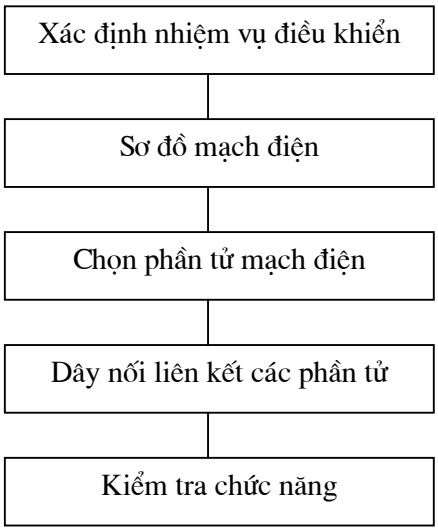
Chương1.Hê thống điều khiển.

1.1.Khái niêm hê thống điều khiển:

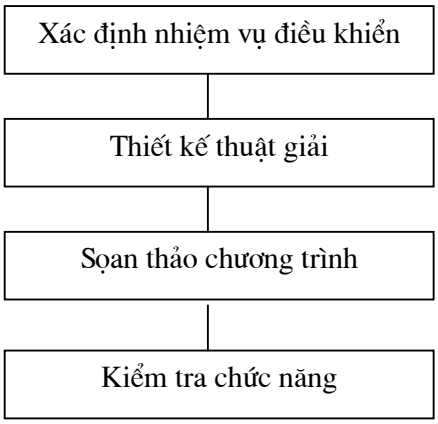
Trong công nghiệp yêu cầu tự động hoá ngày càng tăng, đòi hỏi kỹ thuật điều khiển phải đáp ứng được những yêu cầu đó. Để giải quyết được nhiệm vụ điều khiển người ta có thể thực hiện bằng hai cách: thực hiện bằng Rơle, khởi động từ ... hoặc thực hiện bằng chương trình nhớ. Hệ điều khiển bằng Rơle và hệ điều khiển bằng lập trình có nhớ khác nhau ở phần xử lý: thay vì dùng Rơle, tiếp điểm và dây nối trong phương pháp lập trình có nhớ chúng được thay bằng cách mạch điện tử. Như vậy thiết bị PLC làm nhiệm vụ thay thế phần mạch điện điều khiển trong khâu xử lý số liệu. Nhiệm vụ của sơ đồ mạch điều khiển sẽ được xác định bằng một số hữu hạn các bước thực hiện xác định gọi là "***chương trình"***. Chương trình này mô tả các bước thực hiện gọi là tiến trình điều khiển, tiến trình này được lưu vào bộ nhớ nên được gọi là "***điều khiển lập trình có nhớ"***. Trên cơ sở khác nhau của khâu xử lý số liệu ta có thể biểu diễn hai hệ điều khiển như sau:

Các bước thiết lập sơ đồ điều khiển bằng Rơle:



Hình 1-1:lưu đồ điều khiển dùng Rơle

Các bước thiết lập sơ đồ điều khiển bằng PLC:



Hình 1-2: Lưu đồ điều khiển bằng PLC

Khi thay đổi nhiệm vụ điều khiển người ta cần thay đổi mạch điều khiển bằng cách lắp lại mạch, thay đổi phần tử mới đối với hệ thống điều khiển bằng Rơle điện. Trong khi đó khi thay đổi nhiệm vụ điều khiển ta chỉ cần thay đổi chương trình soạn thảo đối với hệ điều khiển bằng lập trình có nhớ.

Sự khác nhau giữa hệ điều khiển bằng Rơle điện và lập trình có nhớ có thể minh hoạ bằng một ví dụ sau:

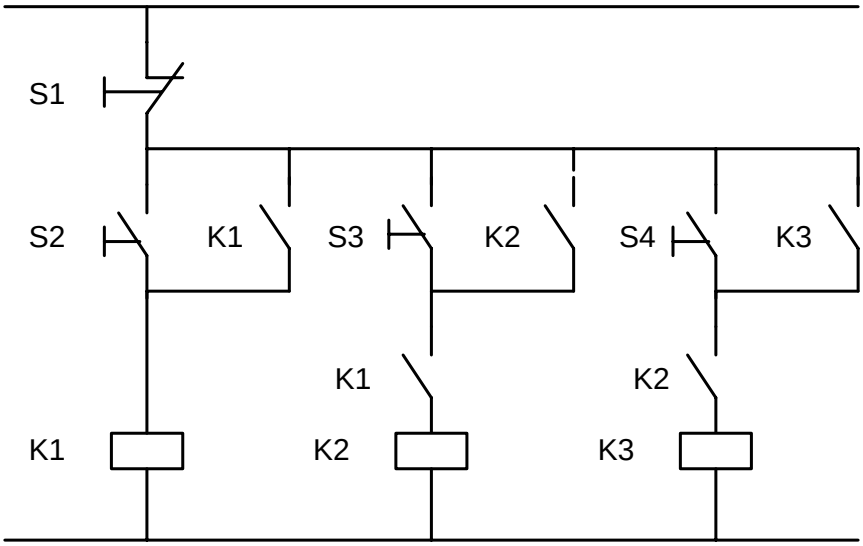
Điều khiển hệ thống 3 máy bơm nước qua 3 khởi động từ K1, K2, K3. Trình tự điều khiển như sau: Các máy bơm hoạt động tuần tự nghĩa là K1 đóng trước tiếp đến là K2 rồi cuối cùng là K3 đóng.

Để thực hiện nhiệm vụ theo yêu cầu trên mạch điều khiển ta thiết kế như sau:

Trong đó các nút ấn S1, S2, S3, S4 là các phần tử nhập tín hiệu.

Các tiếp điểm K1, K2, K3 và các mối liên kết là các phần xử lý.

Các khởi động từ K1, K2, K3 là kết quả xử lý.



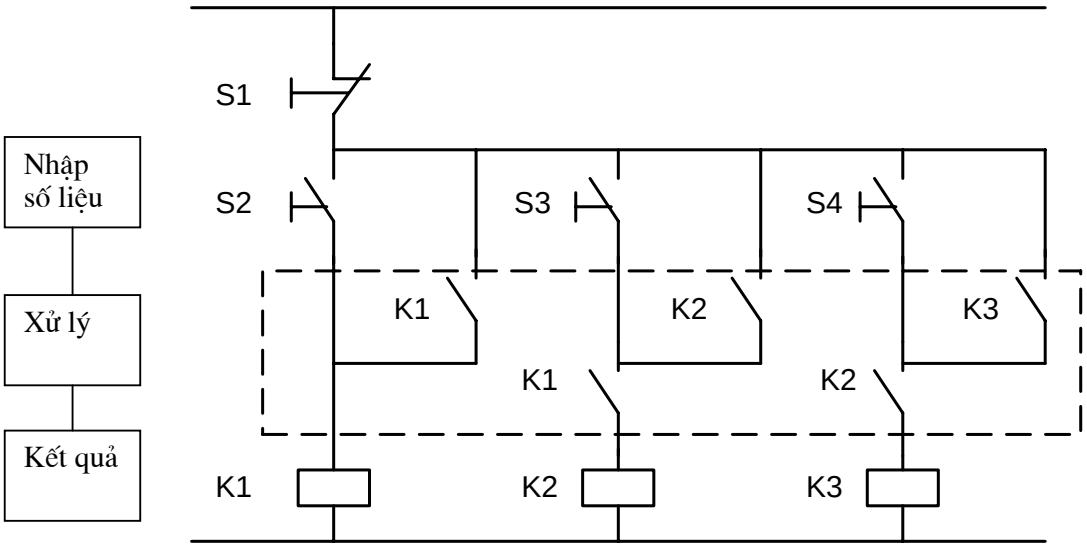
Hình 1-3:Sơ đồ điều khiển

Nếu ta thay bằng thiết bị điều khiển PLC ta có thể mô tả như sau:

-Tín hiệu vào: S1, S2, S3, S4 vẫn giữ nguyên.

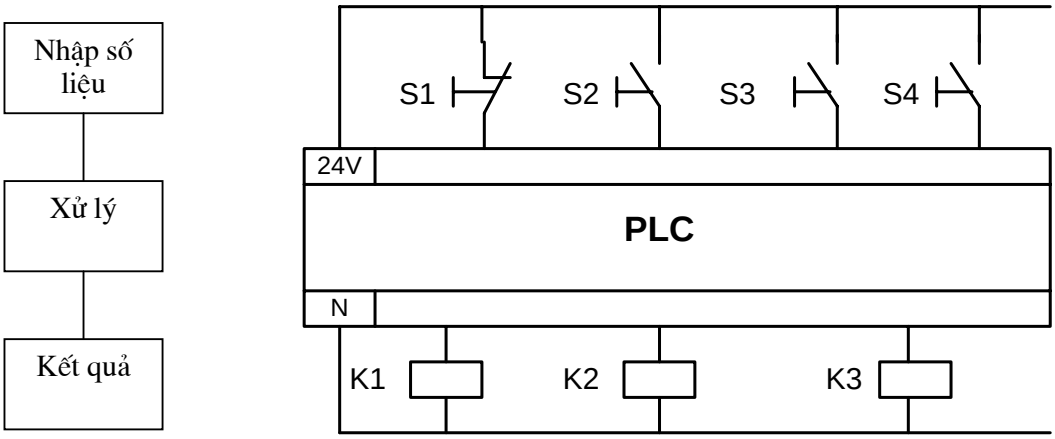
-Tín hiệu ra: K1, K2, K3 là các khởi động từ vẫn giữ nguyên.

-Phần tử xử lý:được thay thế bằng PLC.



Hình 1-4

Khi thực hiện bằng chương trình điều khiển có nhớ PLC ta chỉ cần thực hiện nối mạch theo sơ đồ sau:



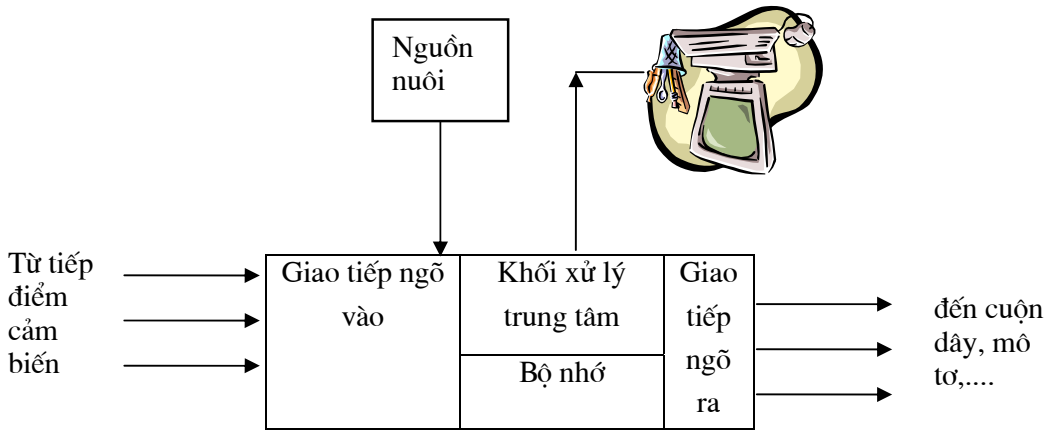
Hình 1-5:Sơ đồ nối dây thực hiện bằng PLC

Nếu bây giờ nhiệm vụ điều khiển thay đổi ví dụ như các bơm 1,2,3 hoạt động theo nguyên tắc là chỉ một trong số các bơm được hoạt động độc lập. Như vậy đối với mạch điều khiển dùng Rơle ta phải tiến hành lắp giáp lại toàn bộ mạch điều khiển, trong khi đó đối với mạch điều khiển dùng PLC thì ta lại chỉ cần soạn thảo lại chương trình rồi nạp lại vào CPU thì ta sẽ có ngay một sơ đồ điều khiển theo yêu cầu nhiệm vụ mới mà không cần phải nối lại dây trên mạch điều khiển.

Như vậy một cách tổng quát có thể nói hệ thống điều khiển PLC là tập hợp các thiết bị và linh kiện điện tử. Để đảm bảo tính ổn định, chính xác và an toàn.. trong quá trình sản xuất, các thiết bị này bao gồm nhiều chủng loại, hình dạng khác nhau với công suất từ rất nhỏ đến rất lớn. Do tốc độ phát triển quá nhanh của công nghệ và để đáp ứng được các yêu cầu điều khiển phức tạp nên hệ thống điều khiển phải có hệ thống tự động hoá cao. Yêu cầu này có thể thực hiện được bằng hệ lập trình có nhớ PLC kết hợp với máy tính, ngoài ra còn cần có các thiết bị ngoại vi khác như: Bảng điều khiển, động cơ, cảm biến, tiếp điểm, công tắc tơ,...

Khả năng truyền dữ liệu trong hệ thống rất rộng thích hợp cho hệ thống xử lý và cũng rất linh động trong các hệ thống phân phối .

Mỗi một thành phần trong hệ thống điều khiển có một vai trò quan trọng như được trình bày trong hình vẽ sau.



Hình 1-6: Mô hình hệ thống điều khiển PLC

Hệ thống PLC sẽ không cảm nhận được thế giới bên ngoài nếu không có các cảm biến, và cũng không thể điều khiển được hệ thống sản xuất nếu không có các động cơ, xy lanh hay các thiết bị ngoại vi khác nếu cần thiết có thể sử dụng các máy tính chủ tại các vị trí đặc biệt của dây chuyền sản xuất .

1.1.1.Hệ thống điều khiển PLC điển hình :

Trong hệ thống điều khiển PLC các phần tử nhập tín hiệu như : chuyển mạch, nút ấn, cảm biến, ... được nối với đầu vào của thiết bị PLC. Các phần tử chấp hành như : đèn báo, rơ le, công tắc tơ,... được nối đến lối ra của PLC tại các đầu nối.

Chương trình điều khiển PLC được soạn thảo dưới các dạng cơ bản (sẽ được trình bày ở phần sau) sẽ được nạp vào bộ nhớ bên trong PLC, sau đó tự động thực hiện tuần tự theo một chuỗi lệnh điều khiển được xác định trước .

Hệ còn cho phép công nhân vận hành thao tác bằng tay các tiếp điểm, nút dừng khẩn cấp để đảm bảo tính an toàn trong các trường hợp xảy ra sự cố.

1.1.2.Vai trò của PLC:

PLC được xem như trái tim trong một hệ thống điều khiển tự động đơn lẻ với chương trình điều khiển được chứa trong bộ nhớ của PLC, PC thường xuyên kiểm tra trạng thái của hệ thống thông qua các tín hiệu hồi tiếp từ thiết bị nhập để từ đó có thể đưa ra những tín hiệu điều khiển tương ứng đến các thiết bị xuất.

PLC có thể được sử dụng cho những yêu cầu điều khiển đơn giản và được lập đi lập lại theo chu kỳ, hoặc liên kết với máy tính chủ khác hoặc máy tính chủ thông qua một kiểu hệ thống mạng truyền thông để thực hiện các quá trình xử lý phức tạp.

Tín hiệu vào.

Mức độ thông minh của một hệ thống điều khiển phụ thuộc chủ yếu vào khả năng của PLC để đọc được các dữ liệu khác nhau từ các cảm biến cũng như bằng các thiết bị nhập bằnh tay .

Tiêu biểu cho các thiết bị nhập bằng tay như : Nút ấn, bàn phím và chuyển mạch. Mặt khác, để đo, kiểm tra chuyển động, áp suất, lưu lượng chất lỏng ... PLC phải nhận các tín hiệu từ các cảm biến. Ví dụ : Tiếp điểm hành trình, cảm biến quang điện ... tín hiệu đưa vào PLC có thể là tín hiệu số (Digital) hoặc tín hiệu tương tự (Analog), các tín hiệu này được giao tiếp với PLC thông qua các Modul nhận tín hiệu vào khác nhau khác nhau DI (vào số) hoặc AI (vào tương tự)....

Đối tượng điều khiển .

Một hệ thống điều khiển sẽ không có ý nghĩa thực tế nếu không giao tiếp được với thiết bị xuất, các thiết bị xuất thông dụng như: Môtơ, van, Rơle, đèn báo, chuông điện,... cũng giống như thiết bị nhập, các thiết bi xuất được nối đến các ngõ ra của Modul ra (Output). Các Modul ra này có thể là DO (Ra số) hoặc AO (ra tương tự).

1.1.3.Cấu tạo PLC.

Thiết bị điều khiển lập trình PLC bao gồm khối xử lý trung tâm (CPU) trong đó có chứa chương trình điều khiển và các Modul giao tiếp vào/ra có nhiệm vụ liên kết trực tiếp đến các thiết bị vào/ra, sơ đồ khối cấu tạo PLC được vẽ như hình 1-6.

Khối xử lý trung tâm : là một vi xử lý điều khiển tất cả các hoạt động của PLC như: Thực hiện chương trình, xử lý vào/ra và truyền thông với các thiết bị bên ngoài.

Bộ nhớ: có nhiều các bộ nhớ khác nhau dùng để chứa chương trình hệ thống là một phần mềm điều khiển các hoạt động của hệ thống, sơ đồ LAD, trị số của

Timer, Counter được chứa trong vùng nhớ ứng dụng, tùy theo yêu cầu của người dùng có thể chọn các bộ nhớ khác nhau:

- Bộ nhớ ROM: là loại bộ nhớ không thay đổi được, bộ nhớ này chỉ nạp được một lần nên ít được sử dụng phổ biến như các loại bộ nhớ khác .
- Bộ nhớ RAM: là loại bộ nhớ có thể thay đổi được và dùng để chứa các chương trình ứng dụng cũng như dữ liệu, dử liệu chứa trong Ram sẽ bị mất khi mất điện. Tuy nhiên, điều này có thể khắc phục bằng cách dùng Pin.
- Bộ nhớ EPROM: Giống như ROM, nguồn nuôi cho EPROM không cần dùng Pin, tuy nhiên nội dung chứa trong nó có thể xoá bằng cách chiếu tia cực tím vào một cửa sổ nhỏ trên EPROM và sau đó nạp lại nội dung bằng máy nạp.
- Bộ nhớ EEPROM: kết hợp hai ưu điểm của RAM và EPROM, loại này có thể xóa và nạp bằng tín hiệu điện. Tuy nhiên số lần nạp cũng có giới hạn.

1.1.4.Ưu nhược điểm của hệ thống :

Trong giai đoạn đầu của thời kỳ phát triển công nghiệp vào khoảng năm 1960 và 1970, yêu cầu tự động của hệ điều khiển được thực hiện bằng các Rơle điện từ nối nối với nhau bằng dây dẫn điện trong bảng điều khiển, trong nhiều trường hợp bảng điều khiển có kích thước quá lớn đến nỗi không thể gắn toàn bộ lên trên tường và các dây nối cũng không hoàn toàn tốt vì thế rất thường xảy ra trục trặc trong hệ thống. Một điểm quan trong nữa là do thời gian làm việc của các Rơle có giới hạn nên khi cần thay thế cần phải ngừng toàn bộ hệ thống và dây nối cũng phải thay mới cho phù hợp, bảng điều khiển chỉ dùng cho một yêu cầu riêng biệt không thể thay đổi tức thời chức năng khác mà phải lắp giáp lại toàn bộ, và trong trường hợp bảo trì cũng như sửa chữa cần đòi hỏi thợ chuyên môn có tay nghề cao. Tóm lại hệ điều khiển Rơle hoàn toàn không linh động.

**Tóm tắt nhược điểm của hệ thống điều khiển dùng Rơle:*

- Tốn kém rất nhiều dây dẫn .
- Thay thế rất phức tạp.
- Cần công nhân sửa chữa tay nghề cao.
- Công suất tiêu thụ lớn .
- Thời gian sửa chữa lâu.
- Khó cập nhật sơ đồ nên gây khó khăn cho công tác bảo trì cũng như thay thế.

Ưu điểm của hệ điều khiển PLC:

Sự ra đời của hệ điều khiển PLC đã làm thay đổi hẳn hệ thống điều khiển cũng như các quan niệm thiết kế về chúng, hệ điều khiển dùng PLC có nhiều ưu điểm như sau:

- Giảm 80% Số lượng dây nối.
- Công suất tiêu thụ của PLC rất thấp .
- Có chức năng tự chuẩn đoán do đó giúp cho công tác sửa chữa được nhanh chóng và dễ dàng.
- Chức năng điều khiển thay đổi dễ dàng bằng thiết bị lập trình (máy tính, màn hình) mà không cần thay đổi phần cứng nếu không có yêu cầu thêm bớt các thiết bị xuất nhập.
- Số lượng Rơle và Timer ít hơn nhiều so với hệ điều khiển cổ điển.
- Số lượng tiếp điểm trong chương trình sử dụng không hạn chế.
- Thời gian hoàn thành một chu trình điều khiển rất nhanh (vài mS) dẫn đến tăng cao tốc độ sản xuất .
- Chi phí lắp đặt thấp .
- Độ tin cậy cao.
- Chương trình điều khiển có thể in ra giấy chỉ trong vài phút giúp thuận tiện cho vấn đề bảo trì và sửa chữa hệ thống.

1.1.5.ứng dụng của hệ thống điều khiển PLC:

Từ các ưu điểm nêu trên, hiện nay PLC đã được ứng dụng trong rất nhiều lĩnh vực khác nhau trong công nghiệp như:

- Hệ thống nâng vận chuyển.
- Dây chuyền đóng gói.
- Các ROBOT lắp giáp sản phẩm .
- Điều khiển bơm.
- Dây chuyền xử lý hoá học.
- Công nghệ sản xuất giấy .
- Dây chuyền sản xuất thuỷ tinh.
- Sản xuất xi măng.
- Công nghệ chế biến thực phẩm.
- Dây chuyền chế tạo linh kiện bán dẫn.
- Dây chuyền lắp giáp Tivi.
- Điều khiển hệ thống đèn giao thông.

- Quản lý tự động bãi đậu xe.
- Hệ thống báo động.
- Dây truyền may công nghiệp.
- Điều khiển thang máy.
- Dây chuyền sản xuất xe Ôtô.
- Sản xuất vi mạch.
- Kiểm tra quá trình sản xuất .

1.2 Hê thống điều khiển PLC S7-300.

1.2.1.Cấu trúc phần cứng của hê thống PLC S7-300.

