng thái với ngõ vào EN. Loại toán hạng là dữ liệu dạng BOOL, toán hạng là địa chỉ dạng I, Q, M, L, D.

FC <n>: Chỉ số của khối FC, phụ thuộc vào từng loại CPU. Biểu diễn toán hạng là dữ liệu dạng khối FC.

Hoạt động: Lệnh CALL-FC dùng để gọi chức năng (FC) ở dạng có điều kiện hoặc không có điều kiện. Lệnh gọi chỉ được thực thi nếu kết quả phép toán Logic RLO=1. Nếu lệnh CALL-FC được thực thi thì:

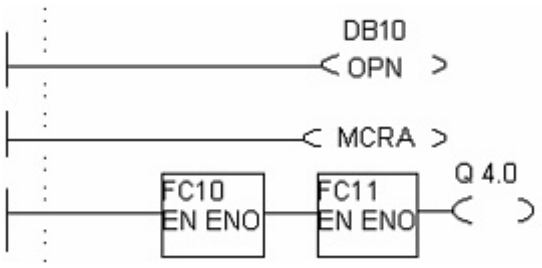
- Địa chỉ của lệnh gọi được lưu trữ.
- Vùng dữ liệu cục bộ trước được thay thế bằng vùng dữ liệu cục bộ hiện hành.
- Bit MA (bit kích hoạt vùng MCR) được dịch vào thanh ghi con trỏ B.
- Vùng dữ liệu cục bộ mới của lệnh gọi chức năng được tạo ra.

Sau lệnh gọi khối, chương trình tiếp tục xử lý trong chức năng đã được gọi. Bit BR được quét để mà nhận biết trạng thái ngõ ra ENO. Người sử dụng phải đánh dấu trạng thái cần thiết vào bit BR bằng lệnh lưu (SAVE) để nhận biết lỗi. Nếu gọi chức năng và có khai báo trong bảng khai báo biến có các ngõ IN, OUT, IN-OUT thì các ngõ này được thêm vào trong chương trình gọi khối như là một ngõ vào/ra bình thường. Khi gọi khối chức năng FC chúng ta phải đánh dấu các ngõ vào/ra tại lúc gọi.

Lệnh này tác động lên thanh ghi trạng thái như sau:

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
Không điều kiện	x	-	-	-	0	0	x	x	x
Có điều kiện	-	-	-	-	0	0	x	x	x

Ví dụ:



Chương trình trên được viết trong khối chức năng FB bởi người sử dụng. Khi

chương trình thực hiện đến lệnh OPN thì khối DB10 được mở và chức năng MCR được kích hoạt. Sau quá trình thực thi thì lệnh gọi không điều kiện khối FC10 được thực thi và xảy ra các quá trình như sau:

- Khối dữ liệu DB10 được đưa vào khối FB, khối dữ liệu cục bộ được gọi từ lệnh gọi khối FB được lưu. Bit MA (Dùng kích hoạt chức năng MCR) được dịch vào con trỏ B và khi đó đặt mức 0 cho lệnh gọi FC10. Chương trình xử lý tiếp tục trong FC10. Nếu chức năng MCR được kích hoạt bởi khối FC10 thì nó phải kích hoạt lại cùng với FC10. Trạng thái của kết quả phép toán Logic RLO phải được lưu vào bit BR của thanh ghi trạng thái bằng lệnh SAVE để xác định lỗi trong quá trình gọi khối FB. Khi FC10 thực hiện xong thì chương trình quay lại gọi khối FB. Bit MA được lưu trữ . Sau khi FC10 thực thi xong thì chương trình xử lý tiếp tục gọi khối FB và nếu:

. ENO=1 thì FC11 được thực thi.
. ENO=0 thì chương trình sẽ thực hiện tiếp ở Nextword kế.
. Nếu FC11 xử lý đúng thì ngõ ra ENO=1, vì vậy Q4.0=1.

v. LỆNH GỌI KHỐI CHỨC NĂNG HỆ THỐNG (SFB_CALL SYSTEM FB FROM BOX)

Ký hiệu:

LAD/FBD	STL
DB <n> SFB <n> EN ENO	CALL SFB <n>,DB<n>

Ký hiệu trên còn phụ thuộc vào từng loại khối SFB mà nó có thể có nhiều ngõ vào/ra. Một khối SFB bao giờ cũng phải có ngõ EN, ENO, tên hoặc số của SFB.

EN là ngõ vào cho phép lệnh gọi hoạt động (Enable Input). Loại toán hạng là dữ liệu dạng BOOL, toán hạng là địa chỉ dạng I, Q, M, L, D.

ENO là ngõ ra cho phép các lệnh sau lệnh gọi mà có nối với ngõ ENO hoạt động (Enable Output). Ngõ ENO có cùng trạng thái với ngõ vào EN. Loại toán hạng là dữ liệu dạng BOOL, toán hạng là địa chỉ dạng I, Q, M, L, D.

SFB <n>, DB <n>: Chỉ số của khối SFB/DB, phụ thuộc vào từng loại CPU. Biểu diễn toán hạng là dữ liệu dạng khối SFB/DB.

Hoạt động: Lệnh CALL-SFB dùng để gọi khối chức năng hệ thống (SFB) ở dạng

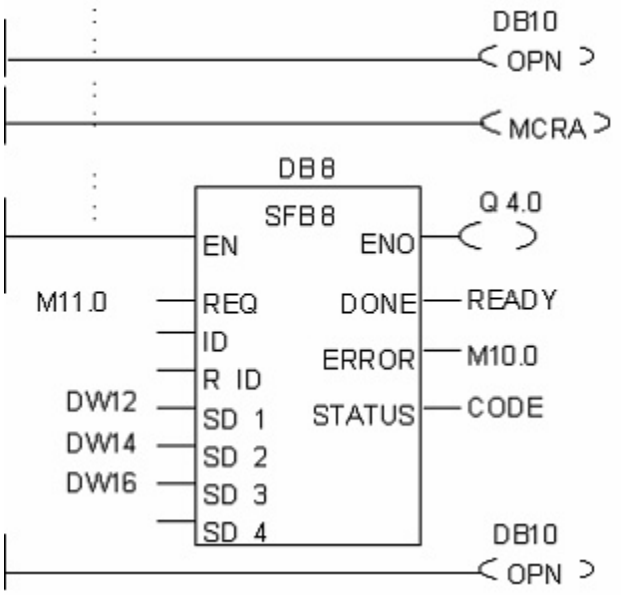
có điều kiện hoặc không có điều kiện. Lệnh gọi chỉ được thực thi nếu kết quả phép toán Logic RLO=1. Nếu lệnh CALL-SFB được thực thi thì:

- Địa chỉ của lệnh gọi được lưu trữ.
- Dữ liệu của hai khối dữ liệu hiện hành (DB toàn cục và DB cục bộ) được lưu.
- Vùng dữ liệu cục bộ trước được thay thế bằng vùng dữ liệu cục bộ hiện hành.
- Bit MA (bit kích hoạt vùng MCR) được dịch vào thanh ghi con trỏ B.
- Vùng dữ liệu cục bộ của chức năng gọi khối mới được tạo ra.

Sau lệnh gọi khối, chương trình tiếp tục xử lý trong khối chức năng hệ thống đã được gọi. Ngõ ra ENO=1 khi khối SFB được gọi (EN=1) và không có lỗi. Lệnh này tác động lên thanh ghi trạng thái như sau:

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
Không điều kiện	x	-	-	-	0	0	x	x	x
Có điều kiện	-	-	-	-	0	0	x	x	x

Ví dụ:



Chương trình trên được viết trong khối chức năng FB bởi người sử dụng. Khi chương trình thực hiện đến lệnh OPN thì khối DB10 được mở và chức năng MCR được kích hoạt. Sau quá trình thực thi thì lệnh gọi không điều kiện khối SFB8 được thực thi và xảy ra các quá trình như sau:

• Khối dữ liệu DB10 được đưa vào khối FB, khối dữ liệu cục bộ được gọi từ lệnh gọi khối FB được lưu. Bit MA (Dùng kích hoạt chức năng MCR) được dịch vào con trỏ B và khi đó đặt mức 0 cho lệnh gọi SFB8. Chương trình xử lý tiếp tục trong SFB8. Nếu chức năng MCR được kích hoạt bởi khối SFB8 thì nó phải kích hoạt lại cùng với SFB8. Khi SFB8 thực hiện xong thì chương trình quay lại gọi khối FB. Bit MA được lưu trữ và khối dữ liệu cục bộ do người dùng viết trong khối FB được mở trở lại. Nếu SFB8 xử lý đúng thì ENO=1, vì vậy Q4.0=1.

VI. LỆNH GỌI CHỨC NĂNG HỆ THỐNG (SFC_CALL SYSTEM FC FROM BOX)

Ký hiệu:

LAD/FBD	STL
SFC <n> EN ENO	CALL SFC <n>

Ký hiệu trên còn phụ thuộc vào từng loại khối SFC mà nó có thể có nhiều ngõ vào/ra. Một khối SFC bao giờ cũng phải có ngõ EN, ENO, tên hoặc số của SFC.

EN là ngõ vào cho phép lệnh gọi hoạt động (Enable Input). Loại toán hạng là dữ liệu dạng BOOL, toán hạng là địa chỉ dạng I, Q, M, L, D.

ENO là ngõ ra cho phép các lệnh sau lệnh gọi mà có nối với ngõ ENO hoạt động (Enable Output). Ngõ ENO có cùng trạng thái với ngõ vào EN. Loại toán hạng là dữ liệu dạng BOOL, toán hạng là địa chỉ dạng I, Q, M, L, D.

SFC <n>: Chỉ số của khối SFC, phụ thuộc vào từng loại CPU. Biểu diễn toán hạng là dữ liệu dạng khối SFC.

Hoạt động: Lệnh CALL-SFC dùng để gọi chức năng hệ thống (SFC) ở dạng có điều kiện hoặc không có điều kiện. Lệnh gọi chỉ được thực thi nếu kết quả phép toán Logic RLO=1. Nếu lệnh CALL-SFC được thực thi thì:

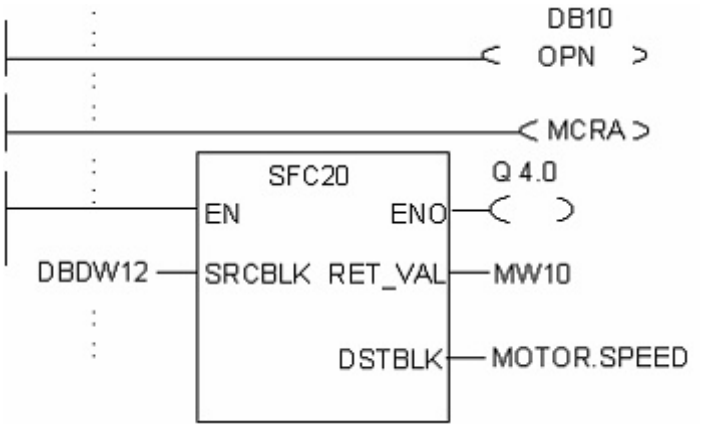
- Địa chỉ của lệnh gọi được lưu trữ.
- Vùng dữ liệu cục bộ trước được thay thế bằng vùng dữ liệu cục bộ hiện hành.
- Bit MA (bit kích hoạt vùng MCR) được dịch vào thanh ghi con trỏ B.
- Vùng dữ liệu cục bộ mới của lệnh gọi chức năng được tạo ra.

Sau lệnh gọi khối, chương trình tiếp tục xử lý trong chức năng hệ thống SFC đã được gọi. Ngõ ra ENO=1 khi khối SFC được gọi (EN=1) và không có lỗi. Bit BR được quét để mà nhận biết trạng thái ngõ ra ENO. Người sử dụng phải đánh dấu trạng thái cần thiết vào bit BR bằng lệnh lưu (SAVE) để nhận biết lỗi.

Lệnh này tác động lên thanh ghi trạng thái như sau:

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
Không điều kiện	x	-	-	-	0	0	x	x	x
Có điều kiện	-	-	-	-	0	0	x	x	x

Ví dụ:



Chương trình trên được viết trong khối chức năng FB bởi người sử dụng. Khi chương trình thực hiện đến lệnh OPN thì khối DB10 được mở và chức năng MCR được kích hoạt. Sau quá trình thực thi thì lệnh gọi không điều kiện khối SFC20 được thực thi và xảy ra các quá trình như sau:

- Khối dữ liệu DB10 được đưa vào khối FB, khối dữ liệu cục bộ được gọi từ lệnh gọi khối FB được lưu. Bit MA (Dùng kích hoạt chức năng MCR) được dịch vào con trỏ B và khi đó đặt mức 0 cho lệnh gọi SFC20. Chương trình xử lý tiếp tục trong SFC20. Nếu chức năng MCR được kích hoạt bởi khối SFC20 thì nó phải kích hoạt lại cùng với SFC20. Khi SFC20 thực hiện xong thì chương trình quay lại gọi khối FB. Bit MA được lưu trữ . Ngõ ra Q4.0 phụ thuộc vào ngõ ENO:

. ENO=1 thì Q4.0=1.
. ENO=0 thì Q4.0=0.

CHƯƠNG 5

BẢNG KÝ HIỆU (SYMBOL TABLE)

CHƯƠNG 5 BẢNG KÝ HIỆU (SYMBOL TABLE)

I. KHÁI NIỆM VỀ ĐỊA CHỈ TUYỆT ĐỐI, ĐỊA CHỈ TƯƠNG ĐỐI

1.1. ĐỊA CHỈ TUYỆT ĐỐI:

Địa chỉ tuyệt đối là địa chỉ trực tiếp.

Ví dụ: I0.0, Q4.0 là địa chỉ trực tiếp.

Khi dùng địa chỉ trực tiếp thì không cần bảng ký hiệu, nhưng chương trình sẽ khó đọc.

1.2. ĐỊA CHỈ KÝ HIỆU

Địa chỉ ký hiệu là địa chỉ sử dụng các ký hiệu thay vì sử dụng địa chỉ tuyệt đối.

Ví dụ: DONGCO, NGOVAO1 là các địa chỉ ký hiệu.

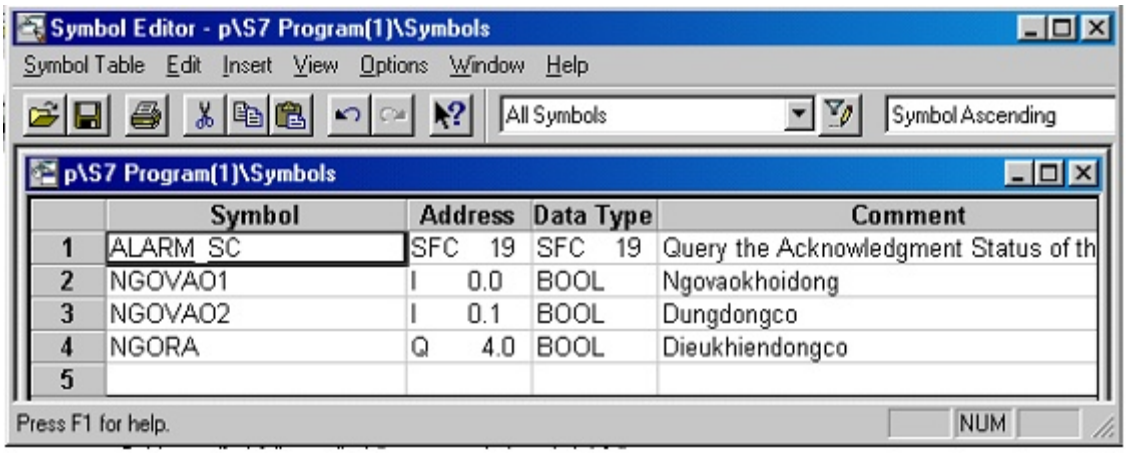
Các ký hiệu cho các ngõ vào, ngõ ra, timer, bit nhớ và các khối được lưu trữ trong bảng ký hiệu. Các ký hiệu được đánh dấu bằng dấu ngoặc kép. Khi nhập các tên ký hiệu thì không cần phải đánh dấu ngoặc kép mà trình soạn thảo sẽ thêm vào sau khi nhập xong.

Có hai loại ký hiệu là ký hiệu chung và ký hiệu cục bộ.

Ký hiệu chung được khai báo trong bảng ký hiệu, có thể sử dụng cho tất cả các khối của chương trình. Tên trong bảng ký hiệu phải là duy nhất, tức là một tên ký hiệu chỉ được xuất hiện một lần trong bảng ký hiệu.

Ký hiệu cục bộ được khai báo trong phần khai báo của khối, chúng chỉ có thể dùng trong khối đó. Tên ký hiệu giống nhau cũng có thể sử dụng trong phần khai báo của khối khác.

II. CÁC THÀNH PHẦN CHÍNH CỦA BẢNG KÝ HIỆU



Bảng ký hiệu có các thành phần giống như các phần mềm khác như thanh tiêu đề, thanh menu, thanh công cụ và có thành phần quan trọng nhất là để soạn thảo ký hiệu.

Bảng soạn thảo ký hiệu có các hàng và các cột. Các cột gồm **Symbol** chứa tên ký hiệu, **Address** chứa địa chỉ, **Data Type** chứa loại dữ liệu, **Comment** chứa lời bình cho ký hiệu (có thể có hoặc không). Mỗi một ký hiệu chiếm một hàng trong bảng và cuối mỗi bảng có một hàng để trống để có thể thêm vào một ký hiệu mới.

Trong bảng ký hiệu chúng ta có thể tìm những địa chỉ và thay thế nó bằng một địa chỉ khác. Lọc những ký hiệu để xem, quan sát và có thể liên kết chúng với nhau theo thứ tự tên, địa chỉ, loại dữ liệu, lời bình, hoạt động điều khiển và giám sát, giao tiếp, lời nhắn... hoặc sắp xếp dữ liệu theo thứ tự tên, bảng chữ cái tăng dần hoặc giảm dần. Có thể xuất hoặc nhập bảng ký hiệu đến một chương trình khác theo những file có định dạng sau:

- Định dạng ASCII (*.ASC) như Notepad, Word.
- Định dạng dữ liệu thay thế (*.DIF) như Excel.
- Định dạng hệ thống dữ liệu (*.SDF) như Access.
- Danh sách cài đặt (*.SEQ) như STEP 5.

III. SOẠN THẢO KÝ HIỆU TRONG LAD/STL/FBD

Có thể soạn thảo ký hiệu trong LAD/STL/FBD bằng hai cách: Vào menu **Insert/Symbol** hoặc nhấp chuột vào toán tử cần soạn thảo ký hiệu và sau đó chọn **Insert Symbol** để cho phép cài đặt tên ký hiệu tương ứng với các địa chỉ tyệt đối. Tên ký hiệu được nhập vào bảng ký hiệu một cách tự động. Nếu tên ký hiệu được nhập trùng với tên đã có sẵn trong bảng ký hiệu thì nó sẽ hiển thị trong một màu khác,

chúng không thể sử dụng lại trong bảng ký hiệu.

Trong trình soạn thảo LAD/STL/FBD có thể quan sát địa chỉ theo địa chỉ ký hiệu hoặc địa chỉ tuyệt đối bằng cách vào menu **View** chọn **Sympolic Representation**. Để hiển thị thông tin cài đặt về địa chỉ ký hiệu và địa chỉ tuyệt đối được sử dụng trong nextword bằng cách vào menu **View** chọn **Sympolic Information**.

CHƯƠNG 6

CÁC KHỐI TỔ CHỨC

CHƯƠNG 6

CÁC KHỐI TỔ CHỨC

I. TỔNG QUAN VỀ KHỐI TỔ CHỨC OB

Khối OB (Organization Block): Là khối tổ chức và quản lý chương trình, có nhiều khối OB với các dữ liệu khác nhau và chúng chỉ được gọi bởi hệ điều hành. Khối tổ chức là khối giao tiếp giữa hệ điều hành của CPU với chương trình người dùng. Những khối tổ chức OB được sử dụng để thực hiện những công việc sau:

- Tại thời điểm CPU khởi động.
- Quá trình thực hiện theo chu kỳ và theo thời gian.
- Khi có lỗi xảy ra.
- Khi xảy ra ngắt phần cứng.

Những khối tổ chức được thực hiện theo mức độ ưu tiên mà chúng cho phép. Khối OB nào có mức độ ưu tiên cao hơn thì khối đó sẽ được thực hiện trước.

Khối được CPU xử lý thường xuyên và theo chu kỳ là khối OB1, chương trình người dùng sẽ được chứa trong khối này. Còn các khối OB khác thì làm các nhiệm vụ khác như: Ngắt thời điểm, ngắt thời gian trễ, ngắt chu kỳ, ngắt phần cứng, ngắt lỗi không đồng bộ, ngắt lỗi đồng bộ, khởi động.

PLC S7-300 có các loại khối tổ chức sau: OB1 (thực hiện chương trình theo chu

kỳ), OB10 (thực hiện ngắt thời điểm), OB20 (thực hiện ngắt theo thời gian trễ), OB35 (thực hiện ngắt chu kỳ), OB40 (thực hiện ngắt phần cứng), OB80 đến OB87 (xác định các lỗi không đồng bộ), OB100 (là OB khởi động), OB121, OB 122 (xác định các lỗi đồng bộ).

Bảng các loại khối OB, quyền ưu tiên và chức năng của các loại OB trong S7-300:

Loại OB	Chức năng	Quyền ưu tiên
OB1	Thực hiện chương trình theo chu kỳ lặp lại	1
OB10	Thực hiện ngắt thời điểm	2
OB20	Thực hiện ngắt theo thời gian trễ	3
OB35	Thực hiện ngắt chu kỳ	12
OB40	Thực hiện ngắt phần cứng	16
OB80÷OB87	Xác định các lỗi không đồng bộ	26/28
OB100, OB102	OB khởi động	27
OB121, OB122	Xác định các lỗi đồng bộ	OB tương ứng bị ngắt bởi lỗi

Quyền ưu tiên 26/28: Ưu tiên là 26 nếu lỗi xảy ra trong khi một OB có quyền ưu tiên thấp hơn được xử lý. Ưu tiên là 28 nếu lỗi xảy ra trong khi một OB khởi động (có quyền ưu tiên 27) đang được xử lý.

II. KHỐI OB XỬ LÝ THEO CHU KỲ LẶP (OB1)

Khối OB1 được thực hiện trong hệ điều hành của CPU S7 một cách liên tục theo chu kỳ. Khi OB1 đã được thực hiện xong thì OB1 được khởi động lại bởi hệ điều hành. OB1 được thực thi khi quá trình khởi động xong. Chúng ta có thể gọi các chức năng và khối chức năng khác như FB, FC, SFB, SFC vào khối OB1.

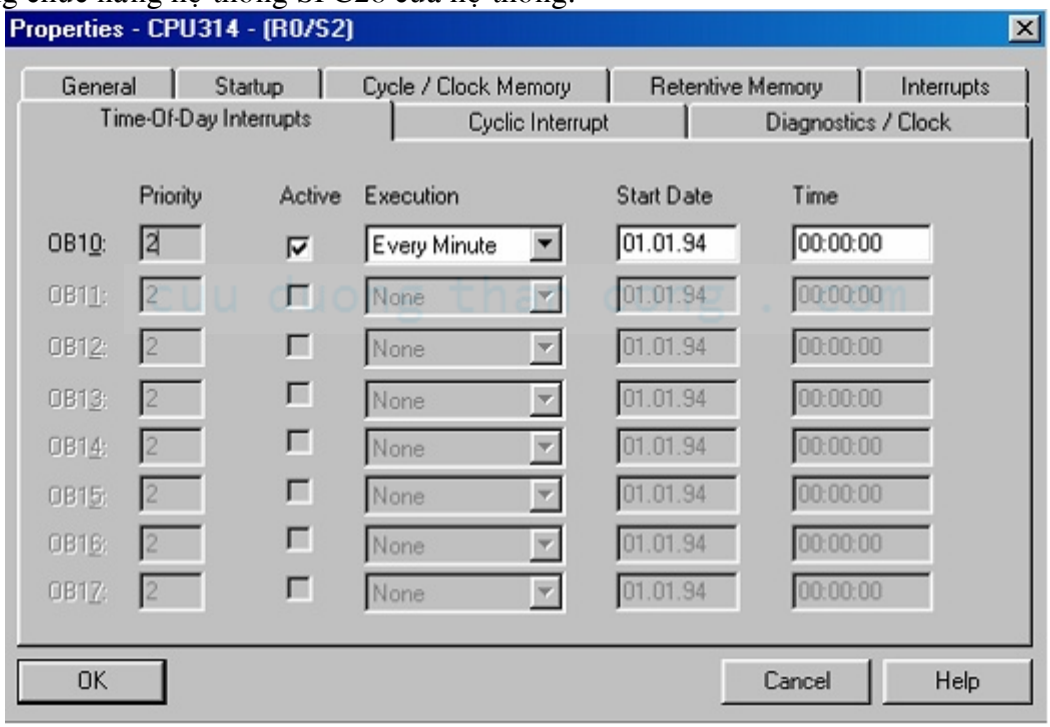
Khi một OB khác được gọi từ hệ điều hành thì quá trình xử lý theo chu kỳ bị ngắt vì khối OB1 có quyền ưu tiên thấp nhất. Bất kỳ một khối OB nào khác cũng có thể ngắt chương trình chính và tự xử lý chương trình của nó, sau khi xử lý xong thì nó trả về OB1 ngay vị trí mà nó ngắt. Nếu một OB có quyền cao hơn OB đang xử lý được gọi thì OB đang xử lý bị ngắt ngay sau khi hoàn thành câu lệnh. Hệ điều hành nhớ thanh ghi ngăn xếp cho khối bị ngắt. Các thông tin của thanh ghi này sẽ được khôi phục lại khi hệ điều hành bắt đầu xử lý lại khối bị ngắt.

Khi OB1 thực thi xong thì hệ điều hành gởi dữ liệu cục bộ đến bộ đệm ngõ ra và cập nhật thông tin từ bộ đệm ngõ vào trước khi khởi động lại khối OB1.

S7-300 có bộ giám sát thời gian quét cực đại và cực tiểu. Nếu thời gian quét nằm ngoài vùng này thì sẽ có một lỗi sảy ra. Thời gian quét cho phép này có thể thay đổi bằng cách sử dụng chức năng hệ thống SFC43. Nếu có lỗi sảy ra thì hệ điều hành gọi khối OB80 (OB xác định thời gian lỗi), nếu OB80 không được lập trình thì CPU chuyển sang chế độ STOP. Thời gian quét cho phép cực đại là 150ms.

III. KHỐI OB XỬ LÝ NGẮT THỜI ĐIỂM (OB10)

Ngắt thời được sử dụng cho một chương trình xác định được gọi trong OB10 ở một điểm thời gian xác định hay khởi động theo chu kỳ tại điểm thời gian này như mỗi phút, mỗi giờ, mỗi ngày, mỗi tuần, mỗi tháng, cuối mỗi tháng, mỗi năm. Ngắt thời điểm được thiết kế bằng công cụ "HW-Config" của chương trình STEP7 hoặc sử dụng chức năng hệ thống SFC28 của hệ thống.



Nếu hộp điều khiển **Active** được tác động thì OB ngắt thời điểm sẽ được khởi động tự động ở mỗi lần khởi động lại toàn bộ.

Các SFC được sử dụng để điền khiển các ngắt thời điểm là:

- SFC28: Đặt ngày, giờ khởi động và chu kỳ.
- SFC29: Xoá ngắt thời điểm.
- SFC30: Tác động ngắt thời điểm.

- SFC31: Hỏi ngắt thời điểm.

Chúng ta có thể tích cực hoặc huỷ bỏ chế độ ngắt thời điểm bằng những hàm có sẵn trong hệ điều hành và do đó không cần phải chuyển CPU về chế độ STOP.

Hàm SFC39 có tác dụng che ngắt.

Hàm SFC40 có tác dụng bỏ mặt nạ che ngắt.

Hàm SFC41 có tác dụng che tất cả các ngắt có mức ưu tiên cao hơn tín hiệu ngắt đang được xử lý.

Hàm SFC42 có tác dụng bỏ mặt nạ che tất cả các ngắt có mức ưu tiên cao hơn tín hiệu ngắt đang được xử lý.

IV. NGẮT THỜI GIAN TRỄ (OB20)

Chương trình ngắt thời gian trễ (OB20) được xử lý với một sự làm chậm xác định sau một sự cố xác định xảy ra. OB20 chỉ có thể được kích hoạt bằng việc gọi chức năng hệ thống SFC32. SFC32 cũng được sử dụng cho việc đặt thời gian trễ. Chức năng ngắt thời gian trễ chỉ có thể thực thi khi CPU được đặt ở chế độ RUN. Ngoài chức năng hệ thống SFC32, các SFC sau cũng được sử dụng cho mối quan hệ với ngắt thời gian trễ:

- SFC33: Xoá ngắt thời gian trễ.

- SFC34: Hỏi ngắt thời gian trễ.

V. NGẮT CHU KỲ (OB35)

Ngắt chu kỳ cho phép xử lý một khối trong một khoảng thời gian cố định. Chuẩn thời gian cho OB35 là 100ms, nghĩa là cứ 100ms thì khối OB35 được gọi và thực hiện một lần. Có thể điều chỉnh giá trị này trong phạm vi từ 1ms đến 1 phút.

Khoảng thời gian ngắt chu kỳ được đặt cố định và nó được lặp lại sau mỗi khoảng thời gian đặt trước. Thời gian gián đoạn cụ thể lớn hơn thời gian được yêu cầu cho xử lý. Hệ điều hành sẽ gọi OB35 tại thời điểm xác định, nếu trong khoảng thời gian cho phép OB35 xử lý mà OB35 vẫn còn hoạt động thì hệ điều hành sẽ gọi khối OB80 (OB ngắt chu kỳ). Ở tại thời điểm chạy chương trình OB ngắt chu kỳ không thể được điều

khiển bởi các chức năng hệ thống.

Chúng ta có thể tích cực hoặc huỷ bỏ chế độ ngắt theo chu kỳ bằng những hàm có sẵn trong hệ điều hành và do đó không cần phải chuyển CPU về chế độ STOP.

- Hàm SFC39 có tác dụng che ngắt.

- Hàm SFC40 có tác dụng bỏ mặt nạ che ngắt.

- Hàm SFC41 có tác dụng che tất cả các ngắt có mức ưu tiên cao hơn tín hiệu ngắt đang được xử lý.

- Hàm SFC42 có tác dụng bỏ mặt nạ che tất cả các ngắt có mức ưu tiên cao hơn tín hiệu ngắt đang được xử lý.

VI. NGẮT PHẦN CỨNG (OB40)

Chương trình trong OB ngắt phần cứng được xử lý ngay lập tức khi xảy ra một sự cố xác định. Ngắt phần cứng được tác động từ các modul đặt biệt khác nhau.

Tín hiệu nào cần được khởi động có thể được xác định ở các tham số của các modul tín hiệu (DI, DO, AI, AO) bằng công cụ "HW-Config".

Ở modul CP và FM thì đặt tính ngắt phần cứng được đặt trong phần trợ giúp của phần mềm cấu hình của các modul tương ứng.

Ngắt phần cứng còn có thể kích hoạt bằng các hàm chức năng hệ thống sau:
- Hàm SFC55 để ghi tham số đặt cấu hình cho modul.

- Hàm SFC56 để sửa đổi một vài tham số cấu hình của modul trong chế độ RUN.

- Hàm SFC57 để sửa đổi toàn bộ tham số cấu hình của modul.

Trong khoảng thời gian thực hiện chương trình của OB40 thì hệ thống sẽ không nhận và không xử lý bất cứ một tín hiệu ngắt cứng nào khác.

Chúng ta có thể sử dụng OB40 để xác định giá trị giới hạn thích hợp ở một modul. Nếu giá trị đo vượt quá giá trị giới hạn thì khối OB40 được gọi. Việc dùng khối OB 40 thì không cần viết chương trình trong các khối khác. Chương trình trong khối OB40 có thể được sử dụng để tạo ra một ngắt hay điều khiển dự án.

Tín hiệu ngắt cứng có thể:

- Che ngắt nhờ hàm SFC39.
- Bỏ mặt nạ che ngắt nhờ hàm SFC40.
- Che tất cả các ngắt có mức ưu tiên cao hơn tín hiệu ngắt đang được xử lý nhờ hàm SFC41.
- Bỏ mặt nạ che tất cả các ngắt có mức ưu tiên cao hơn tín hiệu ngắt đang được xử lý nhờ hàm SFC42.

VII. NGẮT CHUẨN ĐOÁN, NGẮT LỖI KHÔNG ĐỒNG BỘ (OB80...OB87)

Lỗi không đồng bộ là lỗi trong nhiệm vụ của CPU. Chúng xảy ra không đồng bộ trong xử lý chương trình và không được sắp xếp vào một vị trí chương trình xác định.

Nếu một lỗi trong chế độ RUN được phát hiện và OB báo lỗi tương thích đã được lập trình, thì nó được gọi và xử lý chương trình của nó. Chương trình này có thể: Đóng mạch cho một còi báo hiệu, tạo việc sao chép dữ liệu và tiếp theo tạo lệnh STOP, một chương trình để ghi lại sự xảy ra thường xuyên của lỗi, không đặt CPU vào chế độ STOP.

Nếu OB báo lỗi cho một lỗi thông thường không tồn tại, thì CPU tự động chuyển sang chế độ STOP. Tất cả các tín hiệu ngắt báo lỗi không đồng bộ đều có thể được che hoặc bỏ mặt nạ che nhờ sử dụng các hàm chức năng hệ thống:

- Hàm SFC39 có tác dụng che ngắt.
- Hàm SFC40 có tác dụng bỏ mặt nạ che ngắt.
- Hàm SFC41 có tác dụng che tất cả các ngắt có mức ưu tiên cao hơn tín hiệu ngắt đang được xử lý.
- Hàm SFC42 có tác dụng bỏ mặt nạ che tất cả các ngắt có mức ưu tiên cao hơn tín hiệu ngắt đang được xử lý.

Bảng các OB báo lỗi không đồng bộ:

OB	Kiểu lỗi	Ưu tiên	Ví dụ
OB80	Lỗi thời gian	26	Thời gian quét chu trình vượt qua cực đại

OB81	Mất nguồn cung cấp	26/28	Hư hỏng pin Backup
OB82	Ngắt chuẩn đoán		Đứt dây ở ngõ vào của modul có khả năng phỏng đoán
OB83	Ngắt do chèn/di chuyển modul		Di chuyển một modul tín hiệu đang ở chế độ hoạt động
OB84	Lỗi phần cứng CPU		Mức tín hiệu không đúng ở đầu giao tiếp MPI
OB85	Lỗi xử lý chương trình		Lỗi trong việc cập nhật dự án (thiếu modul)
OB86	Sai giá đỡ mở rộng		Hư hỏng thiết bị mở rộng hay DB Slave
OB87	Lỗi truyền thông		Lỗi trong khi đọc tín hiệu truyền thông

VIII. CÁC KHỐI TỔ CHỨC KHỞI ĐỘNG (OB100, OB102)

Các khối tổ chức được khởi động riêng bởi hệ điều hành. Có nhiều sự cố khởi động khác nhau mà nó sẽ dẫn đến việc khởi động các khối tổ chức liên quan với nhóm ưu tiên tương ứng. Các khối tổ chức có thể chứa một chương trình điều khiển bình thường, cũng như một bản khai báo. Đối với S7-300 có khởi động lại toàn bộ được thực hiện bởi OB100 và khởi động nguội được thực bởi khối OB102 (chỉ có đối với CPU 318-2).

CPU thực hiện khởi động khi:

- Cấp nguồn cho CPU.
- Khi chuyển công tắt từ vị trí STOP sang vị trí RUN hoặc RUN-P.
- Sau khi có yêu cầu sử dụng chức năng giao tiếp.

Khởi động lại toàn bộ (OB100): Khi thực hiện khởi động lại toàn bộ thì bộ đệm (PI), timer, bộ đếm và bit nhớ không được nhớ bởi pin Backup (Non-retentive) bị xoá và việc xử lý chương trình trong OB1 bắt đầu ở câu lệnh đầu tiên.

Khởi động nguội (OB102): Dùng để khởi động lại khi nguồn bị mất. Nó được tạo ra trong cấu hình phần cứng (HW-Configuration) khi gán các thông số cho CPU. Khi khởi động nguội thì tất cả các bit nhớ, bộ đệm (PI), timer, bộ đếm bị xoá. Khối dữ liệu vẫn duy trì giá trị của nó và chương trình được xử lý lại với câu lệnh đầu tiên trong OB1 sau khi khối khởi động OB102 được xử lý.

IX. CÁC OB BÁO LỖI ĐỒNG BỘ (OB121, OB122)

Những lỗi không đồng bộ có thể được nằm ở một vị trí xác định trong chương trình

nếu lỗi xảy ra trong khi xử lý một câu lệnh xác định. Các OB báo lỗi được gọi đáp ứng với lỗi đồng bộ được xử lý như là một phần của chương trình có cùng quyền ưu tiên với khối đang được xử lý khi lỗi được phát hiện. Nếu khối báo lỗi không đồng bộ không được khai báo thì khi xảy ra lỗi CPU sẽ tự động chuyển từ chế độ RUN sang chế độ STOP. Chương trình S7 cung cấp 3 SFC dùng để che hoặc bỏ mặt nạ che tín hiệu ngắt báo lỗi lập trình và lỗi truy xuất:

- Hàm SFC36 có tác dụng che ngắt.

- Hàm SFC37 có tác dụng bỏ mặt nạ che ngắt.

- Hàm SFC38 có tác dụng đọc nội dung thanh ghi báo kiểu lỗi lập trình gặp phải

OB121: Dùng để báo lỗi lập trình. Hệ điều hành sẽ gọi khối OB121 khi có lỗi xảy ra liên quan đến việc xử lý chương trình. Ví dụ như một khối không tồn tại trong CPU mà được gọi vào trong chương trình thì khối OB121 sẽ được gọi.

OB122: Dùng để báo lỗi truy xuất. Hệ điều hành sẽ gọi khối OB122 khi có lỗi xảy ra trong khi dữ liệu được truy cập đến một modul nào đó. Ví dụ như một modul bị thiếu hay không tồn tại trong chương trình (ví dụ modul I/O không tồn tại) thì khối OB122 sẽ được gọi.

CÁC CHỨC NĂNG HỆ THỐNG (SFC) VÀ KHỐI CHỨC NĂNG HỆ THỐNG (SFB)

CHƯƠNG 6

CHƯƠNG 6

CÁC CHỨC NĂNG HỆ THỐNG (SFC) VÀ KHỐI CHỨC NĂNG HỆ THỐNG (SFB)

I. GIỚI THIỆU CHUNG

Khối SFC (System Functon): Chức năng hệ thống là một chức năng đặt biệt được tích hợp trong hệ điều hành của CPU S7 mà có thể được gọi giống như một chức năng FC vào trong chương trình người sử dụng khi cần thiết.

Khối SFB (System Functon Block): Khối chức năng hệ thống là một khối chức năng đặt biệt được tích hợp trong hệ điều hành của CPU S7 mà có thể được gọi giống như một khối chức năng FB vào trong chương trình người sử dụng khi cần thiết.

Trong cấu trúc chương trình dạng hình thang LAD luôn có hai ngõ EN và ENO. Ngõ vào EN dùng để cho phép các khối SFC và SFB hoạt động. Ngõ ra ENO sẽ có cùng trạng thái với ngõ vào EN nếu lệnh được thực hiện không có lỗi. Chúng ta có thể nhận biết có hay không có lỗi sảy ra bằng việc sử dụng khối SFC và SFB.

Khi sử dụng khối SFC để xác định lỗi trong quá trình thực hiện chương trình trong trong hệ điều hành CPU S7-300. Nếu có lỗi sảy ra thì chúng ta có thể xác định bằng cách kiểm tra bit BR trong thanh ghi trạng thái hoặc tại ngõ ra RET-VAL (Return Value_ giá trị trả về) của khối SFC, giá trị trả về có dạng dữ liệu dạng số nguyên (INT). Khi có lỗi thì bit BR=0 và giá trị trả về RET-VAL là một số âm (bit tín hiệu là 1). Khi không có lỗi thì bit BR=1 và giá trị trả về RET-VAL là một số dương (bit tín hiệu là 0). Có hai loại lỗi là lỗi chung và lỗi đặt biệt tại ngõ ra RET-VAL. Lỗi chung là lỗi mà tất cả các khối SFC có thể xuất ra và lỗi đặt biệt là lỗi mà các khối SFC có thể xuất ra mà liên quan đến một chức năng đặt biệt nào đó.

Cấu trúc của một mã lỗi chức năng hệ thống định dạng mã thập lục phân (Hexadecimal) như sau:

Các bit từ 0 đến 7 chỉ số sự kiện hoặc nhóm lỗi và lỗi đơn. Các bit từ 8 đến 14 chỉ dang lỗi, nếu x=0 chỉ dạng lỗi là đặt biệt, nếu x>0 chỉ dạng lỗi là lỗi chung.

Bảng giới thiệu những chức năng hệ thống SFC, tên hình thức và chức năng của khối SFC:

SFC	Tên hình thức	Chức năng
SFC0	SET-CLR	Đặt thời gian cho hệ thống
SFC1	READ-CLR	Đọc thời gian của hệ thống
SFC2	SET-RTM	Đặt bộ đo thời gian hoạt động
SFC3	CTRL-RTM	Khởi động, ngừng bộ đo thời gian hoạt động
SFC4	READ-RTM	Đọc giá trị bộ đo thời gian hoạt động
SFC5	GADR-LGC	Hỏi địa chỉ của một kênh
SFC6	RD-SINFO	Đọc thông tin về khối OB khởi động
SFC7	DP-PRAL	Tạo ngắt phần cứng trên khối DP chủ
SFC9	EN-MSG	Cho phép liên lạc giữa các khối, biểu tượng và trạng thái các nhóm với nhau
SFC10	DIS-MSG	Không cho phép liên lạc giữa các khối, biểu tượng và trạng thái các nhóm với nhau
SFC11	DPSYC-FR	Đồng bộ những nhóm của DP tớ
SFC13	DPNRM-DG	Đọc dữ liệu chuẩn đoán của DP tớ
SFC14	DPTD-DAT	Đọc dữ liệu cổng phân bố (DP) tớ chuẩn
SFC15	DPWR-DAT	Ghi dữ liệu cổng phân bố (DP) tớ chuẩn
SFC17	ALARM-SQ	Tạo thông tin nhận biết giữa các khối liên hệ
SFC18	ALARM-S	Tạo thông tin nhận biết cố định giữa các khối liên hệ
SFC19	ALARM-SC	Xem trạng thái nhận biết dữ liệu ngõ vào
SFC20	BLKMOV	Copy dữ liệu trong bảng biến
SFC21	FILL	Thiết lập giá trị ban đầu cho một vùng nhớ
SFC22	CREAT-DB	Tạo khối DB
SFC23	DEL-DB	Xóa khối DB
SFC24	TEST-DB	Kiểm tra khối DB
SFC25	COMPRESS	Nén vùng nhớ của chương trình người sử dụng
SFC26	UPDAT-PI	Cập nhật giá trị đến bộ đệm ngõ vào
SFC27	UPDAT-PO	Cập nhật giá trị đến bộ đệm ngõ ra
SFC28	SET-TINT	Đặt ngắt thời gian của ngày
SFC29	CAN- TINT	Bỏ việc ngắt thời gian của ngày
SFC30	ACT- TINT	Kích hoạt việc ngắt thời gian của ngày
SFC31	QRY- TINT	Hỏi ngắt thời gian của ngày

SFC32	SRT-DINT	Khởi động ngắt thời gian trễ
SFC33	CAN- DINT	Bỏ ngắt thời gian trễ
SFC34	QRY- DINT	Hỏi ngắt thời gian trễ
SFC35	MP-ALM	Tạo ngắt do xử lý nhiều phép tính
SFC36	MSK-FLT	Tạo mặt nạ những lỗi đồng bộ
SFC37	DMSK-FLT	Bỏ mặt nạ những lỗi đồng bộ
SFC38	READ-ERR	Đọc giá trị thanh ghi lỗi
SFC39	DIS-IRT	Bỏ những ngắt mới và những lỗi không đồng bộ
SFC40	EN-IRT	Cho phép những ngắt mới và những lỗi không đồng bộ
SFC41	DIS-AIRT	Làm trễ những ngắt có mức ưu tiên cao hơn và những lỗi không đồng bộ
SFC42	EN-AIRT	Cho phép những ngắt có mức ưu tiên cao hơn và những lỗi không đồng bộ
SFC43	RE-TRIGR	Kích lại chu kỳ giám sát
SFC44	REPL-VAL	Truyền giá trị đến thanh ghi ACCU1

SFC46	STP	Chuyển CPU về chế độ STOP
SFC47	WAIT	Làm trễ quá trình thực thi chương trình của người sử dụng
SFC48	SNC-RTCB	Làm đồng bộ những đồng hồ tớ
SFC49	LGC-GADR	Đọc địa chỉ của modul chỉ định
SFC50	RD-LGADR	Đọc tất cả địa chỉ của một modul
SFC51	RDSYSST	Đọc tất cả hoặc một phần trạng thái của hệ thống
SFC52	WR-USMSG	Ghi dữ liệu chuẩn đoán do người dùng định nghĩa đến bộ đệm chuẩn đoán
SFC54	RD-PARM	Đọc giá trị đã được định nghĩa
SFC55	WR- PARM	Ghi giá trị động
SFC56	WR-DPARM	Ghi giá trị mặt định
SFC57	PARM-MOD	Đánh dấu thông tin đến một modul
SFC58	WR-REC	Ghi vào bộ ghi dữ liệu
SFC59	RD-REC	Đọc giá trị bộ ghi dữ liệu
SFC60	GD-SND	Gởi dữ liệu cục bộ (GD)
SFC61	GD-RCV	Lấy một phần dữ liệu của khối dữ liệu cục bộ (GD)
SFC62	CONTROL	Đọc trạng thái của SFB truyền thông cục bộ
SFC63	AB-CALL	Đồng bộ mã của khối
SFC64	TIME-TCK	Đọc thời gian hệ thống
SFC65	X-SENT	Gởi dữ liệu đến bộ giao tiếp bên ngoài trạm S7 cục bộ

SFC66	X-RCV	Nhận dữ liệu từ bộ giao tiếp bên ngoài trạm S7 cục bộ
SFC67	X-GET	Đọc dữ liệu từ bộ giao tiếp bên ngoài trạm S7 cục bộ
SFC68	X-PUT	Ghi dữ liệu đến bộ giao tiếp bên ngoài trạm S7 cục bộ
SFC69	X-ABORT	Ngưng một kết nối truyền thông đang tồn tại bên ngoài trạm S7
SFC72	I-GET	Đọc dữ liệu từ bộ giao tiếp bên trong trạm S7 cục bộ
SFC73	I-PUT	Ghi dữ liệu đến bộ giao tiếp bên trong trạm S7 cục bộ
SFC74	I-ABORT	Ngưng một kết nối truyền thông đang tồn tại bên trong trạm S7
SFC79	SET	Đặt dữ liệu một vùng ngõ ra
SFC80	RSET	Xóa dữ liệu một vùng ngõ ra
SFC81	UBLKMOV	Di chuyển khối mà không thể ngắt
SFC90	H-CTRL	Điều khiển hoạt động trong hệ thống H

Chức năng hệ thống SFC63 chỉ tồn tại trong CPU 614.

Bảng giới thiệu những khối chức năng hệ thống SFB, tên hình thức và chức năng của khối SFB:

SFB	Tên Hình Thức	Chức Năng
SFB0	CTU	Đếm lên
SFB1	CTD	Đếm xuống
SFB2	CTUD	Đếm lên/xuống
SFB3	TP	Tạo một xung
SFB4	TON	Tạo một xung với thời gian trễ cạnh lên

SFB5	TOF	Tạo một xung với thời gian trễ cạnh xuống
SFB8	USEND	Không chỉ định gởi dữ liệu
SFB9	URCV	Không chỉ định nhận dữ liệu
SFB12	BSENT	Gởi đoạn dữ liệu
SFB13	BRCV	Nhận đoạn dữ liệu
SFB14	GET	Đọc dữ liệu từ bộ điều khiển từ xa của CPU
SFB15	PUT	Ghi dữ liệu đến bộ điều khiển từ xa của CPU
SFB16	PRINT	Gởi dữ liệu đến máy in
SFB19	START	Khởi động lại hoàn toàn bằng bộ điều khiển từ xa
SFB20	STOP	Chuyển về chế độ STOP từ bộ điều khiển từ xa
SFB21	RESUME	Khởi động lại bằng bộ điều khiển từ xa
SFB22	STATUS	Hỏi tín hiệu của bộ điều khiển từ xa

SFB23	USTATUS	Nhận tín hiệu của bộ điều khiển từ xa
SFB29	HS-COUNT	Bộ đếm (có tốc độ cao, các chức năng được tích hợp)
SFB30	FREQ-MES	Đo tần số
SFB32	DRUM	Sắp xếp trật tự
SFB33	ALARM	Tạo thông tin của những khối liên hệ có hiển thị nhận biết
SFB34	ALARM-8	Tạo thông tin của những khối liên hệ không cho 8 tín hiệu
SFB35	ALARM-8P	Tạo thông tin của những khối liên hệ cho 8 tín hiệu
SFB36	NOTIFY	Tạo thông tin của những khối liên hệ không hiển thị nhận biết
SFB37	AR-SENT	Gởi dữ liệu kích hoạt
SFB38	HSC-A-B	Đếm A/B
SFB39	POS	Kích cạnh lên
SFB41	CONT-C	Điều khiển tuần tự
SFB42	CONT-S	Điều khiển theo bước
SFB43	PULSEGEN	Tạo xung

Những khối SFB29 và SFB30 chỉ tồn tại trên CPU 312 IFM và CPU 314 IFM. Những khối SFB38, SFB39, SFB41, SFB42 và SFB43 chỉ tồn tại trên CPU 314 IFM.

II. NHỮNG CHỨC NĂNG HỆ THỐNG PHỤC VỤ VIỆC COPY DỮ LIỆU

2.1. COPY DỮ LIỆU DÙNG SFC20

SFC20 dùng để copy dữ liệu tại một vùng bộ nhớ (dữ liệu nguồn) đến một vùng bộ nhớ khác (dữ liệu đích).

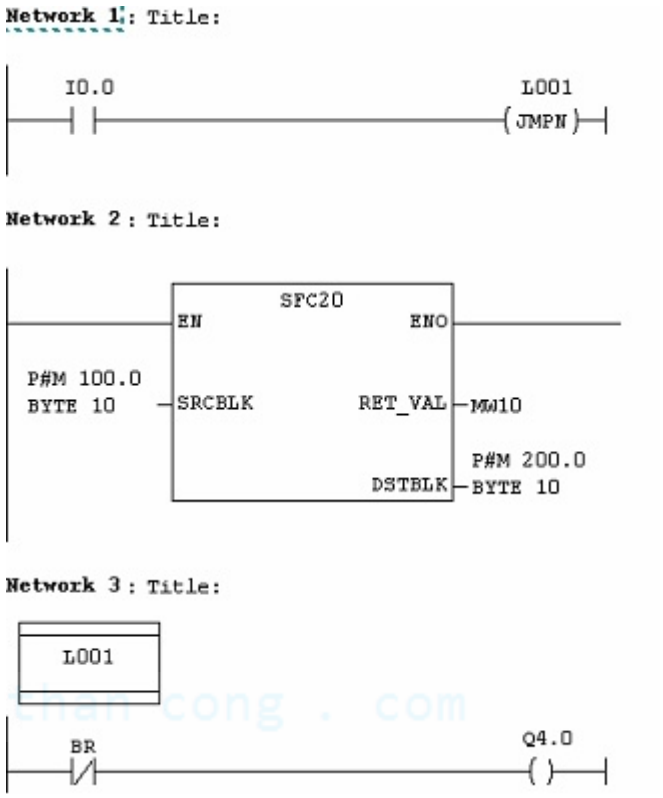
SFC20 có thể copy dữ liệu từ tất cả các vùng nhớ ngoại trừ vùng nhớ trong các khối FB, SFB, FC, SFC, OB, SDB, bộ đếm, timer, vùng nhớ của thiết bị ngoại vi.

SFC20 copy dữ liệu theo thứ tự từ địa chỉ thấp nhất đến địa chỉ cao nhất.

Bảng các tham biến vào ra của hàm SFC20 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
SRCBLK	Ngõ vào	ANY	I,Q,M, D,L	Vùng nhớ cần copy
RET-VAL	Ngõ ra	INT	I,Q,M, D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
DSTBLK	Ngõ ra	ANY	I,Q,M, D,L	Vùng nhớ nơi dữ liệu cần copy đến

Ví dụ: Chương trình copy một vùng dữ liệu sang vùng dữ liệu khác



Chương trình sẽ copy 10 byte từ byte nhớ có địa chỉ 100 đến 10 byte từ byte nhớ có địa chỉ 200. I0.0 dùng để cho phép SFC20 làm việc, nếu I0.0=1 thì SFC20 được xử lý, nếu I0.0=0 thì lệnh nhảy sẽ nhảy đến nhãn L001 và đọc giá trị bit nhớ BR đến ngõ ra Q4.0. Q4.0 để nhận biết SFC0 làm việc có lỗi hay không. Khi SFC20 làm việc thì nó sẽ ghi dữ liệu có địa chỉ từ byte nhớ 100 đến ngõ ra có địa chỉ từ byte nhớ 200. Nếu trong quá trình SFC20 làm việc mà có lỗi thì mã lỗi được lưu trong byte nhớ MW10 và bit nhớ BR=0.

2.2. COPY DỮ LIỆU KHÔNG THỂ NGẮT DÙNG SFC81

SFC81 dùng để copy dữ liệu tại một vùng bộ nhớ (dữ liệu nguồn) đến một vùng bộ nhớ khác (dữ liệu đích). Trong quá trình copy thì không được ngắt bởi các SFC khác.

SFC81 có thể copy dữ liệu từ tất cả các vùng nhớ ngoại trừ vùng nhớ trong các khối FB, SFB, FC, SFC, OB, SDB, bộ đếm, timer, vùng nhớ của thiết bị ngoại vi, những khối dữ liệu không được liên kết.

Chiều dài dữ liệu có thể copy là 512 byte.

Bảng các tham biến vào ra của hàm SFC81 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
SRCBLK	Ngõ vào	ANY	I,Q,M, D,L	Vùng nhớ cần copy
RET-VAL	Ngõ ra	INT	I,Q,M, D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
DSTBLK	Ngõ ra	ANY	I,Q,M, D,L	Vùng nhớ nơi dữ liệu cần copy đến

2.3. THIẾT LẬP GIÁ TRỊ BAN ĐẦU TRONG MỘT VÙNG NHỚ SỬ DỤNG SFC21

SFC21 dùng để tạo dữ liệu ban đầu tại một vùng bộ nhớ (dữ liệu đích) với dữ liệu chứa trong một vùng bộ nhớ khác (dữ liệu nguồn). SFC21 copy vào vùng dữ liệu đích cho tới khi vùng nhớ này đầy.

SFC21 có thể copy dữ liệu từ tất cả các vùng nhớ ngoại trừ vùng nhớ trong các khối FB, SFB, FC, SFC, OB, SDB, bộ đếm, timer, vùng nhớ của thiết bị ngoại vi.

SFC21 copy dữ liệu theo thứ tự từ địa chỉ thấp nhất đến địa chỉ cao nhất.

Bảng các tham biến vào ra của hàm SFC21 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
BVAL	Ngõ vào	ANY	I,Q,M, D,L	Vùng nhớ chứa dữ liệu sẽ được thiết lập giá trị ban đầu cho vùng nhớ đích
RET-VAL	Ngõ ra	INT	I,Q,M, D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
BLK	Ngõ ra	ANY	I,Q,M, D,L	Vùng nhớ nơi cần thiết lập giá trị ban đầu

2.4. TẠO KHỐI DỮ LIỆU DÙNG SFC22

SFC22 dùng để tạo khối dữ liệu mà không chứa dữ liệu ban đầu. SFC22 tạo khối DB có số DB được giới hạn bởi giới hạn dưới (LOW-LIMIT) giới hạn trên (UP-LIMIT). Chúng ta không thể tạo khối DB có số đã tồn tại trang chương trình người sử dụng. Chiều dài của DB phải được tính theo byte. SFC22 có thể được ngắt bởi khối OB có mức ưu tiên ngắt cao hơn. Nếu SFC22 được gọi lại trong khối OB có mức ưu tiên ngắt cao hơn thì sẽ tác động đến mã lỗi là 8091.

Bảng các tham biến vào ra của hàm SFC22 như sau:

Tên biến	Loại	Kiểu dữ	Vùng nhớ	Ý nghĩa

	biến	liệu		
LOW-LIMIT	Ngõ vào	WORD	I,Q,M,D,L, hằng số	Giá trị giới hạn dưới là số nhỏ nhất trong vùng mà có thể xác định cho khối DB
UP-LIMIT	Ngõ vào	WORD	I,Q,M,D,L, hằng số	Giá trị giới hạn trên là số lớn nhất trong vùng mà có thể xác định cho khối DB
COUNT	Ngõ vào	WORD	I,Q,M,D,L, hằng số	Tổng số byte dữ liệu dành cho khối dữ liệu (số này tính theo byte- cực đại là 65534 byte)
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
DB-MUMB ER	Ngõ ra	WORD	I,Q,M,D,L	Số của khối DB, nếu có lỗi sảy ra thì số DB-MUMBER=0

2.5. XÓA KHỐI DỮ LIỆU DÙNG KHỐI SFC23

SFC23 dùng để xóa khối dữ liệu trong bộ nhớ làm việc của CPU. Khối DB được xóa phải không được mở và có mức ưu tiên thấp hơn. CPU sẽ ngừng hoạt động khi SFC23 được gọi. SFC23 có thể được ngắt bởi một khối có mức ưu tiên cao hơn. Nếu khối SFC23 được gọi lại thì một mã nhận biết sẽ được gởi đến ngõ ra RET-VAL.

Bảng các tham biến vào ra của hàm SFC23 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
DB-MUMBER	Ngõ vào	WORD	I,Q,M,D,L, hằng số	Số của khối OB cần được xóa
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

2.6. KIỂM TRA KHỐI DB DÙNG SFC24

Dùng SFC24 để biết được thông tin về khối dữ liệu trong bộ nhớ làm việc của CPU. Lệnh này chỉ dùng để đọc, không thể sửa đổi.

Bảng các tham biến vào ra của hàm SFC24 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
DB-MUMBER	Ngõ vào	WORD	I,Q,M,D,L, hằng số	Số của khối OB cần được kiểm tra
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
DB-LENGTH	Ngõ ra	WORD	I,Q,M,D,L	Số byte dữ liệu được lựa chọn chưa trong khối DB
WRITE-PROT	Ngõ ra	BOOL	I,Q,M,D,L	Thông tin về khối DB

2.7. NÉN BỘ NHỚ NGƯỜI SỬ DỤNG DÙNG SFC25

Những lỗ trống có thể xuất hiện trong bộ nhớ chương trình và bộ nhớ làm việc nếu khối dữ liệu được xóa và nạp lại nhiều lần. Những lỗ trống này làm giảm vùng nhớ của CPU.

Dùng SFC25 để nén vùng RAM trong bộ nhớ chương trình và bộ nhớ làm việc. Chức năng nén này giống như khi khởi động ở chế độ RUN. Nếu lệnh này đã được gọi và vẫn còn thực thi thì nếu gọi lại một lần nữa khối SFC25 thì sẽ có một thông báo lỗi xuất hiện. Nếu chiều dài khối dữ liệu lớn hơn 1000 byte thì chức năng nén này không thực hiện được.

Bảng các tham biến vào ra của hàm SFC25 như sau:

Tên biến Loại biến

Kiểu dữ liệu

Vùng nhớ

Ý nghĩa

RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
BUSY	Ngõ ra	BOOL	I,Q,M,D,L	Cho biết chức năng nén sử dụng SFC25 vẫn còn hoạt động (BUSY=1)

DONE	Ngõ ra	BOOL	I,Q,M,D,L	Cho biết chức năng nén sử dụng SFC25 đã hoàn thành (DONE=1)

2.8. CHUYỂN GIÁ TRỊ THAY THẾ VÀO THANH GHI TÍCH LŨY ACCU1 DÙNG SFC44

SFC44 dùng để truyền giá trị đến thanh ghi tích lũy ACCU1 với mức ưu tiên ngắt do lỗi gây ra. Chỉ có thể gọi khối SFC44 trong những OB lỗi đồng bộ (OB121, OB122).

Bảng các tham biến vào ra của hàm SFC25 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
VAL	Ngõ vào	DWORD	I,Q,M,D,L, hằng số	Giá trị cần thay thế
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

III. NHỮNG KHỐI SFC DÙNG CHO VIỆC ĐIỀU KHIỂN QUÁ TRÌNH THỰC THI CHƯƠNG TRÌNH

3.1. KHỞI ĐỘNG LẠI CHU KỲ GIÁM SÁT DÙNG SFC43

SFC43 dùng để khởi động lại chu kỳ giám sát của quá trình thực thi chương trình.

SFC43 không có toán hạng và nó cũng không có thông tin lỗi.

3.2. CHUYỂN CPU VỀ CHẾ ĐỘ STOP DÙNG SFC46

SFC46 dùng để chuyển CPU về chế độ dừng STOP.

SFC46 không có toán hạng và nó cũng không có thông tin lỗi.

3.3. LÀM TRỄ QUÁ TRÌNH THỰC THI CHƯƠNG TRÌNH DÙNG SFC47

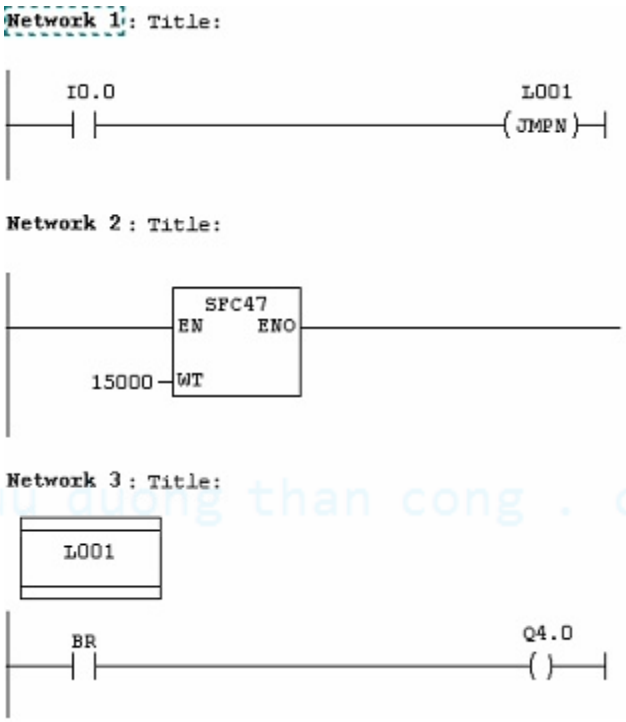
Có thể dùng SFC47 để lập trình thời gian trễ hoặc thời gian chờ trong chương trình người sử dụng. Thời gian trễ có thể lên đến 32767µs. Thời gian trễ nhỏ nhất phụ thuộc vào từng loại CPU.