Thông thường, để tăng tính mềm dẻo trong ứng dụng thực tế mà ở đó phần lớn các đối tượng điều khiển có số tín hiệu đầu vào, đầu ra cũng như chủng loại tín hiệu vào/ra khác nhau mà các bộ điều khiển PLC được thiết kế không bị cứng hoá về cấu hình. Chúng được chia nhỏ thành các modul. Số các Modul được sử dụng nhiều hay ít tuỳ theo từng yêu cầu công nghệ, song tối thiểu bao giờ cũng phải có một Modul chính là các modul CPU, các modul còn lại là các modul truyền nhận tín hiệu đối với đối tượng điều khiển, các modul chức năng chuyên dụng như PID, điều khiển động cơ, Chúng được gọi chung là Modul mở rộng. Tất cả các modul được gá trên những thanh ray (*RACK*).

Modul CPU:

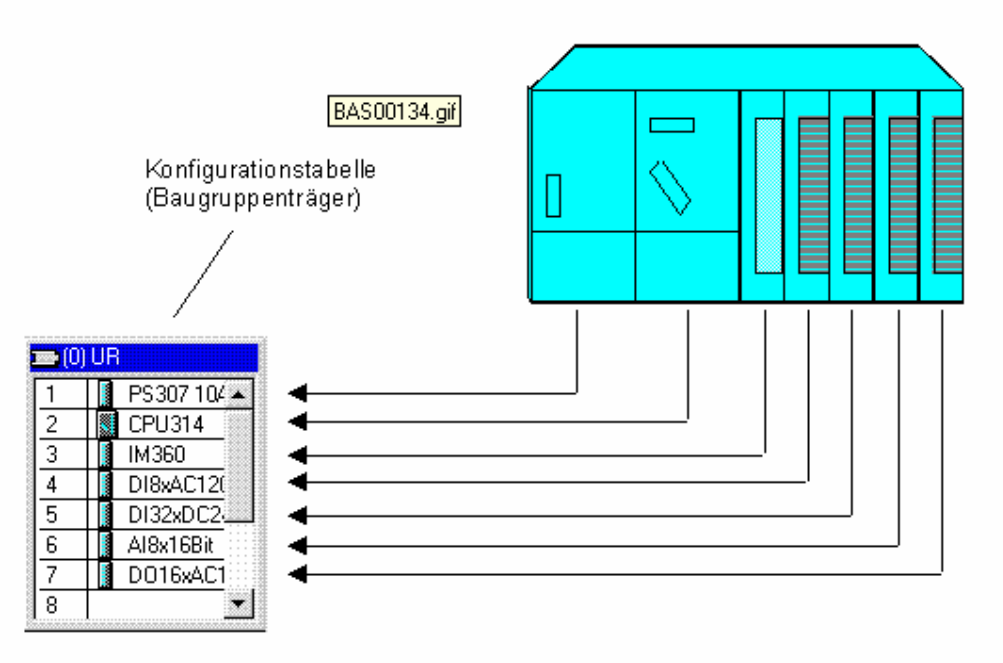
Là modul có chứa bộ vi xử lý, hệ điều hành, bộ nhớ, các bộ thời gian, bộ đếm, cổng truyền thông (chuẩn tryền ***RS485***) và có thể còn có một vài cổng vào ra số (Digital). Các cổng vào ra có trên modul CPU được gọi là cổng vào ra onboard.

Trong PLC S7-300 có nhiều loại modul CPU khác nhau. Nói chung chúng được đặt tên theo bộ vi xử lý có trong nó như: CPU312, modul CPU 314, Modul CPU 315,... Những modul cùng sử dụng một loại bộ vi xử lý, nhưng khác nhau về cổng vào/ra **onboard** cũng như các khối làm việc đặc biết được tích hợp sẵn trong thư viện của hệ điều hành phục vụ việc sử dụng các cổng vào/ra **onboard** này sẽ được phân biệt với nhau trong tên gọi bằng cách thêm cụm chữ cái IFM (Intergated Function Module) ví dụ CPU 312IM, modul CPU 314 IFM.

Ngoài ra có các loại modul CPU với hai cổng truyền thông, trong đó cổng truyền thông thứ hai có chức năng chính là việc phục vụ nối mạng phân tán. Tất nhiên được cài sẵn trong hệ điều hành các loại Modul CPU được phân biệt với các CPU khác bằng thêm cụm từ DP trong tên gọi. Ví dụ Modul CPU 315-DP.

Modul mở rộng: các modul mở rộng được chia làm 5 loại chính:

1/ PS(*Power supply*): modul nguồn nuôi. Có 3 loại 2A ,5A và 10A.



Hình 1-7:Sơ đồ bố trí một trạm PLC(S7-300).

2/ SM: Modul mở rộng cổng rín hiệu vào ra , bao gồm:

a) DI(*Digital input*): Modul mở rộng cổng vào số. Số các cổng vào của modul này có thể là 8, 16, 32 tuỳ thuộc vào từng loại modul.

b) DO(*Digital output*) Modul mở rộng cổng ra số. Số các cổng ra của modul này có thể là 8, 16, 32 tuỳ thuộc vào từng loại modul.

c) DI/DO: (*Digital input/ Digital output*): modul mở rổng các cổng vào/ra số số các cổng vào/ra có thể là 8 vào/8 ra hoặc 16 vào/16 ra tuỳ thuộc vào từng loại modul.

d) AI(*Analog Input*): Modul mở rổng các cổng vào tương tự. Về bản chất chúng chính là những bộ chuyển đổi tương tự-số (AD), tức là mỗi tín hiệu tương tự được chuyển thành một tín hiệu số (nguyên) có độ dài 12 bít, số các cổng vào có thể là 2, 4 hoặc 8 tuỳ thuộc vào từng loại Modul.

e) AO(*Analog ouput*): Modul mở rộng các cổng ra tín hiệu tương tự. Chúng chính là các bộ chuyển đổi số - tương tự (DA). Số các cổng ra tương tự có thể là 2 hoặc 4 tuỳ thuộc từng loại modul.

f) AI/AO (*Analog input/Analog output*): Modul mở rộng các cổng vào ra tương tự. Số các cổng có thể là 4 vào/2 ra hoặc 4 vào/4 ra tuỳ thuộc vào tùng loại modul.

3/ IM (*Interface module*): Modul ghép nối. Đây là loại modul chuyên dụng có nhiệm vụ nối từng nhóm các modul mở rộng lại với nhau thành một khối và được quản lý chung bởi một modul CPU. Thông thường các modul mở rộng được gá liền với nhau trên một thanh đỡ gọi là Rack. Trên mỗi một Rack chỉ có thể gá được nhiều nhất 8 modul mở rộng (không kể modul CPU, Modul nguồn nuôi). Một modul PU S7-300 có thể làm việc trực tiếp được với nhiều nhất 4 RACKS và các Racks này phải được nối với nhau bằng modul IM.

4/ FM (*Function modul*): modul có chức năng điều khiển riêng , ví dụ Modul chức năng điều khiển động cơ bước , modul điều khiển động cơ Servo, modul PID, modul điều khiển vòng kín.

5/ CP (*communication modul*): Modul phục vụ truyền thông trong mạng giữa các PLC với nhau hoặc giữa PLC với máy tính.

1.2.2.Kiểu dữ liệu và phân chia bộ nhớ:

1-Phân loai:

Một chương ttrình trong S7-300 có thể sử dụng các kiểu dữ liệu sau:

1/ BOOL: với dung lượng là 1 bit và có giá trị là 0 hoặc 1 (đúng hoặc sai). Đây là kiểu dữ liệu biến có hai giá trị.

2/ BYTE: gồm 8 bits, thường được dùng để biểu diễn một số nguyên dương trong khoảng từ 0 đến 255 hoặc mã ASCII của một ý tự.

Ví dụ: B#16#14 nghĩa là số nguyên 14 viết theo hệ đếm cơ số 16 có độ dài 1 byte.

3/ WORD: gồm 2 byte, để biểu diễn số nguyên dương từ 0 đến 65535 (2^{16} - 1).

4/DWORD: Là từ kép có giá trị là: 0 đến 2^{32}-1.

5/ INT: cũng có dung lượng là 2 bytes, dùng để biểu diễn một số nguyên trong khoảng -32768 đến 32767 hay (2^{-15}...2^{15}-1).

6/ DINT: gồm 4 bytes, dùng để biểu diễn số nguyên từ -2147483648 đến 2147483647 hay: (2^{-31}....2^{31}-1).

7/ REAL: gồm 4 bytes, dùng để biểu diễn một số thực dấu phảy động có giá trị là: -3,4E^{38}.....3,4E^{38}.

Ví dụ: 1.234567e+13

8/ S5t (hay S5Time): khoảng thời gian, được tính theo giờ/phút/giây: (-2^{-31}+ 2^{31}-1 ms).

Ví dụ: S5t#2h_3m_0s_5ms.

Đây là lệnh tạo khoảng thời gian la 2 tiếng ba phút và 5 mili giây.

9/TOD: Biểu diễn giá trị tức thời tính theo Giờ/phút/giây.

Ví dụ: TOD#5:30:00 là lệnh khai báo giá trị thời gian trong ngày là 5 giờ 30 phút.

10/ DATE: Biểu diễn thời gian tính theo năm / ngày / tháng.

Ví dụ: DATE#2003-6-12

Là lệnh khai báo ngày12 tháng 6 năm 2003.

11/ CHAR: biểu diễn một hoặc nhiều ký tự (nhiều nhất là 4 ký tự) (ASCII - code).

Ví dụ: ABCD

2-Sử dụng và khai báo các dạng tín hiệu:

Trong quá trình thực hiện cấu trúc của tín hiệu số được biểu diễn dưới dạng:

1/ Bit : (ví dụ I0.0) dùng để biểu diễn số nhị phân (có 2 giá trị 1 hoặc 0).

1

2/ Byte : (ví dụ MB0) Một Byte gồm có 8 bits. Ví dụ giá trị của 8 cổng vào (IB0) hoặc 8 cổng ra (QB1),... được gọi là một byte:

0	0	1	1	0	1	0	1

3/ Word: (ví dụ MW0= MB0 + MB1) Một Word gồm có 2 Byte như vậy một Word có độ dài 16 bits.

0	0	1	0	1	1	1	0	0	0	1	0	1	1	1	0

Byte 0 *Byte 1*

4/ Doppelword: (ví dụ MD0 = MW0 + MW2): có độ dài 2 từ hoặc 4 Byte tức là 32 bits.

1.2.3.Cấu trúc bô nhớ của CPU của S7-300:

Được chia ra làm 3 vùng chính:

1) Vùng chứa chương trình ứng dụng: vùng nhớ chương trình được chia làm 3 miền:

   a/ OB: Miền chứa chương trình tổ chức (các chương trình này sẽ được giới thiệu ở mục 1.2.5).

   b/ FC: (*Funktion*): miền chứa chương trình con được tổ chức thành hàm có biến hình thức để trao đổi dữ liệu với chương trình đã gọi nó.

   c/ FB: (*Funktion Block*): Miền chứa chương trình con, được tổ chức thành hàm và có khả năng trao đổi dữ liệu với bất cứ một khối chương trình nào khác. Các dữ liệu này phải được xây dựng thành một khối dữ liệu riêng (gọi là *DB-Data block*).

2) Vùng chứa các tham số của hệ điều hành và chương trình ứng dụng, được phân chia thành 7 miền khác nhau, bao gồm:

a. I (*Procees image input*): miền bộ đệm các dữ liệu cổng vào số. Trước khi thực hiện chương trình, PLC sẽ đọc giá trị logic của tất cả các đầu vào và cất giữ chúng trong vùng nhớ I. Thông thường chương trình ứng dụng không đọc trực tiếp trạng thái logic của cổng vào số mà chỉ lấy dữ liệu của cổng vào từ bộ đệm I.

b. Q (*Process image output*): miền bộ đệm các cổng ra số. Kết thúc giai đoạn thực hiện chương trình sẽ chuyển giá trị logic của bộ đệm tới các cổng ra số. Thông thường không trực tiếp gán giá trị tới tận cổng ra mà chỉ chuyển chúng vào bộ đệm Q.

c. M: Miền các biến cờ. Chương trình ứng dụng sử dụng vùng nhớ này để lưu giữ các tham số cần thiết và có thể truy cập nó theo bit (M), byte (MB), từ (MW) hay từ kép (MD).

d. T: Miền nhớ phục vụ bộ thời gian (TIME) bao gồm việc lưu giữ giá trị thời gian đặt trước (PV-preset value), giá trị đếm thời gian tức thời (CV-Curren value) cũng như các giá trị logic đầu ra của bộ thời gian.

e. C: Miền nhớ phục vụ bộ đếm (counter) bao gồm việc lưu giữ giá trị đặt trước (PV), và giá trị đếm tức thời (CV) và giá trị logic đầu ra của bộ đếm.

f. PI: Miền địa chỉ cổng vào của các modul tương tự. Các giá trị tương tự tại cổng vào của modul tương tự sẽ được đọc và chuyển tự động theo những địa chỉ. Chương trình ứng dụng có thể truy nhập miền nhớ PI theo từng byte (PIB), từng từ (PIW) hoặc theo từ kép (PID).

g. PQ: miền địa chỉ cổng ra cho các modul tương tự. Các gía trị theo những địa chỉ này được modul tương tự chuyển tới các cổng ra tương tự. Chương trình ứng dụng có thể truy cập miền nhớ PQ theo từng byte (PQB), từng từ (PQW) hay từng từ kép (PQD)

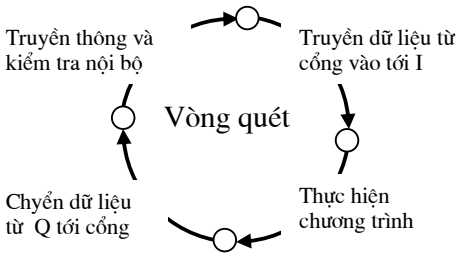
3) Vùng chứa các khối dữ liệu: được chia làm hai loại:

a. DB (*Data block*): miền chứa các dữ liệu được tổ chức thành khối. Kích thước cũng như số lượng do người sử dụng qui định, phù hợp với từng bài toán điều khiển. Chương trình có thể truy cập miền này theo từng bit (DBX), byte (DBB), từ (DBW) hoặc từ kép (DBD).

b. L (*Local data block*) : miền giữ liệu địa phương, được các khối chương trình OB, FB, FC tổ chức và sử dụng cho các biến nháp tức thời và trao đổi giữ liệu của biến hình thức của chương trình với những khối chương trình đã gọi nó. Nội dung của một số dữ liệu trong miền nhớ này sẽ bị xoá khi kết thúc chương trình tương ứng trong OB, FB, FC. Miền này có thể truy cập từ chương trình theo bit (L), byte (LB), từ (LW) hay từ kép (LD).

1.2.4.Vòng quét của chương trình:

SPS (PLC) thực hiện các công việc (bao gồm cả chương trình điều khiển) theo chu trình lặp. Mỗi vòng lặp được gọi là một vòng quét (scancycle). Mỗi vòng quét được bắt đàu bằng việc chuyển dữ liệu từ các cổng vào số tới vùng bộ đệm ảo I, tiếp theo là giai đoạn thực hiện chương trình. Trong từng vòng quét , chương trình được thực hiện từ lệnh đầu tiên đến lệnh kết thúc của khối OB1. Sau giai đoạn thực hiện chương trình là giai đoạn chuyển các nội dung của bộ đệm ảo Q tới các cổng ra số. Vòng quét được kết thúc bằng giai đoạn xử lý các yêu cầu truyền thông (nếu có) và kiểm tra trạng thái của CPU. Mỗi vòng quét có thể mô tả như sau:



Hình1-8: Quá trình hoạt động của một vòng quét.

<u>Chú ý</u> : Bộ đệm I và Q không liên quan tới các cổng vào/ra tương tự nên các lệnh truy nhập cổng tương tự được thực hiện trực tiếp với cổng vật lý chứ không thông qua bộ đệm.

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

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

Nếu sử dụng các khối chương trình đặc biệt có chế độ ngắt, ví dụ khối OB40, OB80,... Chương trình của các khối đó sẽ được thực hiện trong vòng quét khi xuất hiện tín hiệu báo ngắt cùng chủng loại. Các khối chương trình này có thể thực hiện tại mọi vòng quét chứ không phải bị gò ép là phải ở trong giai đoạn thực hiện chương trình. Chẳng hạn một tín hiệu báo ngắt xuất hiện khi PLC đang ở giai đoạn truyền thông và kiểm tra nội bộ, PLC sẽ tạm dừng công việc truyền thông, kiểm tra, để thực hiện ngắt như vậy, thời gian vòng quét sẽ càng lớn khi càng có nhiều tín hiệu ngắt xuất hiện trong vòng quét. Do đó để nâng cao tính thời gian thực cho chương trình điều khiển, tuyệt đối không nên viết chương trình xử lý ngắt quá dài hoặc quá lạm dụng việc sử dụng chế độ ngắt trong chương trình điều khiển.

Tại thời điểm thực hiện lệnh vào/ra, thông thường lệnh không làm việc trực tiếp với cổng vào/ra mà chỉ thông qua bộ nhớ đệm của cổng trong vùng nhớ tham số. Việc truyền thông giữa bộ đêm ảo với ngoại vi trong giai đoạn 1 và 3 do hệ điều hành CPU quản lý. ở một số modul CPU, khi gặp lệnh vào/ra ngay lập tức hệ thống sẽ cho dừng mọi công việc khác, ngay cả chương trình xử lý ngắt, để thực hiện với cổng vào/ra.

1.2.5. Những khối OB đăc biệt:

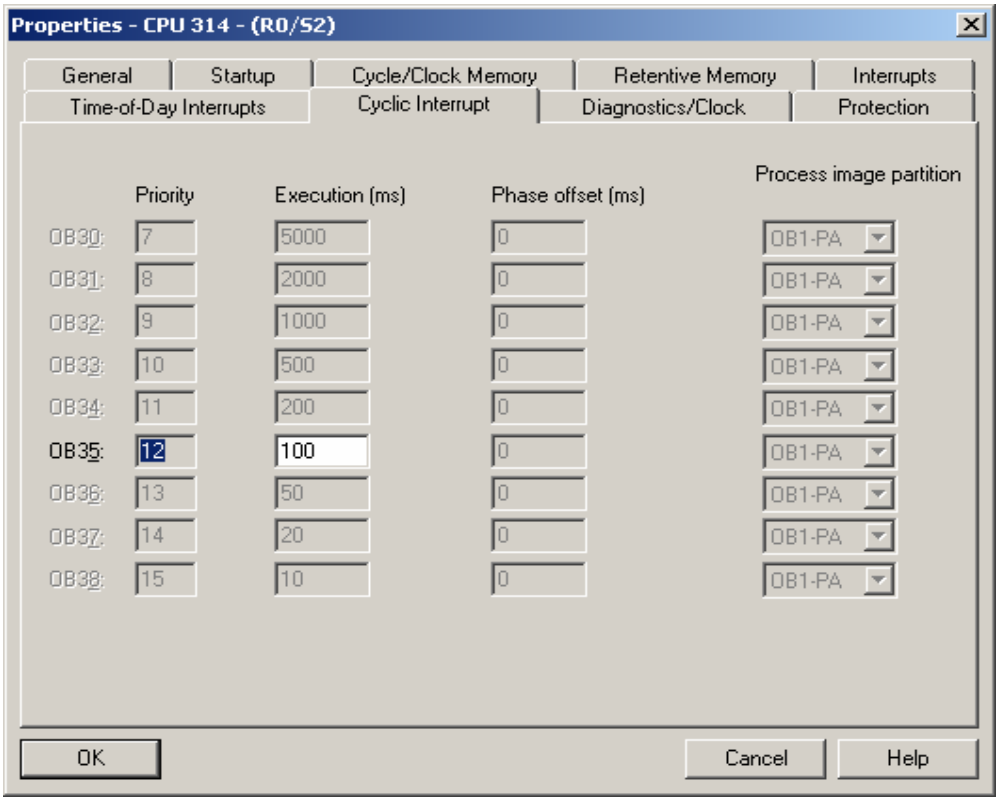
Khối OB1 có chức năng quản lý chính trong toàn bộ chương trình, có nghĩa là nó sẽ thực hiện một cách đều đặn ở từng vòng quét trong khi thực hiện chương trình. Ngoài ra Step7 còn có rất nhiều các khối OB đặc biệt khác và mỗi khối OB đó có một nhiệm vụ khác nhau, ví dụ các khối OB chứa các chương

trình ngắt của các chương trình báo lỗi ,....Tuỳ thuộc vào từng loại CPU khác nhau mà có các khối OB khác nhau. Ví dụ các khối OB đặc biệt.

1. OB10: (*Time of Day Interrupt*): Chương trình trong khối OB10 sẽ được thực hiện khi giá trị của đồng hồ thời gian thực nằm trong một khoảng thời gian đã qui định. OB10 có thể được gọi một lần, nhiều lần cách đều nhau từng phút, từng giờ, từng ngày,....Việc qui định thời gian hay số lần gọi OB10 được thực hiện bằng chương trình hệ thống SFC28 hoặc trong bảng tham số modul CPU nhờ phần mềm Step7.
2. OB20: (*Time Delay Interrupt*): chương trình trong khối OB20 sẽ được thực hiện sau một khoảng thời gian trễ đặt trước kể từ khi gọi chương trình hệ thống SFC32 để đăt thời gian trễ.
3. OB35: (*Cyclic Interrupt*): Chương trình OB35 sẽ được thực hiện cách đều nhau một khoảng thời gian cố định. Mặc định khoảng thời gian này là 100ms, xong ta có thể thay đổi trong bảng đặt tham số cho CPU nhờ phần mềm Step7.
4. OB40 (*Hardware Interrupt*): Chương trình trong khối OB40 sẽ được thực hiện khi xuất hiện một tín hiệu báo ngắt từ ngoại vi đưa vào CPU thông qua các cổng vào/ra số onboard đặc biệt, hoặc thông qua các modul SM, CP, FM.
5. OB80: (*cycle Time Fault*): Chương trình sẽ được thực hiện khi thời gian vòng quét (scan time) vượt qua khoảng thời gian cực đại đã qui định hoặc khi có một tín hiệu ngắt gọi một khối OB nào đó mà khối OB này chưa kết thúc ở lần gọi trước. Mặc định, scan time cực đại là 150ms, nhưng có thể thay đổi tham số nhờ phần mềm Step7.
6. OB81(*Power Supply Fault*): nếu có lỗi về phần nguồn cung cấp thì sẽ gọi chương trình trong khối OB81.
7. OB82: (*Diagnostic Interrupt*) chương trình trong khối này sẽ được gọi khi CPU phát hiện có lỗi từ các modul vào/ra mở rộng. Với điều kiện các modul vào/ra này phải có chức năng tự kiểm tra mình.
8. OB85 (*Not Load Fault*): CPU sẽ gọi khối OB85 khi phát hiện chương trình ứng dụng có sử dụng chế độ ngắt nhưng chương trình xử lý tín hiệu ngắt lại không có trong khối OB tương ứng.
9. OB87 (*Communication Fault*): Chương trình trong khối này sẽ được gọi khi CPU phát hiện thấy lỗi trong truyền thông.
10. OB100 (*Start Up Information*): Khối này sẽ được thực hiện một lần khi CPU chuyển trạng thái từ STOP sang trạng thái RUN.