SFC47 có thể bị ngắt bởi một khối OB có mức ưu tiên ngắt cao hơn. SFC47 không có thông tin lỗi.

Bảng các tham biến vào ra của hàm SFC47 như sau: Ví dụ: Chương trình thực hiện

việc làm trễ 15ms, sau 15ms thì các chương trình khác sẽ làm việc bình thường. Chương trình này sử dụng SFC47.

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
WT	Ngõ vào	INT	I,Q,M,D,L, hằng số	Ngõ vào WT chứa giá trị thời gian trễ có đơn vị là µs.



Nếu I0.0=1 thì SFC47 được xử lý, nếu I0.0=0 thì chương trình sẽ nhảy đến nhãn L001 và thực hiện việc đọc nội dung trong bit BR đến ngõ Q4.0. Khi SFC47 hoạt động thì sau thời gian 15ms=15000*0-6s các chương trình khác sẽ được thực hiện. Nếu có lỗi trong quá trình xử lý thì bit nhớ BR=0.

IV. NHỮNG KHỐI SFC GIÁM SÁT THỜI GIAN CỦA HỆ THỐNG

4.1. ĐẶT THỜI GIAN DÙNG SFC0

SFC0 dùng để đặt thời gian và ngày giờ cho CPU. Khi SFC0 được gọi thì ngày, giờ của hệ thống được hoạt động.

Nếu đồng hồ là đồng hồ chủ thì CPU sẽ khởi động đồng bộ với thời gian khi SFC0 được gọi.

Chúng ta phải tạo dữ liệu DT với FC3 (D-TOD-DT) trước khi truyền giá trị thời

gian từ ngõ vào PDT.

Bảng các tham biến vào ra của hàm SFC0 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
PDT	Ngõ vào	DT	D, L	Ngõ vào dùng để đặt thời gian và ngày. Cách khai báo như sau: DT#năm-tháng-ngày-giờ:phút:giây. Ví dụ: DT#1999-1-15-10:45:30

RET-VAL Ngõ ra INT

I,Q,M,D,L

Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

4.2. ĐỌC THỜI GIAN DÙNG SFC1

SFC1 dùng để đọc ngày giờ hiện hành của đồng hồ hệ thống của CPU. Bảng các tham biến vào ra của hàm SFC1 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
CDT	Ngõ ra	DT	D,L	Ngày giờ hiện hành được xuất ra ngõ CDT

4.3. ĐỒNG BỘ HÓA NHỮNG ĐỒNG HỒ TỚ DÙNG SFC48

Đồng bộ hóa những đồng hồ tớ dùng để truyền ngày, giờ từ những đồng hồ chủ trên những đoạn bus đến tất cả các đồng hồ tớ trên đoạn bus này.

Dùng SFC48 chúng ta sẽ đồng bộ hóa tất cả các đồng hồ tớ trên một đoạn bus. Việc đồng bộ hóa chỉ thành công khi SFC48 được gọi trên một CPU nơi mà được cài đặt.

Bảng các tham biến vào ra của hàm SFC48 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa

RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

4.4. VÍ DỤ

Chương đặt thời gian cho CPU lúc 7 giờ 30 phút ngày 17 tháng 1 năm 2004. Chương trình này có thể được viết trong khối OB1 hoặc một khối FC rồi gọi vào khối OB1. Chương trình này sử dụng SFC0 để đặt thời gian cho CPU và SFC1 để đọc thời gian từ CPU.

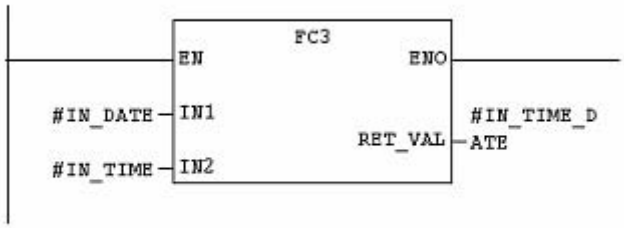
Address	Declaration	Name	Type	Start value	Comment
	in				
	out				
	in_out				
0.0	temp	IN_TIME	TIME_OF_DAY		
4.0	temp	IN_DATE	DATE		
6.0	temp	CUR_RETVAL	INT		
8.0	temp	OUT_TIME	TIME_OF_DAY		
12.0	temp	OUT_DATE	DATE		
14.0	temp	OUT_TIME_DATE	DATE_AND_TIME		
22.0	temp	IN_TIME_DATE	DATE_AND_TIME		
	temp				

Network 1

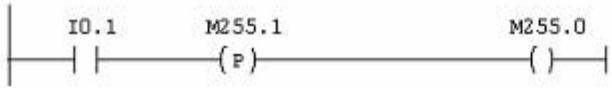
```
      L     D#2004-1-17
      AN    I     0.0
      JC    N1
      L     MW    0
N1:   T     #IN_DATE
      L     TOD#7:30:0.0
      AN    I     0.0
      JC    N2
      L     MD    2

N2:   T     #IN_TIME
```

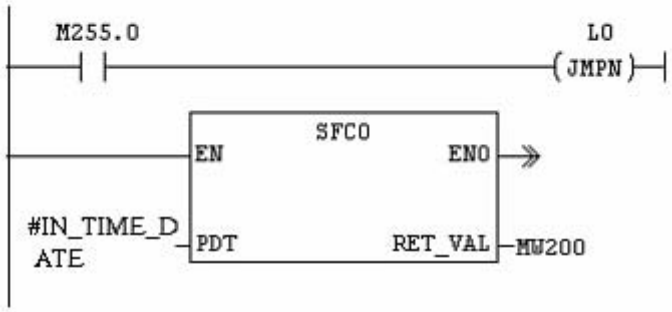
Network 2 : Title:



Network 3 : Title:

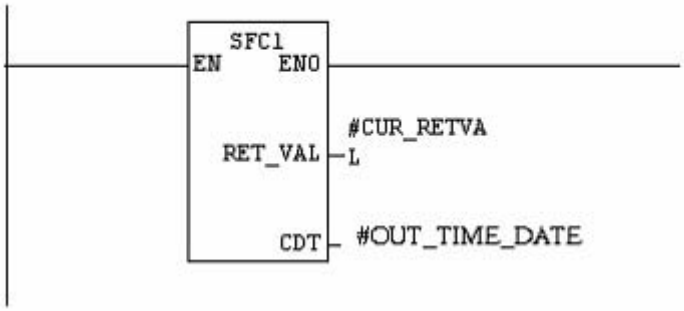


Network 4 : Title:

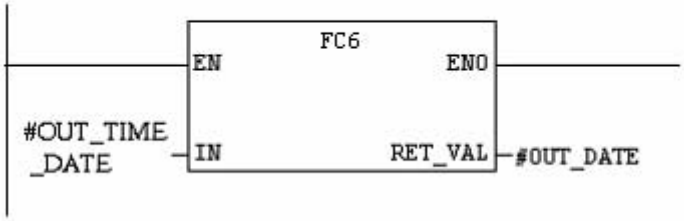


Network 5 : Title:

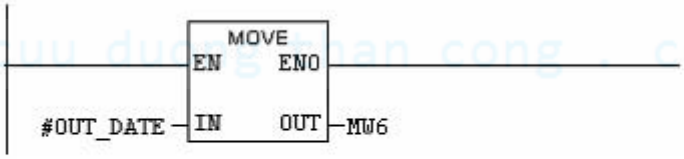
Network 6 : Title:



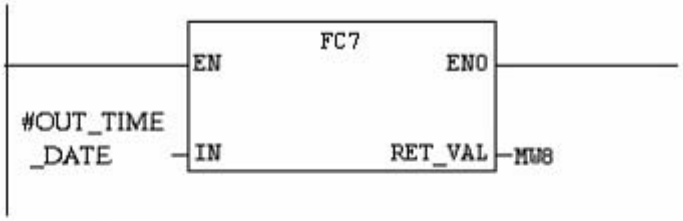
Network 7 : Title:



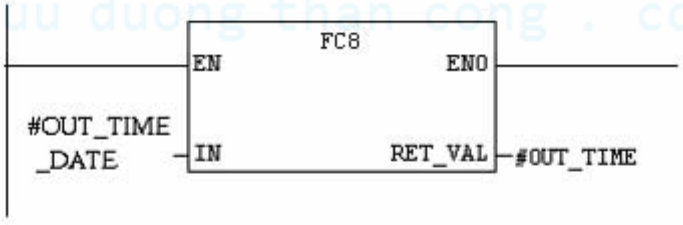
Network 8 : Title:



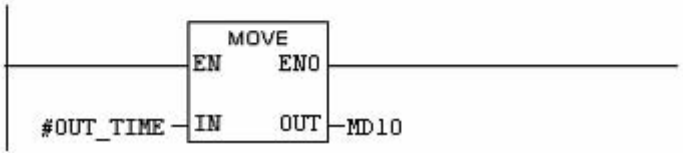
Network 9 : Title:



Network 10 : Title:



Network 11 : Title:



Chương trình này có thể đặt thời gian và ngày tháng cố định hoặc có thể thay đổi nhờ vào ngõ vào I0.0. I0.0=0 thì thời gian và ngày tháng được đặt cố định, I0.0=1 thì thời gian và ngày tháng có thể thay đổi bằng cách thay đổi giá trị ở ngõ vào MW0 dùng để thay đổi ngày tháng và MD2 dùng để thay đổi thời gian. Giá trị thời gian phải nằm trong giới hạn từ ngày 1 tháng 1 năm 1990 đến ngày 31 tháng 12 năm 2089.

FC3 dùng để chuyển đổi thời gian và ngày tháng dạng DATE và TIME_OF_DAY (TOD) sang dạng DATE And TIME (DT) tại ngõ ra RET_VAL.

Nếu I0.1 chuyển từ 0 lên 1 thì SFC0 hoạt động đặt thời gian cho hệ thống. SFC1 để đọc thời gian của hệ thống. FC6 để chuyển đổi dạng DT sang dạng ngày tháng (DATE). FC7 dùng để chuyển đổi dạng DT sang dạng ngày tháng và thời gian. FC8 dùng để chuyển đổi dạng DT sang dạng thời gian (TOD). Dạng dữ liệu thời gian được đưa ra ngõ MD10 bằng lệnh nạp và truyền dữ liệu.

V. NHỮNG SFC ĐO THỜI GIAN HOẠT ĐỘNG CỦA HỆ THỐNG

5.1. GIỚI THIỆU

Chúng ta có thể dùng các SFC2, SFC3, SFC4 để đặt, dừng, đọc việc đo giá trị thời gian hoạt động của CPU.

Khi SFC đo thời gian hoạt động được khởi động thì nó bắt đầu đếm tại giá trị được ghi sau cùng. Nếu chúng ta muốn nó khởi động tại một giá trị đặt trước thì chúng ta phải dùng SFC2. Nếu CPU chuyển sang chế độ dừng hoặc bộ đo thời gian hoạt động dừng thì CPU sẽ ghi nhận giá trị thời gian hiện hành lúc nó bị dừng. Khi khởi động lại CPU thì phải khởi động lại bộ đo thời gian bằng cách dùng SFC3.

Vùng giá trị đo được nằm trong khoảng từ 0 đến 32767 giờ.

5.2. KÍCH HOẠT BỘ ĐO THỜI GIAN DÙNG SFC2

SFC2 dùng để đặt bộ đo thời gian hoạt động của CPU với một giá trị được lựa chọn. Số lượng bộ đo thời gian phụ thuộc vào từng loại CPU. Trong STEP7 có 8 bộ đo thời gian từ 0 đến 7.

Bảng các tham biến vào ra của hàm SFC2 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
NR	Ngõ vào	BYTE	I,Q,M,D,L, hằng số	Ngõ vào chứa số của bộ đo thời gian mà chúng ta muốn sử dụng (từ 0 đến 7)

PV	Ngõ vào	INT	I,Q,M,D,L, hằng số	Ngõ vào chứa giá trị đặt trước cho bộ đo thời gian
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

5.3. KHỞI ĐỘNG- DỪNG BỘ ĐO THỜI GIAN DÙNG SFC3

SFC3 dùng để khởi động- dừng bộ đo thời gian hoạt động của CPU. Bảng các tham biến vào ra của hàm SFC3 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
NR	Ngõ vào	BYTE	I,Q,M,D,L, hằng số	Ngõ vào chứa số của bộ đo thời gian mà chúng ta muốn sử dụng (từ 0 đến 7)
S	Ngõ vào	BOOL	I,Q,M,D,L, hằng số	Ngõ vào S để khởi động-dừng bộ đo thời gian. Khi S=0 thì bộ đếm thời gian dừng, khi S=1 thì bộ đếm thời gian hoạt động
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

5.4. ĐỌC GIÁ TRỊ BỘ ĐO THỜI GIAN DÙNG SFC4

SFC4 dùng để đọc giá trị bộ đo thời gian hoạt động của CPU. SFC4 cung cấp giá trị thời gian hiện hành và trạng thái của bộ đếm như trạng thái dừng hoặc đang đếm. Nếu bộ đếm chạy lâu hơn 32767 giờ thì bộ đếm dừng và có một thông báo lỗi tràn.

Bảng các tham biến vào ra của hàm SFC4 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
NR	Ngõ vào	BYTE	I,Q,M,D,L, hằng số	Ngõ vào chứa số của bộ đo thời gian mà chúng ta muốn sử dụng (từ 0 đến 7)
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
CQ	Ngõ ra	BOOL	I,Q,M,D,L	Ngõ ra cho biết bộ đếm đang hoạt động hay ngừng. CQ=0 cho biết bộ đếm dừng, CQ=1 cho biết bộ đếm đang hoạt động
CV	Ngõ ra	INT	I,Q,M,D,L	Ngõ ra chỉ giá trị hiện hành của bộ

				đếm

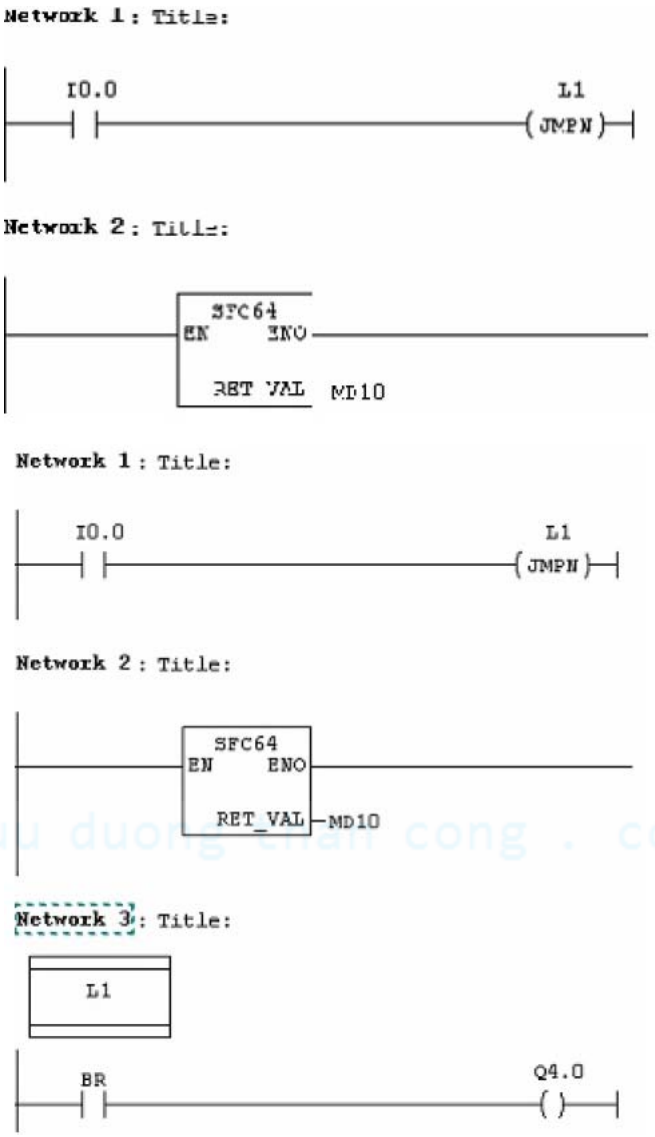
5.5. ĐỌC THỜI GIAN CỦA HỆ THỐNG DÙNG SFC64

SFC64 dùng để đọc thời gian hệ thống của CPU. SFC64 thực chất là bộ đếm thời gian nằm trong khoảng từ 0 đến 231-1=2147483647ms. Khi hệ thống đếm tràn thì nó sẽ quay lại đếm từ 0. Đối với S7-300 thì độ phân giải của bộ đo thời gian hệ thống là 10ms. SFC64 không có thông báo lỗi.

Bảng các tham biến vào ra của hàm SFC64 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
RET-VAL	Ngõ ra	TIME	I,Q,M,D,L	Ngõ ra chứa thời gian của hệ thống có giá trị nằm trong khoảng từ 0 đến (231-1)ms

Ví dụ: Đọc thời gian của hệ thống dùng SFC64

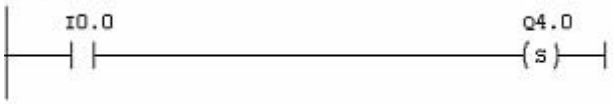


Nếu I0.0=1 thì lệnh SFC64 được kích hoạt, thời gian của hệ thống được gởi ra ngoài ngõ MD10. Nếu I0.0=0 thì lệnh nhảy JMPN sẽ nhảy đến nhãn L1 và đọc nội dung của bit BR trong thanh ghi trạng thái ra ngõ Q4.0.

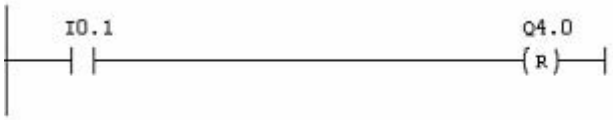
5.6. VÍ DỤ

Chương trình xác định thời gian của động cơ. Ngõ ra Q4.0 dùng để diều khiển động cơ. I0.0 để khởi động động cơ, I0.1 để dừng động cơ, I0.2 để kích hoạt bộ đo thời gian, Q4.1 để hiển thị trạng thái của bộ đo thời gian, IW8 là ngõ vào chứa giá trị đặt trước cho bộ đếm, ngõ ra QW16 chứa giá trị thời gian hiện hành của bộ đo. Chương trình đượcviết trong khối OB1.

Network 1

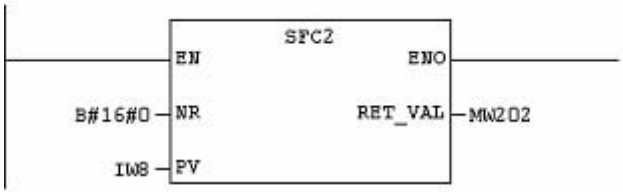


Network 2 : Title:

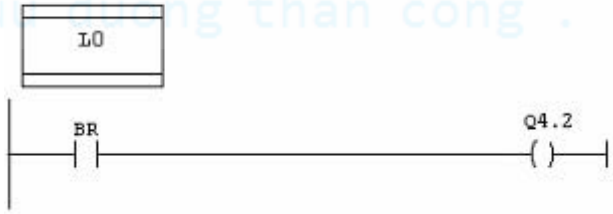


Network 3 : Title:

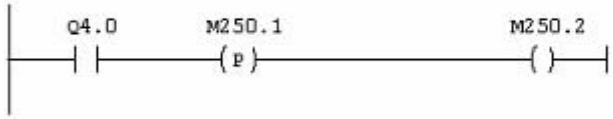
Network 4 : Title:



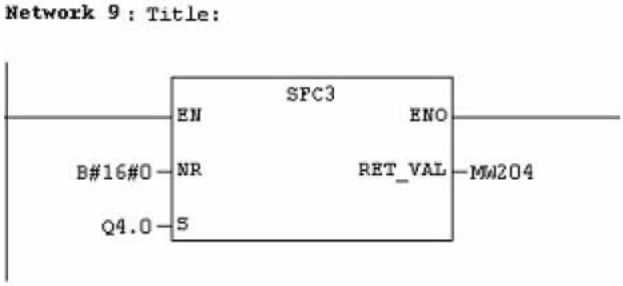
Network 5 : Title:

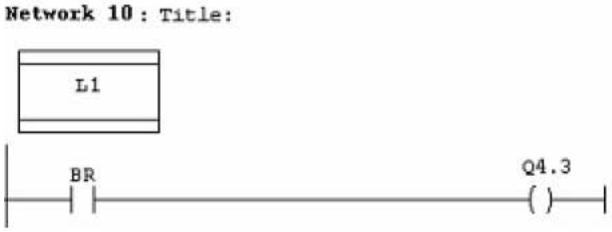


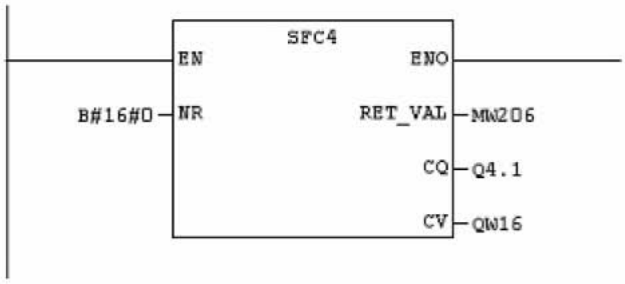
Network 6 : Title:



Network 7 : Title:







Nếu I0.2=1 thì SFC hoạt động có tác dụng kích bộ đến hoạt động, ngõ vào NR chứa số của bộ đếm. Khi đó nếu động cơ hoạt động (Q4.0=1) thì SFC3 cho phép bộ đếm làm việc. SFC4 đọc thời gian hoạt động của động cơ ra ngõ CV đến ngõ ra QW16. Khi động cơ ngừng hoạt động thì SFC3 có tác dụng làm bộ đếm ngưng đếm.

VI. NHỮNG SFC PHỤC VỤ NGẮT THỜI ĐIỂM

6.1. GỚI THIỆU CHUNG

Khi tín hiệu báo ngắt thời điểm phát ra, hệ thống sẽ gọi khối OB10 để xử lý. Trước khi gọi khối OB10 thì phải có các điều kiện sau:

- Xác định thời điểm báo tín hiệu ngắt bằng phần mềm Simatic Manager hoặc nhờ hàm hệ thống SFC28.
- Hủy bỏ tín hiệu ngắt đang tích cực nhờ hàm SFC29.
- Tích cực ngắt nhờ phần mềm Simatic Manager hoặc nhờ hàm hệ thống SFC30.
- Xem trạng thái tín hiệu ngắt nhờ hàm SFC31.

Trước khi sử dụng các hàm hệ thống trên thì khối OB10 phải có trong bộ nhớ của CPU. Nếu không có khối OB10 trong quá trình thực hiện thì hệ điều hành sẽ gọi khối OB85 để xử lý thiếu khối OB và nếu không có khối OB85 thì CPU sẽ chuyển sang chế độ STOP.

Khi đã tích cực tín hiệu ngắt tại thời điểm cho trước mà vì một lý do nào đó ta lại chỉnh đồng hồ thời gian của CPU thì có thể sẽ gây ra nguy cơ đồng hồ được chỉnh tiến hoặc lùi qua thời điểm phát tín hiệu ngắt. Trong trường hợp này hệ điều hành sẽ gọi khối OB80 để thực hiện chương trình xử lý lỗi không đồng bộ về thời gian và nếu cũng không tìm thấy khối OB80 thì CPU sẽ chuyển sang chế độ STOP.

1. **6.2. XÁC ĐỊNH THỜI ĐIỂM NGẮT DÙNG SFC28**
2. **6.3. HỦY BỎ TÍN HIỆU NGẮT DÙNG SFC29**

Hàm SFC28 dùng để khai báo thời điểm ngắt cũng như tần suất phát tín hiệu ngắt (một lần, nhiều lần theo từng phút, giờ, ngày, tuần, tháng, năm).

Bảng các tham biến vào ra của hàm SFC28 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
OB-NR	Ngõ vào	INT	I,Q,M,D,L, hằng số	Tên khối OB chứa chương trình xử lý ngắt (ví dụ OB10)
SDT	Ngõ vào	DT	D,L,hằng số	Thời điểm bắt đầu phát tín hiệu ngắt. Cách khai báo dạng: DT#năm-ngày-tháng-giờ:phút:giây:mili giây

PERIOD	Ngõ vào	WORD	I,Q,M,D,L, hằng số	Tần suất phát tín hiệu ngắt: W#16#0000: Một lần W#16#0201: Mỗi phút một lần W#16#0401: Mỗi giờ một lần W#16#1001: Mỗi ngày một lần W#16#1202: Mỗi tuần một lần W#16#1401: Mỗi tháng một lần W#16#1801: Mỗi năm một lần
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

Hàm SFC29 dùng để hủy bỏ tín hiệu ngắt thời điểm. Để tích cực ngắt trở lại ta phải gọi lại hàm SFC28 để đặt lại thời điểm phát tín hiệu ngắt. Bảng các tham biến vào ra của hàm SFC29 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
OB-NR	Ngõ vào	INT	I,Q,M,D,L, hằng số	Tên khối OB chứa chương trình xử lý ngắt (ví dụ OB10)
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

6.4. KÍCH HOẠT TÍN HIỆU NGẮT DÙNG SFC30

Hàm SFC30 dùng để kích hoạt tín hiệu ngắt thời điểm tại thời điểm định trước đã được khai báo. Mặt dù tín hiệu ngắt được khai báo bởi hàm SFC28 nhưng chỉ thực sự được tích cực khi sử dụng thêm hàm SFC30. Hàm SFC30 không thể bị ngắt.

Bảng các tham biến vào ra của hàm SFC30 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
OB-NR	Ngõ vào	INT	I,Q,M,D,L, hằng số	Tên khối OB chứa chương trình xử lý ngắt (ví dụ OB10)
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

6.5. KIỂM TRA TRẠNG THÁI TÍN HIỆU NGẮT DÙNG SFC31

Hàm SFC31 dùng để xem, kiểm tra trạng thái tín hiệu ngắt thời điểm của các khối xử lý tín hiệu báo ngắt theo thời điểm định trước (OB10). Kết quả được gởi ra ngõ

STATUS với ý nghĩa của các bit được biểu diễn trong bảng dưới đây. Hàm SFC31 không bị ngắt trong quá trình làm việc.

Bit	Giá trị	Giải thích
0	0	Tín hiệu được tích cực bởi hệ điều hành
1	0	Chấp nhận tín hiệu ngắt mới
2	1	Tín hiệu ngắt đã được tích cực
3	-	-
4	1	Đã tìm thấy OB
5	0	Tín hiệu ngắt đã bị hủy

Bảng các tham biến vào ra của hàm SFC31 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
OB-NR	Ngõ vào	INT	I,Q,M,D,L, hằng số	Tên khối OB cần xác định trạng thái (ví dụ OB10)
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
STATUS	Ngõ ra	WORD	I,Q,M,D,L	Chứa trạng thái ngắt thời điểm

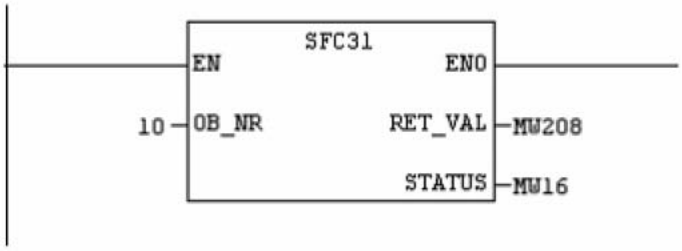
VÍ DỤ

Chương trình điều khiển ngõ ra chỉ ở mức 1 trong thời gian từ 5 giờ sáng thứ 2 đến 20 giờ chiều thứ 6. Chương trình được viết trong cá khối FC12, OB1, OB10. Khối OB1 gọi khối FC12 và khối OB10 để phục vụ ngắt thời điểm.
Chương trình trong khối FC1 như sau: Bảng khai báo các biến:

Address	Declaration	Name	Type	Start value	Comment
	in				
	out				
	in_out				
0.0	temp	IN_TIME	TIME_OF_DAY		
4.0	temp	IN_DATE	DATE		
6.0	temp	OUT_TIME_DATE	DATE_AND_TIME		
14.0	temp	OK_FLAG	BOOL		

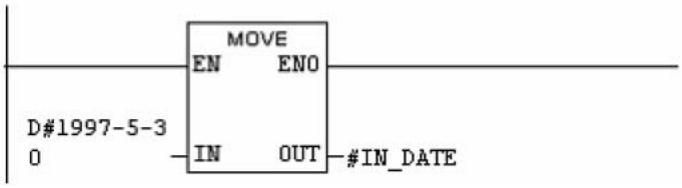
FC12 : Title:

Network 1 : Title:

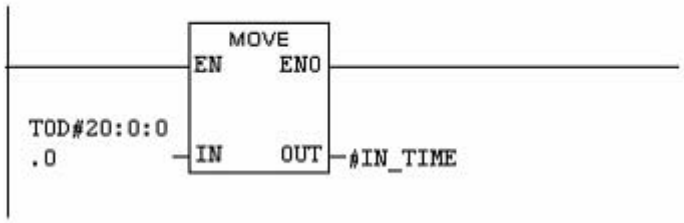


Network 2 : Title:

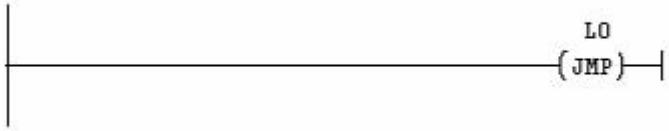
Network 3 : Title:



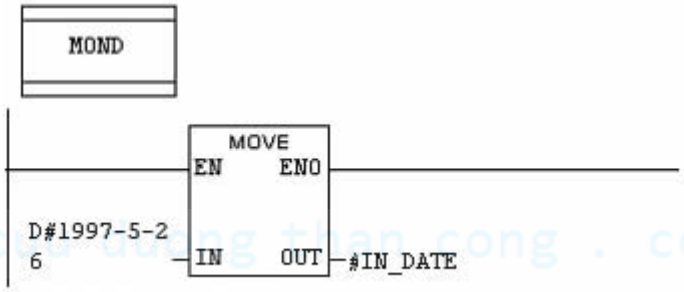
Network 4 : Title:



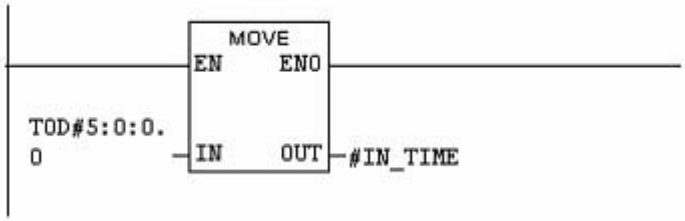
Network 5 : Title:



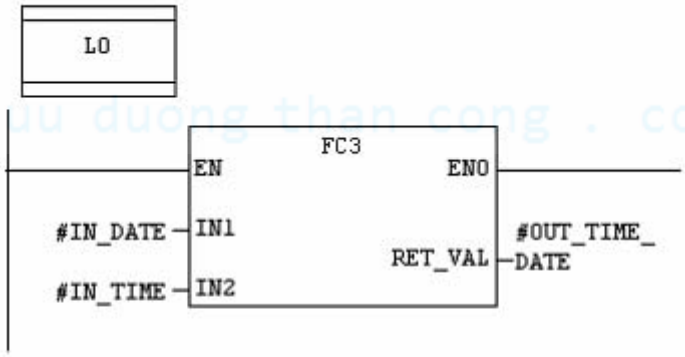
Network 6 : Title:



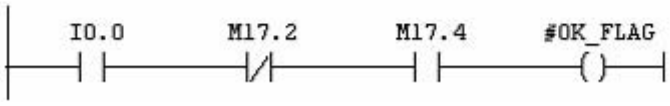
Network 7 : Title:



Network 8 : Title:

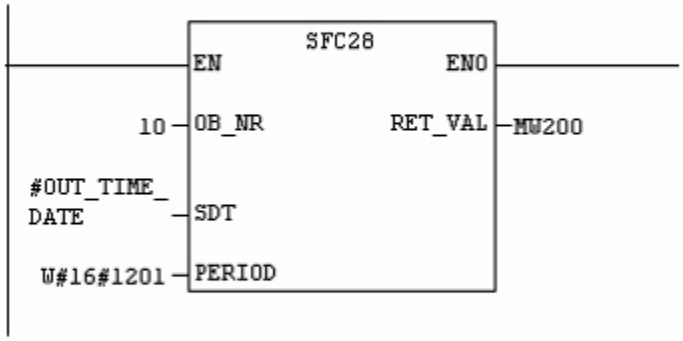


Network 9 : Title:



Network 10 : Title:

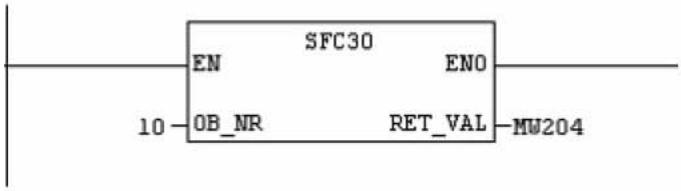
Network 11: Title:



Network 12 : Title:

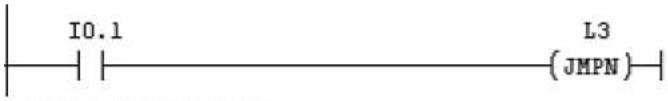
Network 13 : Title:

Network 14 : Title:

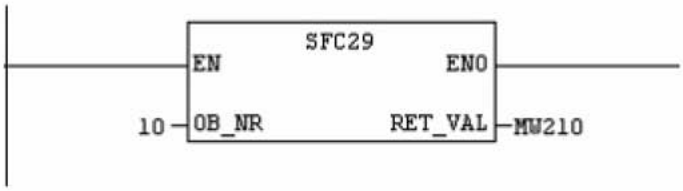


Network 15 : Title:

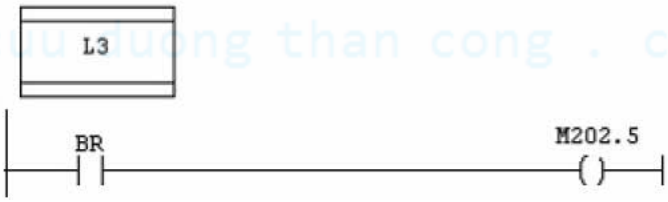
Network 16 : Title:



Network 17 : Title:



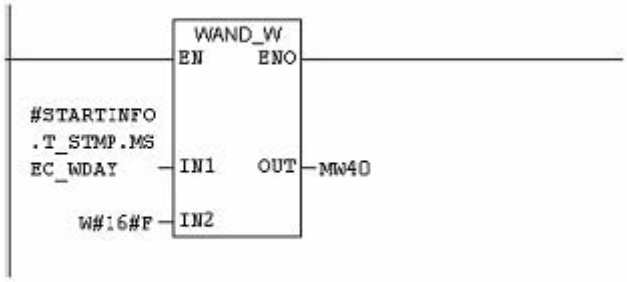
Network 18 : Title:



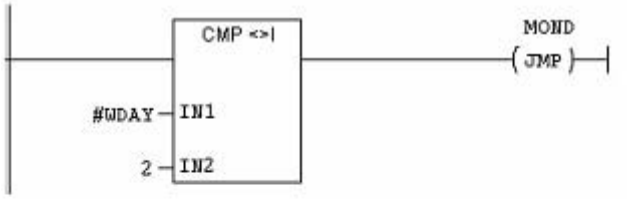
Chương trình trong khối OB10 như sau:

OB10 : "Time of Day Interrupt"

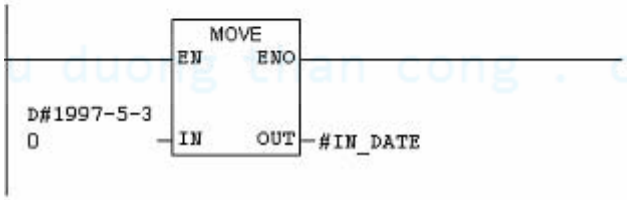
Network 1 : Title:



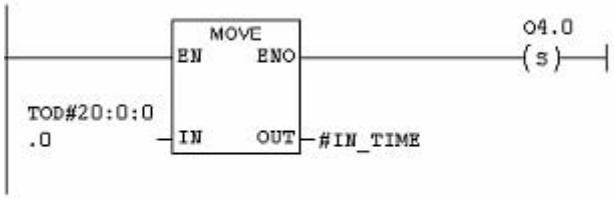
Network 2 : Title:



Network 3 : Title:

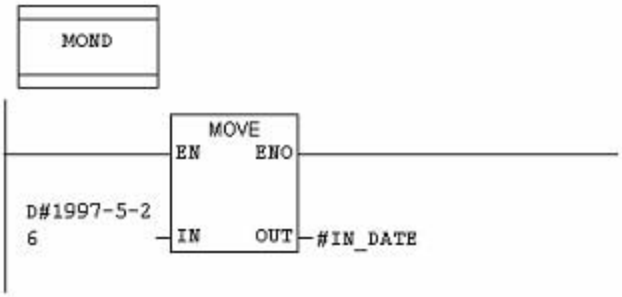


Network 4 : Title:

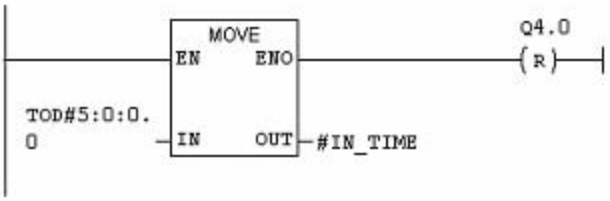


Network 5 : Title:

Network 6 : Title:



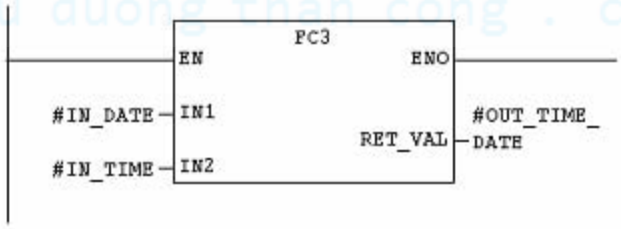
Network 7 : Title:



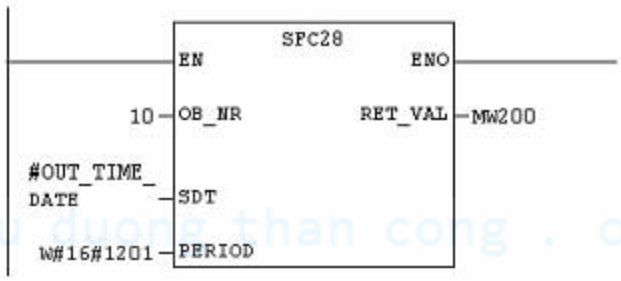
Network 8 : Title:

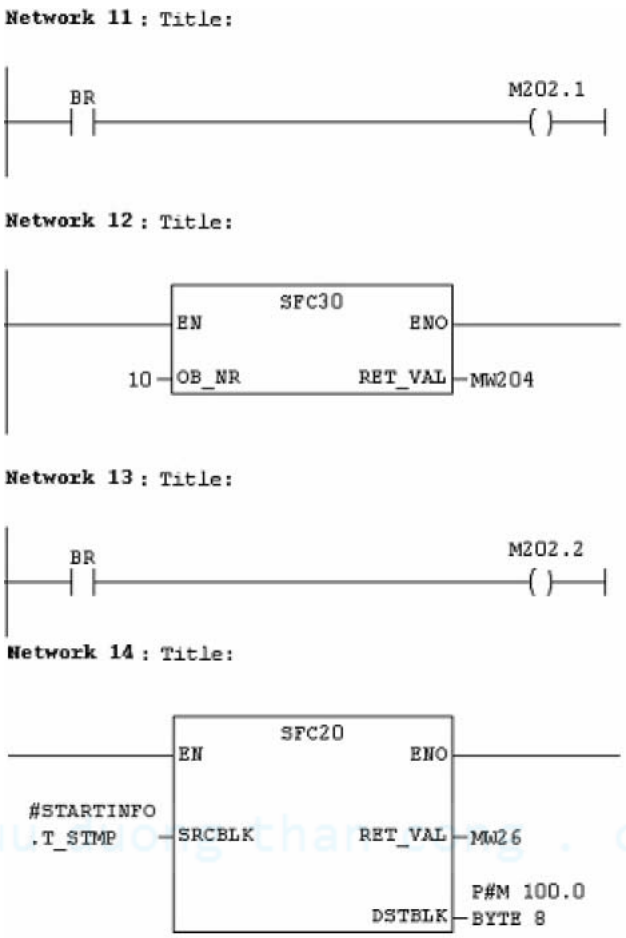
```
L1:   NOP   0
```

Network 9 : Title:



Network 10 : Title:





I0.0 là ngõ vào cho phép ngắt thời điểm, I0.1 là ngõ vào hủy bỏ ngắt thời điểm, Q4.0 là ngõ ra được đặt và xóa bởi OB10, MW16 chỉ trạng thái ngõ ra SFC37. Chương trình trong FC12 có tác dụng đọc trạng thái ngắt thời điểm, đặt , kích hoạt và hủy bỏ ngắt thời điểm. Chương trình trong khối OB10 có tác dụng tạo ra ngắt thời điểm, khối FC3 trong OB10 có tác dụng chuyển đổi thời gian và ngày tháng riêng rẽ thành thời gian và ngày tháng chung. Trong khối OB10 thì SFC28 dùng để đặt ngắt thời điểm, SFC30 kích hoạt ngắt thời điểm, SFC20 dùng để truyền thông tin của khối OB10 đến vùng nhớ 100 đến 107. Chương trình trong khối OB1 chỉ dùng để gọi khối FC12 làm việc theo chu kỳ.

VII. NHỮNG SFC PHỤC VỤ NGẮT THỜI GIAN TRỄ

7.1. GIỚI THIỆU

Có 3 chức năng hệ thống để qui định chế độ làm việc cho những khối ngắt thời gian trễ (OB20) đó là: SFC32 để khởi động ngắt thời gian trễ, SFC33 để bỏ ngắt thời gian trễ, SFC34 để xem kết quả ngắt thời gian trễ. Sau khi gọi SFC32 thì hệ điều hành sẽ bị ngắt sau một khoảng thời gian đã được thiết lập, khi đó khối OB xác định

ngắt thời gian trễ được gọi (OB20).

Trước khi hệ điều hành gọi khối OB 20 thì phải thỏa các điều kiện sau:

- OB20 phải được gọi bởi SFC32.
- OB20 không bị hủy bỏ bởi phần mền Simatic Manager.
- OB20 phải tồn tại trong CPU.

Nếu SFC32 đã được gọi và ta gọi lại SFC32 thì ngắt thời gian trễ khởi động trở lại.

Nếu SFC32 đã được gọi và OB20 không tồn tại thì hệ điều hành tạo ra một lỗi có mức ưu tiên cao hơn tức là gọi khối OB85, nếu khối OB85 không tồn tại thì CPU chuyển sang chế độ STOP.

Nếu SFC32 đã được gọi và ngắt đã được khởi động trong khối OB khởi động và thời gian trễ đã được thiết lập trước khi CPU chuyển sang chế độ RUN thì khối OB20 sẽ được gọi sau khi CPU chuyển sang chế độ RUN.

Nếu thời gian trễ đã được thiết lập và khối OB phục vụ ngắt thời gian trễ vẫn còn đang thực thi thì hệ điều hành tạo ra một lỗi thời gian (gọi khối OB80). Nếu khối OB80 không tồn tại thì CPU chuyển sang chế độ STOP.

Trong quá trình CPU khởi động thì tất cả các khối SFC phục vụ ngắt thời gian trễ bị xoá. Muốn gọi khối OB20 thì SFC ngắt thời gian trễ phải được thiết lập và CPU ở chế độ RUN. Nếu SFC ngắt thời gian trễ đã được thiết lập và CPU không ở chế độ RUN, OB20 sẽ được gọi khi CPU chuyển sang chế độ RUN trước khi lệnh đầu tiên của OB1 được thực thi.

7.2. KHỞI ĐỘNG NGẮT THỜI GIAN TRỄ DÙNG SFC32

SFC32 dùng để khởi động ngắt thời gian trễ, được gọi vào trong khối OB20, OB20 chỉ có thể được kích hoạt bằng việc gọi chức năng hệ thống SFC32. SFC32 được kích hoạt bằng tín hiệu đặt tại ngõ SIGN.

Bảng các tham biến vào ra của hàm SFC32 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
OB-NR	Ngõ vào	INT	I,Q,M,D,L, hằng số	Tên khối OB cần khởi động sau thời gian trễ (OB20)
DTINE	Ngõ vào	TIME	I,Q,M,D,L, hằng số	Thời gian trễ cần đặt từ 1 đến 60000ms

SIGN	Ngõ vào	WORD	I,Q,M,D,L, hằng số	Tín hiệu người sử dụng xác định để khởi động OB ngắt thời gian trễ
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

7.3. XEM TRẠNG THÁI NGẮT THỜI GIAN TRỄ DÙNG SFC34

SFC34 dùng để xem trạng thái ngắt thời gian trễ. Những ngắt thời gian trễ được quản lý trong OB20.

Bảng các tham biến vào ra của hàm SFC34 như sau: Bảng chứa trạng thái ngắt thời gian trễ:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
OB-NR	Ngõ vào	INT	I,Q,M,D,L, hằng số	Tên khối OB cần xem trạng thái ngắt thời gian trễ (OB20)
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
STATUS	Ngõ ra	WORD	I,Q,M,D,L	Chứa trạng thái ngắt thời gian trễ

Bit	Giá trị	Giải thích
0	0	Ngắt thời gian trễ được tích cực bởi hệ điều hành
1	0	Những ngắt mới đã được loại bỏ
2	0	Những ngắt thời gian trễ không được tích cực hoặc thiết lập
3	-	-
4	0	OB ngắt thời gian trễ không được nạp
5	0	Tín hiệu ngắt thời gian trễ đã bị hủy

7.4. HUỶ BỎ NGẮT THỜI GIAN TRỄ DÙNG SFC33

SFC33 dùng để huỷ bỏ ngắt thời gian trễ. Khi đó OB20 không được gọi. Bảng các tham biến vào ra của hàm SFC34 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
OB-NR	Ngõ vào	INT	I,Q,M,D,L, hằng số	Tên khối OB cần huỷ bỏ ngắt thời gian trễ (OB20)

RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

VÍ DỤ

Chương trình làm trễ có các trạng thái như sau: nếu I0.0 được đặt thì ngõ ra Q4.0 sẽ được đặt sau 10 giây, mỗi lần I0.0 được đặt thì thời gian trễ được khởi động trở lại. Nếu I0.1 được đặt trong 10 giây thì khối OB20 không được gọi, Q4.0 không được đặt. Nếu I0.2 được đặt thì Q4.0 được reset lại. Chương trình được viết trong khối OB1 và OB20.

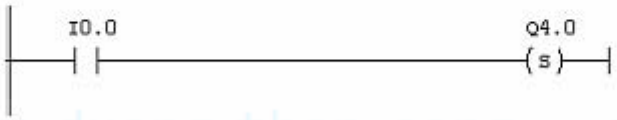
Chương trình trong khối OB20 như sau:

Bảng khai báo biến:

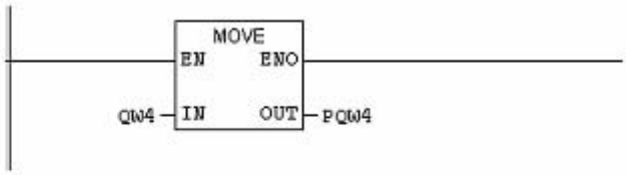
Address	Declaration	Name	Type	Start	Comment
0.0	temp	OB20_EV_CLASS	BYTE		Bits 0-3 = 1 (Coming even
1.0	temp	OB20_STRT_INF	BYTE		16#21 (OB 20 has started)
2.0	temp	OB20_PRIORITY	BYTE		3 (Priority of 1 is lowes
3.0	temp	OB20_OB_NUMBR	BYTE		20 (Organization block 20
4.0	temp	OB20_RESERVED_1	BYTE		Reserved for system
5.0	temp	OB20_RESERVED_2	BYTE		Reserved for system
6.0	temp	OB20_SIGN	WORD		Identifier input (SIGN) a
8.0	temp	OB20_DTIME	TIME		Delay time (DTIME) input
12.0	temp	OB20_DATE_TIME	DATE_AND_TI		Date and time OB20 starte
20.0	temp	STARTINFO	STRUCT		
+0.0	temp	SIGN	WORD		
+2.0	temp	T_STMP	STRUCT		
+0.0	temp	SECONDS	BYTE		
+2.0	temp	MSEC_WDAY	WORD		
=4.0	temp		END_STRUCT		
=6.0	temp		END_STRUCT		

OB20 : "Time Delay Interrupt"

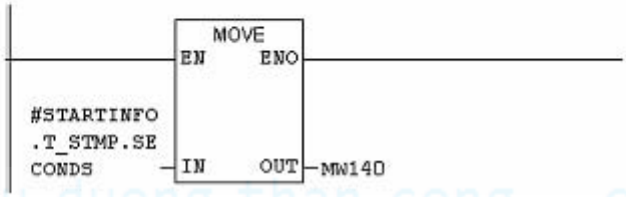
Network 1 : Title:

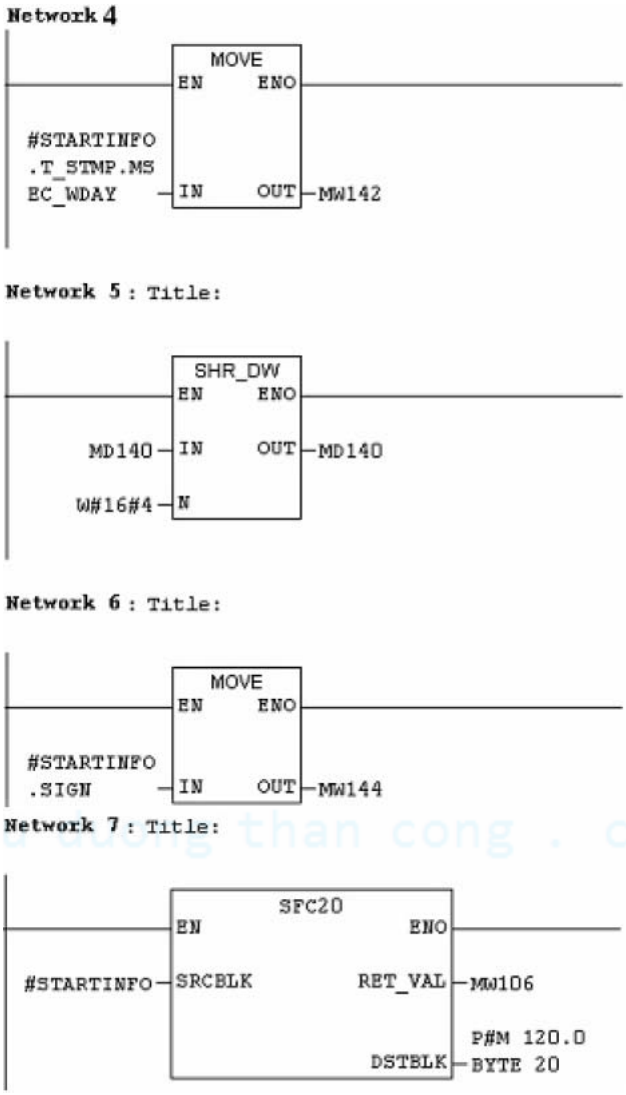


Network 2 : Title:



Network 3 : Title:





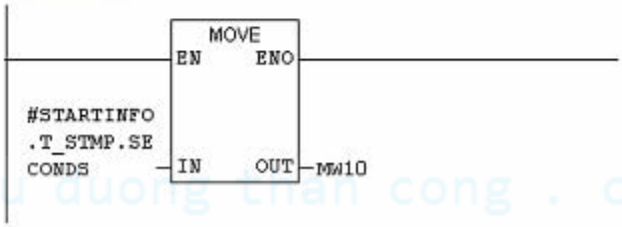
Chương trình trên có tác dụng đặt ngõ ra Q4.0 lên 1, đọc thời gian hiện hành và lưu trữ thông tin khởi động trong vùng nhớ tạm. Lệnh dịch trong netword 5 có tác dụng dịch 4 bit sang phải để thiết lập ngày của tuần và thời gian tính theo mili giây được viết vào MW142 dưới dạng mã BCD. SFC20 có tác dụng copy thông tin khởi động đến vùng nhớ từ MB120 đến MB 139.

Chương trình trong khối OB1 như sau:

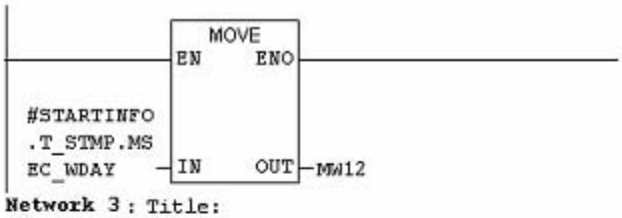
Bảng khai báo biến:

Address	Declara	Name	Type	Start	Comment
0.0	temp	OB1_EV_CLASS	BYTE		Bits 0-3 = 1 (Coming event),
1.0	temp	OB1_SCAN_1	BYTE		1 (Cold restart scan 1 of OB
2.0	temp	OB1_PRIORITY	BYTE		1 (Priority of 1 is lowest)
3.0	temp	OB1_OB_NUMBR	BYTE		1 (Organization block 1, OB1
4.0	temp	OB1_RESERVED_1	BYTE		Reserved for system
5.0	temp	OB1_RESERVED_2	BYTE		Reserved for system
6.0	temp	OB1_PREV_CYCLE	INT		Cycle time of previous OB1 s
8.0	temp	OB1_MIN_CYCLE	INT		Minimum cycle time of OB1 (m
10.0	temp	OB1_MAX_CYCLE	INT		Maximum cycle time of OB1 (m
12.0	temp	OB1_DATE_TIME	DATE_AND_TI		Date and time OB1 started
20.0	temp	STARTINFO	STRUCT		
+0.0	temp	T_STMP	STRUCT		
+0.0	temp	SECONDS	BYTE		
+2.0	temp	MSEC_WDAY	WORD		
=4.0	temp		END_STRUCT		

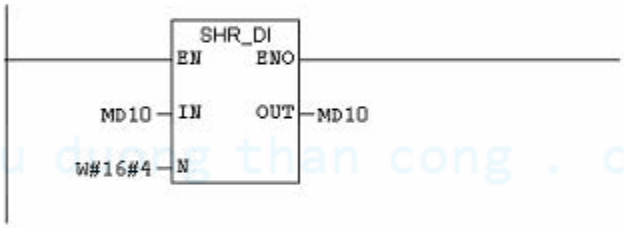
Network 1: Title:



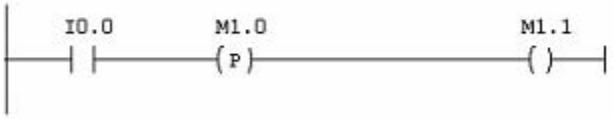
Network 2 : Title:



Network 3 : Title:

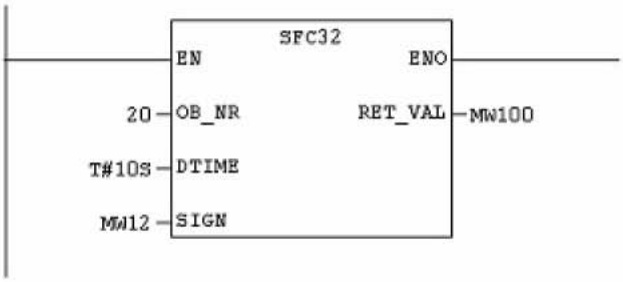


Network 4 : Title:



Network 5 : Title:

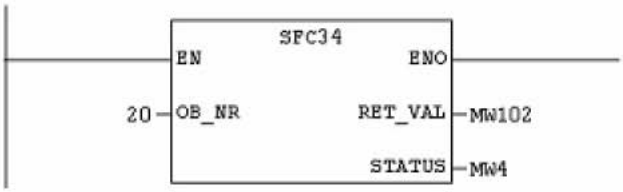
Network 6 : Title:



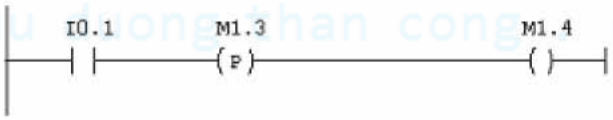
Network 7 : Title:

```
LO:   NOP   0
```

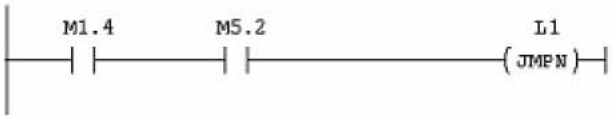
Network 8 : Title:



Network 9 : Title:



Network 10 : Title:



Network 11 : Title:

Network 12 : Title:

```
L1:   NOP   0
```

Network 13 : Title:

OB1 có tác dụng đọc thời gian hiện hành, khởi động, hủy bỏ ngắt thời gian trễ và reset ngõ ra Q4.0 phụ thuộc vào trạng thái I0.2. Lệnh dịch phải ở network 3 có tác

dụng thiết lập ngày của tuần và ghi giá trị giây vào MW12. SFC32 để khởi động ngắt thời gian trễ và gọi khối OB20 để phục vụ ngắt thời gian trễ. SFC34 để xem trạng thái của quá trình ngắt thời gian trễ và hiển thị thông tin lỗi nếu có. SFC33 để hủy bỏ ngắt thời gian trễ mà được thực hiện bởi khối OB20.

VIII. NHỮNG SFC PHỤC VỤ BÁO LỖI ĐỒNG BỘ

8.1. GỚI THIỆU CHUNG

Có 3 chức năng hệ thống SFC phục vụ cho việc xử lý tín hiệu báo lỗi đồng bộ là SFC36 dùng để đặt mặt nạ che ngắt, SFC37 dùng để bỏ mặt nạ che ngắt, SFC38 dùng để đọc nội dung thanh ghi báo lỗi tín hiệu.

Thông thường, khi gặp tín hiệu báo lỗi đồng bộ (lỗi lập trình, lỗi truy nhập modul...), hệ điều hành sẽ kiểm tra tín hiệu đó có bị che hay không: Nếu không bị che, hệ thống sẽ gọi khối OB chứa chương trình xử lý lỗi tương ứng. Khối này do người sử dụng viết tuỳ theo yêu cầu của bài toán điều khiển. Trong trường hợp không tìm thấy khối OB này, hệ điều hành chuyển CPU về chế độ STOP.

Nếu bị che hệ thống không gọi khối OB xử lý lỗi. Mã báo kiểu lỗi được hệ điều hành ghi vào thanh ghi báo lỗi đồng bộ. Nội dung thanh ghi này có thể được đọc nhờ chức năng SFC38.

Có 2 loại lỗi đồng bộ là lỗi lập trình (do OB121 xử lý) và lỗi truy nhập modul (do OB122 xử lý).

8.2. MẶT NẠ CHE NHỮNG LỖI ĐỒNG BỘ DÙNG SFC36

Hàm SFC36 có tác dụng che một số kiểu lỗi đồng bộ. Muốn che kiểu lỗi lập trình thì khai báo tại ngõ vào PRGFLT-SET-MASK, Muốn che kiểu lỗi truy cập modul thì khai báo tại ngõ vào ACCFLT-SET-MASK.

Hàm sẽ có 2 giá trị trả về là PRGFLT- MASKED và ACCFLT- MASKED để thông báo những kiểu lỗi đồng bộ nào đã được che mặt nạ và những kiểu lỗi nào chưa che mặt nạ. Hình thức thông báo của hàm là các bit tương ứng trong thanh ghi của kiểu tín hiệu báo lỗi đồng bộ đã được che mặt nạ sẽ có giá trị là 1.

Hàm SFC36 không bị ngắt.

Bảng các tham biến vào ra của hàm SFC36 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa

PRGFLT-SET -MASK	Ngõ vào	DWORD	I,Q,M,D,L, hằng số	Đánh dấu kiểu lỗi lập trình sẽ được che
ACCFLT-SET-MASK	Ngõ vào	DWORD	I,Q,M,D,L, hằng số	Đánh dấu kiểu lỗi truy cập sẽ được che
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Thông tin lỗi. W#16#0000: Không có lỗi xảy ra. W#16#0001: Có ít nhất một lỗi được che.

PRGFLT-MASKED	Ngõ ra	DWORD	I,Q,M,D,L	Mã báo các kiểu lỗi lập trình đã được che
ACCFLT-MASKED	Ngõ ra	DWORD	I,Q,M,D,L	Mã báo các kiểu lỗi truy cập đã được che

8.3. BỎ MẶT NẠ CHE NHỮNG LỖI ĐỒNG BỘ DÙNG SFC37

Hàm SFC37 có tác dụng bỏ mặt nạ che một số kiểu lỗi đồng bộ. Những kiểu tín hiệu báo lỗi đồng bộ cần được gỡ bỏ mặt nạ che phải được đánh đánh dấu bằng giá trị logic 1 vào bit tương ứng của 2 tham biến hình thức đầu vào của hàm SFC37 là PRGFLT-RESET-MASK và ACCFLT-RESET-MASK. Hàm SFC37 không bị ngắt.

Hàm sẽ có 2 giá trị trả về là PRGFLT- MASKED và ACCFLT- MASKED để thông báo những kiểu lỗi đồng bộ nào có mặt nạ che và những kiểu lỗi nào không có mặt nạ. Hình thức thông báo của hàm là các bit tương ứng trong thanh ghi của kiểu tín hiệu báo lỗi đồng bộ đã được che mặt nạ sẽ có giá trị là 1.