11. OB121: (*Synchronouns error*): Khối này sẽ được gọi khi CPU phát hiện thấy lỗi logic trong chương trình như đổi sai kiểu dữ liệu hoặc lỗi truy nhập khối DB, FC, FB không có trong bộ nhớ của CPU.
12. OB122 (*Synchronouns error*): Khối này sẽ được thực hiện khi CPU phát hiện thấy lỗi truy nhập Modul trong chương trình, ví dụ trong chương trình có lệnh truy nhập modul mở rộng nhưng lại không có modul này.

Để thực hiện thay đổi các chức năng của các khối OB trong CPU ta chỉ cần kích đúp chuột trái vào vị tí CPU trong bảng cấu hình cứng của Project khi đó trên màn hình sẽ xuất hiện một cửa sổ như sau:



Hình 1-9

Chú ý không phải tất cả các CPU đều có các khối OBs như đã giới thiệu. Số lượng và chủng loại khối OB tuỳ thuộc vào từng loại CPU.

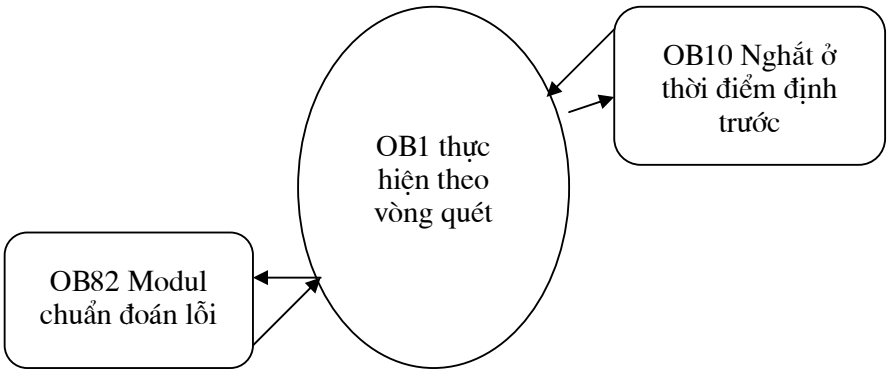
Chương 2: Kỹ thuật lập trình:

2.1.Giới thiệuchung:

2.1.1.Lập trình tuyến tính và lập trình có cấu trúc:

Phần bộ nhớ của CPU dành cho chương trình ứng dụng có tên gọi là logic Block. Như vậy logic block là tên chung để gọi tất cả các khối bao gồm những khối chương trình tổ chức OB, khối chương trình FC, khối hàm FB. Trong các loại khối chương trình đó thì chỉ có khối duy nhất khối OB1 được thực hiện trực tiếp theo vòng quét. Nó được hệ điều hành gọi theo chu kỳ lặp với khảng thời gian không cách đều nhau mà phụ thuộc vào độ dài của chương trình. Các loại khối chương trình khác không tham gia vào vòng quét.

Với tổ chức chương trình như vậy thì phần chương trình trong khối OB1 có đầy đủ điều kiện của một chương trình điều khiển thời gian thực và toàn bộ chương trình ứng dụng có thể chỉ cần viết trong OB1 là đủ như hình vẽ sau. Cách tổ chức chương trình với chỉ một khối OB1 duy nhất như vậy được gọi là lập trình tuyến tính.



Hình 2-1: Sơ đồ khối kiểu lập trình tuyến tính

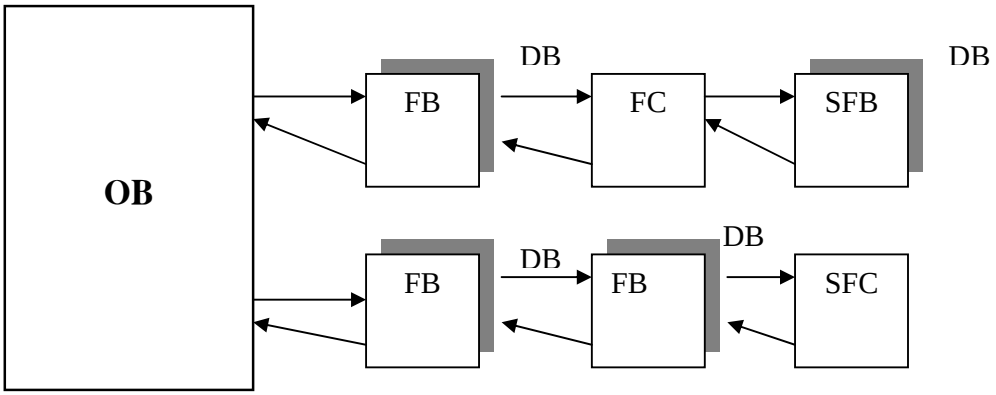
Khối OB1 được hệ thống gọi xoay vòng liên tục theo vòng quét.

Các khối OB khác không tham gia vào vòng quét được gọi bằng những tín hiệu báo ngắt. S7-300 có nhiều tín hiệu báo ngắt như tín hiệu báo ngắt khi có sự cố nguồn nuôi, có sự cố chập mạch ở các modul mở rộng, tín hiệu báo ngắt theo chu kỳ thời gian, và mỗi loại tín hiệu báo ngắt như vậy cũng chỉ có khả năng gọi

một khối OB nhất định. Ví dụ tín hiệu báo ngắt sự cố nguồn nuôi chỉ gọi khối OB81, tín hiệu báo ngắt truyền thông chỉ gọi khối OB87.

Mỗi khi xuất hiện tín hiệu báo ngắt hệ thống sẽ dừng công việc đang thực hiện lại, chẳng hạn như tạm dừng việc thực hiện chương trình trong OB1, và chuyển sang thực hiện chương trình xử lý ngắt tong các khối OB tương ứng. Ví dụ khi đang thực hiện chương trình trong khối OB1 mà xuất hiện ngắt báo sự cố truyền thông, hệ thống sẽ tạm dừng việc thực hiện chương trình trong OB1 lại để gọi chương trình trong khối truyền thông OB87. Chỉ sau khi đã thực hiện xong chương trình trong khối OB87 thì hệ thống mới quay trở về hực hiện tiếp tục phần chương trình còn lại trong OB1.

Với kiểu lập trình có cấu trúc thì khác vì toàn bộ chương trình điều khiển được chia nhỏ thành các khối FC và FB mang một nhiệm vụ cụ thể riêng và được quản lý chung bởi những khối OB. Kiểu lập trình này rất phù hợp cho những bài toán phức tạp, nhiều nhiệm vụ và lại rất thuận lợi cho việc sửa chữa sau này.



Hình 2-2: Sơ đồ kiểu lập trình có cấu trúc.

OB: Organization Block
FB: Function Block
FC: Function
SFB: System Function block
SFC: System function
SDB: System Data Block
DB: Data block
Chú ý: Bao giờ ***FB*** cũng sử dụng chung với ***DB***.

2.1.2.Qui trình thiết kế chương trình điều khiển dùng PLC:

Qui trình thiết kế hệ thống điều khiển dùng PLC bao gồm các bước sau:

1.Xác định qui trình điều khiển:

Điều đầu tiên cần biết là đối tượng điều khiển của hệ thống, mục đích cính của PLC là phải điều khiển được các thiết bị ngoại vi. Các chuyển động của đối tượng điều khiển được kiểm tra thường xuyên bởi các thiết bị vào, các thiết bị nạy gửi tín hiệu đến PLC và tiếp theo đó PLC sẽ đưa tín hiêu điều khiển đến các thiết bị để điều khiển chuyển động của đối tượng. Để đơn giản, qui trình điều khiển có thể mô tả theo lưu đồ (hình vẽ 2-3).

2.xác định tín hiệu vào ra:

Bước thứ hai là phải xác định vị trí kết nối giữa các thiết bị vào ra với PLC. Thiết bị vào có thể là tiếp điểm, cảm biến, Thiết bị ra có thể là Rơle điện từ, Motor, đèn, Mỗi vị trí kết nối được đánh số tương tự ứng với PLC sử dụng.

3.Soan thảo chương trình:

Chương trình điều khiển được soạn thảo dưới dạng lưu đồ hình thang như đã trình bày ở bước 1.

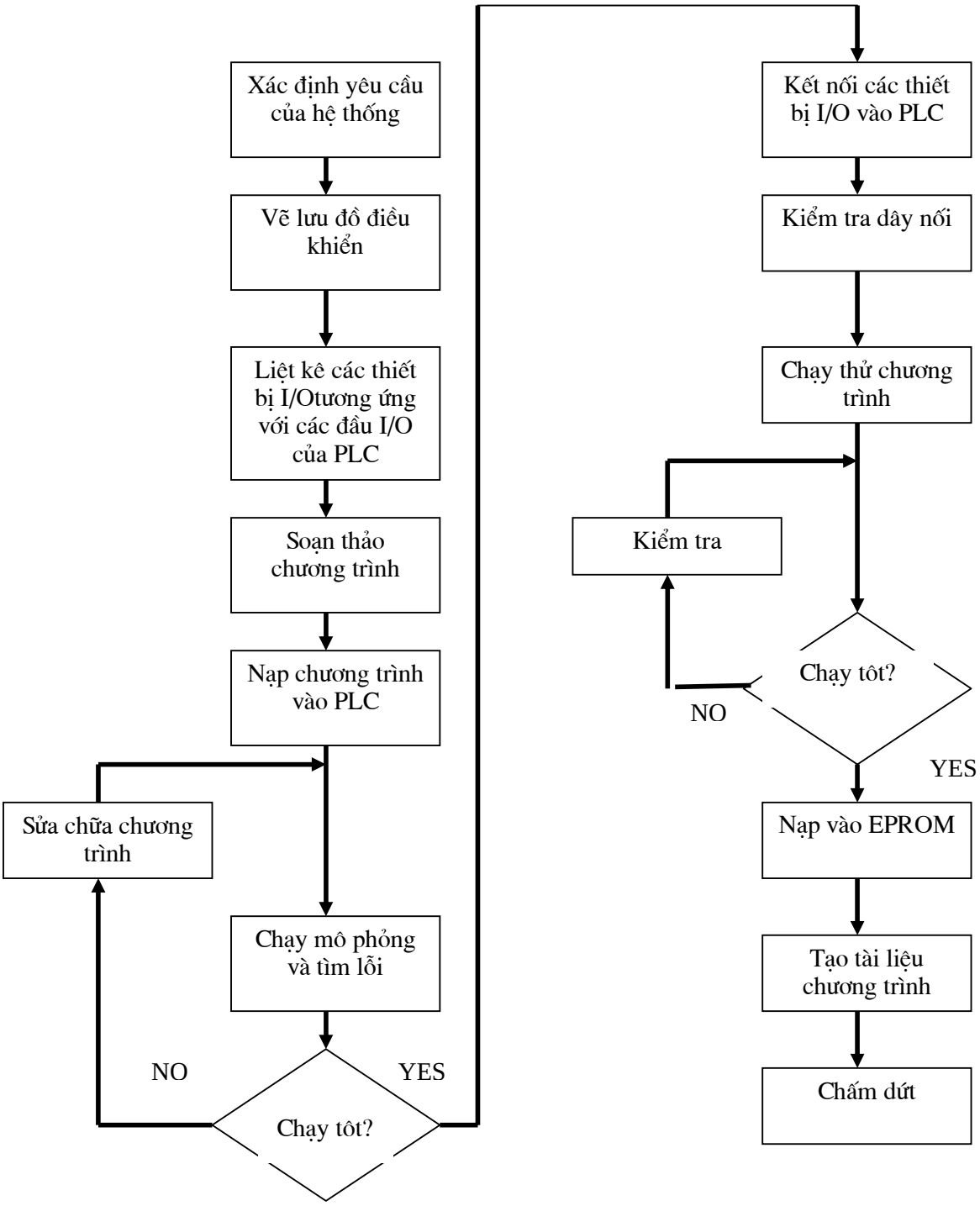
4.Nap chương trình vào bộ nhớ:

Cấp nguồn cho PLC, cài đặt cấu hình khối giao tiếp I/O nếu cần (Phụ thuộc vao từng loại PLC). Sau đó nạp chương trình soạn thảo trên màn hình vào bộ nhớ của PLC. Sau khi hoàn tất nên kiểm tra lỗi bằng chức năng tự chuẩn đoán và nếu có thể thì chạy chương trình mô phỏng hoạt động của hệ thống (Ví dụ chương trình S7-SIM, S7- VISU,...).

5.Chay chương trình:

Trước khi khởi động hệ thống cần phải chắc chắn dây nối từ PLC đến các thiết bị ngoại vi là đúng, trong quá trình chạy kiểm tra có thể cần thiết phải thực hiện các bước tinh chỉnh hệ thống nhằm đảm bảo an toàn khi đưa vào hoạt động thực tế.

Qui trình thiết kế hệ thống điều khiển bằng PLC:



Hình 2-3: Qui trình thiết kế một hệ thống điều khiển tự động.

2.2.Các ngôn ngữ lập trình:

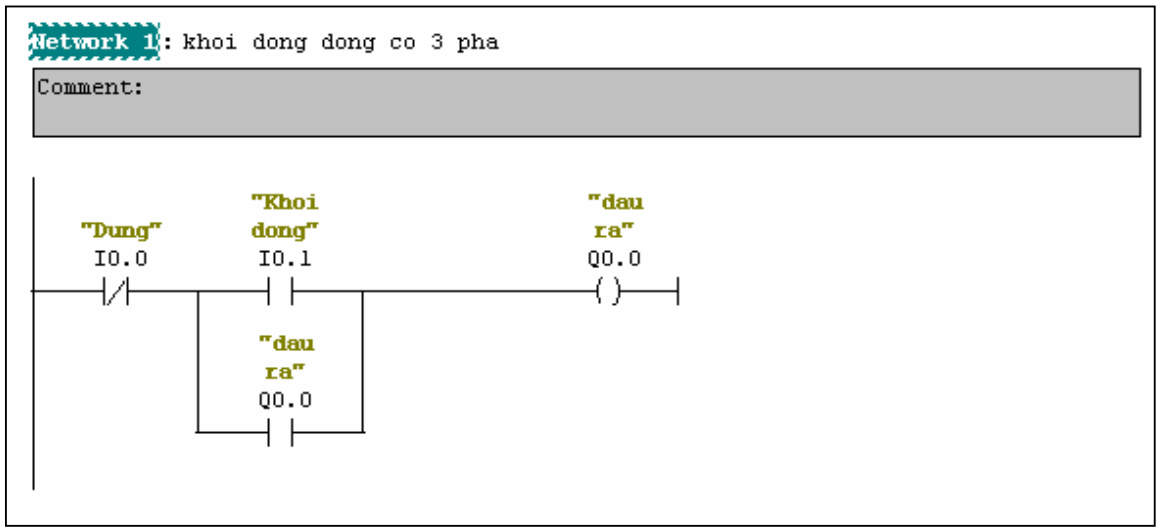
Đối với PLC S7-300 có thể sử dụng 6 ngôn ngữ để lập trình.

1/ Ngôn ngữ lập trình LAD:

Với loại ngôn ngữ này rất thích hợp với người quen thiết kế mạch điều khiển logic

chương trình được viết dưới dạng liên kết giữa các công tắc:

ví dụ:



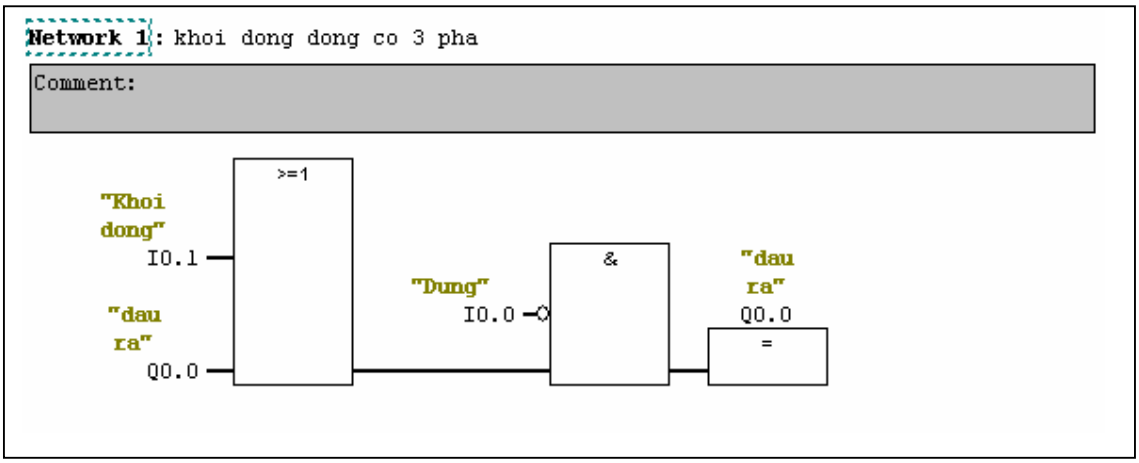
Hình 2-4: ví dụ kiểu lập trình LAD.

2/ Ngôn ngữ lập trình FBD :

Loại ngôn ngữ này thích hợp cho những người quen sử dụng và thiết kế mạch điều khiển số.

Chương trình được viết dưới dạng liên kết của các hàm logic kỹ thuật số:

Ví dụ:

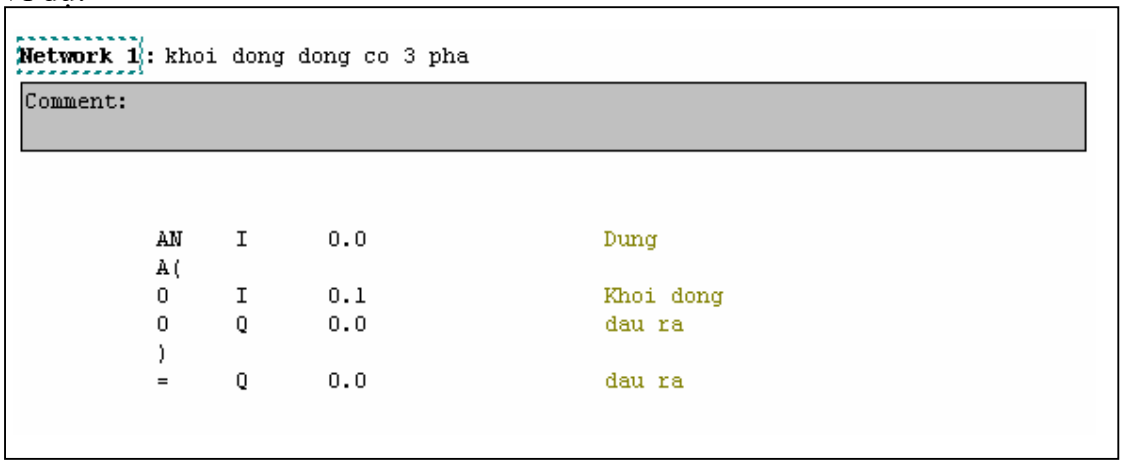


Hình 2-5: Ví dụ kiểu lập trình FBD.

3/ Ngôn ngữ lập trình STL

Đây là ngôn ngữ lập trình thông thường của máy tính. Một chương trình được ghép bởi nhiều lệnh theo một thuật toán nhất định, mỗi lệnh chiếm một hàng và đều có cấu trúc chung là : "tên lệnh" + "toán hạng".

Ví dụ:



Hình 2-6: Ví dụ kiểu lập trình STL.

4/ Ngôn ngữ lập trình SCL (Structured Control Language):

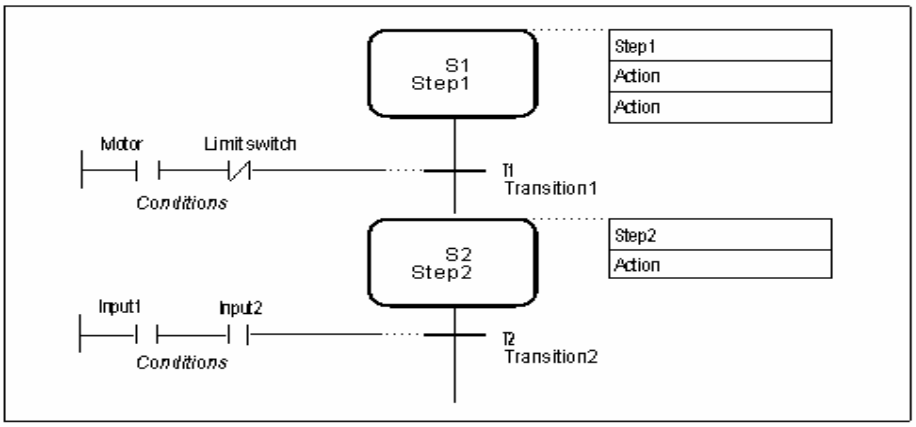
Kiểu viết chương trình này sử dụng ngôn ngữ PASCAL. Rất phù hợp cho những người đã viết các chương trình bằng ngôn ngữ máy tính.

ví dụ:

```
FUNCTION_BLOCK FB 20
VAR_INPUT
ENDWERT : INT;
END_VAR
VAR_IN_OUT
IQ1 : REAL;
END_VAR
VAR
INDEX : INT;
END_VAR
BEGIN
CONTROL := FALSE;
FOR INDEX := 1 TO ENDWERT DO
IQ1 := IQ1 * 2;
IF IQ1 > 10000 THEN
CONTROL = TRUE
END_IF;
END_FOR;
END_FUNCTION_BLOCK
```

5/ Ngôn ngữ lập trình : S7-Graph.