Bảng các tham biến vào ra của hàm SFC37 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
PRGFLT-RESET-MASK	Ngõ vào	DWORD	I,Q,M,D,L, hằng số	Đánh dấu kiểu lỗi lập trình sẽ được bỏ mặt nạ che
ACCFLT-RESET-MASK	Ngõ vào	DWORD	I,Q,M,D,L, hằng số	Đánh dấu kiểu lỗi truy cập sẽ được bỏ mặt nạ che

RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Thông tin lỗi. -W#16#0000: Tất cả các kiểu lỗi đã được bỏ mặt nạ che. -W#16#0001: Có ít nhất một lỗi không được gỡ bỏ mặt nạ che.
PRGFLT-MASKED	Ngõ ra	DWORD	I,Q,M,D,L	Mã báo các kiểu lỗi lập trình đã được che
ACCFLT-MASKED	Ngõ ra	DWORD	I,Q,M,D,L	Mã báo các kiểu lỗi truy cập đã được che

8.4. ĐỌC THANH GHI LỖI ĐỒNG BỘ DÙNG SFC38

Hàm SFC38 có dùng để đọc nội dung của thanh ghi lỗi. Đây là thanh ghi báo sự xuất hiện lỗi đồng bộ. Khi xuất hiện một tín hiệu báo lỗi, cho dù tín hiệu này có được che hay không, tức là hệ thống có xử lý tín hiệu báo lỗi kiểu đó hay không, mã báo sự xuất hiện kiểu lỗi đó vẫn được hệ điều hành ghi vào bit tương ứng trong thanh ghi báo xuất hiện lỗi. Khi sử dụng SFC38 để đọc nội dung thanh ghi báo sự xuất hiện lỗi thì sau khi đọc nội dung của thanh ghi, SFC38 sẽ xoá nội dung của thanh ghi này luôn.

Bảng các tham biến vào ra của hàm SFC38 như sau:

Tên biến

Loại biến

Kiểu dữ liệu

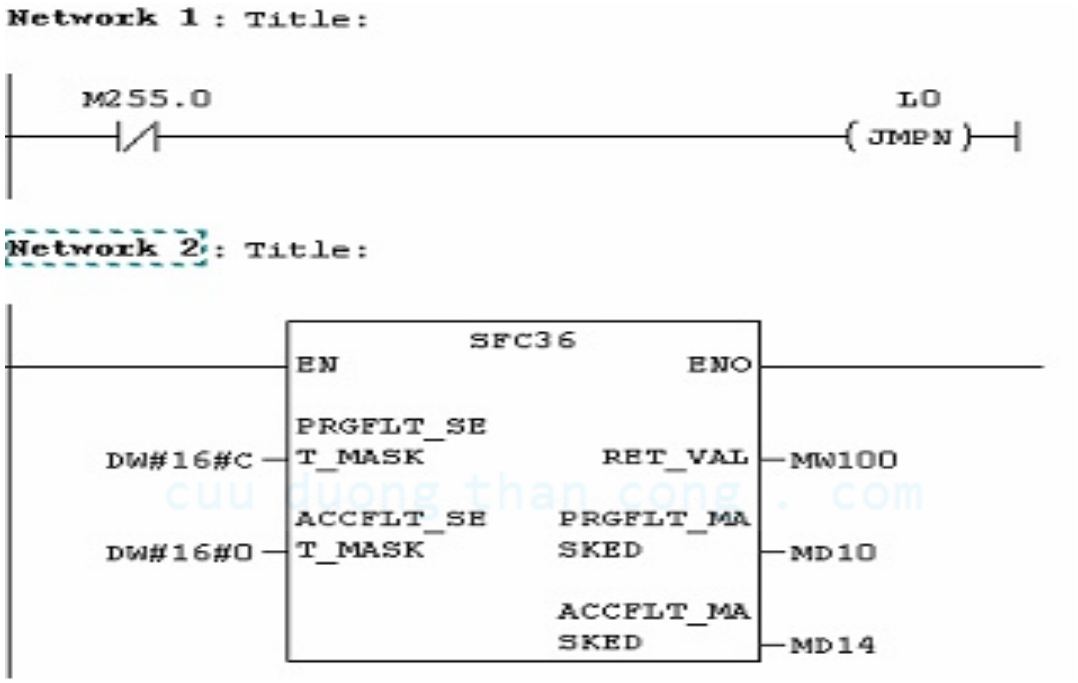
Vùng nhớ

Ý nghĩa

PRGFLT-Q UERY	Ngõ vào	DWORD	I,Q,M,D,L, hằng số	Đánh dấu kiểu lỗi lập trình cần kiểm tra
ACCFLT-QUERY	Ngõ vào	DWORD	I,Q,M,D,L, hằng số	Đánh dấu kiểu lỗi truy cập cần kiểm tra
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Thông tin lỗi. -W#16#0000: Tất cả các kiểu lỗi cần kiểm tra đã bị che. -W#16#0001: Có ít nhất một lỗi cần kiểm tra không bị che.

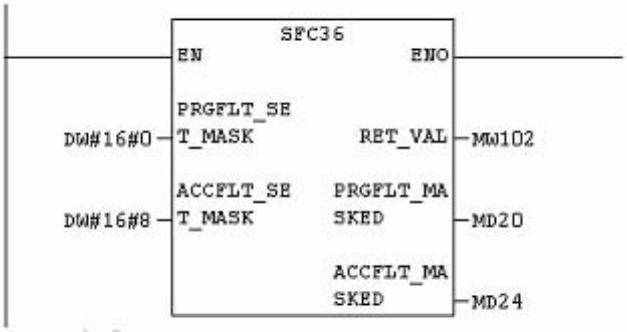
PRGFLT-ES R	Ngõ ra	DWORD	I,Q,M,D,L	Mã báo các kiểu lỗi lập trình đã xuất hiện.
ACCFLT-ESR	Ngõ ra	DWORD	I,Q,M,D,L	Mã báo các kiểu lỗi truy cập đã xuất hiện.

VÍ DỤ



Network 3 : Title:

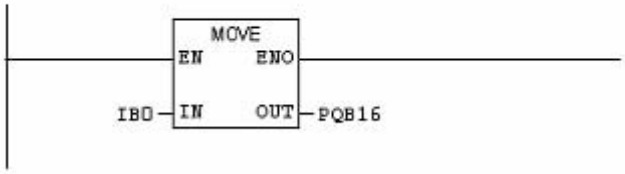
Network 4 : Title:



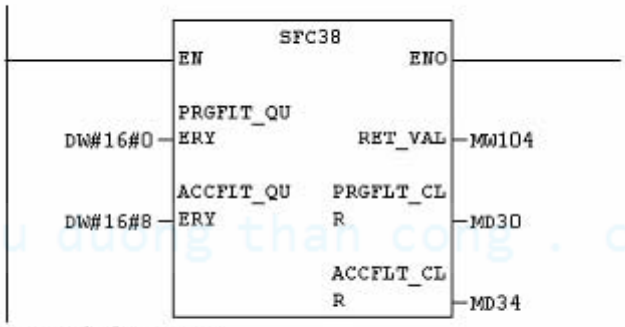
Network 5 : KET THUC CHUONG TRINH NEU M 27.3=1

```
AN    M     27.3
BEC
```

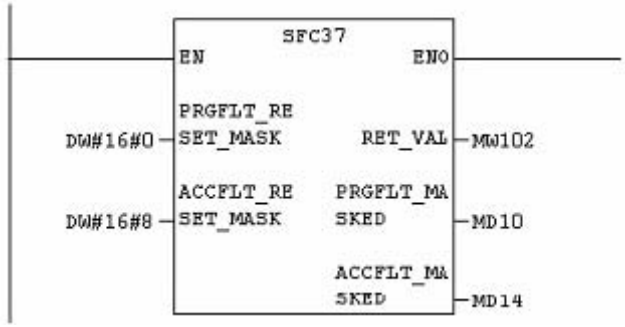
Network 6 : Title:



Network 7 : Title:



Network 8 : Title:



Nếu bit nhớ M225.0=1 thì SFC36 (Netword 2) được gọi khi đó các lỗi lập trình đọc và ghi được che và các lỗi truy cập không được che. Ngõ ra MW100 chứa thông tin lỗi, MD10 chứa mã lỗi lập trình, MD14 chứa mã lỗi truy cập.

Nếu SFC36 (Netword 4) được gọi khi đó các lỗi lập trình không được che và lỗi truy cập ghi được che. Ngõ ra MW102 chứa thông tin lỗi, MD20 chứa mã lỗi lập trình, MD24 chứa mã lỗi truy cập.

SFC38 dùng để đọc các lỗi đồng bộ. Tất cả các lỗi lập trình không được đọc và lỗi truy cập ghi được đọc. Ngõ ra MW104 chứa thông tin lỗi, MD30 chứa mã lỗi lập trình, MD34 chứa mã lỗi truy cập.

SFC37 được gọi có tác dụng bỏ mặt nạ che các lỗi đồng bộ. Khi đó các lỗi lập trình không được bỏ mặt nạ che và lỗi truy cập ghi được bỏ mặt nạ che. Ngõ ra MW100 chứa thông tin lỗi, MD10 chứa mã lỗi lập trình, MD14 chứa mã lỗi truy cập.

IX. NHỮNG SFC PHỤC VỤ BÁO LỖI KHÔNG ĐỒNG BỘ

9.1. GIỚI THIỆU CHUNG

Có 4 chức năng hệ thống SFC phục vụ cho việc xử lý tín hiệu báo lỗi không đồng bộ là SFC39 dùng để đặt mặt nạ che ngắt, SFC40 dùng để bỏ mặt nạ che ngắt, SFC41 dùng để đặt mặt nạ che ngắt tất cả các ngắt có mức ưu tiên cao hơn mức ưu tiên của khối chương trình chứa lệnh gọi hàm, SFC42 dùng để bỏ mặt nạ che ngắt tất cả các ngắt có mức ưu tiên cao hơn mức ưu tiên của khối chương trình chứa lệnh gọi hàm.

Các tín hiệu báo ngắt, báo lỗi được xếp theo từng nhóm như sau:

- Nhóm ngắt tại một thời điểm định trước, chương trình xử lý nằm trong khối OB10.
- Nhóm ngắt theo thời gian trễ, chương trình xử lý nằm trong khối OB20.
- Nhóm ngắt theo chu kỳ, chương trình xử lý nằm trong khối OB35.
- Nhóm ngắt phần cứng, chương trình xử lý nằm trong khối OB40.
- Nhóm ngắt truyền thông, chương trình xử lý nằm trong khối OB50, OB51.
- Nhóm ngắt lỗi không đồng bộ, chương trình xử lý nằm trong khối OB80 đến OB87.
- Nhóm ngắt lỗi đồng bộ, chương trình xử lý nằm trong khối OB121 và OB 122.

9.2. MẶT NẠ CHE NHỮNG NGẮT VÀ LỖI KHÔNG ĐỒNG BỘ DÙNG SFC39

Hàm SFC39 có tác dụng che một tín hiệu ngắt nhất định, một nhóm các tín hiệu ngắt hoặc che tất cả các tín hiệu ngắt và tín hiệu báo lỗi không đồng bộ. Khi một tín hiệu ngắt hay báo lỗi được dương mặt nạ che, hệ thống sẽ không để ý tới tín hiệu đó nữa, tức là không gọi khối OB tương ứng chứa chương trình xử lý tín hiệu ngắt, báo lỗi này cho tới khi mặt nạ che được bỏ đi nhờ hàm SFC40. Khi dùng SFC39 thì tất cả các ngắt sảy ra đều bị xoá.

Bảng các tham biến vào ra của hàm SFC39 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
MODE	Ngõ vào	BYTE	I,Q,M,D,L, hằng số	Loại tín hiệu ngắt, báo lỗi cần được che
OB-NR	Ngõ vào	INT	I,Q,M,D,L, hằng số	Tên của khối OB của tín hiệu ngắt, báo lỗi cần được che
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

9.3. BỎ MẶT NẠ CHE NHỮNG LỖI KHÔNG ĐỒNG BỘ DÙNG SFC40

Hàm SFC40 có tác dụng bỏ mặt nạ che của một tín hiệu ngắt, một nhóm các tín hiệu ngắt, tất cả các tín hiệu ngắt và tín hiệu báo lỗi không đồng bộ.

Khi một tín hiệu ngắt hay báo lỗi không đồng bộ được gỡ bỏ mặt nạ che, hệ thống sẽ gọi khối OB tương ứng chứa chương trình xử lý mỗi khi xuất hiện tín hiệu ngắt, báo lỗi không đồng bộ.

Bảng các tham biến vào ra của hàm SFC40 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
MODE	Ngõ vào	BYTE	I,Q,M,D,L, hằng số	Loại tín hiệu ngắt, báo lỗi được bỏ mặt nạ che
OB-NR	Ngõ vào	INT	I,Q,M,D,L, hằng số	Tên khối OB của tín hiệu ngắt, báo lỗi được bỏ mặt nạ che
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

9.4. MẶT NẠ CHE NHỮNG NGẮT VÀ LỖI KHÔNG ĐỒNG BỘ DÙNG SFC41

SFC41 dùng để đặt mặt nạ che ngắt tất cả các ngắt, tín hiệu không đồng bộ có mức ưu tiên cao hơn mức ưu tiên của khối OB chứa lệnh gọi hàm.

Trong một khối chương trình hàm SFC41 có thể được gọi nhiều lần. Số lần gọi được hệ điều hành đếm và ghi nhận lại dưới dạng tham trị trả về của hàm.

Khi một tín hiệu ngắt hay báo lỗi được dương mặt nạ che, hệ thống sẽ không để ý tới tín hiệu đó nữa, tức là không gọi khối OB tương ứng chứa chương trình xử lý tín hiệu ngắt, báo lỗi này cho tới khi mặt nạ che được bỏ đi nhờ hàm SFC42 hoặc khi khối chương trình chứa lệnh gọi hàm đã được thực hiện xong.

Bảng các tham biến vào ra của hàm SFC41 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Chỉ số lần gọi hàm SFC41 (hoạt động xử lý ngắt chỉ được hoạt động lại khi RET-VAL=0

9.5. BỎ MẶT NẠ CHE NHỮNG LỖI KHÔNG ĐỒNG BỘ DÙNG SFC42

Hàm SFC42 có tác dụng bỏ mặt nạ che của một tín hiệu ngắt, tín hiệu báo lỗi không đồng bộ có mức ưu tiên cao hơn mức ưu tiên của khối OB chứa lệnh gọi hàm.

Mỗi lần gọi khối SFC41 thì phải gọi khối SFC42 nếu muốn bỏ mặt nạ che tất cả các ngắt, tín hiệu không đồng bộ. Hàm SFC42 không bị ngắt.

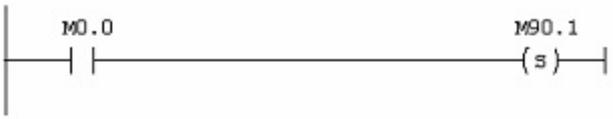
Bảng các tham biến vào ra của hàm SFC42 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	-Số lần hàm SFC42 còn cần phải gọi để bỏ mặt nạ che tất cả các ngắt và tín hiệu không đồng bộ. -Hoặc là giá trị W#16#8080 nếu hàm SFC42 được gọi khi tất cả các tín hiệu ngắt và tín hiệu không đồng bộ đang ở trạng thái tích cực.

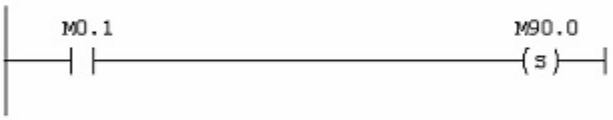
VÍ DỤ

Chương trình che và bỏ mặt nạ che lỗi không đồng bộ dùng SFC39 và SFC40. Chương trình này được viết trong khối OB1.

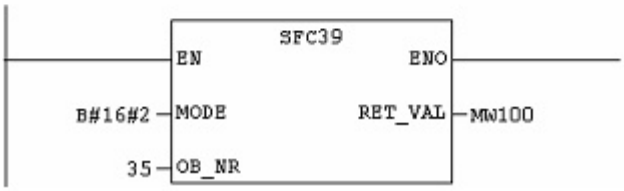
Network 1 : Title:



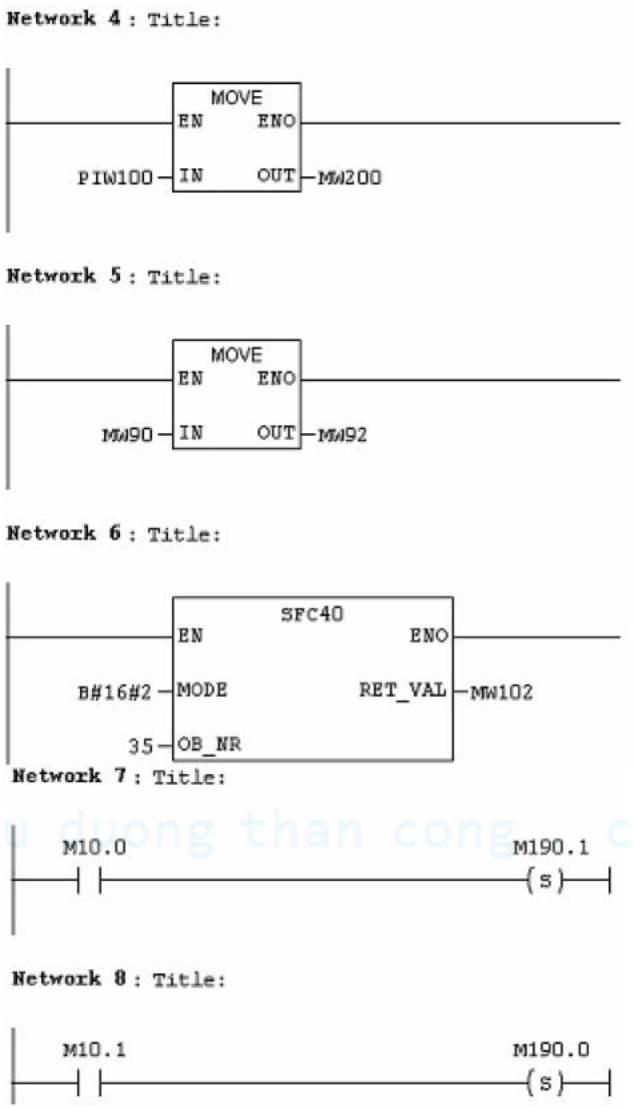
Network 2 : Title:



Network 3 : Title:



cuu duong than cong . com



Network 1 và network 2 là đoạn chương trình dễ bị ngắt nếu có lỗi sảy ra. Các network 3, 4, 5 không bị ngắt bởi các lỗi do khối OB35 báo lỗi gây ra. SFC39 có tác dụng che các lỗi không đồng bộ. Sau khi khối OB40 được gọi thì đoạn chương trình ở network 7, 8 có thể bị ngắt nếu có lỗi.

X. NHỮNG SFC PHỤC VỤ CHUẨN ĐOÁN LỖI HỆ THỐNG

Những CPU đều duy trì dữ liệu trạng thái của bộ điều khiển lập trình. Sử dụng những chức năng chuẩn đoán hệ thống chúng ta có thể đọc ra ngoài những dữ liệu này. Có một vài dữ liệu có thể hiển thị trên bộ điều khiển lập trình nhờ phần mềm STEP 7.

Chúng ta cũng có thể xem những dữ liệu để chuẩn đoán hệ thống bằng cách lập trình sử dụng các chức năng SFC6 và SFC51.

10.1. ĐỌC THÔNG TIN CỦA OB KHỞI ĐỘNG DÙNG SFC6

Dùng SFC6 chúng ta có thể đọc thông tin khởi động về những vấn đề sau:

- OB cuối cùng được gọi mà chưa thực thi xong.
- OB khởi động cuối cùng đã được khởi động.

Trong trường hợp này thì thời gian không được ghi lại. Nếu lệnh gọi khối SFC6 trong khối OB100. OB101 hoặc OB 102 thì những thông tin khởi động giống nhau là giá trị quay về.

Bảng các tham biến vào ra của hàm SFC6 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
TOP-SI	Ngõ ra	STURCT	D,L	Thông tin khởi động của OB hiện hành
START-UP-S I	Ngõ ra	STURCT	D,L	Thông tin khởi động của OB khởi động được khởi động lần cuối.

10.2. ĐỌC TRẠNG THÁI CỦA TOÀN HỆ THỐNG HOẶC MỘT PHẦN HỆ THỐNG DÙNG SFC51

Dùng SFC51 chúng ta có thể đọc trạng thái của toàn hệ thống hoặc một phần hệ thống.

Để đọc trạng thái của hệ thống thì chúng ta phải đặt ngõ vào REQ lên 1, lúc đó SFC51 được gọi. Nếu SFC51 đọc ngay khi có lệnh gọi thì SFC51 báo bận bằng cách gởi đến ngõ ra BUSY giá trị là 1. Nếu BUSY=1 chức năng hệ thống SFC51 đọc chưa hoàn thành.

Bảng các tham biến vào ra của hàm SFC6 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
REQ	Ngõ vào	BOOL	I,Q,M,D,L, hằng số	Khi REQ=1 thì SFC51 được kích hoạt
SSL-ID	Ngõ vào	WORD	I,Q,M,D,L, hằng số	Chứa trạng thái của toàn hệ thống hoặc một phần hệ thống cần được đọc
INDEX	Ngõ vào	WORD	I,Q,M,D,L, hằng số	Loại hoặc số của một đối tượng trong một phần hệ thống

RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
BUSY	Ngõ ra	BOOL	I,Q,M,D,L	BUSY=1 thì quá trình đọc chưa hoàn thành
SSL-HEADE R	Ngõ ra	STRUCT	D,L	Ngõ ra chứa trạng thái của toàn hệ thống hoặc một phần hệ thống

DR	Ngõ ra	ANY	I,Q,M,D,L	Diễn tả vùng chứa trạng thái của toàn hệ thống hoặc một phần hệ thống được đọc nếu: -Nếu chỉ đọc thông tin thì không cần để ý đến ngõ này. -Nếu cần biết số byte thì chúng ta để ý đến ngõ này

10.3. GHI DỮ LIỆU CHUẨN ĐOÁN DO NGƯỜI DÙNG ĐỊNH NGHĨA ĐẾN BỘ ĐỆM CHUẨN ĐOÁN SỬ DỤNG SFC52

SFC 52 cho phép chúng ta ghi dữ liệu chuẩn đoán do người dùng định nghĩa đến bộ đệm chuẩn đoán. Chúng ta cũng có thể gởi dữ liệu chuẩn đoán đến tất cả các trạm. Nếu có lỗi sảy ra thì ngõ ra RET-VAL sẽ cung cấp một thông tin lỗi.

Khi chúng ta gởi dữ liệu chuẩn đoán đến tất cả các trạm thì dữ liệu này sẽ chứa trong bộ đệm gởi và tự động gởi đến các trạm đang làm việc. Chúng ta có thể kiểm tra việc gởi dữ liệu chuẩn đoán này bằng cách dùng SFC51 với giá trị đặt tại các ngõ SSI-ID=W#16#0132 và INDEX=W#16#0005. Dữ liệu chuẩn đoán chỉ có thể gởi đến bộ đệm gởi nến bộ đệm gởi này còn trống. Nếu bộ đệm gởi đầy thì: • Dữ liệu chuẩn đoán không được nhập vào bộ đệm chuẩn đoán. • Ngõ ra RET-VAL cho biết bộ đệm chuẩn đoán đầy (RET-VAL= W#16#8092). Nếu dữ liệu chuẩn đoán được gởi đến các trạm mà các trạm không ở trạng thái

hoạt động thì: • Dữ liệu chuẩn đoán được nhập vào bộ đệm chuẩn đoán. • Ngõ ra RET-VAL cho biết không có trạm nào đang làm việc (RET-VAL= W#16#8091).

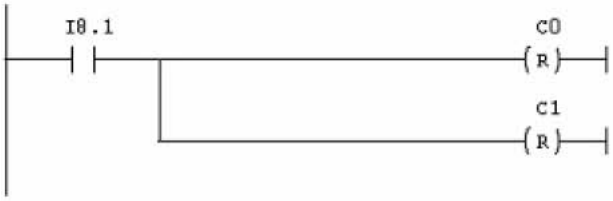
Bảng các tham biến vào ra của hàm SFC52 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
SEND	Ngõ vào	BOOL	I,Q,M,D,L, hằng số	Cho phép gởi dữ liệu chuẩn đoán đến tất cả các trạm
EVENTN	Ngõ vào	WORD	I,Q,M,D,L, hằng số	Dùng để đánh dấu sự kiện ID. Gồm có các loại W#16#8xyz,W#16#9xyz, W#16#Axyz, W#16#Bxyz.
INFO1	Ngõ vào	ANY	I,Q,M,D,L	Chứa thông tin với những loại dữ liệu khác nhau như: WORD, INT, ARRAY[0 đến 1] (dạng chuỗi).
INFO2	Ngõ vào	ANY	I,Q,M,D,L	Chứa thông tin với những loại dữ liệu khác nhau như: DWORD, DINT, REAL, TIME, ARRAY[0 đến 3] (dạng chuỗi).
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi

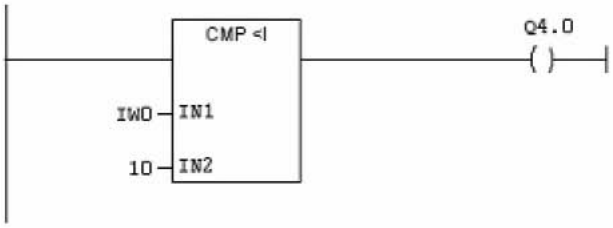
10.4. VÍ DỤ

Chương trình đọc thông tin của hệ thống, so sánh thông tin đó và ghi ra bộ đệm chuẩn đoán hoặc bộ đệm gởi. Chương trình được viết trong khối OB1 như sau:

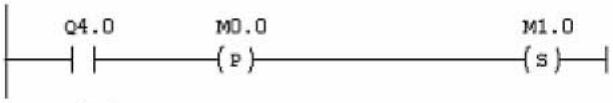
Network 1 : Title:



Network 2 : Title:

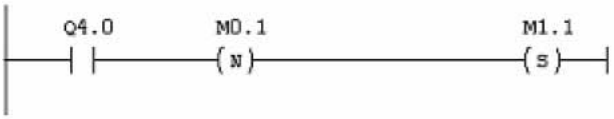


Network 3 : Title:

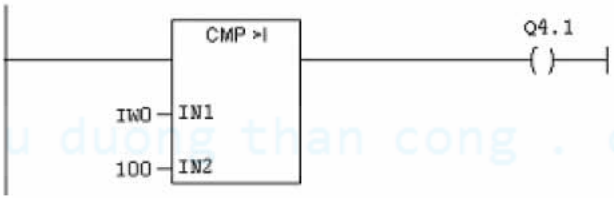


Network 4 : Title:

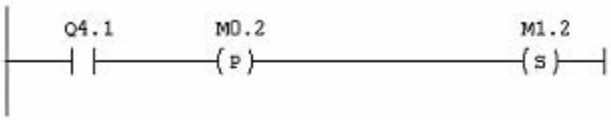
Network 5 : Title:



Network 6 : Title:

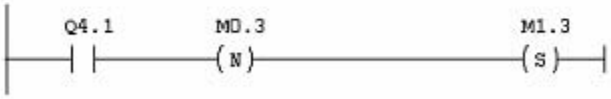


Network 7 : Title:

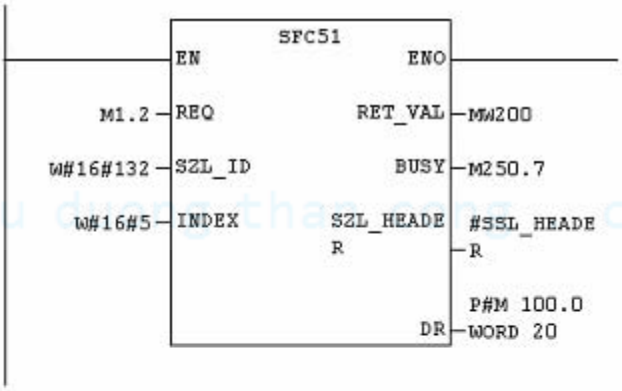


Network 8 : Title:

Network 9 : Title:



Network 10 : Title:

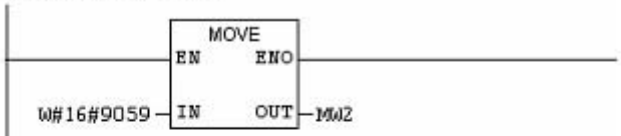


Network 11 : Title:

Network 12 : Title:

```
ON    M    250.6
ON    M    107.0
BEC
```

Network 13 : Title:



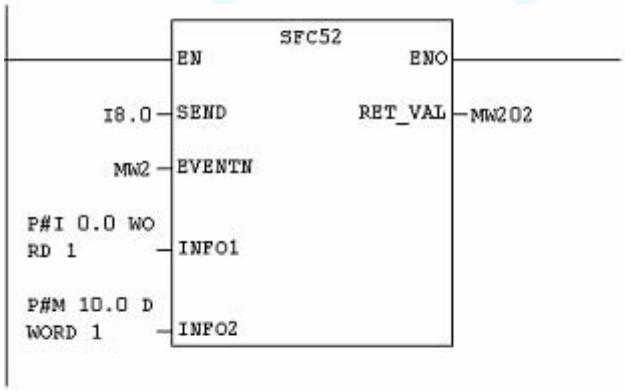
Network 14 : Title:

Network 15 : Title:

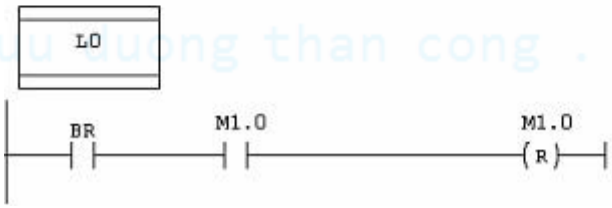
```
LC    C     0
T     MD    10
```

Network 16 : Title:

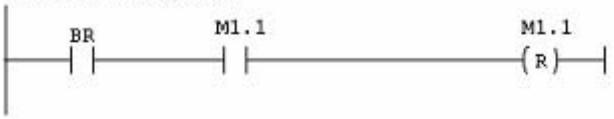
Network 17 : Title:



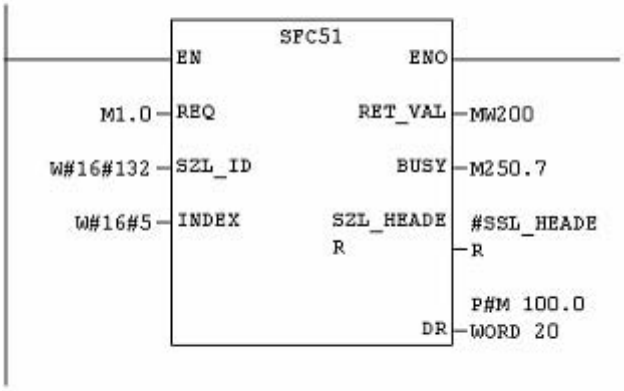
Network 18 : Title:



Network 19 : Title:



Network 20 : Title:

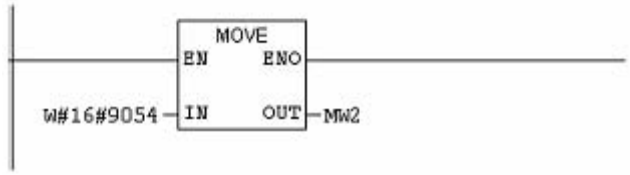


Network 21 : Title:

Network 22 : Title:

```
ON    M    250.6
ON    M    107.0
BEC
```

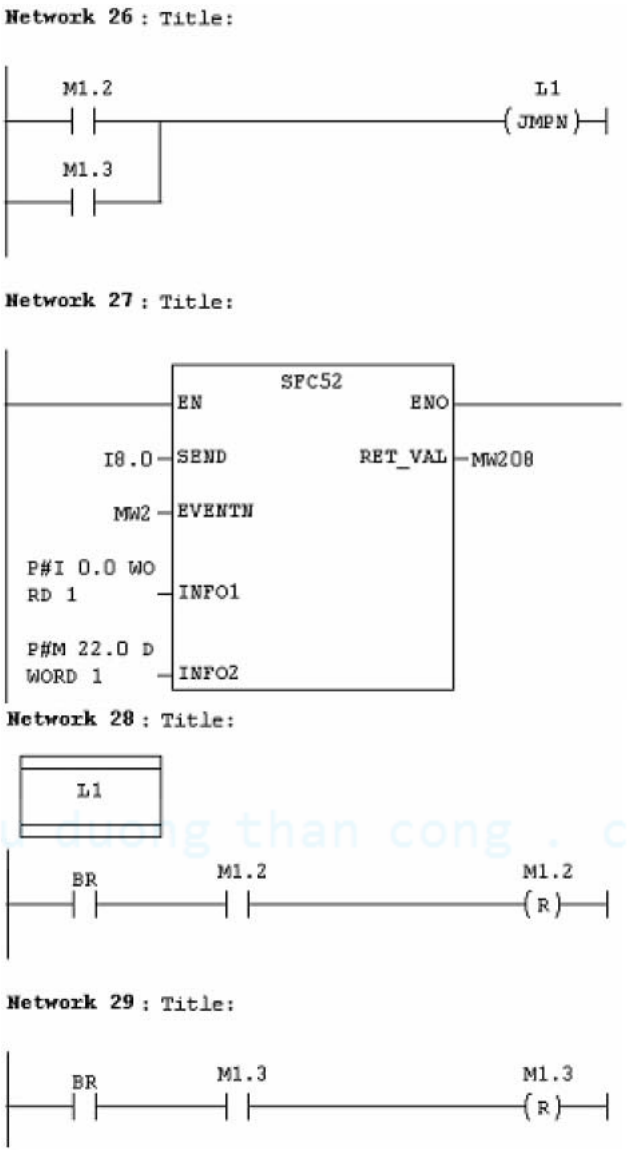
Network 23 : Title:



Network 24 : Title:

Network 25 : Title:

```
LC    C      1
T     MD     22
```



Ngõ vào I8.0 dùng để reset các bộ đếm C0 và C1. Giá trị ngõ vào IW0 được so sánh với 10, nếu IW0 nhỏ hơn 10 thì ngõ ra Q4.0=1 báo giới hạn dưới. Khi Q4.0 chuyển từ 0 lên 1 thì bit nhớ M1.0 lên 1, bộ đếm C0 đếm tăng lên 1. Nếu Q4.0 chuyển từ 1 xuống 0 thì sẽ đặt bit nhớ M1.1 lên 1.