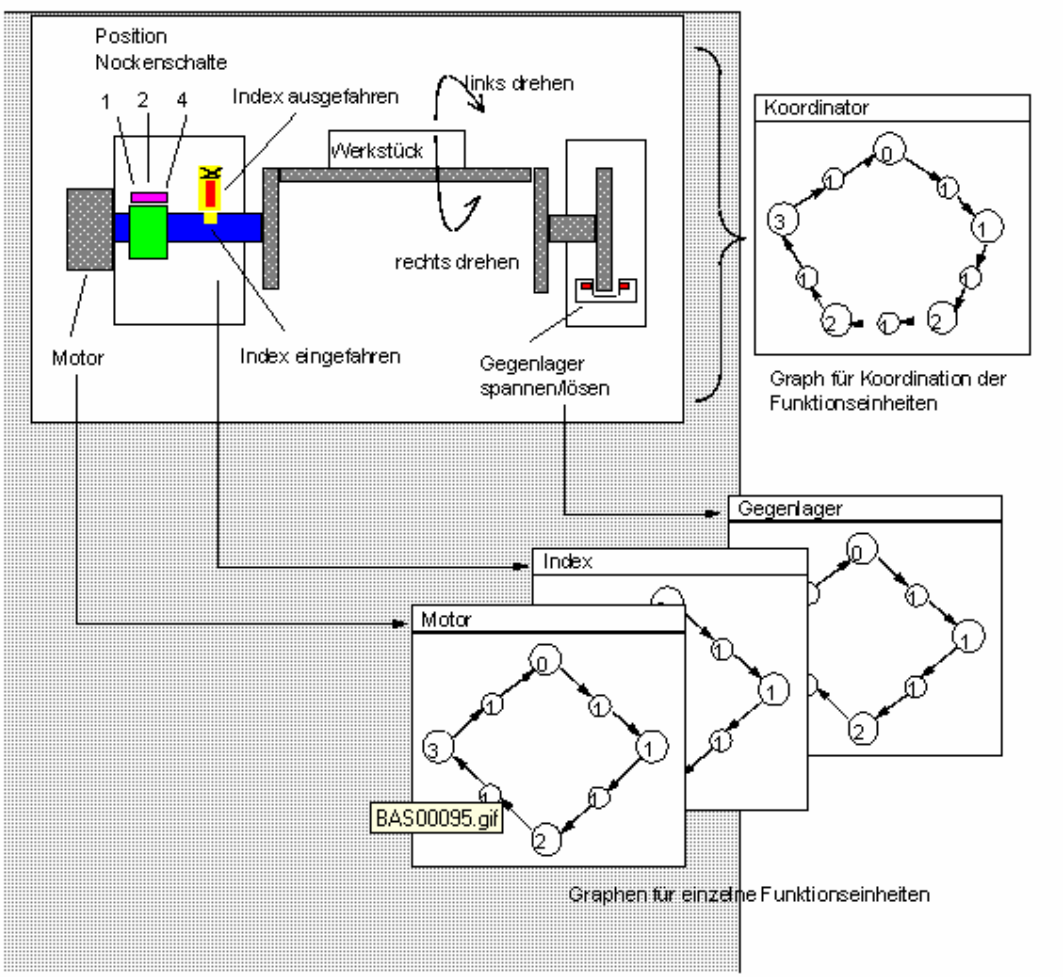
Ví dụ:



Hình2-7: Sơ đồ khối lập trình kiểu S7-Graph.

6/ Ngôn ngữ lập trình : S7-HiGraph.

Đây là một loại ngôn ngữ viết chương trình rất phù hợp cho các bài toán làm việc có tính tuần tự. Tại mỗi thời điểm chỉ có một bước được thực hiện. Với kiểu lập trình này người lập trình phải sử dụng phương pháp lập trình có cấu trúc.
Ví dụ:



Hình 2-8 : Sơ đồ lập trình bằng ngôn ngữ S7-HiGraph.

Trong cuốn tài liệu này sẽ giới thiệu 4 loại ngôn ngữ dùng để lập trình (FBD, STL, LAD và S7GRAPH) trong phần bài tập mẫu.

Chương 3: Cài đặt phần mềm S7-300 và chon chế đô làm viêc

3.1. Giới thiêu chung:

Muốn xây dựng một chương trình điều khiển sử dụng phần mềm Step7 cần thực hiện các thủ tục như sau:

- Khai báo cấu hình cứng cho một trạm PLC thuộc họ Simatic S7-300/400.
- Xây dựng cấu hình mạng gồm nhiều trạm PLC S7-300/400 cũng như thủ tục truyền thông giữa chúng.
- Soạn thảo và cài đặt chương trình điều khiển cho 1 hoặc nhiều trạm.
- Giám sát việc thực hiện chương trình điều khiển trong một trạm PLC và gỡ rối chương trình.

Ngoài ra Step 7 còn có cả một thư viện đầy đủ với các hàm chuẩn hữu ích, phần trợ giúp Online rất mạnh có khả năng trả lời mọi câu hỏi của người sử dụng về cách sử dụng Step 7, về cú pháp lệnh trong lập trình, về xây dựng cấu hình cứng của một trạm cũng như của một mạng gồm nhiều trạm PLC.

3.2. Cài đặt Step7:

3.2.1.Tổng quát về Step 7

Tại việt nam hiện có rất nhiều phiên bản của bộ phần mềm gốc của Step7. Đang được sử dụng nhiều nhất là phiên bản (version) 4.2, 5.0 và 5.1. Trong khi phiên bản 4.2 khá phù hợp cho những PC có cấu hình trung bình (CPU 80586, 90MB còn trống trong ổ cứng, màn hình VGA) nhưng lại đòi hỏi tuyệt đối có bản quyền. Trong khi phiên bản 5.0 và 5.1 mặc dù đòi hỏi máy tính có cấu hình mạnh nhưng lại không đòi hỏi bản quyền một cách tuyệt đối, nghĩa là phiên bản này vẫn làm việc ở một mức hạn chế khi không có bản quyền. Phần lớn các đĩa gốc của Step7 đều có khả năng tự cài đặt chương trình (autorun). Bởi vậy chỉ cần cho đĩa vào ổ CD và thực hiện theo đúng chỉ dẫn hiện trên màn hình. Ta có thể chủ động thực hiện việc cài đặt bằng cách gọi chương trình Setup.exe có trên đĩa. Công việc cài đặt, về cơ bản không khác nhiều so với việc cài đặt các phần mềm ứng dụng khác, tức là cũng bắt đầu bằng việc chọn ngôn ngữ cài đặt (mặc định là tiếng Anh), chọn thư mục đặt trên ổ cứng (mặc định là C:\simens), kiểm tra dung tích còn lại trên ổ cứng, chọn ngôn ngữ sẽ được sử dụng trong quá trình làm việc với Step7 sau này.

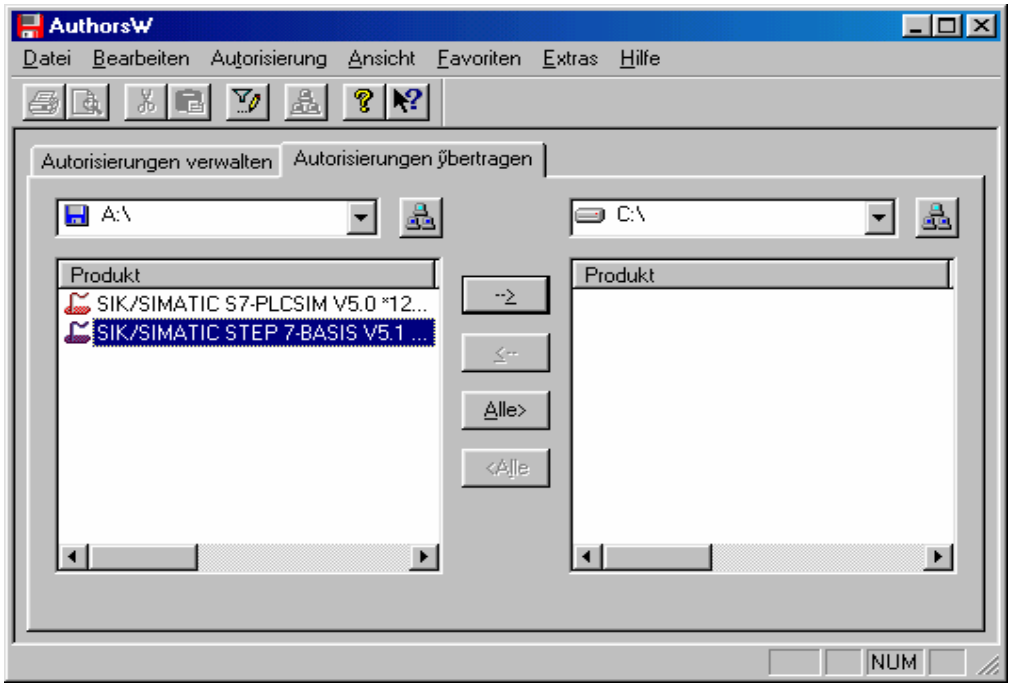
Một số vấn đề cần giải thích rõ thêm khi cài đặt phần mềm Step7:(cuốn tài liệu này hướng dẫn các bạn cài đặt bằng ngôn ngữ tiếng Anh) nhưng về cơ bản cài đặt bằng tiếng Đức cũng không có nhiều điều khác biệt.

3.2.2 Khai báo mã hiệu sản phẩm: mã hiệu sản phẩm luôn đi kèm với sản phẩm và được in ngay trên đĩa chứa bộ cài Step7. Khi trên màn hình xuất hiện cửa sổ yêu cầu cho biết mã hiệu sản phẩm, ta phải điền đầy đủ vào tất cả các thư mục của cửa sổ đó, kể cả địa chỉ người sử dụng sau đó ấn **continue** để tiếp tục.

Hình 3-1: Khai báo mã hiệu của sản phẩm

3.2.3.Chuyển bản quyền: Bản quyền Step7 nằm trên một đĩa mềm riêng (thường có mầu vàng hoặc mầu đỏ). Trong quá trình cài đặt, trên màn hình sẽ xuất hiện yêu cầu chuyển bản quyền sang ổ đích (mặc định là c:\ ) có dạng như sau:

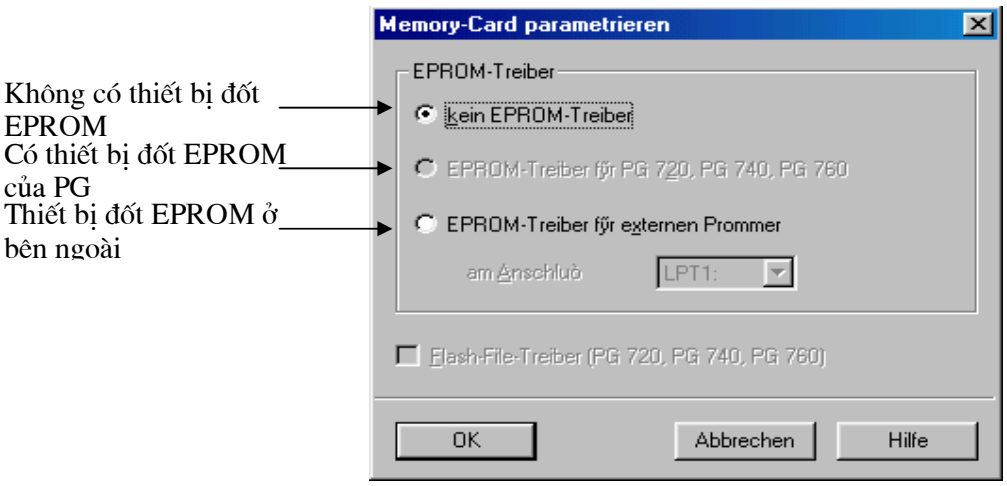
Ta có thể chuyển bản quyền sang ổ đĩa C:\ ngay trong khi cài đặt Step7 bằng cách cho đĩa bản quyền vào ổ đĩa A: rồi ấn phím Authorize. Ta cũng có thể bỏ qua và sẽ chuyển bản quyền sau vào lúc khác bằng cách ấn phím Skip. Trong trường hợp bỏ qua thì sau này, lúc chuyển bản quyền, ta phải sử dụng chương trình truyền bản quyền có tên là AuthorsW.EXE cũng có trên đĩa bản quyền (Ver.4.2) hoặc có cùng trong đĩa CD với phần mềm gốc Step7 (ver5.1).



Hình 3-2: Chuyển bản quyền

Chú ý đĩa mềm chứa bản quyền (Author disk) đã được bảo vệ cấm sao chép. Cho dù bản quyền đã được chuyển từ đĩa mềm sang ổ cứng và trên đĩa mềm không còn bản quyền, nhưng nó vẫn là một đĩa đặc biệt có chỗ chứa bản quyền. Bản quyền khi sao chép sang ổ đĩa cứng sẽ nằm trong thư mục Ax nf zz. Nếu thư mục này bị hỏng, ta sẽ mất bản quyền. Bởi vậy mỗi khi muốn cài đặt lại hệ thống hay dọn dẹp lại ổ đĩa cứng thì trước đó ta phải thực hiện rút bản quyền khỏi ổ đĩa C: và chuyển ngược về ổ đĩa mềm Author cũng bằng chương trình AuthorsW.EXE.

3.2.4.Khai báo thiết bi đốt EPROM: Chương trình step7 có khả năng đốt chương trình ứng dụng lên thẻ EPROM cho PLC. Nếu máy tính PC của ta có thiết bị đốt EPROM thì cần phải thông báo cho Step7 biết khi trên màn hình xuất hiện cửa sổ:



Hình 3-3: Khai báo thiết bị đốt EPROM

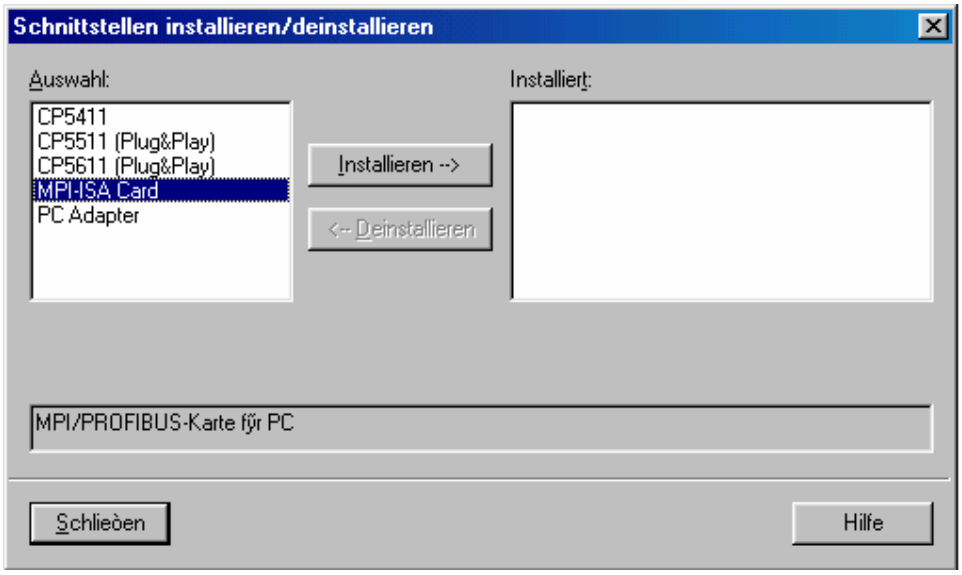
3.2.5. Chon giao diên cho PLC:

Chương trình Step7 được cài đặt trên PC (máy tính cá nhân) hoặc PG (lập trình bằng tay) để hỗ trợ việc soạn thảo cấu hình cứng cũng như chương trình cho PLC, tức là sau đó toàn bộ những gì đã soạn thảo sẽ được dịch sang PLC. Không những thế, Step7 còn có khả năng quan sát việc thực hiện chương trình của PLC. Muốn như vậy ta cần phải có bộ giao diện ghép nối giữa PC với PLC để truyền thông tin, dữ liệu.

Step7 có thể ghép nối với PLC bằng nhiều bộ phương thức ghép nối khác nhau như qua Card MPI, qua bộ chyển đổi PC/PPI, qua thẻ PROFIBUS (CP) nhưng chúng phải được khai báo sử dụng.

Ngay sau khi Step7 được cài đặt xong, trên màn hình xuật hiện cửa sổ thông báo cho ta chọn các bộ giao diện sẽ được sử dụng. Cửa sổ này có dạng sau (hìnhvẽ 3-4):

Muốn chọn bộ giao diện nào, ta đánh dấu bộ giao diện đó ở phía trái rồi ấn phím Install.... Những bộ giao diện đã được chọn sẽ được ghi vào ô bên phải. Sau khi chọn xong các bộ giao diện sử dụng, ta còn phải đặt tham số làm việc cho những bộ giao diện đó bao gồm tốc độ truyền , cổng ghép nối với máy tính. Chẳng hạn khi đã chọn bộ giao diện MPI -ISA Card ta phải đăt tham số làm việc cho nó thông qua cửa sổ màn hình.



Hình3-4: Khai báo dạng kết nối PC với CPU

3.3.Đặt tham số làm việc:

Sau khi cài đặt xong Step7, trên màn hình (Destop) sẽ xuất hiện biểu tượng icon của nó. Đồng thời trong **Menu** của Window cũng có thư mục **Simatic** với tất cả các tên của những thành phần liên quan, từ các phần mềm trợ giúp đến các phần mềm cài đặt cấu hình, chế độ làm việc của Step7.

Khi vừa được cài đặt, step7 có cấu hình mặc định về chế độ làm việc của **Simatic,** chẳng hạn cú pháp các lệnh lại được viết theo tiếng Đức ví dụ như AND thì viết thành UND, muốn chuyển thành dạng thông dụng quốc tế ta phải cài đặt lại cấu hình cho Step7.

Tất nhiên, bên cạnh việc chọn ngôn ngữ cho cú pháp lệnh ta còn có thể sửa đổi nhiều chức năng khác của Step 7 như nơi sẽ chứa chương trình trên đĩa cứng, những thanh ghi sẽ được hiển thị nội dung khi gỡ rối chương trình, song các việc đó không ảnh hưởng quyết định tới việc sử dụng Step7 theo thói quen của ta như ngôn ngữ cú pháp lệnh.

3.4.Soan thảo một Project.

Khái niệm Project không đơn thuần chỉ là chương trình ứng dụng mà rộng hơn bao gồm tất cả những gì liên quan đến việc thiết kế phần mềm ứng dụng để điều khiển, giám sát một hay nhiều trạm PLC. Theo khái niệm như vậy, trong một Project sẽ có:

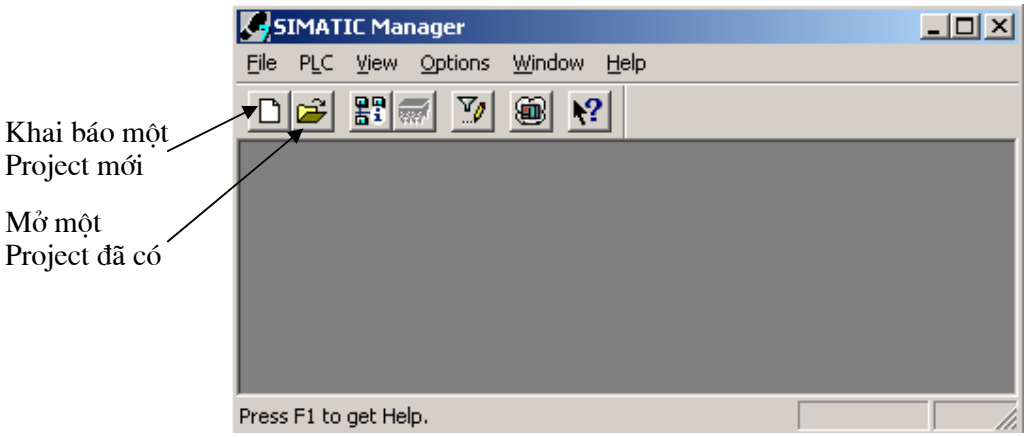
1. Bảng cấu hình cứng về tất cả các module của từng trạm PLC.

2. Bảng tham số xác định chề độ làm việc cho từng module của mỗi trạm PLC.
3. Các Logic block chứa chương trình ứng dụng của từng trạm PLC.
4. Cấu hình ghép nối và truyền thông giữa cac trạm PLC.
5. Các cửa sổ giao diện phục vụ việc giám sát toàn bộ mạng hoặc giám sát từng trạm PLC của mạng.

Ở đây, trong khuôn khổ phần mềm Step7 tôi chỉ giới thiệu việc soạn thảo một Project gồm các phần 1,2,3. Những phần còn lại bạn đọc có thể tham khảo trong cuốn tài liệu khác của cùng tác giả.

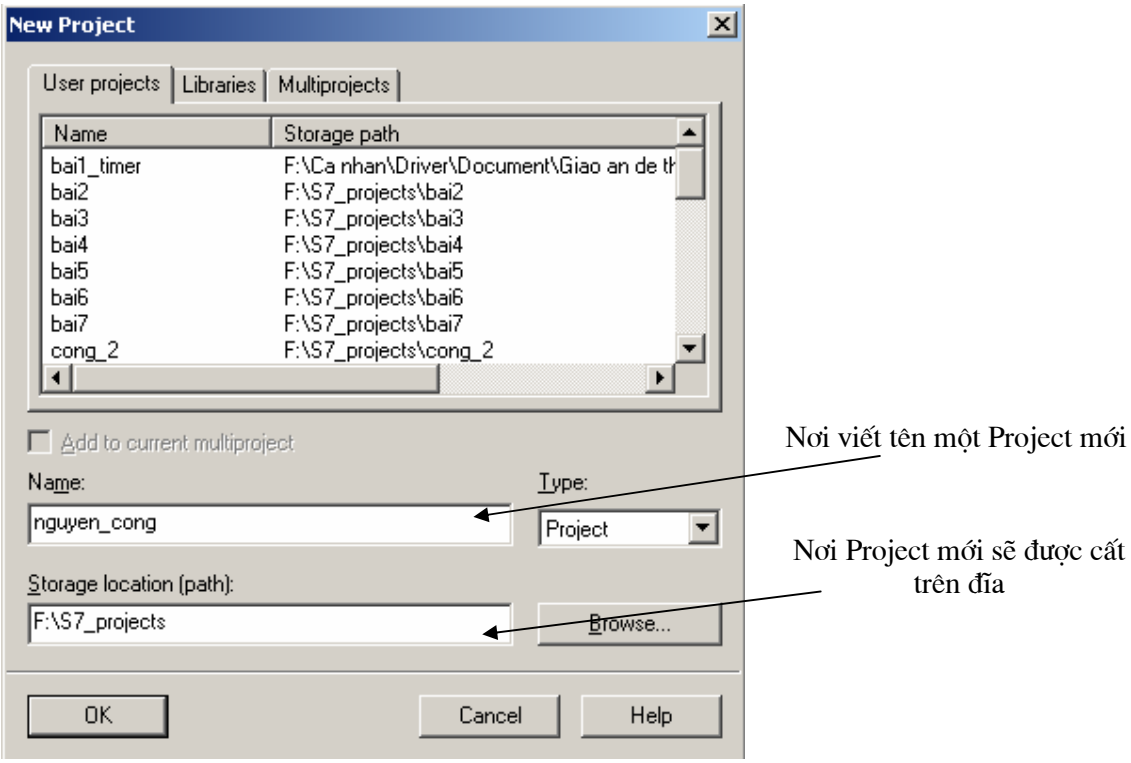
3.4.1.Khai báo và mở một Project mới.