Tương tự lệnh so sánh hơn sẽ so sánh IW0 với 10, nếu IW0 lớn hơn 100 thì ngõ ra Q4.1=1 báo giới hạn trên. Khi Q4.1 chuyển từ 0 lên 1 thì bit nhớ M1.2 lên 1, bộ đếm C1 đếm tăng lên 1. Nếu Q4.1 chuyển từ 1 xuống 0 thì sẽ đặt bit nhớ M1.3 lên 1.

SFC51 dùng để đọc trạng thái của hệ thống và kiểm tra xem bộ đệm gởi có đầy hay chưa. Nếu SFC51 chưa đọc song thì ngõ ra BUSY báo bận (mức logic 1).

Ngõ ra SSL_HEADER của SFC51 được khai báo trong bảng khai báo biến như sau:

CUR_SSL_HEADER : STRUCT

LENTHDR : WORD
N_DR : WORD
: END_STRUCT

Lệnh SFC52 có tác dụng ghi dữ liệu do người dùng định nghĩa ra bộ đệm chuẩn đoán, giá trị cần gửi được đưa ra ngõ RET-VAL.

Lệnh SFC51 ở netword 20 có tác dụng kiểm tra bộ đệm gởi. SFC52 ở netword 27 có tác dụng ghi dữ liệu đến bộ đệm chuẩn đoán, I8.0 dùng để cho phép SFC52 gởi dữ liệu.

XI. NHỮNG SFC VÀ SFB DÙNG ĐỂ CẬP NHẬT DỮ LIỆU VÀO BỘ ĐỆM VÀ XỬ LÝ TÍN HIỆU THEO BIT

11.1. CẬP NHẬT DỮ LIỆU VÀO BỘ ĐỆM NGÕ VÀO DÙNG SFC26

SFC26 dùng để cập nhật dữ liệu vào bộ đệm ngõ vào hoặc một phần bộ đệm ngõ vào. Quá trình cập nhật phải được khai báo bằng phần mềm STEP 7.

Nếu quá trình cập nhật dữ liệu vào bộ đệm ngõ vào tại lúc bắt đầu chu kỳ quét thì sẽ không phụ thuộc vào khối SFC26.

Giá trị khai báo được cập nhật vào bộ đệm ngõ vào phải có địa chỉ không vượt quá vùng giới hạn của bộ đệm ngõ vào.

Bảng các tham biến vào ra của hàm SFC26 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
PART	Ngõ vào	BYTE	I,Q,M,D,L, hằng số	Số của vùng bộ đệm ngõ vào cần cập nhật dữ liệu. PART có giá trị từ 0 đến 15
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
FLADDR	Ngõ ra	WORD	I,Q,M,D,L	Địa chỉ của byte đầu tiên mà gây ra một lỗi nếu có lỗi truy cập

11.2. CẬP NHẬT DỮ LIỆU VÀO BỘ ĐỆM NGÕ RA DÙNG SFC27

SFC27 dùng để cập nhật dữ liệu của bộ đệm ngõ ra hoặc một phần bộ đệm ngõ ra đến modul ngõ ra. Quá trình cập nhật phải được khai báo bằng phần mềm STEP 7.

Nếu quá trình cập nhật dữ liệu của bộ đệm ngõ ra tại cuối chu kỳ quét thì sẽ

không phụ thuộc vào khối SFC27.

Giá trị khai báo được cập nhật từ bộ đệm ngõ ra phải có địa chỉ không vượt quá vùng giới hạn của bộ đệm ngõ ra.

Bảng các tham biến vào ra của hàm SFC27 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
PART	Ngõ vào	BYTE	I,Q,M,D,L, hằng số	Số của vùng bộ đệm ngõ vào cần cập nhật dữ liệu. PART có giá trị từ 0 đến 15
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
FLADDR	Ngõ ra	WORD	I,Q,M,D,L	Địa chỉ của byte đầu tiên mà gây ra một lỗi nếu có lỗi truy cập

11.3. ĐẶT GIÁ TRỊ CỦA MỘT VÙNG DỮ LIỆU NGÕ RA LÊN 1 DÙNG SFC79

Khi gọi SFC79 thì:

- Một vùng bit được chọn tại ngõ vào/ra của thiết bị ngoại vi được đặt.
- Những bit tương ứng trong bộ đệm ngõ ra cũng được đặt bất.

Nếu chức năng master control relay (MCR) không được đặt thì lệnh gọi khối SFC79 không được thực thi. Nếu SFC79 thực thi thì toàn bộ các byte được ghi đến vùng địa chỉ vào/ra. Nếu vùng bit được lựa chọn tại ngõ vào N không nằm ở đầu và cuối một vùng byte cần đặt thì: Những bit nằm ở byte đầu và cuối được truyền đến vùng vào/ra của thiết bị ngoại vi và bộ đệm ngõ vào không được ghi. Những bit nằm trong vùng được chọn được đặt. Bảng các tham biến vào ra của hàm SFC79 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
N	Ngõ vào	INT	I,Q,M,D,L, hằng số	Số của những bit được đặt
RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
SA	Ngõ ra	POINTER	P	Chỉ ra vị trí bit đầu tiên được đặt

11.4. XOÁ GIÁ TRỊ CỦA MỘT VÙNG DỮ LIỆU NGÕ RA DÙNG SFC80

Khi gọi SFC80 thì: • Một vùng bit được chọn tại ngõ vào/ra của thiết bị ngoại vi bị xoá. • Những bit tương ứng trong bộ đệm ngõ ra cũng được xoá bất chấp vùng bộ

đệm ngõ ra có bị xoá hay không. Nếu chức năng master control relay (MCR) không được đặt thì lệnh gọi khối SFC80

không được thực thi. Nếu SFC80 thực thi thì toàn bộ các byte được ghi đến vùng địa chỉ vào/ra. Nếu vùng bit được lựa chọn tại ngõ vào N không nằm ở đầu và cuối một vùng byte

cần xoá thì: • Những bit nằm ở byte

đầu và cuối được truyền đến

vùng vào/ra của thiết bị ngoại vi

và bộ đệm ngõ vào không được

ghi.

• Những bit nằm trong vùng được chọn được đặt. Nếu N=0 thì SFC80 không tác động. Bảng các tham biến vào ra của hàm SFC80 như sau:

Tên biến	Loại biến	Kiểu dữ liệu	Vùng nhớ	Ý nghĩa
N	Ngõ vào	INT	I,Q,M,D,L, hằng số	Số của những bit cần xoá

RET-VAL	Ngõ ra	INT	I,Q,M,D,L	Nếu có lỗi xảy ra thì giá trị trả về chứa mã lỗi
SA	Ngõ ra	POINTER	P	Chỉ ra vị trí bit đầu tiên được xoá

CÁC BÀI TẬP ỨNG DỤNG

CHƯƠNG 7

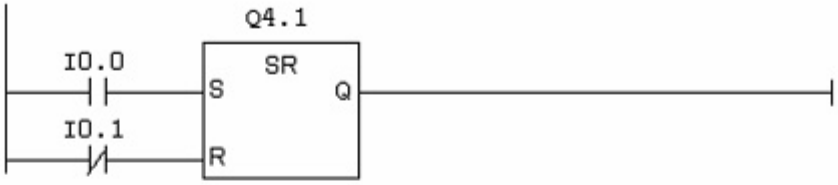
CÁC BÀI TẬP ỨNG DỤNG

Bài 1: Viết chương trình cho hệ thống làm đầy chai với yêu cầu:

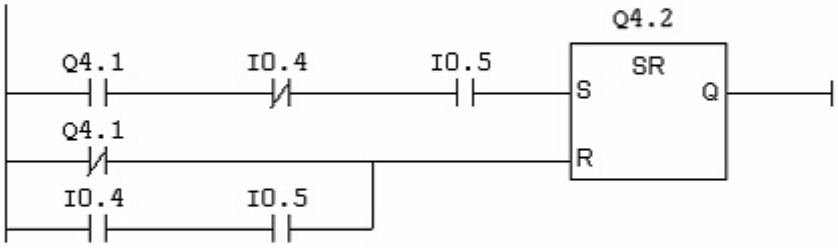
Ngõ vào I0.0 là tiếp điểm thường hở dùng khởi động hệ thống.
Ngõ vào I0.1 là tiếp điểm thường đóng dùng dừng hệ thống.
Khi hệ thống khởi động thì đèn ngõ ra Q4.1 sáng lên.
Khi hệ thống khởi động có thể chọn chế độ làm việc bằng tay hoặc tự động.
Khi chọn I0.4=0 là chế độ tay, I0.4=1 là chế độ tự động.
I0.5 dùng để cho phép các chế độ hoạt động.
Các đèn báo chế độ: Chế độ tay Q4.2, chế độ tự động Q4.3.
Khi thay đổi chế độ thì hệ thống sẽ dừng lại.
Ở chế độ bằng tay thì hệ thống có thể chạy thuận hoặc chạy nghịch bằng công tắc I0.2 và I0.3.

Chương trình điều khiển cho quá trình này như sau: **Bài 2:** Viết chương trình cho hệ thống làm đầy chai với yêu cầu:

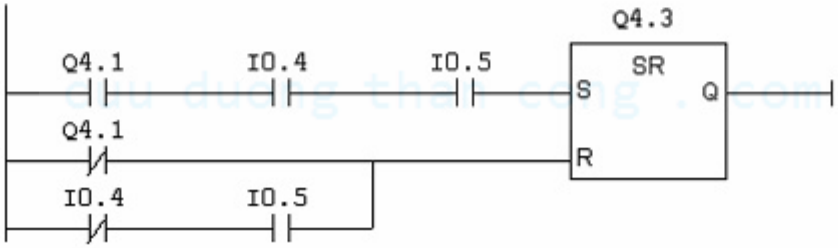
Network 1 : khoi dong/dung he thong



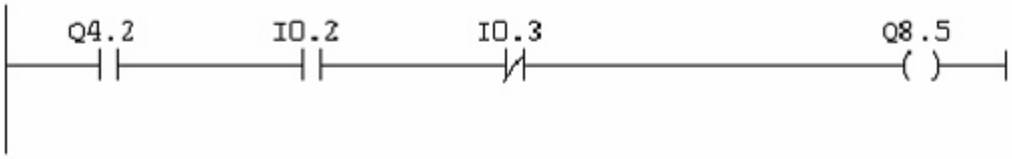
Network 2 : che do tay



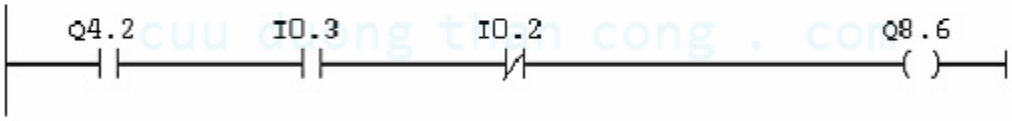
Network 3 : che do tu dong



Network 4 : che do tay quay thuan



Network 5 : che do tay quay nghich



Network 6 : che do tu dong quay thuan

Ngõ vào I0.0 là tiếp điểm thường hở dùng để khởi động hệ thống.

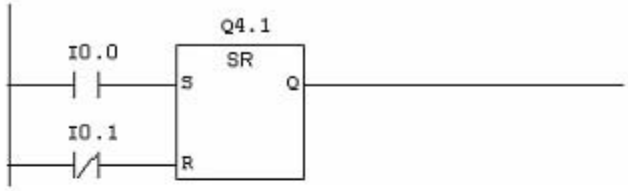
Ngõ vào I0.1 là tiếp điểm thường đóng dùng để dừng hệ thống.
Khi hệ thống khởi động thì đèn ngõ ra Q4.1 sáng lên.
Khi hệ thống khởi động có thể chọn chế độ làm việc bằng tay hoặc tự động.
Khi chọn I0.4=0 là chế độ tay, I0.4=1 là chế độ tự động.
I0.5 dùng để cho phép các chế độ hoạt động.
Các đèn báo chế độ: Chế độ tay Q4.2, chế độ tự động Q4.3.
Khi thay đổi chế độ thì hệ thống sẽ dừng lại.

Ở chế độ bằng tay thì hệ thống có thể chạy thuận hoặc chạy nghịch bằng công tắc I0.2 và I0.3.

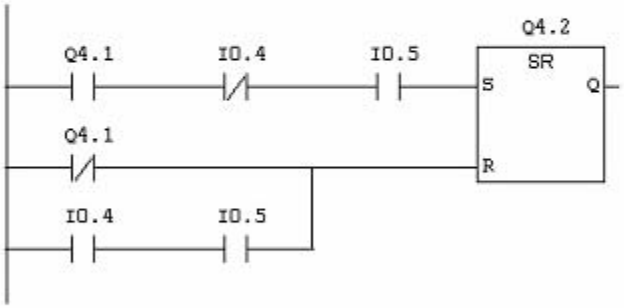
Ở chế độ tự động: Khi động cơ băng chuyền được khởi động (chỉ có quay thuận) thì băng chuyền chạy liên tục cho đến khi bị tắt bằng công tắt I0.1 hoặc khi cảm biến I8.6 phát hiện được chai. Khi chai đã được làm đầy thì băng chuyền tiếp tục chạy cho tới khi tắt bằng công tắt I0.1 hoặc khi cảm biến I8.6 phát hiện được chai tiếp theo. Quá trình làm đầy chai được thực hiện trong 3 giây và được thông báo bằng ngõ ra Q5.0. Hệ thống đếm số chai đầy và chai rỗng nhờ hai cảm biến I8.5 và I8.7. Số chai hư bằng số chai rỗng trừ số chai đầy.

Chương trình điều khiển cho quá trình này như sau:

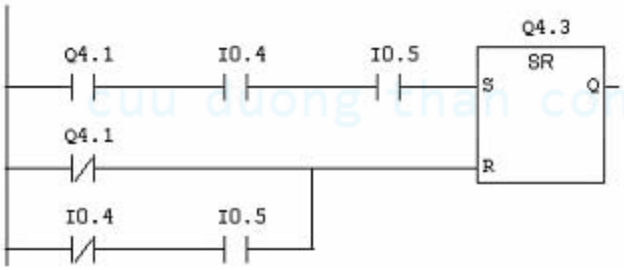
Network 1 : khoi dong/dung he thong



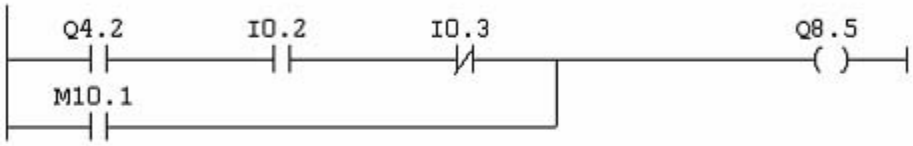
Network 2 : che do tay



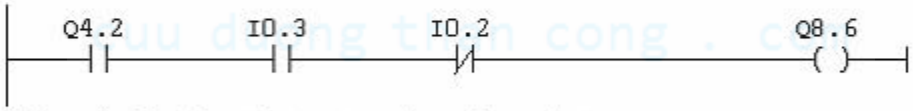
Network 3 : che do tu dong



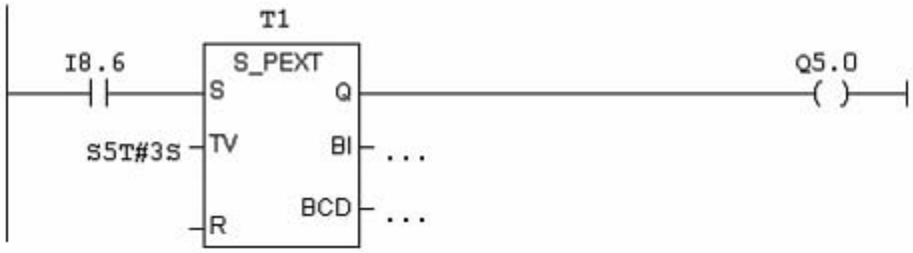
Network 4 : che do tay quay thuan



Network 5 : che do tay quay nghich

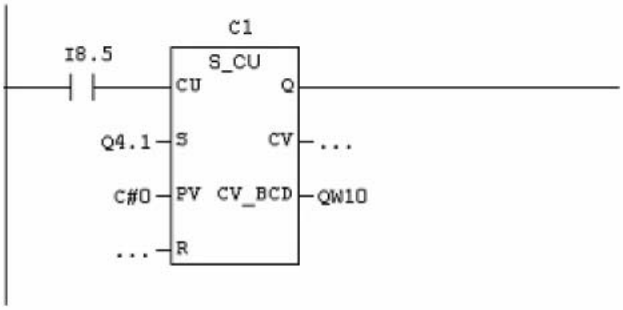


Network 6 : dat thoi gian lam day chai

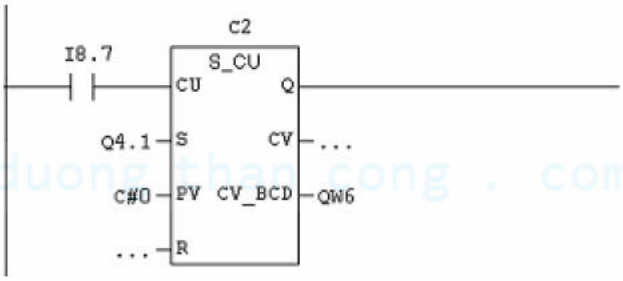


Network 7 : tao bit nho phu o che do tu dong

Network 8 : dem so chai rong



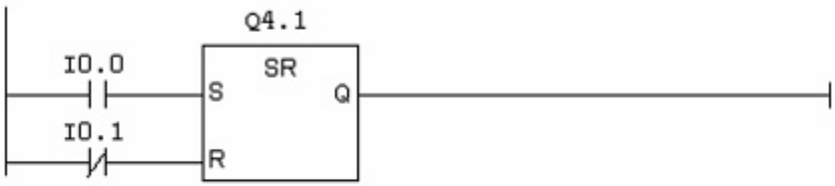
Network 9 : dem so chai day



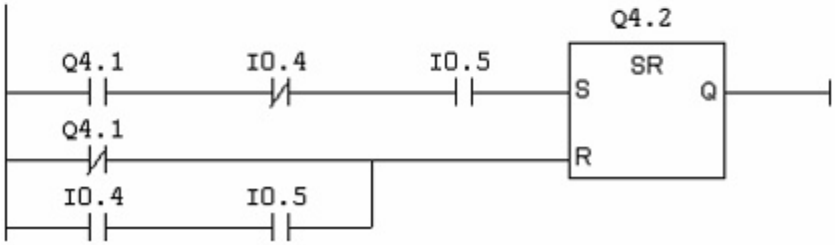
Bài 3: Yêu cầu giống như ở bài tập 2 nhưng bài tập này đếm sản phẩm dùng các lệnh toán học.

Chương trình điều khiển cho quá trình này như sau:

Network 1 : khoi dong/dung he thong

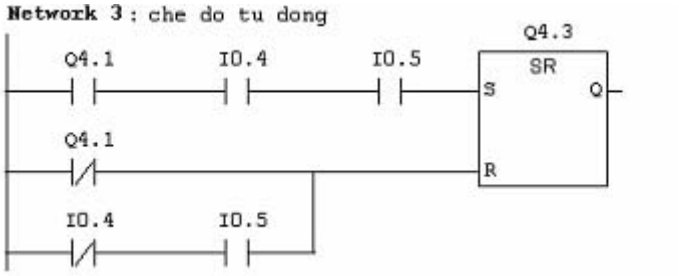


Network 2 : che do tay

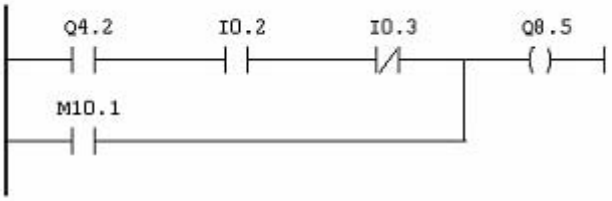


cuu duong than cong . com

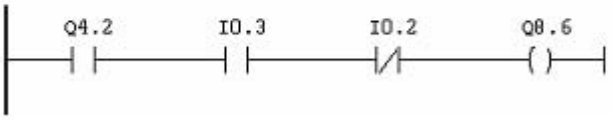
Network 3 : che do tu dong



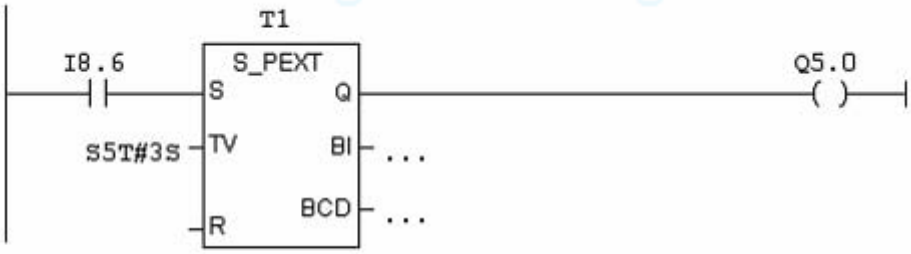
Network 4 : che do tay quay thuan



Network 5 : che do tay quay nghich

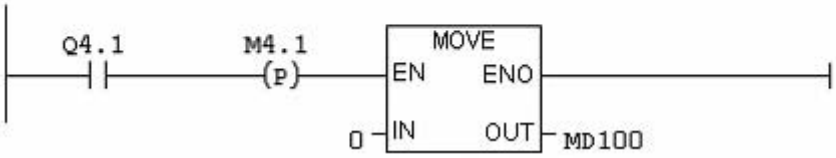


Network 6 : dat thoi gian lam day chai

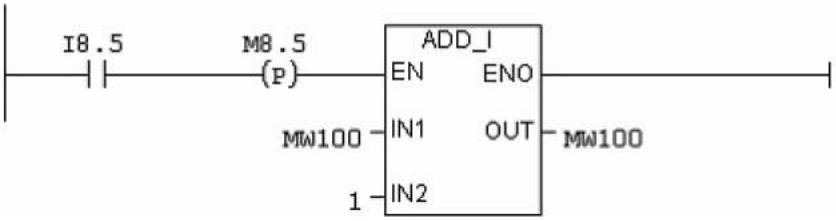


Network 7 : tao bit nho phu c che do tu dong

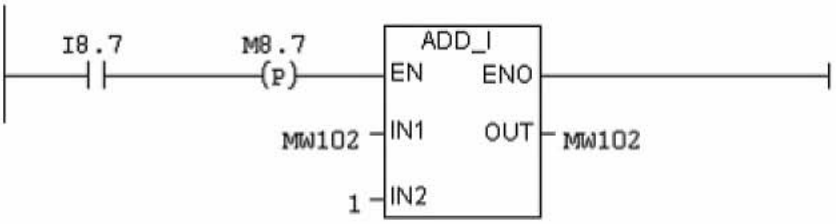
Network 8 : xoa so chai rong, day, hu



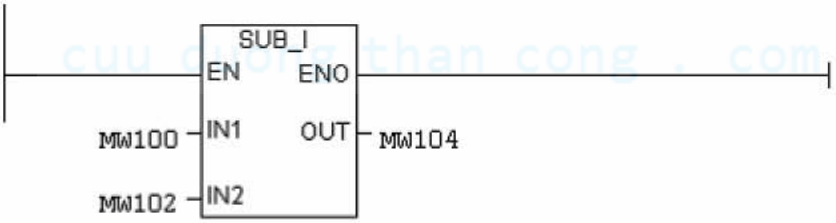
Network 9 : dem so chai rong



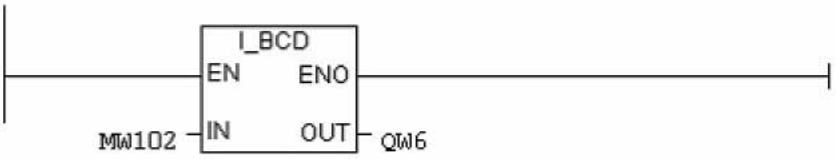
Network 10 : dem so chai day



Network 11 : dem so chai hu



Network 12 : hien thi so chai day



Bài 4: Bài tập này cũng tương tự như bài tập 2 nhưng có thêm các yêu cầu:
Ngõ vào I0.0 là tiếp điểm thường hở dùng để khởi động hệ thống.
Ngõ vào I0.1 là tiếp điểm thường đóng dùng để dừng hệ thống.
Khi hệ thống khởi động thì đèn ngõ ra Q4.1 sáng lên.
Khi hệ thống khởi động có thể chọn chế độ làm việc bằng tay hoặc tự động.
Khi chọn I0.4=0 là chế độ tay, I0.4=1 là chế độ tự động.
I0.5 dùng để cho phép các chế độ hoạt động.

Các đèn báo chế độ: Chế độ tay Q4.2, chế độ tự động Q4.3.

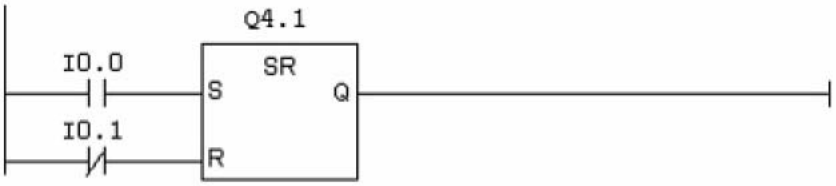
Khi thay đổi chế độ thì hệ thống sẽ dừng lại.

Ở chế độ bằng tay thì hệ thống có thể chạy thuận hoặc chạy nghịch bằng công tắc I0.2 và I0.3.

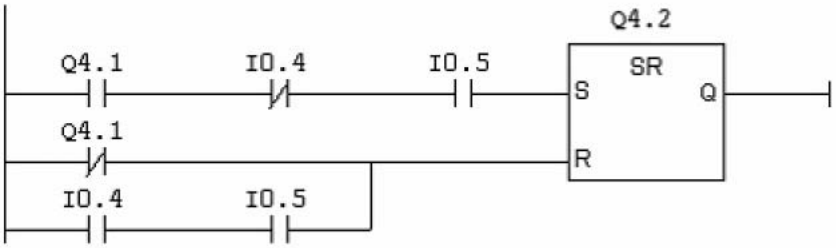
Ở chế độ tự động: Khi động cơ băng chuyền được khởi động (chỉ có quay thuận) thì băng chuyền chạy liên tục cho đến khi bị tắt bằng công tắt I0.1 hoặc khi cảm biến I8.6 phát hiện được chai. Khi chai đã được làm đầy thì băng chuyền tiếp tục chạy cho tới khi tắt bằng công tắt I0.1 hoặc khi cảm biến I8.6 phát hiện được chai tiếp theo. Quá trình làm đầy chai được thực hiện trong 3 giây và được thông báo bằng ngõ ra Q5.0. Hệ thống đếm số chai đầy và chai rỗng nhờ hai cảm biến I8.5 và I8.7. Số chai hư bằng số chai rỗng trừ số chai đầy. Số chai đầy được đưa vào các thùng, mỗi thùng chức 20 chai, số thùng được hiển thị ở QW6.