Để khai báo một Project, từ màn hình chính của Step 7 ta chọn **File-> New** hoặc kích chuột tại biểu tượng "New Project/ Library".



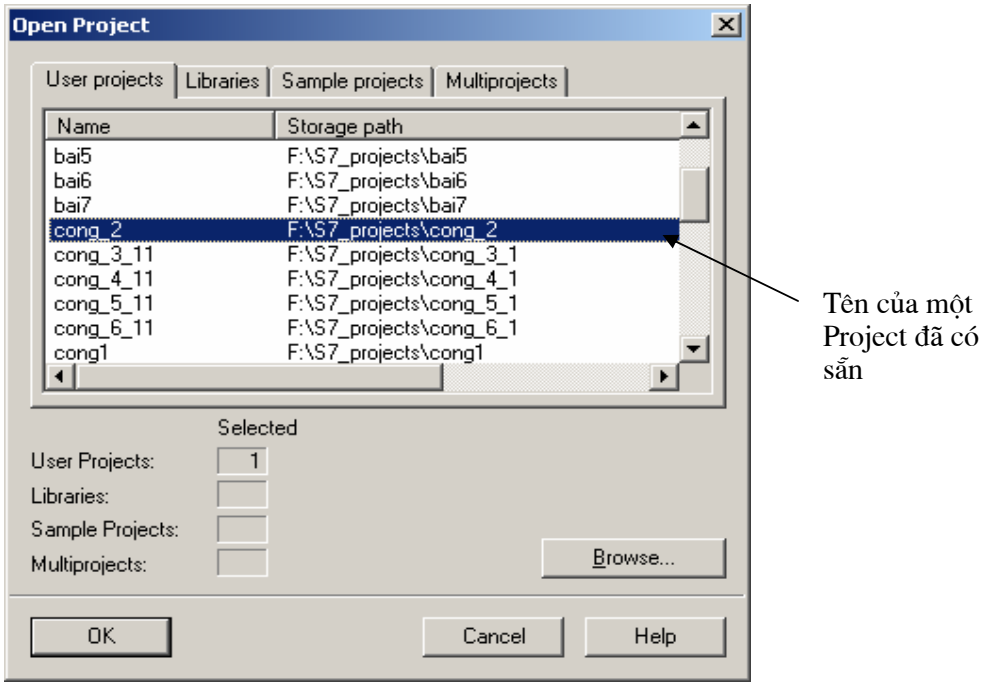
Hình 3-5: Mở một Project mới

Khi đó trên màn hình sẽ xuất hiện hộp hội thoại như hình 3-6. Gõ tên Project rồi ấn phím OK và như vậy ta đã khai báo song một Project mới. Ngoài ra ta còn có thể chọn nơi Project sẽ được cất lên đĩa. Mặc định, nơi cất sẽ là thư mục đã được quy định khi cài đặt Step 7, ở đây là thư mục F:\S7_ projects.



Hình 3-6: Đặt tên cho một Project mới

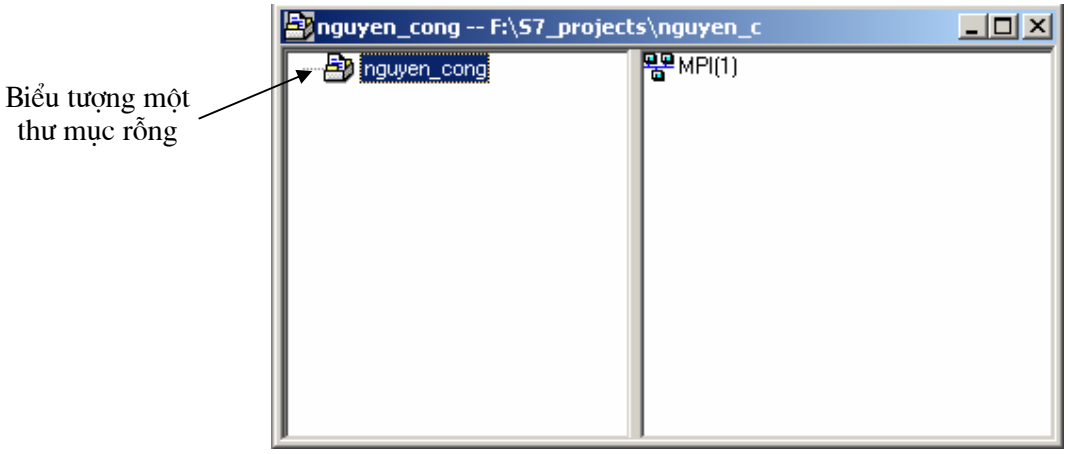
Trong trường hợp muốn mở một Project đã có, ta chọn File -> Open hoặc kích chuột tại biểu tượng "Open Project/ Library" từ cửa sổ chính của Step7 rồi chọn tên Project muốn mở từ hộp hội thoại có dạng như hình 3-7. Cuối cùng ấn phím OK để kết thúc.



Hình 3-7: Mở một Project đã có.

3.4.2. Xây dựng cấu hình cứng cho tram PLC.

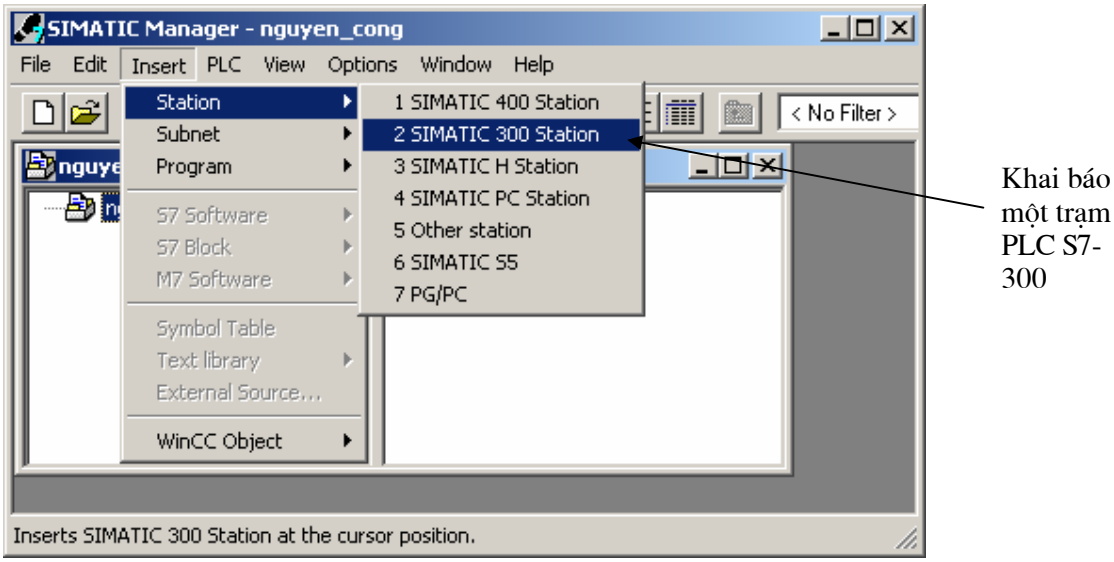
Sau khi khai báo xong một Project mới, trên màn hình sẽ xuất hiện Project đó nhưng ở dạng rỗng (chưa có gì trong project), điều này ta nhận biết được qua biểu tượng thư mục bên cạnh tên Project giống như một thư mục rỗng của Window.



Hình 3-8: Biểu tượng một Project mới.

Công việc tiếp theo ta có thể làm là xây dựng cấu hình cứng cho một trạm PLC. Điều này không bắt buộc, ta có thể không cần khai báo cầu hình cứng cho trạm mà đi ngay vào phần chương trình ứng dụng. Song kinh nghiệm cho thấy công việc này nên làm vì khi có cấu hình trong project, lúc bật nguồn PLC, hệ điều hành của S7-300 bao giờ cũng đi kiểm tra các module hiện có trong trạm, so sánh với cấu hình mà ta xây dựng và nếu phát hiện thấy sự không đồng nhất sẽ phát ngay tín hiệu báo ngắt lỗi hoặc thiếu module chứ không cần phải đợi tới khi thực hiện chương trình ứng dụng.

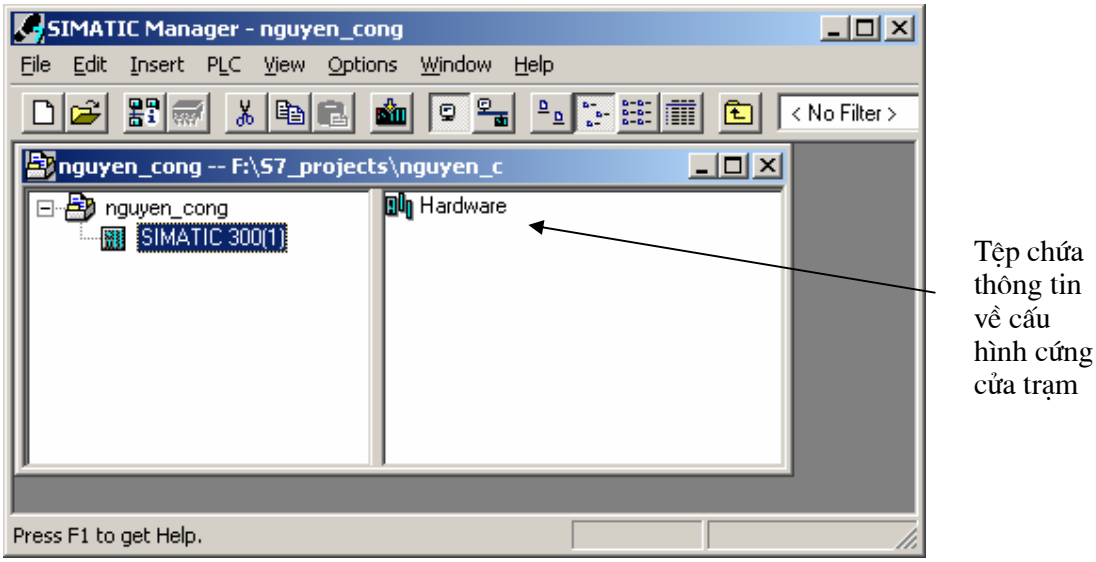
Trước hết ta khai báo cấu hình cứng cho một trạm PLC với simatic S7-300 bằng cách vào: Insert ***-> Station ->Simatic 300- Station:***



Hình 3-9: Khai báo cấu hình cứng cho trạm PLC

Trong trường hợp không muốn khai báo cấu hình cứng mà đi ngay vào chương trình ứng dụng ta có thể chọn thẳng. Động tác này sẽ hữu ích cho những trường hợp một trạm PLC có nhiều phiên bản ứng dụng khác nhau.

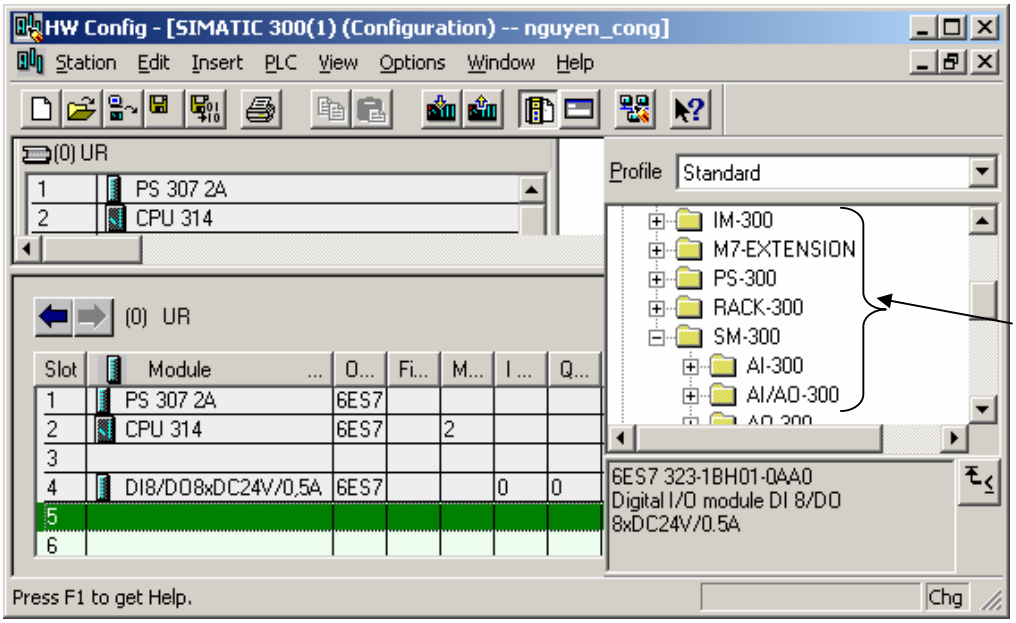
Sau khi đã khai báo một trạm (chèn một Station), thư mục Project chuyển sang dạng không rỗng với thư mục con trong nó tên mặc định là Simatic300(1) chứa tệp thông tin về cấu hình cứng của trạm.



Hình 3-10:Màn hình khai báo cấu hình cứng cho tạm PLC

Để vào màn hình khai báo cấu hình cứng, ta nháy chuột tại biểu tượng **Hardware**. Trong hộp thoại hiện ra ta khai báo thanh Ray (Rack) và các module có trên thanh Ray đó.

Ví dụ:



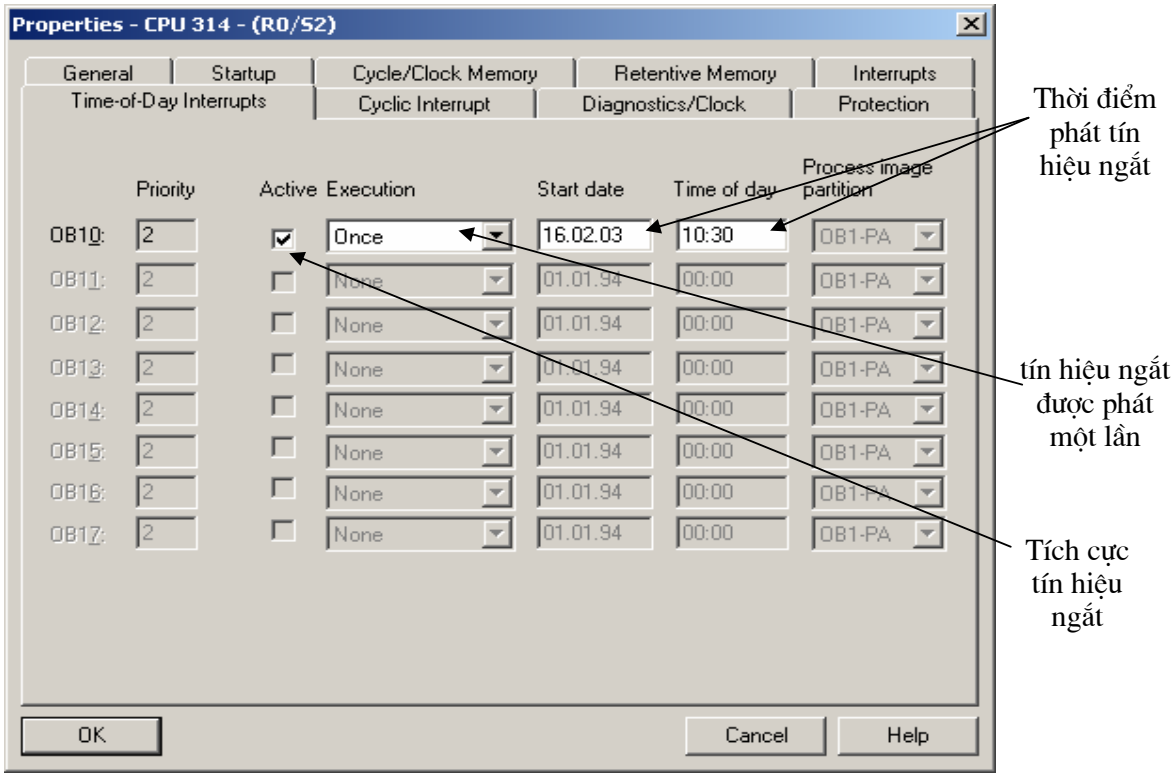
Hình 3-11: Thư viện để lấy các Modul

Step7 giúp việc khai báo cấu hình cứng được đơn giản nhờ bảng danh mục các module của nó. Muốn đưa module nào vào bảng cấu hình ta chỉ cần đánh dấu vị trí nơi module sẽ được đưa vào rồi nháy kép chuột trái tại tên của module đó trong bảng danh mục cac module kèm theo.

3.4.3. Đặt tham số quy định chế độ làm việc cho module.

Với bảng cấu hình cứng phần mềm Step7 cũng xác định luôn cho ta địa chỉ từng module.

Chẳng hạn Step7 có hỗ trợ việc tích cực ngắt theo thời điểm cho module CPU để module này phát một tín hiệu ngắt gọi khối OB10 một lần vào đúng ngày 16/02/2003 lúc 10 giờ 30. Để làm được điều này ta nháy đúp chuột tại tên của module CPU ở vị trí 2 rồi chọn ô Time-Of-Day Interrupt, trên màn hình sẽ xuất hiện hộp hội thoại như hình 3-12. Điền thời điểm, tần suất phát tín hiệu ngắt rồi đánh dấu tích cực chế độ ngắt vào các ô tương ứng trong hộp hội thoại. Cuối cùng ấn phím OK.

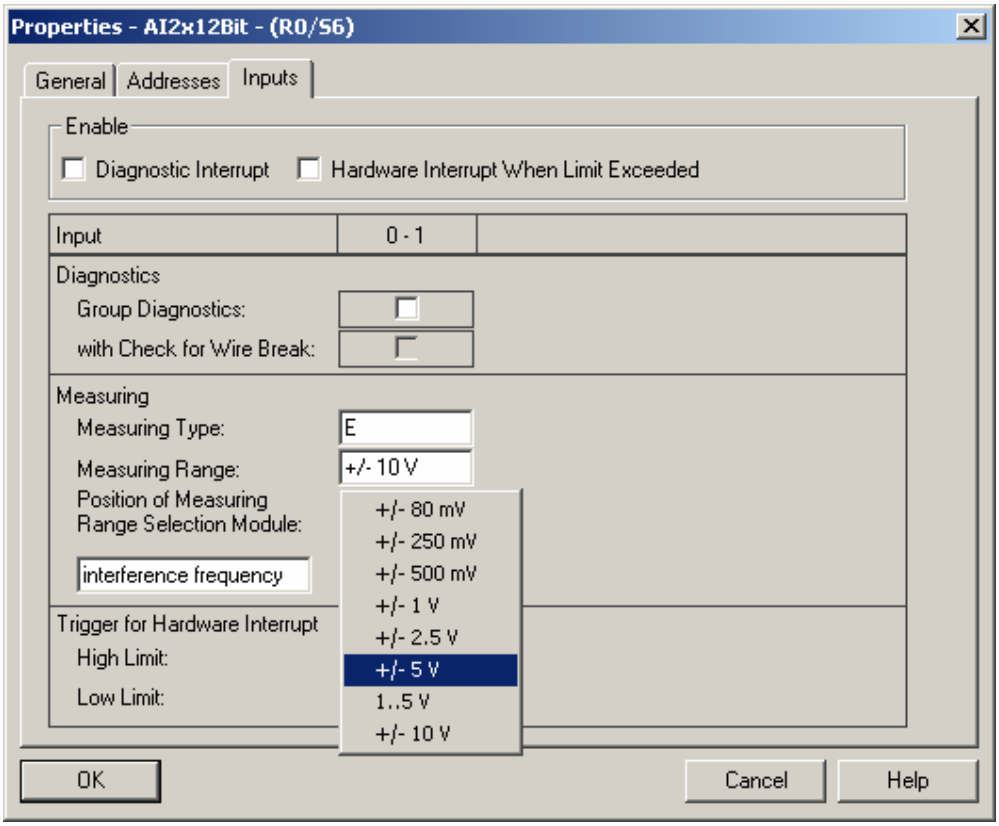


Hình 3-12: Đặt tham số cho Modul CPU

Cũng trong hộp hội thoại ta thấy module CPU314 chỉ cho phép sử dụng OB10 trong số các module OB10 - OB17 với mức ưu tiên là để chứa chương trình xử lý tín hiệu ngắt theo thời điểm.

Các chế độ làm việc khác của module CPU cũng được quy định nhờ Step7. Ví dụ để sửa đổi thời gian vòng quét cực đại cho phép từ giá trị mặc định 150ms thành 100 ms, ta chọn Cycle/Clock memory trong hộp hội thoại rồi sửa nội dung ô Scan time thành 100.

Hoàn toàn tương tự ta cũng có thể sử dụng Step7 để quy định chế độ làm việc cho các module mở rộng khác, như xác định chế độ làm việc với dạng tín hiệu điện áp, với dải ± 5V cho module AI:



Hình 3-13: Đặt chế độ cho Modul Analog

3.4.4. Soan thảo chương trình cho các khối logic.

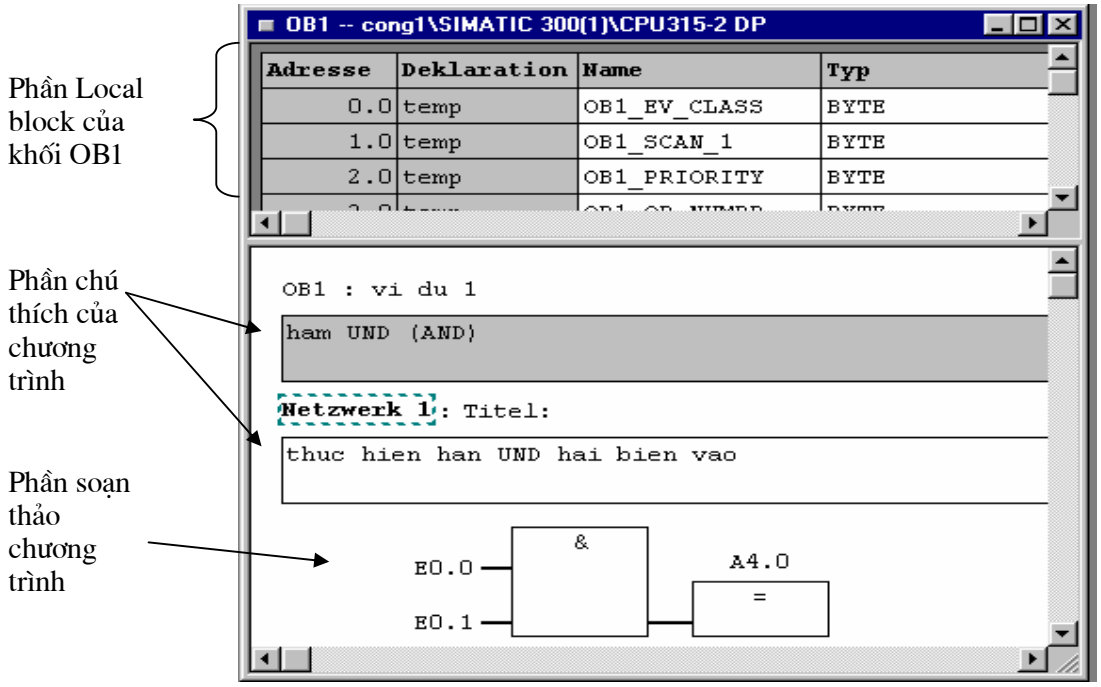
Sau khi khai báo cấu hình cứng cho một trạm PLC và quay trở về cửa sổ chính của Step7 ta thấy trong thư mục Simatic 300(1) bây giờ có thêm các thư mục con và tất nhiên ta có thể đổi tên các thư mục đó.

Tất cả các khối Logic (OB, FC, FB, DB) chứa chương trình ứng dụng sẽ nằm trong thư mục Block. Mặc định trong thư mục này đã có sẵn khối OB1.

1.Soan thảo chương trình cho khối OB1:

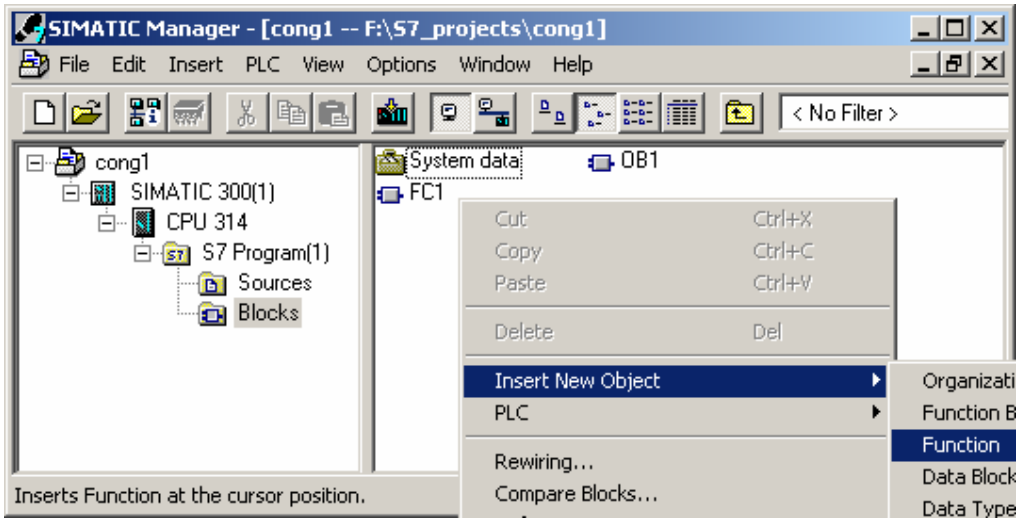
Ta nháy chuột tại biểu tượng OB1 bên nửa cửa sổ bên phải. Trên màn hình sẽ xuất hiện cửa sổ của chế độ soạn thảo chương trình như hình 3-14.

Chức năng chương trình soạn thảo của Step7 về cơ bản cũng giống như các chương trình soạn thảo khác, tức là cũng có các phím nóng để gõ nhanh, có chế độ cắt và dán, có chế độ kiểm tra lỗi cú pháp lệnh.



Hình 3-14: Soạn thảo chương trình trong OB1

Để khai báo va soạn thảo chương trình cho các khối OB khác hoặc cho các khối FC, FB hoặc DB, ta có thể tạo một khối mới ngay trực tiếp từ chương trình soạn thảo bằng cách kích chuột phải vào phần trống như hình vẽ sau:



Hình 3-15: Mở một khối logic khác.

Hoặc cũng có thể chèn thêm khối mới đó trước từ cửa sổ chính của Step7 bằng phím **Insert -> S7 Block** rồi sau đó mới vào soạn thảo chương trình cho khối mới được chèn thêm như dã làm với OB1.

Trong màn hình soạn thảo chương trình cho các khối Logic, ta có thể thay đổi không riêng phần chương trình mà cả phần local block của khối đó bao gồm tên hình thức, kiểu dữ liệu, giá trị ban đầu, Chú ý rằng không được thay đổi 20 bytes đầu trong local block của các khối OB.

Các bước soạn thảo một khối logic cho chương trình ứng dụng được tóm tắt như sau:

- Tạo khối logic hoặc từ cửa sổ màn hình chính của Step7 bằng cách chọn Einfuegen (Insert) trên thanh công cụ rồi vào S7 Block để chọn loại khối logic mong muốn (OB, FB, FC) hoặc vào chương trình soạn thảo rồi từ đó kích biểu tượng New.
- Thiết kế local block cho khối logic vừa tạo.

Với tất cả các khối để hoàn thành công việc thiết kế Local Block ta cần phải chú ý việc khai báo theo bảng sau:

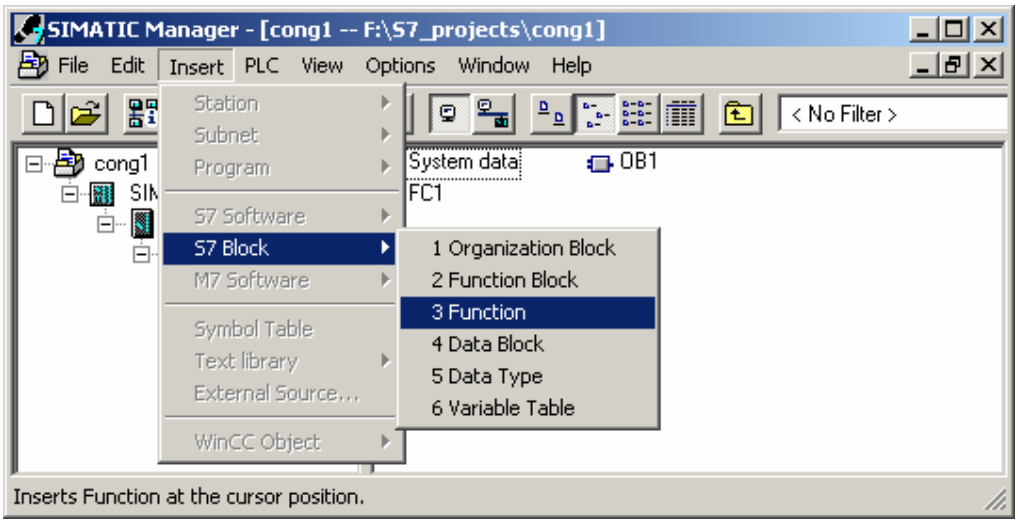
Loại biến	ý nghĩa	chức năng	Khối thực hiện
IN	Nhận các tín hiệu từ đầu vào	đọc	FB, FC
OUT	Xuất các tín hiệu ra	xuất	FB, FC
IN_OUT	Nhận và gửi các tín hiệu	đọc, xuất	FB, FC
STAT	Nội dung của biến hình thức, có khả năng lưu giữ lại khi kết thúc chương trình trong FB	đọc, xuất	FB
TEMP	Biến tạm thời, nội dung sẽ bị mất đi khi kết thúc chương trình trong FB, FC hoặc OB	đọc , xuất	FB, FC, OB

- Soạn thảo chương trình: chương trình có thể được soạn thảo theo rất nhiều ngôn ngữ khác nhau ví dụ: FBD, LAD, STL.... xem trong mục **2.2.**

2.Soan thảo môt chương trình trong khối logic FC1:

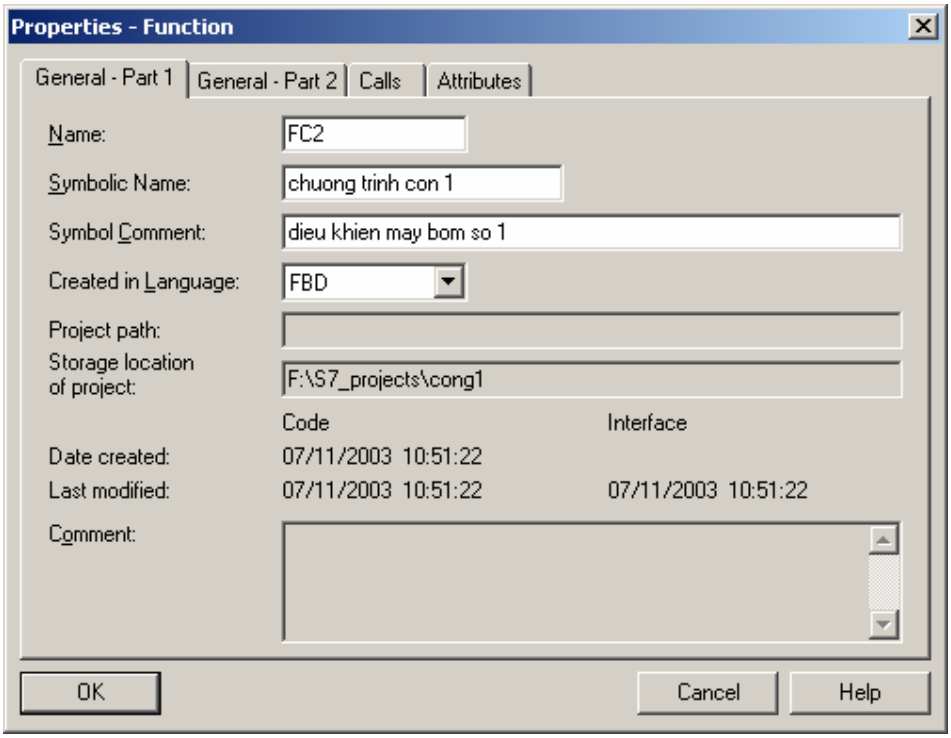
Ta thực hiện các bước như sau:

a/ Tạo khối:



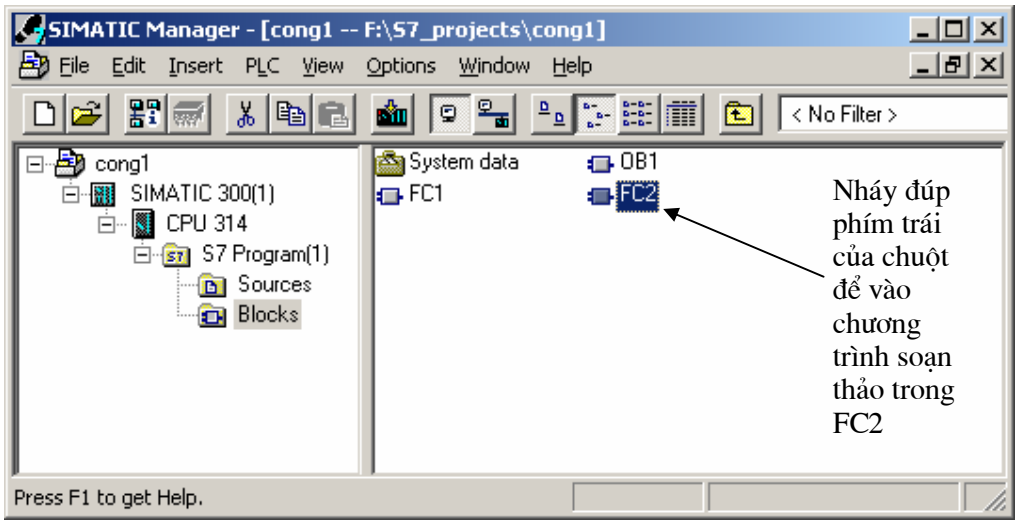
Hình 3-16: Tạo một khối logic mới

Sau khi chọn thư mục như hình vẽ trên trên màn hình sẽ hiện ra một cửa sổ sau:



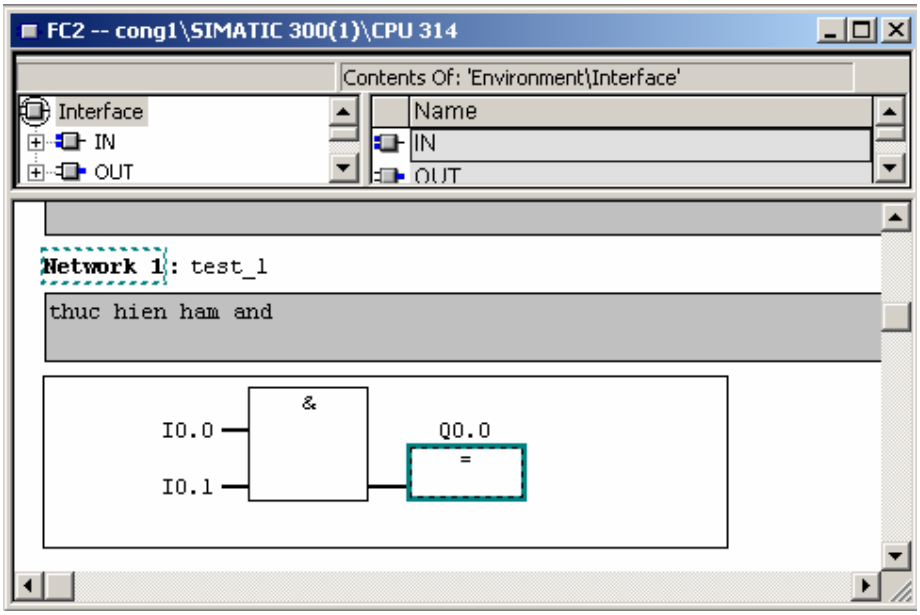
Hình 3-17: Đặt tên và chọn chế độ làm việc cho khối logic mới.

Trong hộp hội thoại cho phép ta chọn tên của FC ví dụ FC2. Trong thực tế Step7 luôn mặc định thứ tự của các FC và ta chỉ cần OK nếu ta chấp nhận tên như đã mặc định, ngoài ra ta còn có thể chọn chế độ viết chương trình trong khối hàm FC2 dưới dạng FBD, LAD hay STL. Cuối cùng ta nhấn nút OK. Trên màn hình sẽ xuất hiện cửa sổ chính của Step7 như sau:



Hình 3-18: Gọi màn hình soạn thảo.

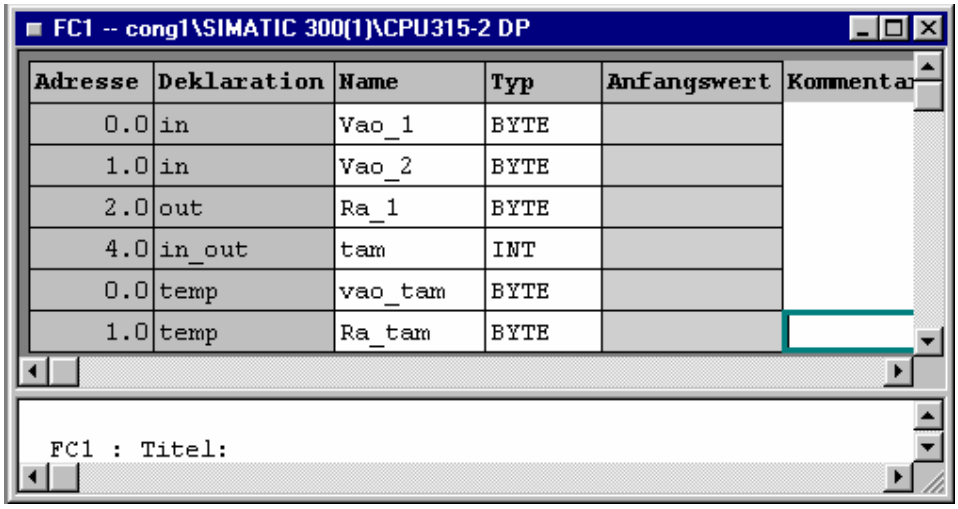
Để soạn thảo chương trình trong FC2 ta chỉ cần nhấy đúp chuột trái vào biểu tượng của FC2 và lập tức sẽ hiện ra cửa sổ soạn thảo chương trình cho FC2:



Hình 3-19: Màn hình soạn thảo của khối Logic FC2.

b/ Xây dựng Local block:

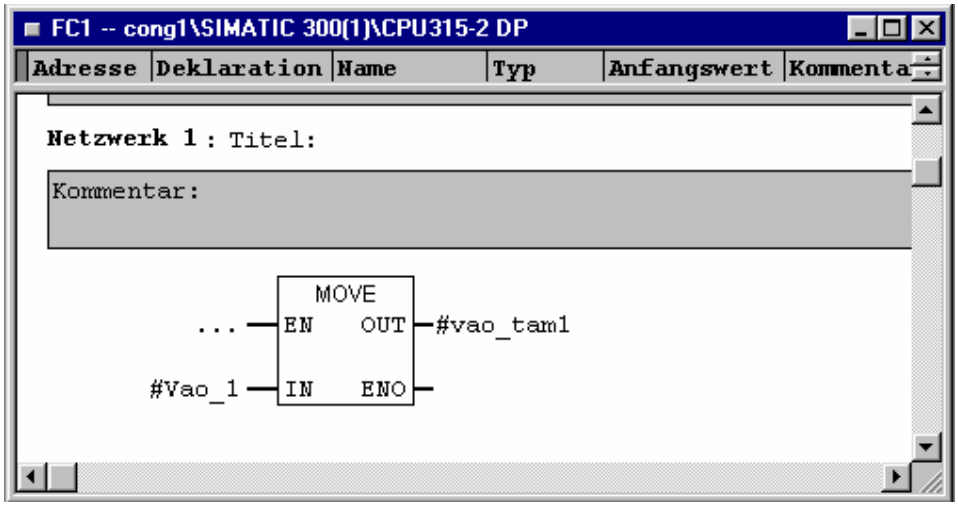
Trong cửa sổ màn hình soạn thảo ta xây dựng local block cho khối FC2 như sau:



Hình 3-20: Nhập dữ liệu vào khối Lokal block của khối FC

c/ Soạn thảo chương trình:

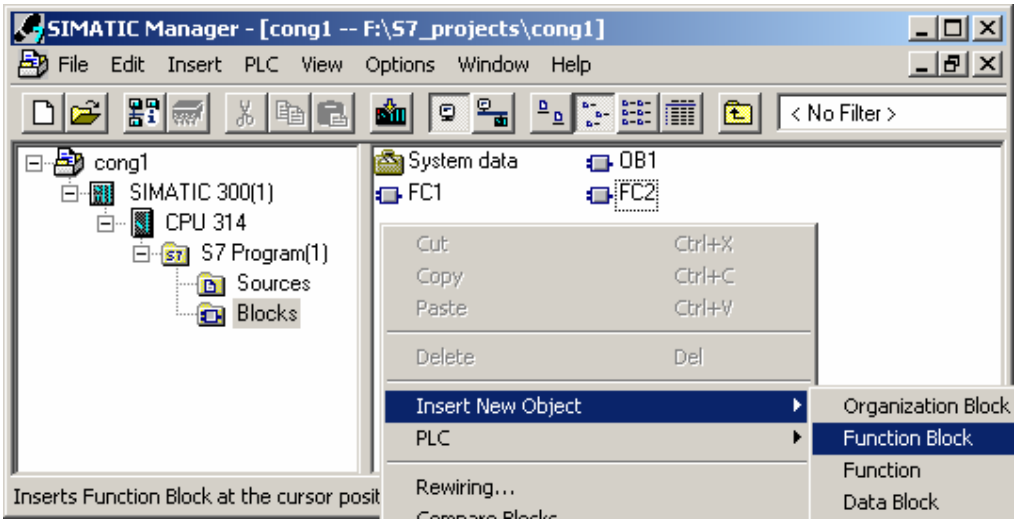
Toàn bộ chương trình có thể viết trong khối logic FC2 như sau:



Hình 3-21: Soạn thảo chương trình trong khối logic FC1.

3.Soan thảo chương trình cho khối FB.

a/Tạo khối FB: Ta có thể tạo khối FB bằng cách từ cửa sổ màn hình chính của Step7 ta dùng chuột phải và chọn các đối tượng như hình sau:

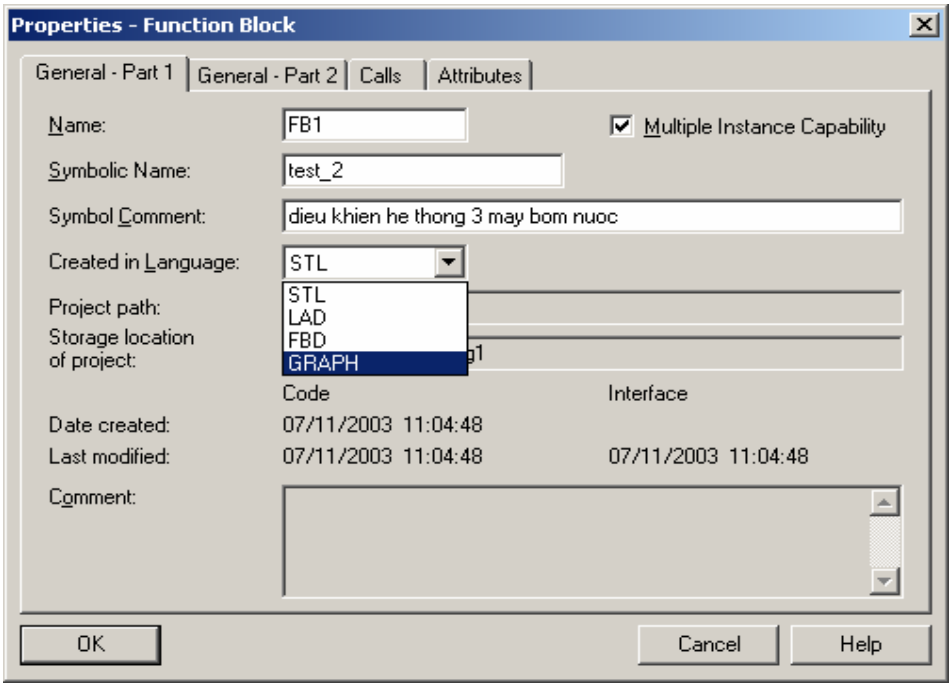


Hình 3-22: Tạo khối FB

Sau khi chọn thư mục ***Funktionsblock*** trên màn hình xuất hiện một cửa sổ:

Trong cửa sổ đó ta cần phải đặt tên cho khối FB mà ta mới chọn ví dụ FB1 (thông thường S7 tự gán cho một tên theo thứ tự mà người lập trình đã chọn khi đó nếu đồng ý ta chỉ cần nhấn nút OK). Ngoài ra ta còn có thể đặt tên cho khối FB; ví dụ: test_1, chọn cách viết chương trình AWL, KOP, FUP hay S7-GRAPH,..... Sau khi đã điền đủ các thông tin vào cửa sổ màn hình ta nhấn nút OK.