Chương trình điều khiển cho quá trình này như sau:

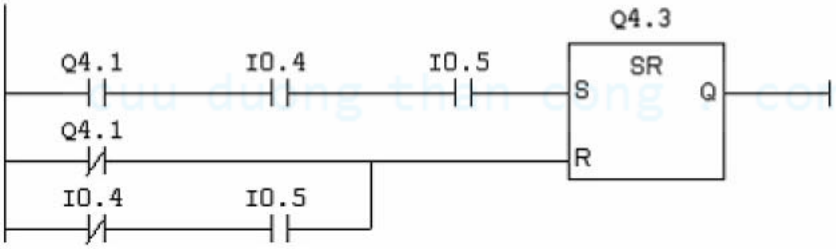
Network 1 : khoi dong/dung he thong



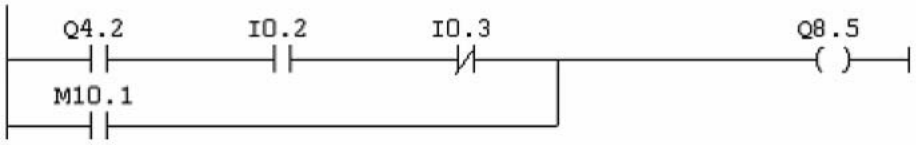
Network 2 : che do tay



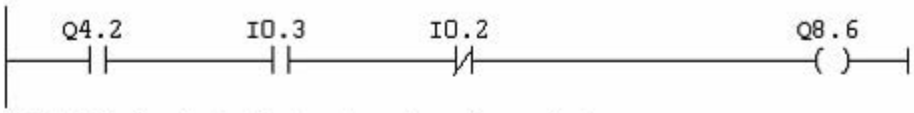
Network 3 : che do tu dong



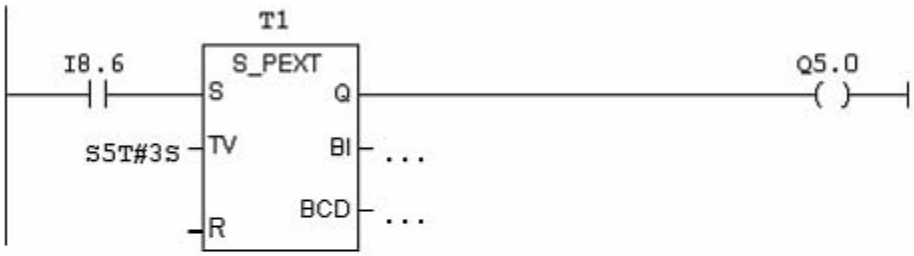
Network 4 : che do tay quay thuan



Network 5 : che do tay quay nghich

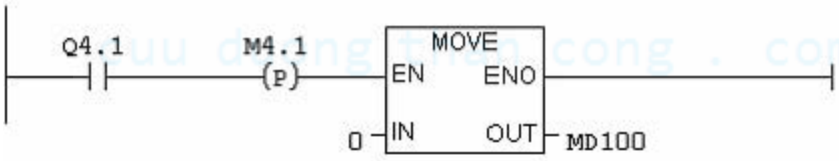


Network 6 : dat thoi gian lam day chai

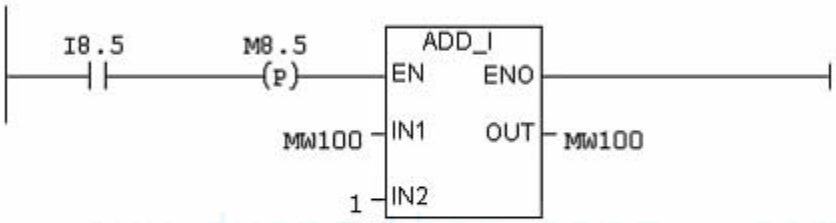


Network 7 : tao bit nho phu o che do tu dong

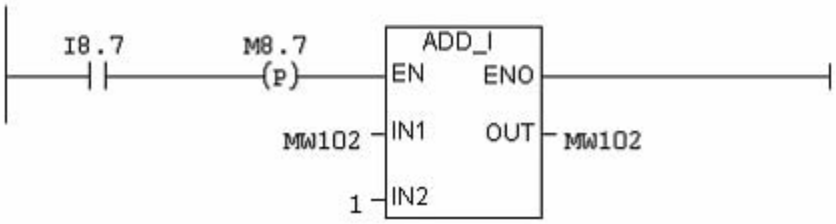
Network 8 : xoa so chai rong, day, hu



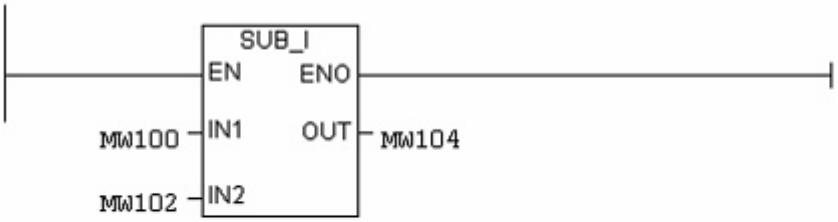
Network 9 : dem so chai rong



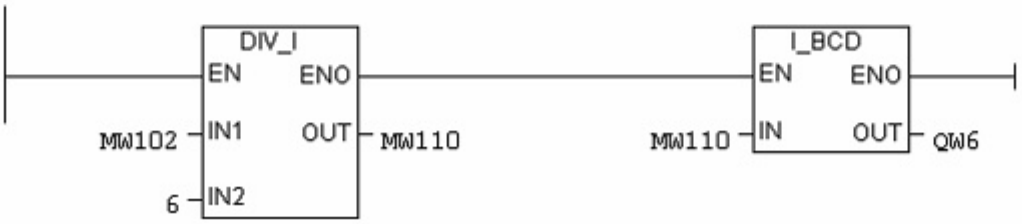
Network 10 : dem so chai day



Network 11 : dem so chai hu



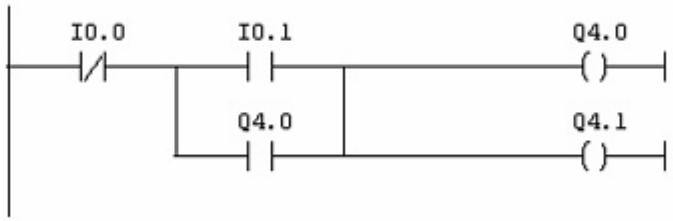
Network 12 : dem so thung bang cach chia so chai day cho 6



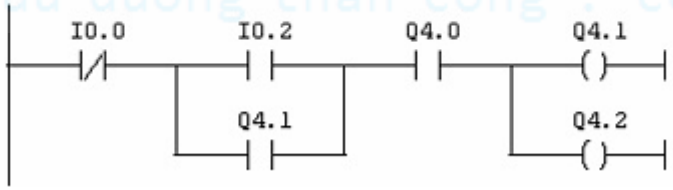
Bài 5: Mạch điều khiển tuần tự cưỡng bức. I0.0 là nút nhấn thường đóng dùng để dừng động cơ. Khi nhấn I0.1 thì ngõ ra Q4.0=1 và tự duy trì, ngõ ra Q4.1=1 tượng trưng cho đèn ngõ ra 1. Khi đó nếu nhấn I0.2 thì ngõ ra Q4.2 =1 và tự duy trì và ngõ ra Q4.3=1 tượng trưng cho đèn ngõ ra 2. I0.2 có tác động chỉ khi I0.1 tác động trước.

Chương trình điều khiển cho quá trình này như sau:

Network 1 : Title:



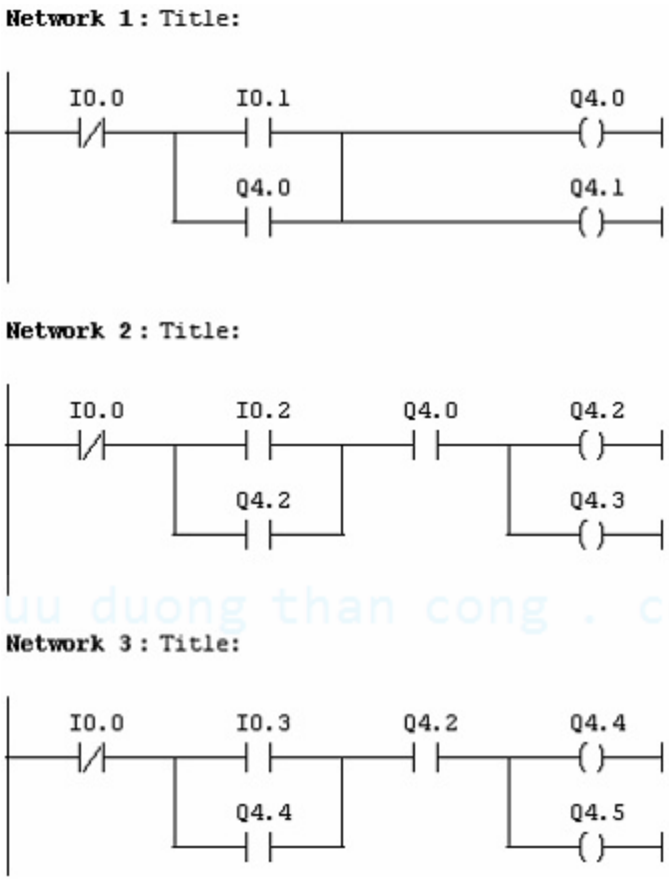
Network 2 : Title:



Bài 6: Mạch tuần tự khởi động bằng tay. Mạch này có các trạng thái như sau:
I0.0 dùng để dừng hệ thống.
Khi nhấn I0.1 thì đèn 1 sáng và tự duy trì.

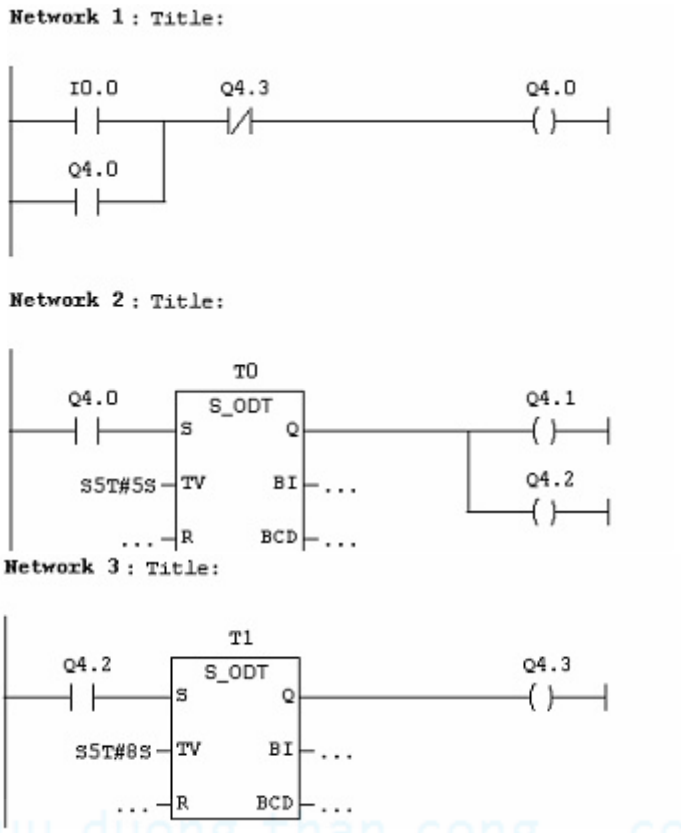
Khi nhấn I0.2 thì đèn 2 sáng và tự duy trì.
Khi nhấn I0.3 thì đèn 3 sáng và tự duy trì.

Chương trình điều khiển cho quá trình này như sau:



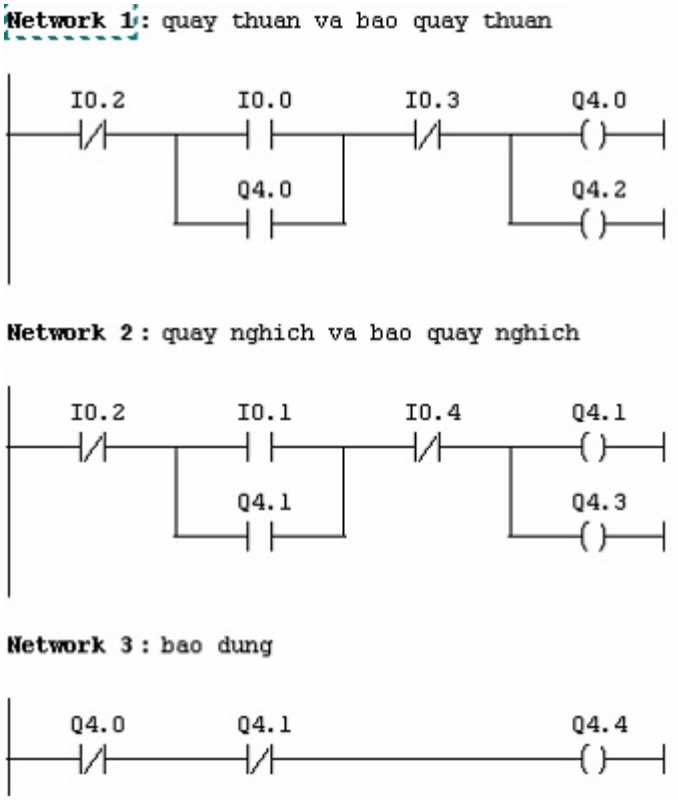
Bài 7: Mạch tự động đóng tuần tự. Mạch này chỉ có một nút nhấn I0.0 dùng để khởi động mạch. Khi nhấn I0.0 thì Q4.0=1 và sau thời gian 5s thì rơle thời gian tác động làm cho ngõ ra Q4.1=1, sau 8s nữa thì rơle thời gian làm cho Q4.0=0 mạch ngưng hoạt động.

Chương trình điều khiển cho quá trình này như sau:



Bài 8: Mạch điều khiển thang máy xây dựng. Khi nhấn nút nhấn nâng thì gàu được nâng lên đến khi đụng công tắc giới hạn trên thì gàu dừng lại. Khi nhấn nút nhấn hạ thì gàu được hạ xuống đến khi đụng công tắc giới hạn dưới thì gàu dừng lại. Trong lúc nâng hoặc hạ, nếu nhấn nút nhấn dừng thì gàu dừng lại.

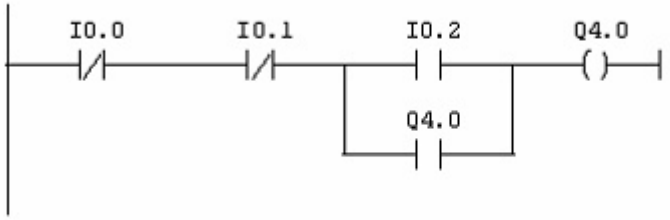
Chương trình điều khiển cho quá trình này như sau:



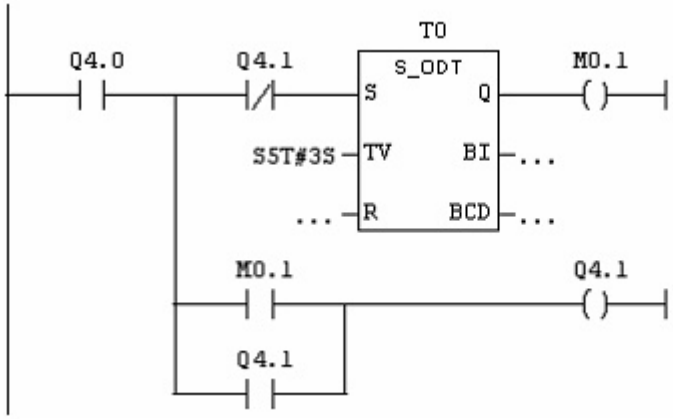
Bài 9: Mạch khởi động cơ xoay chiều 3 pha (động cơ này khởi động bằng cách thêm điện trở phụ mắt nối tiếp với một trong 3 pha của động cơ). Khi nhấn nút nhấn khởi động thì động cơ khởi động với điện trở phụ, sau một thời gian chỉ định trước động cơ làm việc ở trạng thái gần ổn định thì điện trở phụ được nối tắc.

Chương trình điều khiển cho quá trình này như sau:

Network 1: I0.0 cat dong qua tai, I0.1 nut nhan dung, I0.2 mo may



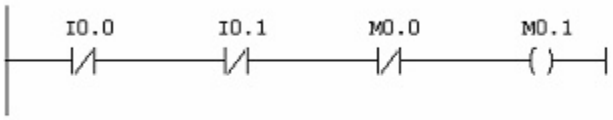
Network 2 : Q4.1 noi tat dien tro phu sau 3 giay



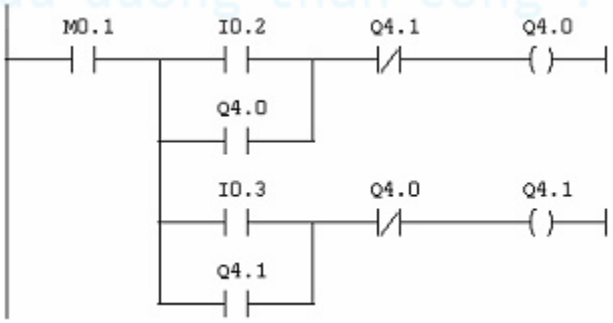
Bài 10: Mạch đổi chiều quay của động cơ xoay chiều 3 pha. Khi cho động cơ quay theo chiều nào đó thì muốn động cơ đổi chiều thì phải dừng động cơ và nhấn nút khởi động theo chiều ngược lại. Trong quá trình đảo chiều quay, có một thời gian trễ xác định để động cơ đảo chiều quay.

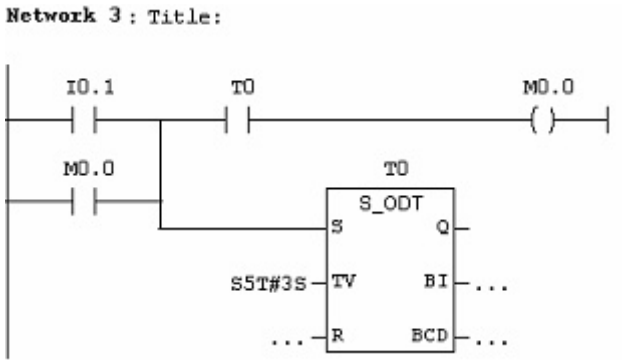
Chương trình điều khiển cho quá trình này như sau:

Network 1 : Title:



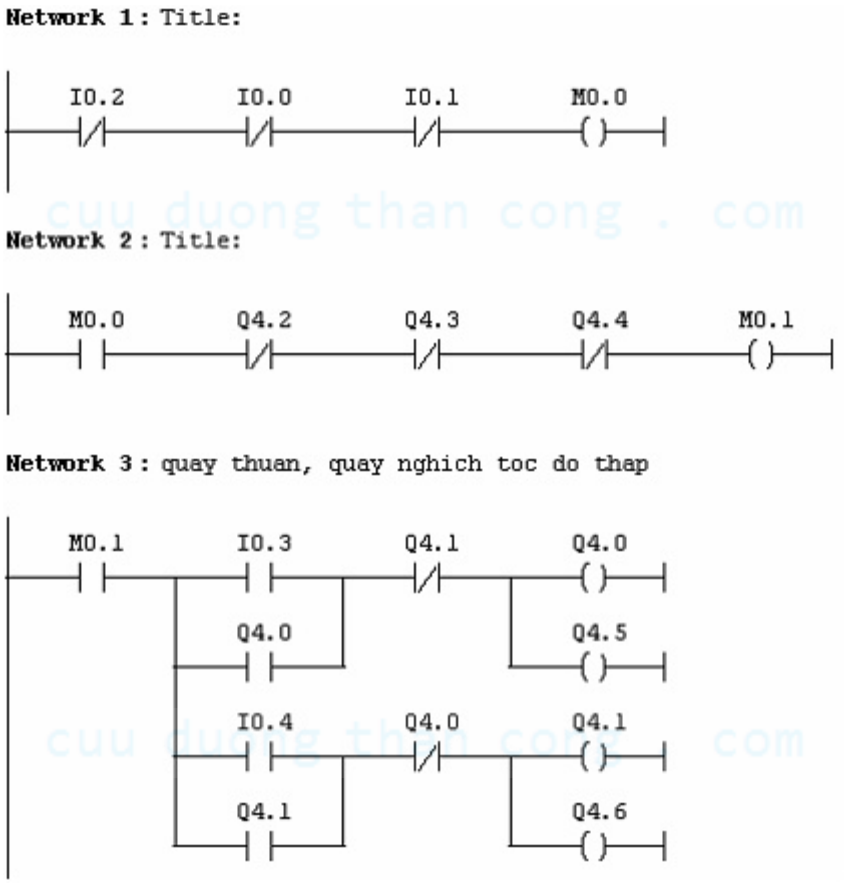
Network 2 : Title:

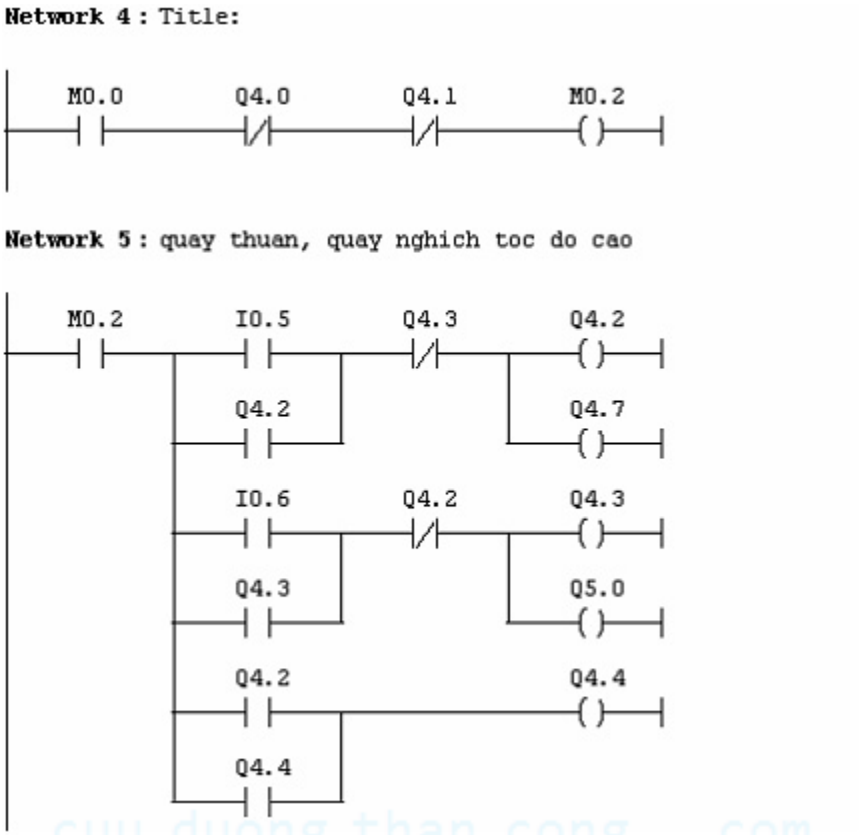




Bài 11: Mạch điều khiển động cơ 2 tốc độ và 2 chiều quay. Khi nhấn I0.0 hoặc I0.1 thì động cơ chạy trái hoặc phải với tốc độ thấp. Khi nhấn I0.2 hoặc I0.3 thì động cơ chạy trái hoặc phải với tốc độ cao.

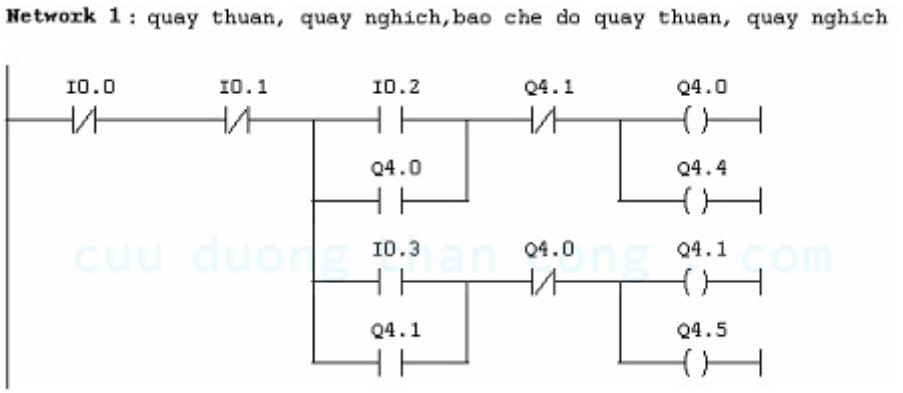
Chương trình điều khiển cho quá trình này như sau:



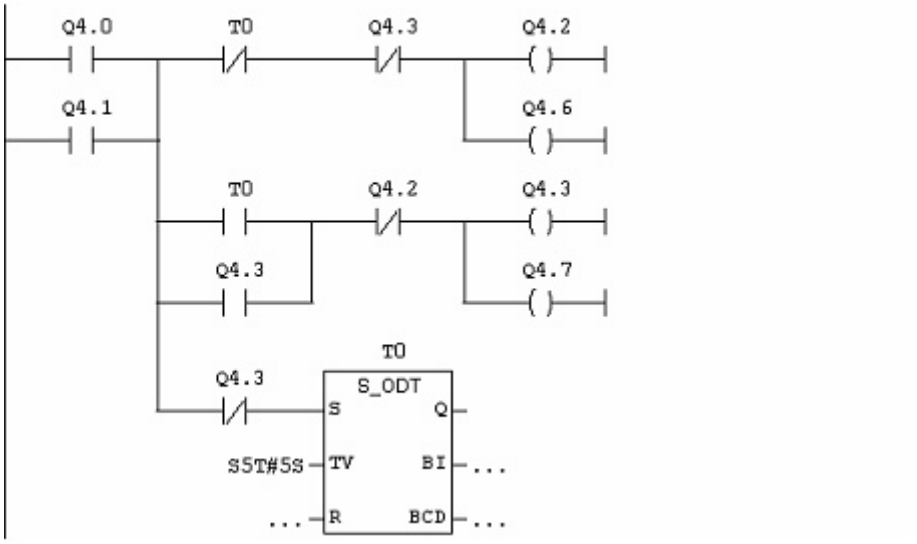


Bài 12: Mạch khởi động sao-tam giác có 2 chiều quay. Khi nhấn cho động cơ quay thuận thì mạch khởi động ở chế độ nối sao, sau một thời gian định trước thì động cơ chuyển sang chế độ nối tam giác. Tương tự cho chiều khởi động ngược lại.

Chương trình điều khiển cho quá trình này như sau:



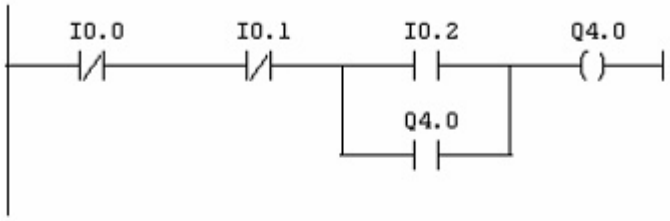
Network 2 : chay sao (Q4.2)-chay tamgiac (Q4.3), bao che do sao-tam giac



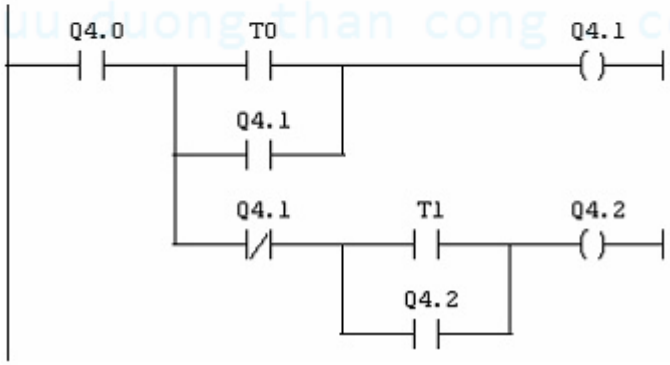
Bài 13: Mạch khởi động sao-tam giác cho động cơ 3 pha có vành trượt. Động cơ 3 pha có vành trượt thường khởi động bằng điện trở phụ được nối với phần ứng. Động cơ được nối với 3 cấp điện trở phụ. Khi nhấn nút nhấn khởi động thì sau một thời gian nhất định thì các cấp điện trở lần lượt được cắt và động cơ làm việc ở chế độ định mức khi các điện trở phụ được cắt hết.

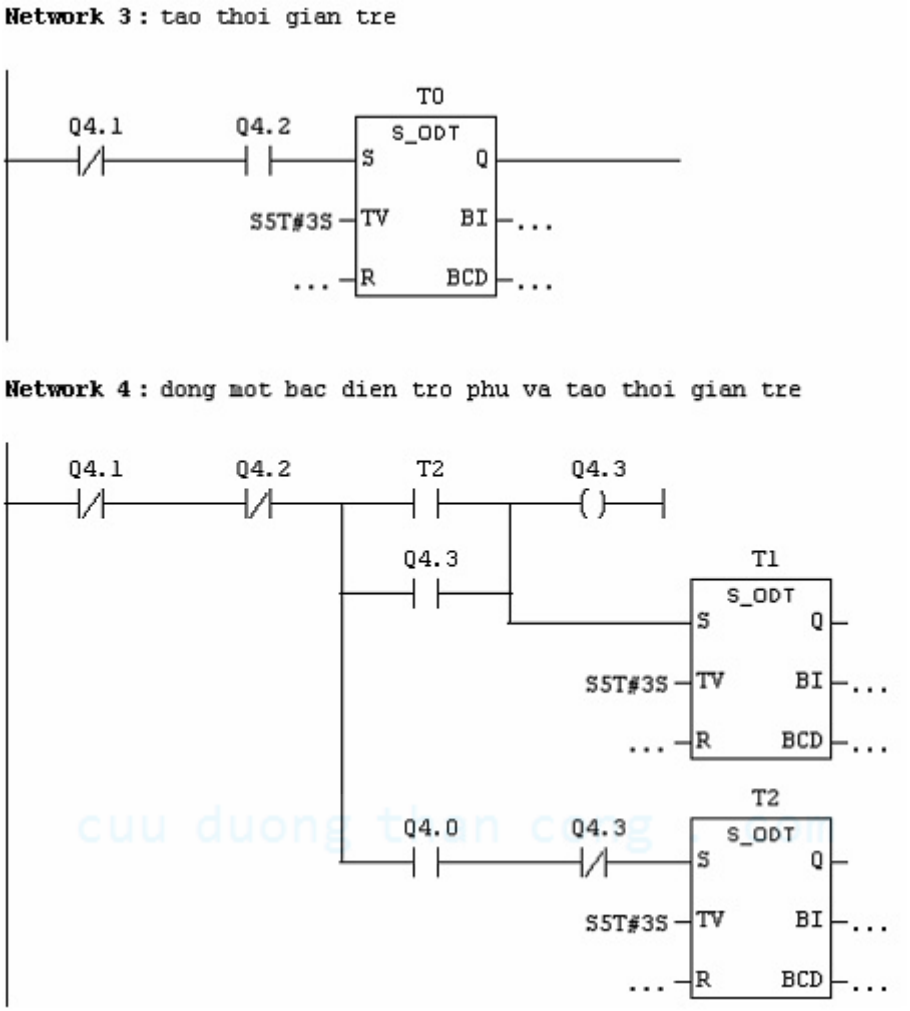
Chương trình điều khiển cho quá trình này như sau:

Network 1 : khoi dong



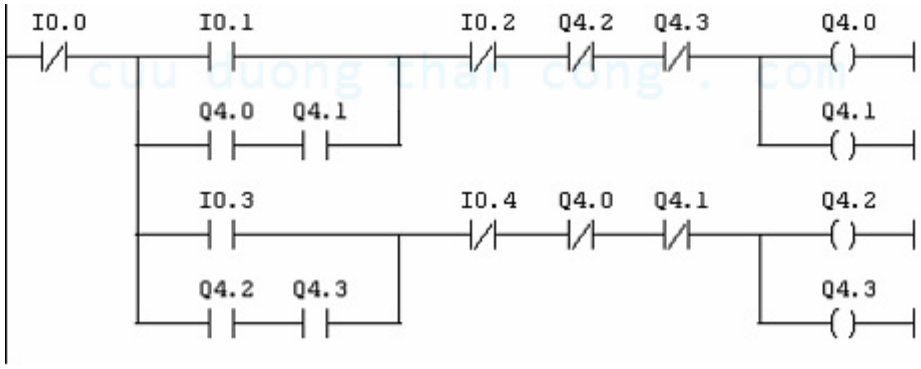
Network 2 : dong hai bac dien tro phu con lai





Bài 14: Mạch điều khiển 2 động cơ làm việc đồng thời theo chiều thuận và nghịch. Khi nhấn nút khởi động theo chiều thuận hay nghịch thì cả 2 động cơ làm việc cùng lúc theo cùng một chiều.