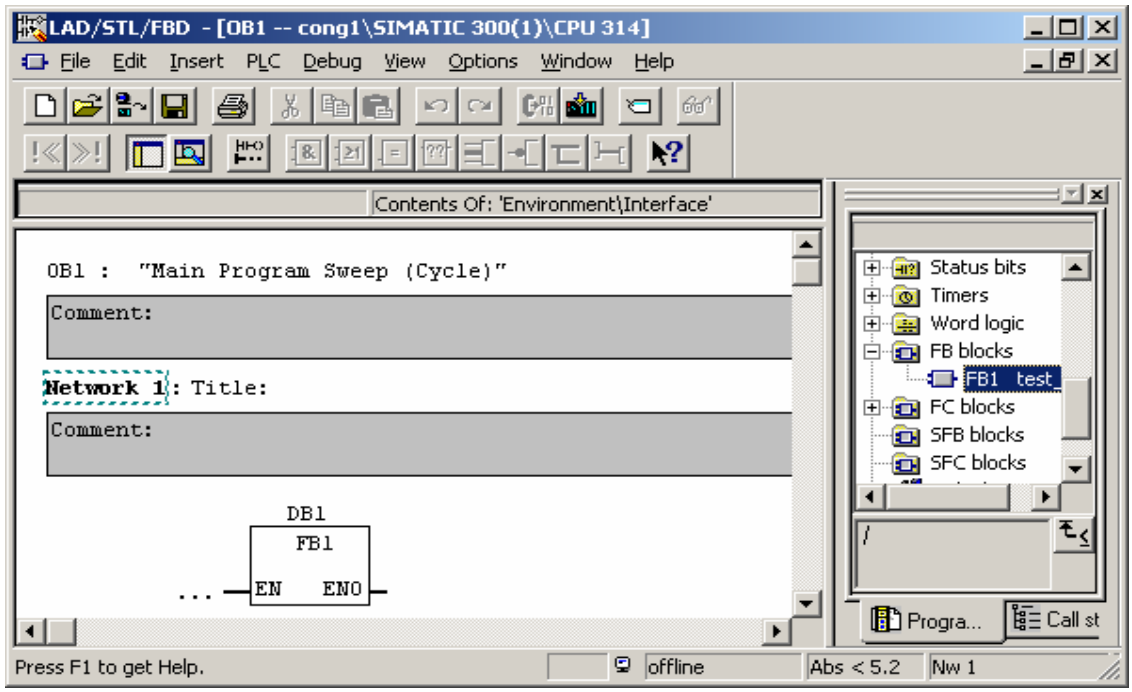
Muốn soạn thảo chương trình trong khối FB ta chỉ cần nhấn đúp chuột trái vào biểu tượng FB trên màn hình chính. Sau khi thực hiện xong bước này ta sẽ có cửa sổ soạn thảo chương trình cho khối FB1 và công việc tiếp theo cũng được thực hiện giống như ta đã thực hiện đối với khối FC ở trên , đó là các bước như xây dựng Local block, soạn thảo chương trình.



Hình 3-23: Chọn ngôn ngữ viết chương trìnhtrong khối FB1

b/Thủ tục gọi khối FB:

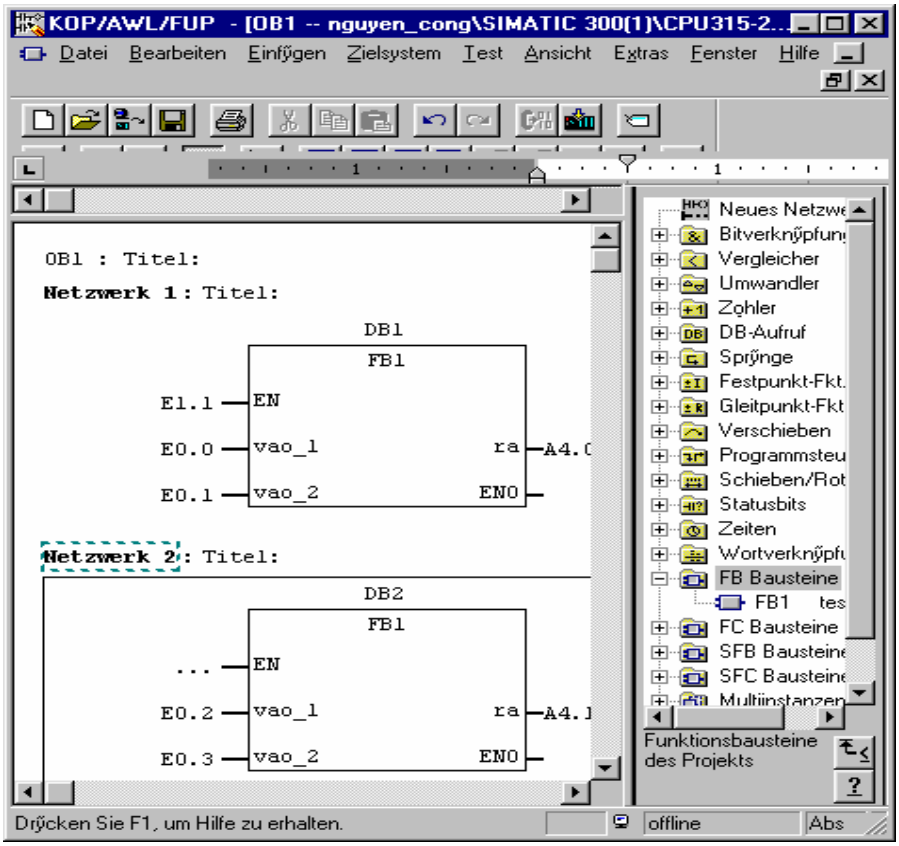
Vì khối FB bao giờ cũng làm việc với khối dữ liệu DB dùng để lưu giữ nội dung các biến kiểu STAT của Local block. Vì vậy để thực hiện việc gọi khối FB ta phải đặt tên cho khối dữ liệu DB tương ứng. Lệnh gọi khối hàm FB như sau:



Hình 3-24:Gọi khối FB1

Tuỳ theo nhu cầu sử dụng mà ta sử dụng một , hai hay nhiều khối DB ta phải đặt tên cho khối DB mà ta vừa chọn ví dụ DB1, DB2,...

Sau khi đã chọn xong bước trên ta có thể soạn thảo chương trình cho khối DB1 và DB2 như sau:

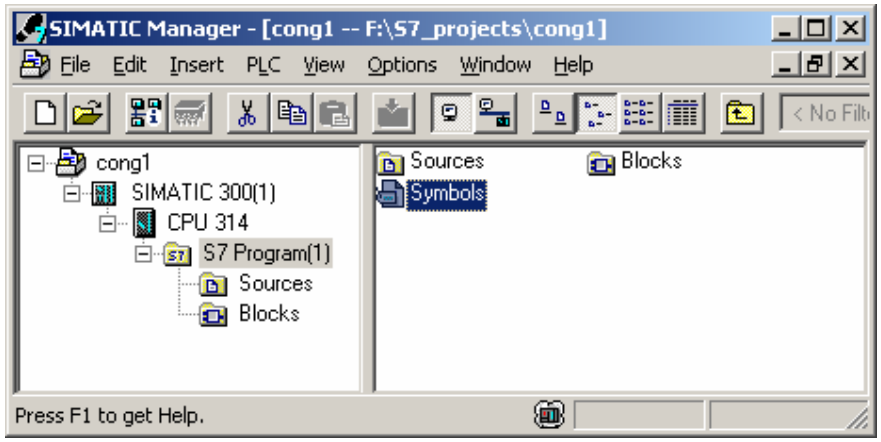


Hình 3-25:Màn hình soạn thảo trong khối FBs.

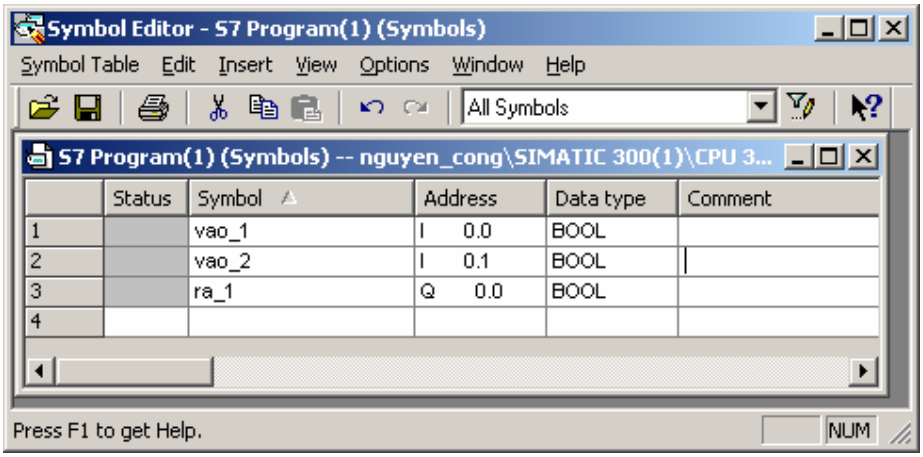
3.4.5.Sử dụng biến hình thức:

Step7 cung cấp một khả năng sử dụng tên hình thức trong lập trình thay vì các ký hiệu địa chỉ , chữ số khối FB, FC,...khó nhớ. Các tên hình thức được thay bởi một địa chỉ hay một tên khối tuỳ ý theo người lập trình tự đặt. Để làm được điều này, người lập trình cần phải khai báo trước trong một bảng có tên là Symbols.

Kích chuột vào thư mục mẹ của Block, ở đây là thư mục với tên mặc định là S7 Program(1), sau đó nháy phím chuột trái tại biểu tượng Symbole như hình vẽ ta sẽ có màn hình soạn thảo bằng các tên hình thức sau:



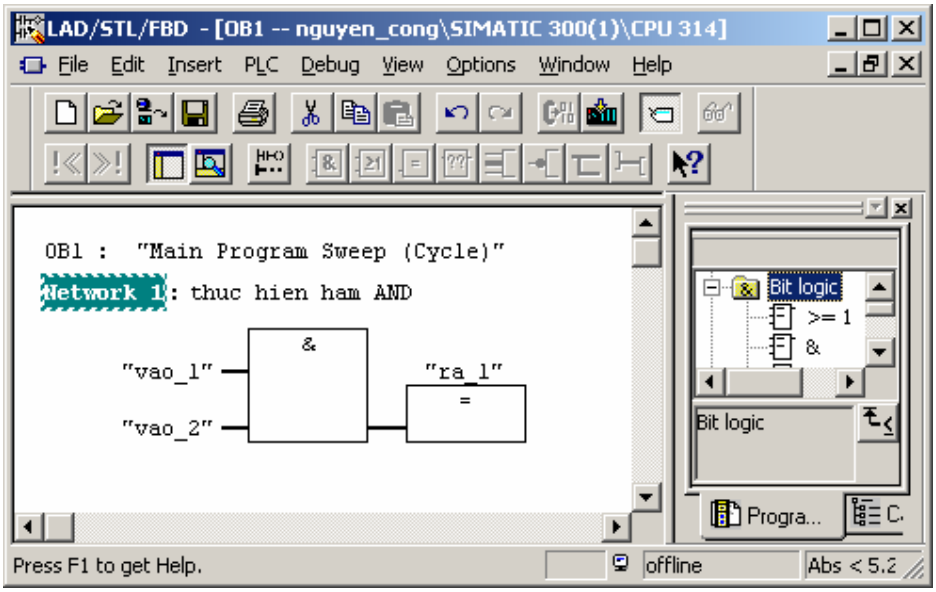
Hình3-26: Sử dụng biến hình thức.



Hình 3-27: Ghi các ký hiệu biến hình thức vào bảng Symbol.

Sau khi điền đầy đủ tên hình thức, địa chỉ ô nhớ mà nó thay thế (hầu hết kiểu dữ liệu đều được S7 tự xác định căn cứ vào địa chỉ ô nhớ) và cất vào Project, ta sẽ quay trở lại màn hình chính của S7. Mở một khối chương trình, ví dụ OB1 và chọn biểu tượng dùng biến hình thức ta sẽ chuyển sang dạng soạn thảo với những biến hình thức như đã đặt sẵn trong bảng Symbole.

ví dụ :



Hình 3-28: Màn hình soạn thảo với các tên biến hình thức.

Muốn quay trở về để sử dụng lại các ký hiệu địa chỉ tuyệt đối ta nhấn lại nút đã chọn ban đầu là biểu tượng này  nằm trên thanh công cụ .

3.5.Nạp chương trình và giám sát việc thực hiện chương trình.

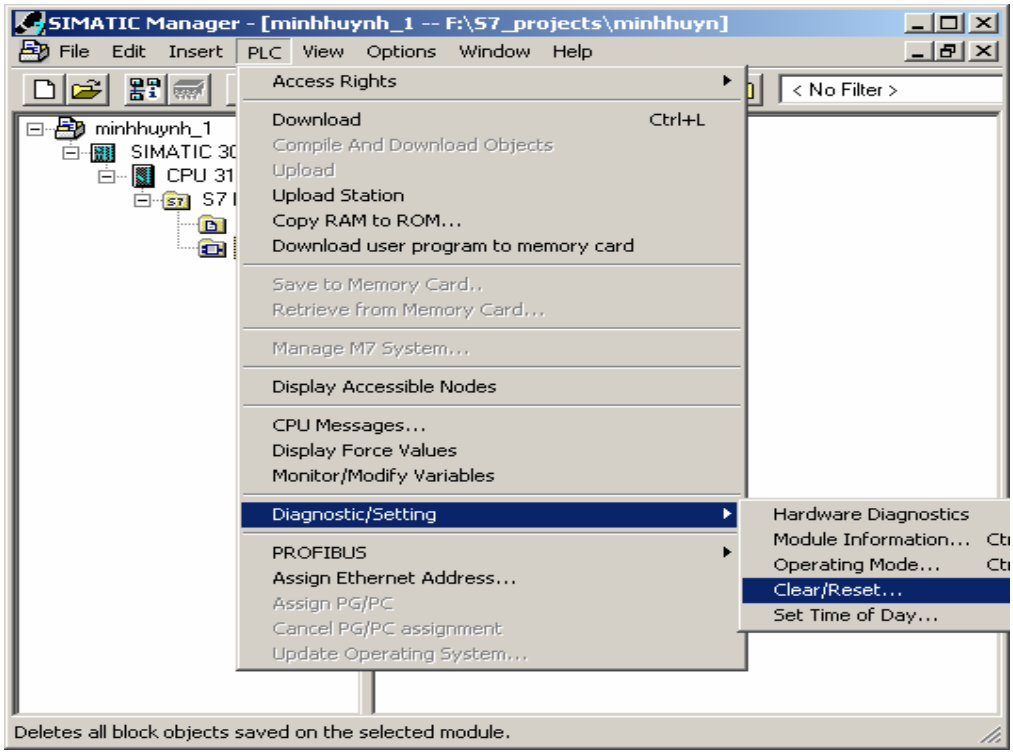
3.5.1. Nạp chương trình soạn thảo từ PC xuống CPU:

Chương trình sau khi đã soạn thảo cần được truyền xuống CPU. Để làm được điều này, ta nhấn chuột trái vào biểu tượng này  trên thanh công cụ và trả lời đầy đủ các câu hỏi. Chú ý khi nạp chương trình cần phải đặt CPU ở trạng thái ***Stop*** hoặc đặt CPU ở trạng thái ***RUN-P***.

3.5.2.Xoá chương trình đã có trong CPU:

Để thực hiện việc nạp chương trình mới từ PC xuống CPU ta cần thực hiện công việc xoá chương trình đã có sẵn trong CPU. Điều này ta thực hiện các bước như sau:

- Đưa trạng thái của CPU về STOP : Từ màn hình chính của Step7 ta chọn lệnh:

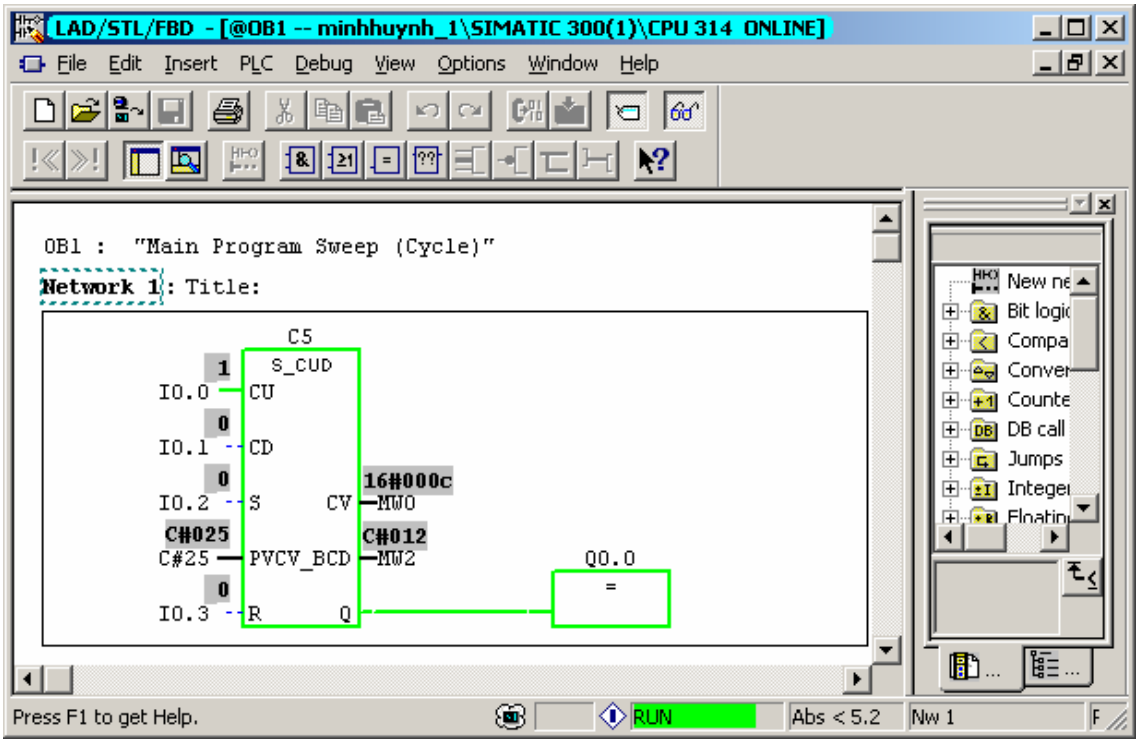


Hình 3-29

3.5.3.Quan sát việc thực hiện chương trình:

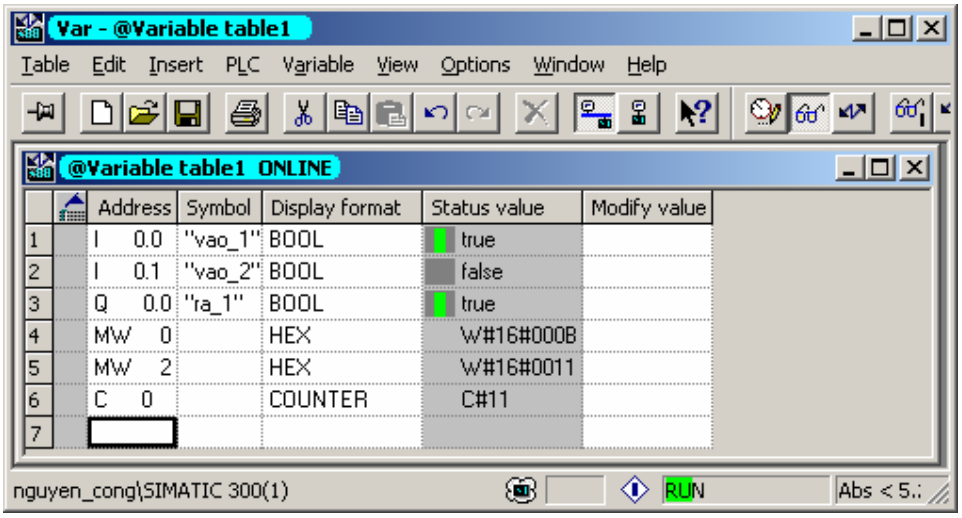
Sau khi đã nạp chương trình soạn thảo xuống CPU lúc này chương trình đã được ghi vào bộ nhớ của CPU. Khi đó ta có thể tách rời PC và CPU của S7 mà chương trình vẫn hoạt động bình thường. Để thực hiện việc quan sát quá trình hoạt động của chương trình và CPU ta sử dụng chức năng giám sát chương trình bằng cách nhấn vào biểu tượng này  trên thanh công cụ. Sau khi chọn chức năng giám sát chương trình này thì trên màn hình sẽ xuất hiện một cửa sổ sau:

Tuỳ theo kiểu viết chương trình mà ta nhận được sự khác nhau về kiểu hiển thị trên màn hình (Dưới đây sử dụng kiểu viết chương trình FBD).



Hình 3-30: Quan sát quá trình hoạt động.

Ngoài ra ta còn có thể quan sát được nội dung của ô nhớ. Những ô nhớ muốn quan sát cần phải khai báo trong bảng Variable.



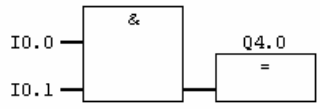
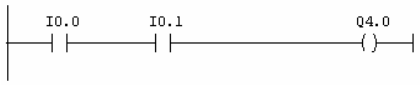
Hình 3-31: Quan sát nội dung của ô nhớ.

Sau khi khai báo tất cả các biến cần quan sát ta kích vào phím quan sát trên màn hình xuất hiện cửa sổ như hình trên. Tuỳ theo yêu cầu mà ta kích vào phím quan sát tương ứng trên màn hình sẽ hiển thị nội dung của ô nhớ tại thời điểm hiện tại hay liên tục quan sát theo từng thời điểm.