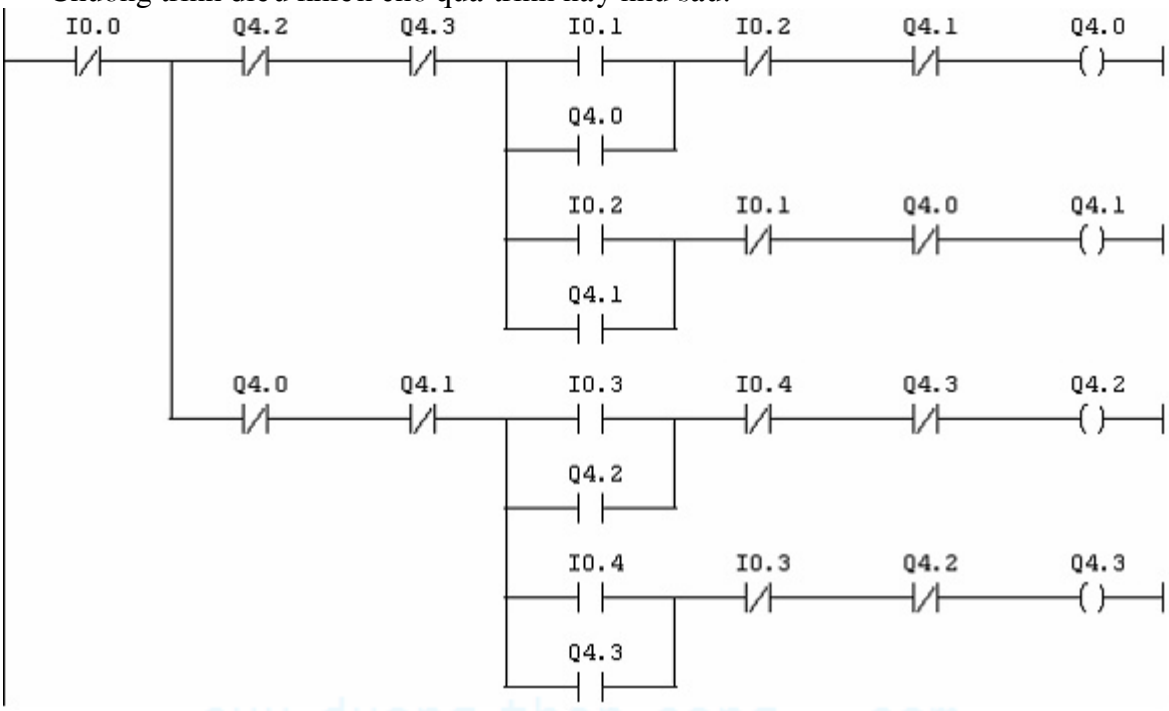
Chương trình điều khiển cho quá trình này như sau:



Bài 15: Mạch điều khiển 2 động cơ làm việc xen kẽ theo chiều thuận và nghịch. Ở

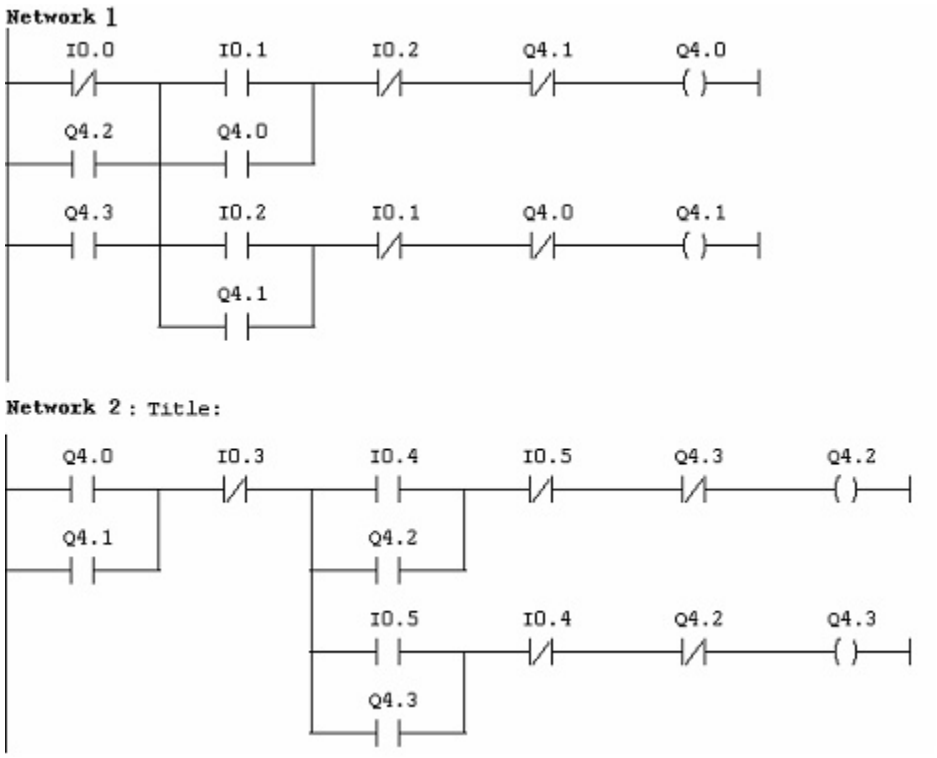
một thời điểm thì chỉ có một động cơ hoạt động theo chiều thuận hoặc chiều nghịch.

Chương trình điều khiển cho quá trình này như sau:



Bài 16: Mạch điều khiển động cơ làm việc theo trình tự. Hai động cơ có thể quay theo chiều thuận hoặc chiều nghịch. Động cơ thứ hai chỉ làm việc khi động cơ thứ nhất làm việc.

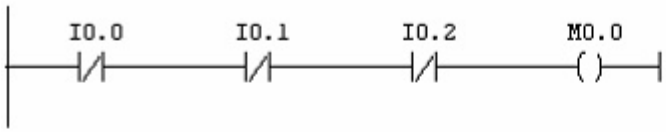
Chương trình điều khiển cho quá trình này như sau:



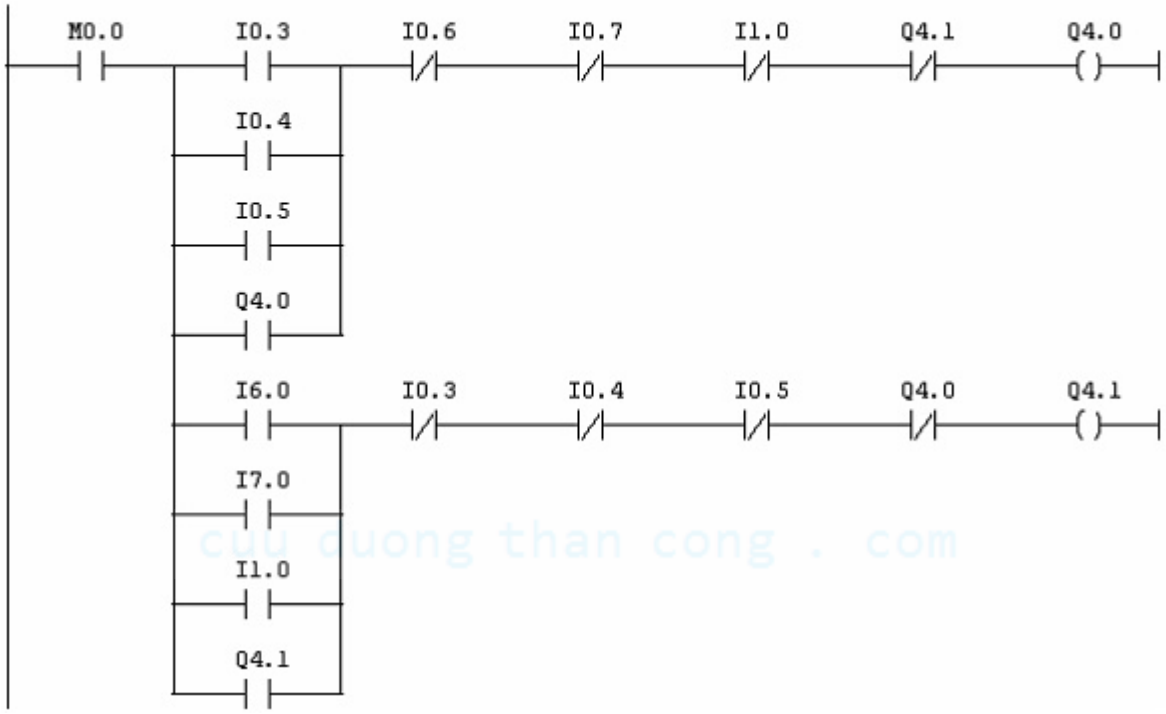
Bài 17: Mạch điều khiển 2 động cơ quay thuận và quay nghịch độc lập nhau ở nhiều vị trí. 2 động cơ có thể làm việc độc lập nhau, mỗi động cơ có thể quay thuận, quay nghịch và được điều khiển ở nhiều nơi.

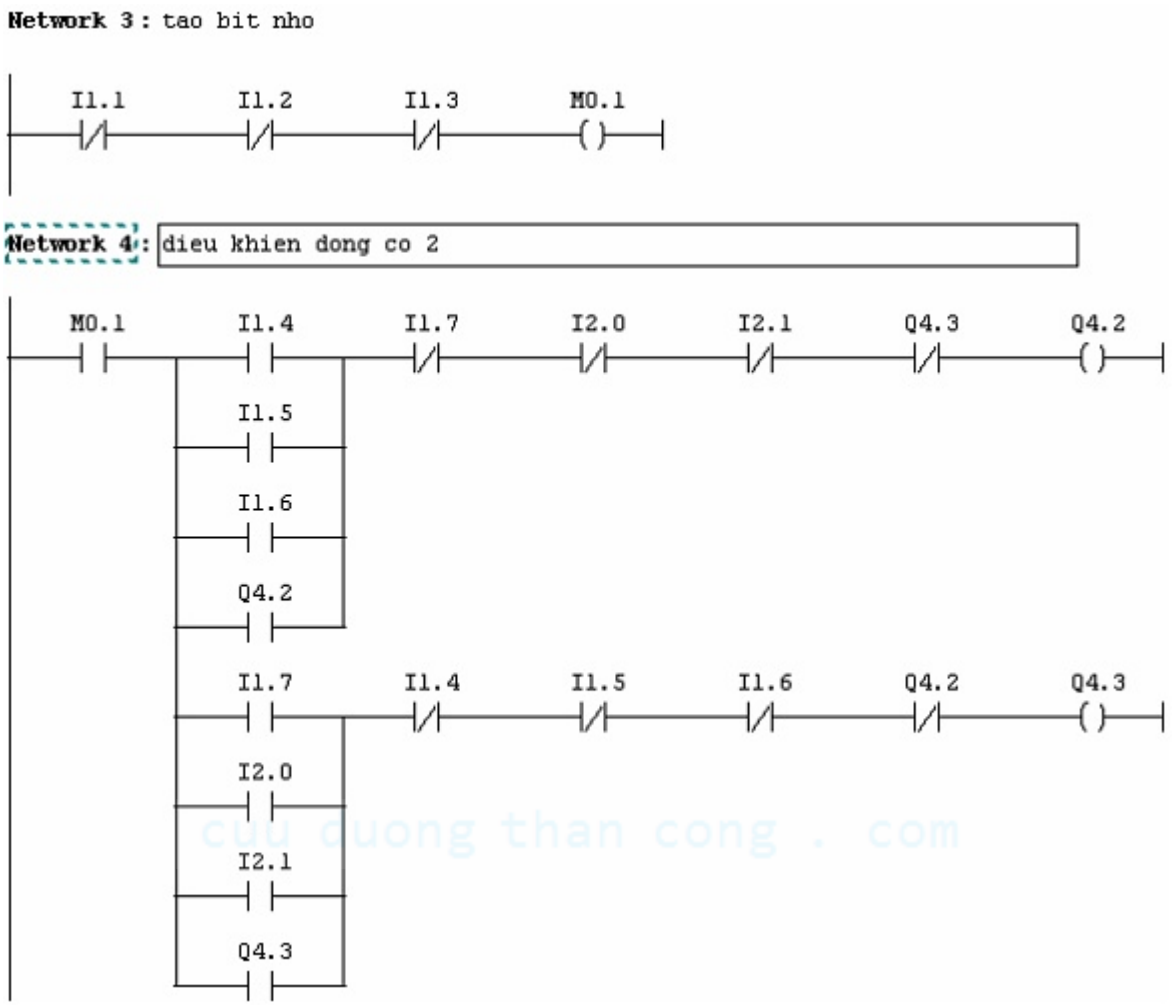
Chương trình điều khiển cho quá trình này như sau:

Network 1 : tao bit nho



Network 2 : dieu khien dong co 1

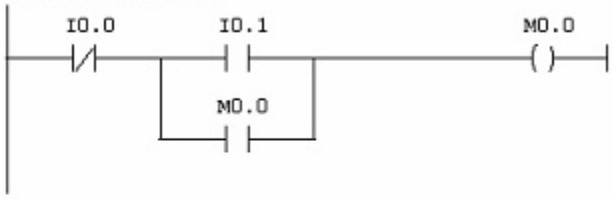




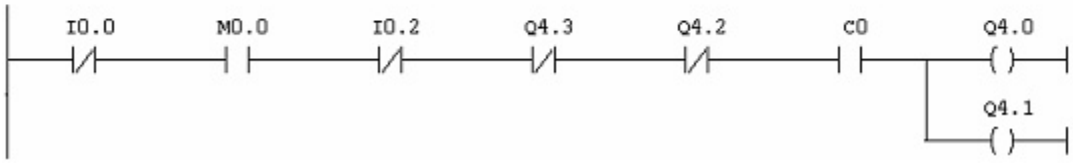
Bài 18: Mạch điều khiển quá trình pha trộn sơn. Có hai loại sơn màu khác nhau được bơm bằng máy bơm 1 và 2 khi sơn đầy thì báo cho 2 máy ngừng hoạt động và máy trộn hoạt động, sau 5s thì máy trộn ngừng. Sau khi trộn xong thì sơn được đưa đến bồn chứa sơn bằng bơm 3 qua van xả. Khi sơn hết thì báo cho máy bơm 3 ngừng hoạt động, van đóng lại và máy bơm 1 và 2 tiếp tục hoạt động 10 lần. Trong quá trình hoạt động, nếu nhấn dừng thì hệ thống dừng lại. I0.0 là nút nhấn dừng cho hệ thống.

Chương trình điều khiển cho quá trình này như sau:

Network 1 : Title:

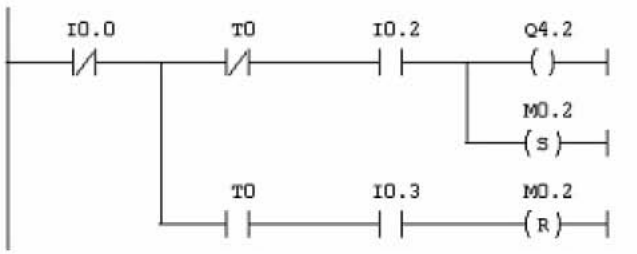


Network 2 : dieu khien bom 1 va bom 2

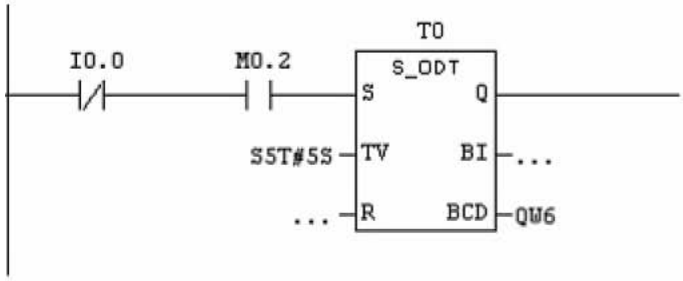


cuu duong than cong . com

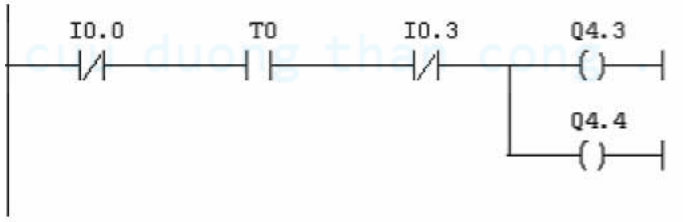
Network 3 : dieu khien may tron



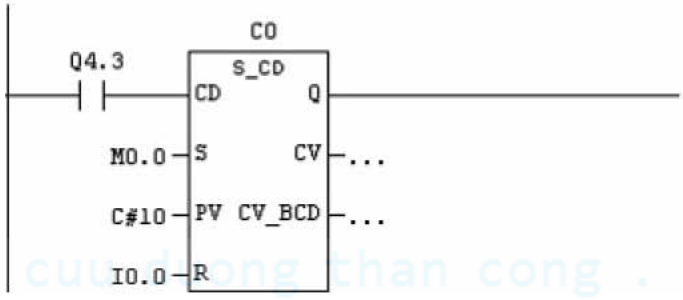
Network 4 : dinh thoi cho may tron



Network 5 : dieu khien van va may bom 3

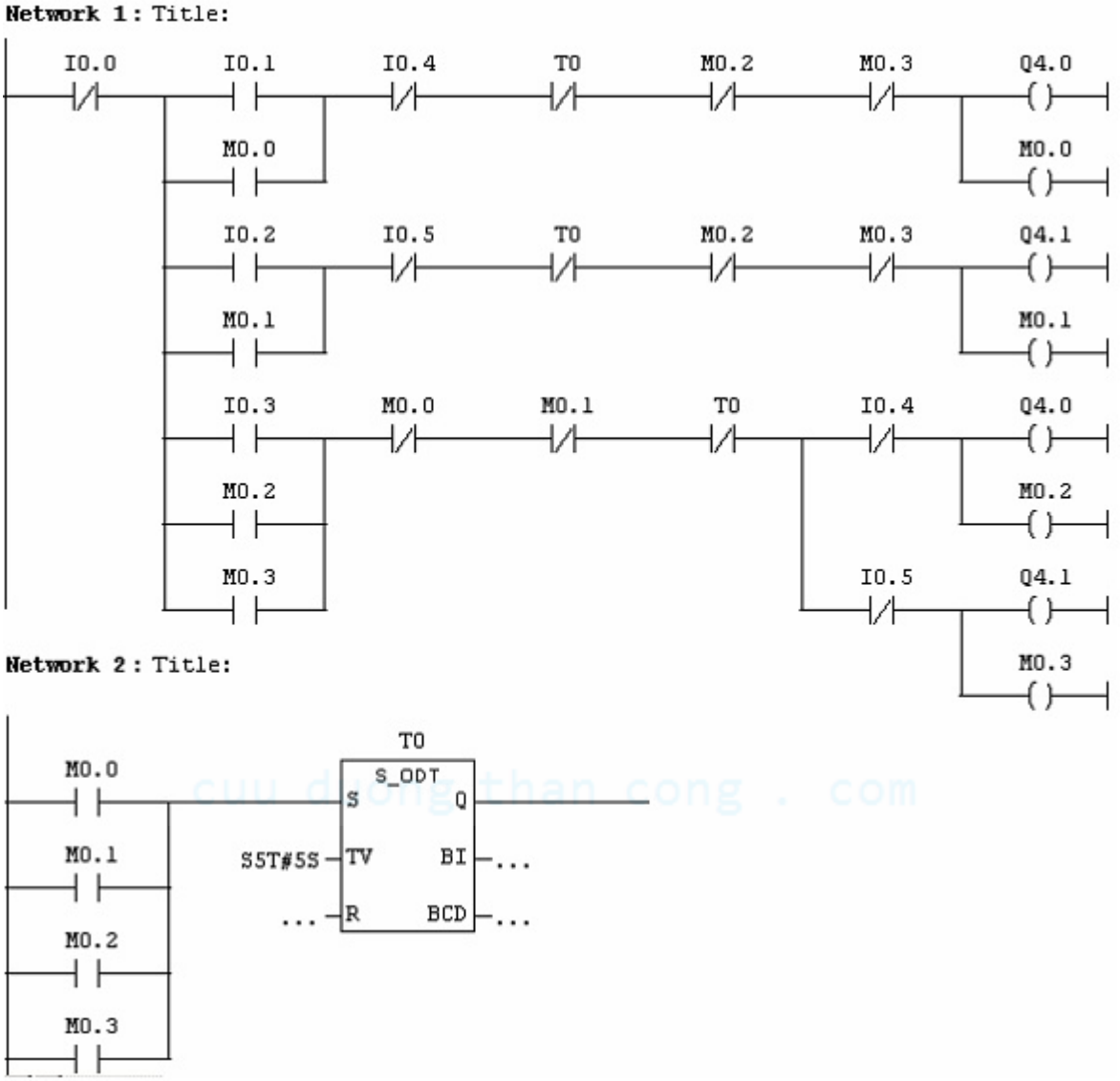


Network 6 : dem so chu ky hoat dong



Bài 19: Mạch điều khiển bồn trộn hoá chất với yêu cầu: Có 2 bồn trộn hoá chất với mỗi bồn được kéo một động cơ. Hoá chất được đổ vào do con người thực hiện. Có thể chọn 1 trong 2 bồn làm việc hoặc cả 2 bồn đều làm việc. Nếu trong quá trình trộn mà nếu có van nào bị hở thì bồn đó ngưng hoạt động. Mỗi bồn hoạt động trong thời gian là 5 giây. Mạch còn có các đèn báo trạng thái của các bồn.

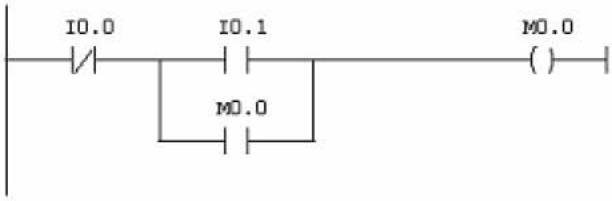
Chương trình điều khiển cho quá trình này như sau:



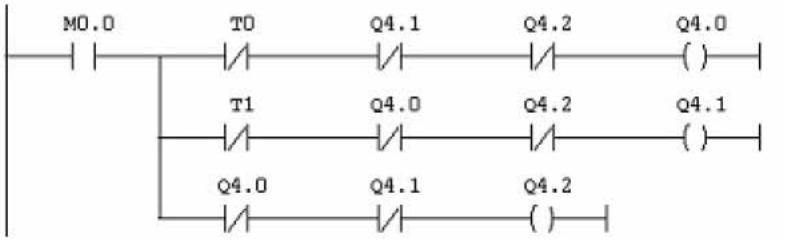
Bài 20: Chương trình điều khiển đèn giao thông cho một ngã tư với đèn xanh sáng 25 giây, đèn vàng sáng 5 giây, đèn đỏ sáng 30 giây.

Chương trình điều khiển cho quá trình này như sau:

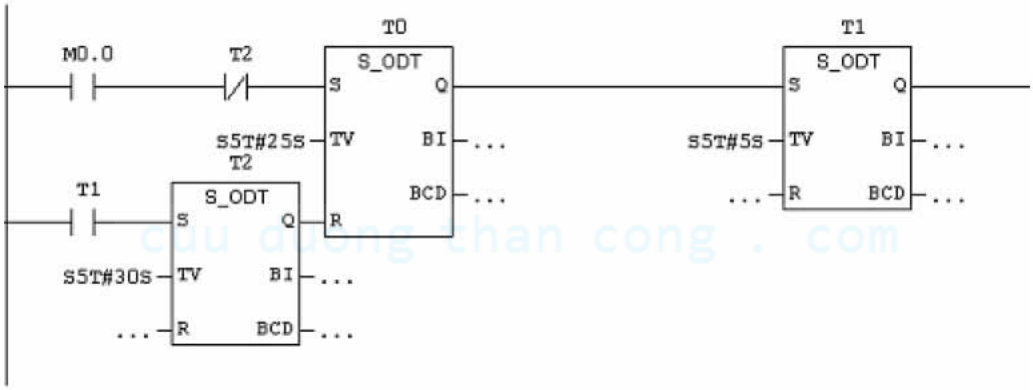
Network 1 : Title:



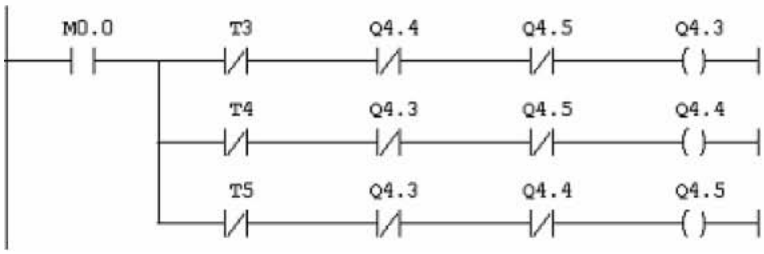
Network 2 : x1, v1, d1



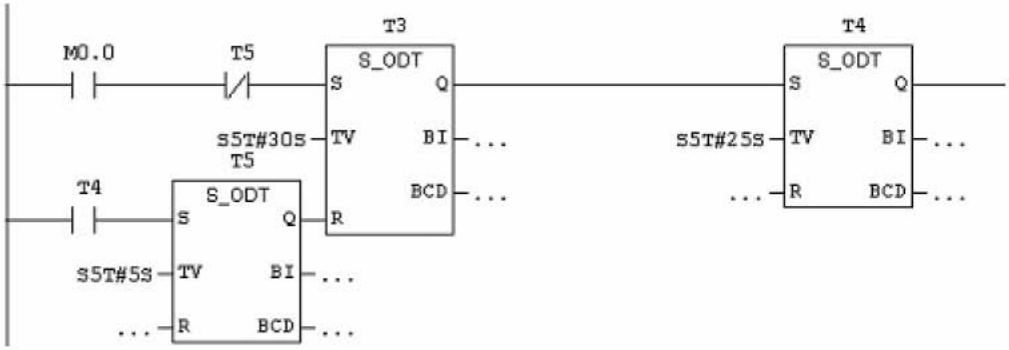
Network 3 : Title:



Network 4 : x2, v2, d2



Network 5 : Title:



Bài 21: Chương trình kiểm tra mức nước trong thùng chứa. Thùng chứa chứa tối đa là 600 lít, nếu mức nước trong thùng nhiều hơn 600 lít thì chương trình sẽ ngưng hoạt động, nếu mức nước trong thùng ít hơn 50 lít thì có tín hiệu báo. Chương trình được viết trong OB1 dùng thêm khối FC105 nằm trong thư viện chuẩn của chương trình STEP7. FC105 có các ngõ vào/ra như sau:

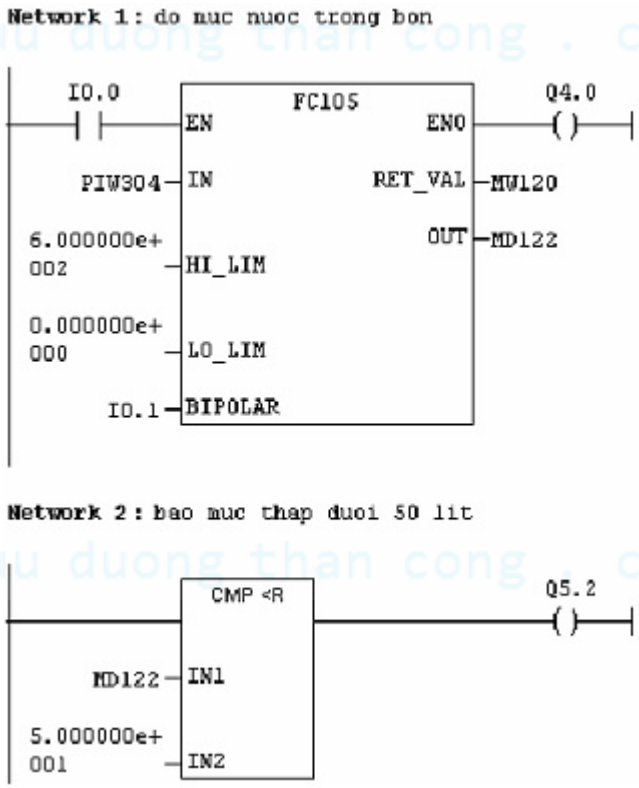
EN: Ngõ vào cho phép FC105 hoạt động.
ENO: Ngõ ra có mức logic 1 khi FC105 hoạt động và không có lỗi.

IN: Ngõ vào dạng số nguyên dùng để đọc tín hiệu vào. Có thể đọc trực tiếp từ modul

Analog hoặc qua giao tiếp dạng số nguyên.

LO_LIM, HI_LIM: Các giới hạn trên và dưới dùng để chuyển đổi đại lượng vật lý được đặt tại các ngõ này.

OUT: Ngõ ra chứa giá trị đọc được từ ngõ vào có dạng số thực.
BIPOLAR: Xác định giá trị âm hoặc giá trị dương được chuyển đổi. BIPOLAR=0 thì chuyển đổi giá trị dương, BIPOLAR=1 thì chuyển đổi giá trị âm.
RET_VAL: Ngõ ra có giá trị 0 nếu lệnh hoạt động không có lỗi.

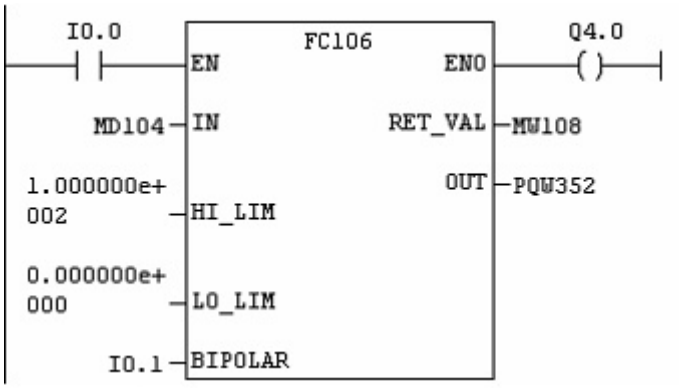


Hoạt động của chương trình trên như sau: Nếu I0.0= thì FC105 hoạt động, khi đó giá trị ngõ vào PIW304 dạng số nguyên được chuyển ra ngoài qua ngõ ra OUT đến

MD122 dạng số thực nếu số lít nhỏ hơn 60. Nếu bồn chứa nhiều hơn 60 lít thì FC105 không hoạt động. Nếu số nước trong bồn nhỏ hơn 50 lít thì ngõ ra Q5.2 sẽ lên 1.

Bài 22: Chương trình chuyển đổi số thực thành số nguyên 16 bit, giá trị ngõ ra này có thể chuyển ra ngoài nhờ modul Analog. Chương trình được viết trong khối OB1 sử dụng FC106. Các ngõ vào/ra của FC106 tương tự như FC105 nhưng chỉ khác nhau ở hai ngõ IN và OUT, FC106 có ngõ vào IN là dạng số thực cần chuyển đổi, ngõ ra OUT có

dạng số nguyên 16 bit.



Nếu I0.0=1 thì FC106 hoạt động. Khí đó số thực có giới hạn từ 0 đến 100 được chuyển thành số nguyên có giá trị từ 0 đến 27648. Nếu I0.1=0 thì số thực có giá trị từ 0 đến 100 được chuyển thành số nguyên có giá trị từ –27648 đến 27648.