CHƯƠNG 4. CÁC HÀM CƠ BẢN:

4.1.Nhóm hàm Logic tiếp điểm:

1/ Hàm AND : Toán hạng là kiểu dữ liệu BOOL hay địa chỉ bit I,Q, M, T, C, D, L

FBD	LAD	STL
		A I 0.0 A I 0.1 = Q 4.0

Hình 4-1: Cách khai báo hàm AND

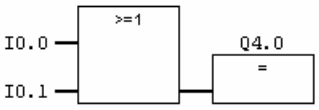
Tín hiệu ra Q4.0 sẽ bằng 1 khi đồng thời tín hiệu I0.0=1 và I0.1=1.

Dữ liệu vào và ra :

Vào: I0.0, I0.1: BOOL

Ra : Q4.0 : BOOL

2/ Hàm OR : *Toán hạng là kiểu dữ liệu BOOL hay địa chỉ bit I,Q, M, T, C, D, L.*

FBD	LAD	STL
		O I 0.0 O I 0.1 = Q 4.0

Hình 4-2: Khai báo hàm OR

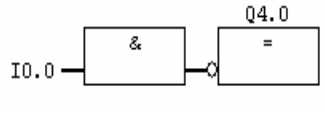
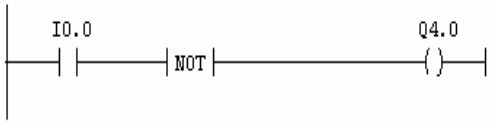
Tín hiệu ra sẽ bằng 1 khi ít nhất có một tín hiệu vào bằng 1.

Dữ liệu vào và ra:

Vào : I0.0, I0.1: BOOL

Ra : Q4.0: BOOL

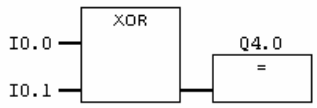
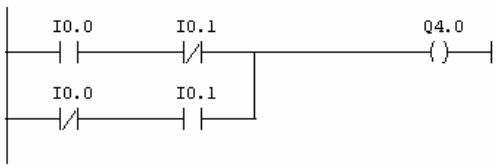
3/ Hàm NOT:

FBD	LAD	STL
		U E 0.0 NOT = A 4.0

Hình 4-3: Khai báo hàm thực hiện chức năng phủ định.

Tín hiệu ra sẽ là nghich đảo của tín hiệu vào.
Dữ liệu vào và ra:
Vào : I0.0 : BOOL
Ra : Q4.0 : BOOL

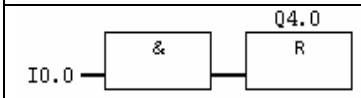
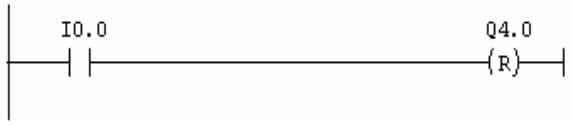
4/ Hàm XOR: Toán hạng là kiểu dữ liệu BOOL hay địa chỉ bit I, Q, M, T, C, D, L.

FBD	LAD	STL
		X I 0.0 X I 0.1 = Q 4.0

Hình 4-4: Khối thực hiện chức năng XOR.

Tín hiệu ra Q4.0= 1 khi I0.0 khác I0.2
Dữ liệu vào và ra:
Vào: I0.0, I0.1 : BOOL
Ra : Q4.0 : BOOL

5/ Lệnh xoá RESET : Toán hạng là địa chỉ bit I, Q, M, T, C, D, L.

FBD	LAD	STL
		A I 0.0 R Q 4.0

Hình 4-5: Khối thực hiện chức năng RESET

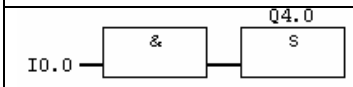
Tín hiệu ra Q4.0 = 0 (Q4.0 sẽ được xoá) khi I0.0 =1 .

Dữ liệu vào và ra:

Vào: I0.0 : BOOL

Ra : Q4.0 : BOOL

6/ Lệnh SET : Toán hạng là địa chỉ bit I, Q, M, T, C, D, L.

FBD	LAD	STL
		A I 0.0 S Q 4.0

Hình 4-6: Khối thực hiện chức năng SET.

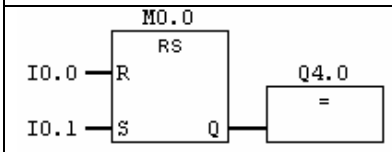
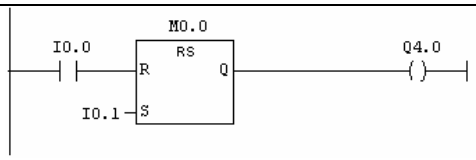
Tín hiệu ra Q4.0 = 1 (Q4.0 sẽ được thiết lập) khi I0.0 =1.

Dữ liệu vào và ra:

Vào I0.0 : BOOL

Ra Q4.0 : BOOL

7/Bộ nhớ RS: Toán hạng là địa chỉ bit I, Q, M, D, L.

FBD	LAD	STL
		A I 0.0 R M 0.0 A I 0.1 S M 0.0 A M 0.0 = Q 4.0

Hình 4-7: Khối thực hiện chức năng RS.

Khi I0.0 = 1 và I0.1 =0 Merker M0.0 bị Reset và đầu ra Q4.0 là "0". Nếu I0.0 = 0 và I0.1 = 1 thì Set cho M0.0 và đầu ra Q4.0 là "1".

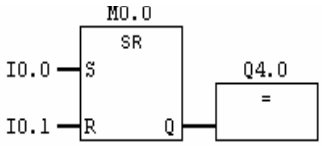
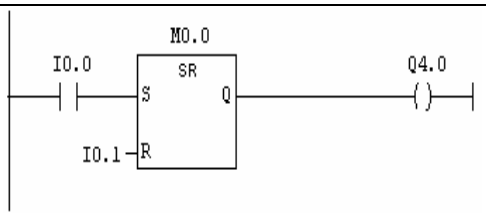
Khi cả hai đầu vào Set va Reset cùng đồng thời =1 thì M0.0 và Q4.0 có giá trị là "1".

Dữ liệu vào và ra:

Vào I0.0, I0.1 : BOOL

Ra Q4.0 : BOOL

8/ Bộ nhớ SR: Toán hạng là địa chỉ bit I, Q, M, D, L

FBD	LAD	STL
		A I 0.0 S M 0.0 A I 0.1 R M 0.0 A M 0.0 = Q 4.0

Hình 4-8: Khối thực hiện chức năng SR

Khi I0.0 = 1 và I0.1 =0 thì Set cho Merker M0.0 và đầu ra Q4.0 là "1". Nếu I0.0 = 0 và I0.0 = 1 thì M0.0 bị Reset và đầu ra Q4.0 là "0".

Khi cả hai đầu vào Set va Reset cùng đồng thời =1 thì M0.0 và Q4.0 có giá trị là "0".

Dữ liệu vào và ra:

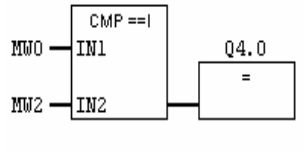
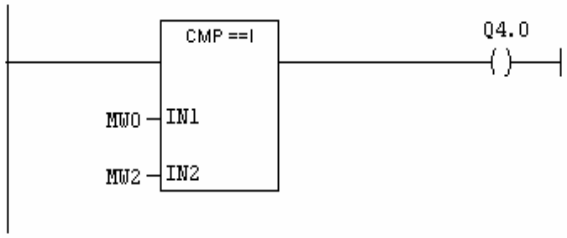
Vào I0.0, I0.1 : BOOL

Ra Q4.0 : BOOL

<u>Chú ý:</u> Trong kỹ thuật số trạng thái của trigơ RS sẽ bị cấm khi R=1 và S=1. Nên ở đây có hai loại bộ nhớ RS và SR là loại Trigơ ưu tiên R hay ưu tiên S

4.2. Nhóm hàm so sánh:

4.2.1.Nhóm hàm so sánh số nguyên 16 bit:

FBD	LAD	STL
		L MW 0 L MW 2 ==I = Q 4.0

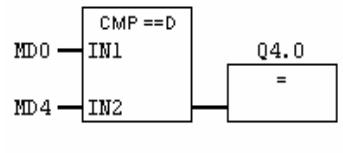
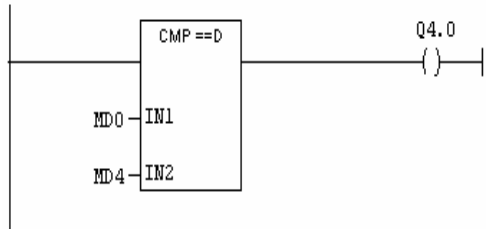
Hình 4-9: Khối thực hiện chức năng so sánh bằng nhau

Có các dạng so sánh hai số nguyên 16 bits như sau :

- Hàm so sánh bằng nhau giữa hai số nguyên 16 bits: ==
- Hàm so sánh khác nhau giữa hai số nguyên 16 bits: <>
- Hàm so sánh lớn hơn giữa hai số nguyên 16 bits: >
- Hàm so sánh nhỏ hơn giữa hai số nguyên 16 bits: <
- Hàm so sánh lớn hơn hoặc bằng nhau giữa hai số nguyên 16 bits: >=
- Hàm so sánh nhỏ hơn hoặc bằng nhau giữa hai số nguyên 16 bits: <=

Trong ví dụ trên đầu ra Q4.0 sẽ là "1" khi MW0 = MW1.

4.2.2. Nhóm hàm so sánh hai số nguyên 32 bits:

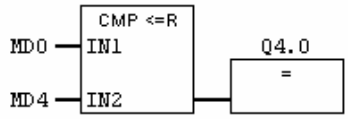
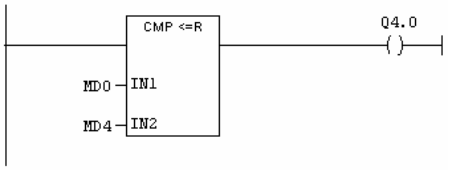
FBD	LAD	STL
		L MD 0 L MD 4 ==D = Q 4.0

Hình 4-10: Khối thực hiện chức năng so sánh

Trong ví dụ trên đầu ra Q4.0 sẽ là "1" khi MD0 = MD4.

- Hàm so sánh bằng nhau giữa hai số nguyên 32 bits: ==
- Hàm so sánh khác nhau giữa hai số nguyên 32 bits: <>
- Hàm so sánh lớn hơn giữa hai số nguyên 32 bits: >
- Hàm so sánh nhỏ hơn giữa hai số nguyên 32 bits: <
- Hàm so sánh lớn hơn hoặc bằng nhau giữa hai số nguyên 32 bits: >=
- Hàm so sánh nhỏ hơn hoặc bằng nhau giữa hai số nguyên 32 bits: <=

4.2.3.Nhóm hàm so sánh các số thực 32 bits

FBD	LAD	STL
		L MD 0 L MD 4 <=R = Q 4.0

Hình 4-11: Khối thực hiện chức năng so sánh hai số thực.

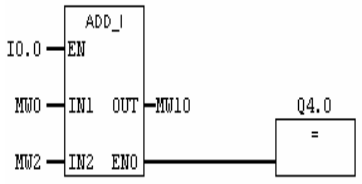
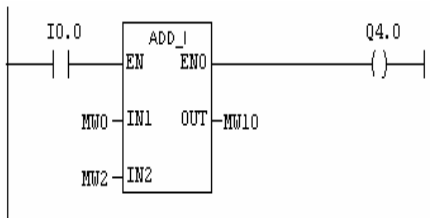
Trong ví dụ trên đầu ra Q4.0 sẽ là "1" khi MD0 < MD1 .
Các dạng so sánh hai số thực 32 bits như sau :

- Hàm so sánh bằng nhau giữa hai số thực 32 bits: ==
- Hàm so sánh khác nhau giữa hai số thực 32 bits: <>
- Hàm so sánh lớn hơn giữa hai số thực 32 bits: >
- Hàm so sánh nhỏ hơn giữa hai số thực 32 bits: <
- Hàm so sánh lớn hơn hoặc bằng nhau giữa hai số thực 32 bits: >=
- Hàm so sánh nhỏ hơn hoặc bằng nhau giữa hai số thực 32bits: <=

4.3.Các hàm toán hoc:

4.3.1. Nhóm hàm làm viêc với số nguyên 16 bits:

1/ Cộng hai số nguyên 16 bits:

FBD	LAD	STL
		A I 0.0 JNB _001 L MW 0 L MW 2 +I T MW 10 AN OV SAVE CLR _001: A BR = Q 4.0

Hình 3-12: Khối thực hiện chức năng cộng hai số nguyên 16 bits.

Dữ liệu vào và ra:

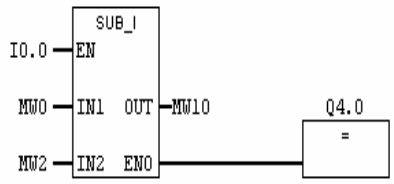
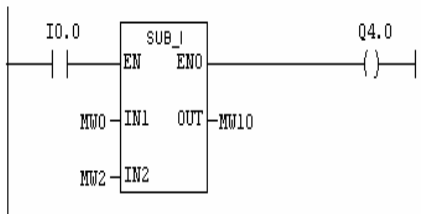
EN: BOOL IN1: INT

IN2: INT OUT: INT ENO: BOOL

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện cộng hai số nguyên 16 bits MW0 với MW2. Kết quả được cất vào MW10.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

2/ Trừ hai số nguyên 16 bits:

FBD	LAD	STL
		A I 0.0 JNB _001 L MW 0 L MW 2 -I T MW 10 AN OV SAVE CLR _001: A BR = Q 4.0

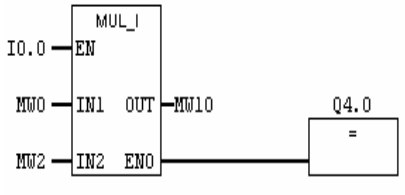
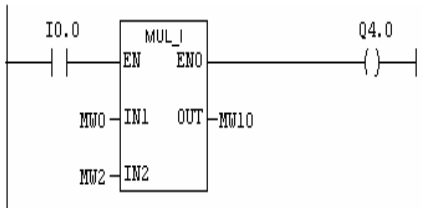
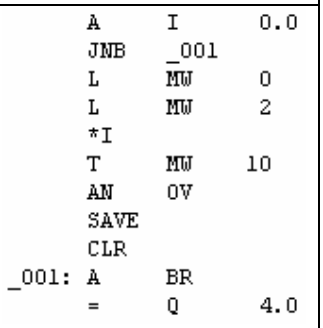
Hình 4-13: Khối thực hiện chức năng trừ hai số nguyên 16 bits

Dữ liệu vào và ra:

EN : BOOL IN1: INT
IN2: INT OUT: INT ENO: BOOL

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện trừ hai số nguyên 16 bits MW0 với MW2. Kết quả được cất vào MW10.
Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

3/ Nhân hai số nguyên 16 bits:

FBD	LAD	STL
		

Hình 4-14: Khối thực hiện chức năng nhân hai số 16 bits.

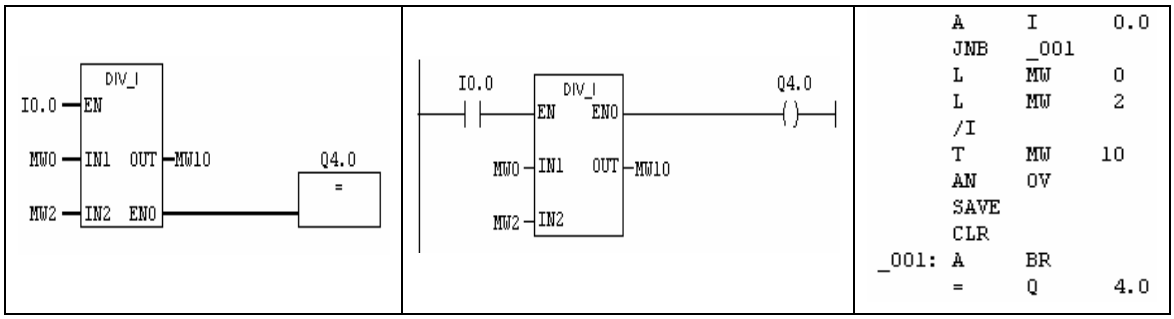
Dữ liệu vào và ra:
EN: BOOL IN1: INT
IN2: INT OUT: IN ENO: BOOL

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện nhân hai số nguyên 16 bits MW0 với MW2. Kết quả được cất vào MW10.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

4/ Chia hai số nguyên 16 bits:

FBD	LAD	STL



Hình 4-15: Khối thực hiện chức năng chia hai số nguyên 16 bits

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện chia hai số nguyên 16 bits MW0 với MW2. Kết quả được cất vào MW10.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

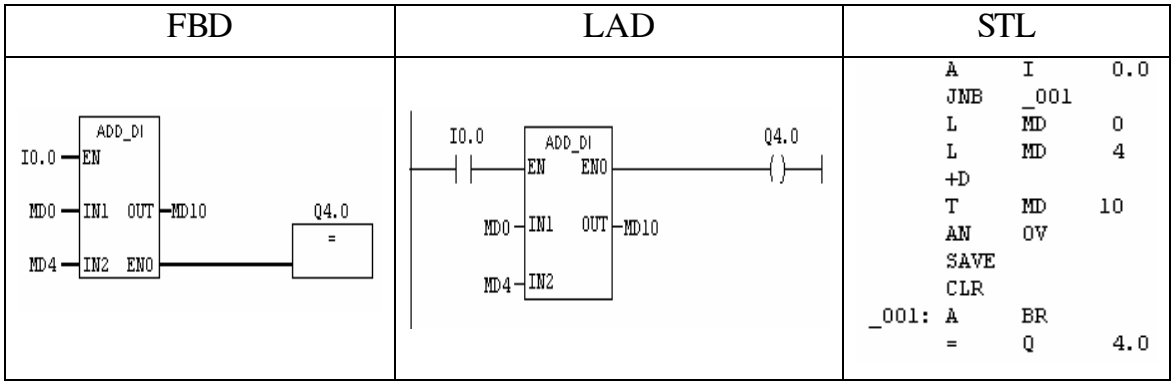
4.3.2.Nhóm hàm làm việc với số nguyên 32 bits:

1/ Cộng hai số nguyên 32 bits:

Dữ liệu vào và ra:

EN: BOOL IN1: DINT

IN2: DINT OUT: DINT ENO: BOOL



Hình 4-16: Khối thực hiện chức năng cộng hai số nguyên 32 bits

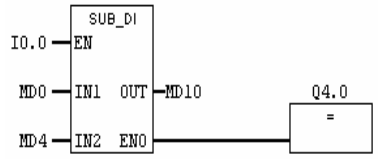
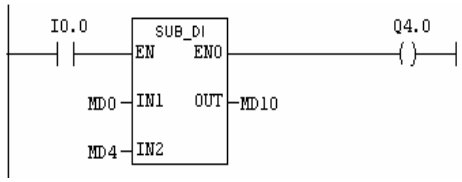
Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện công hai số nguyên 32 bits MD0 với MD4. Kết quả được cất vào MD10.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

2/ Trừ hai số nguyên 32 bits:

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện trừ hai số nguyên 32 bits MD0 với MD4. Kết quả được cất vào MD10.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 0 L MD 4 -D T MD 10 AN OV SAVE CLR _001: A BR = Q 4.0

Hình 4-17: Khối thực hiện chức năng trừ hai số nguyên 32 bits

Dữ liệu vào và ra:

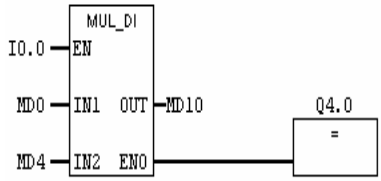
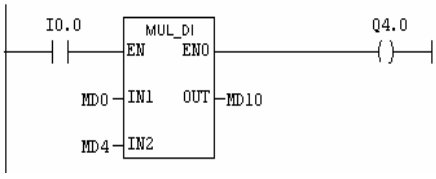
EN: BOOL IN1: DINT

IN2: DIN OUT: DINT ENO: BOOL

3/ Nhân hai số nguyên 32 bits:

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện nhân hai số nguyên 32 bits MD0 với MD4. Kết quả được cất vào MD10.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 0 L MD 4 *D T MD 10 AN OV SAVE CLR _001: A BR = Q 4.0

Hình 4-18: Khối thực hiện chức năng nhân hai số nguyên 32 bit

Dữ liệu vào và ra:

EN: BOOL IN1: DINT

IN2: DINT OUT: DINT ENO: BOOL

4/ Chia hai số nguyên 32 bits :

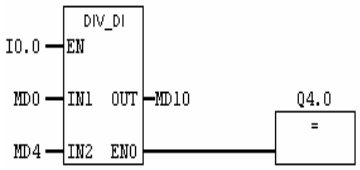
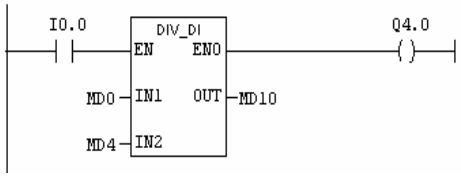
Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện chia hai số nguyên 32 bits MD0 với MD4. Kết quả được cất vào MD10.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

Dữ liệu vào và ra:

EN: BOOL IN1: DINT

IN2: DINT OUT: DINT ENO: BOOL

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 0 L MD 4 /D T MD 10 AN OV SAVE CLR _001: A BR = Q 4.0

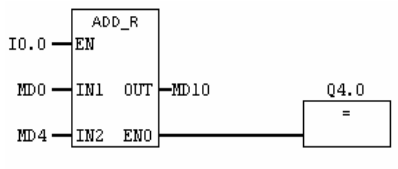
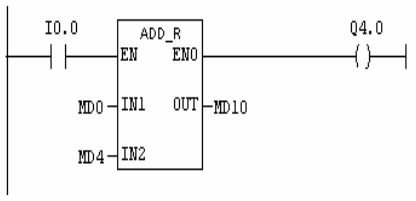
Hình 4-19: Khối thực hiện chức năng chia hai số nguyên 32 bits

4.3.3.Nhóm hàm làm việc với số thực:

1/ Cộng hai số thực:

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện cộng hai số thực MD0 + MD4. Kết quả được cất vào MD10.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 0 L MD 4 +R T MD 10 AN OV SAVE CLR _001: A BR = Q 4.0

Hình 4-20: Khối thực hiện chức năng cộng hai số thực

Dữ liệu vào và ra:

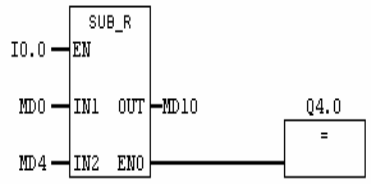
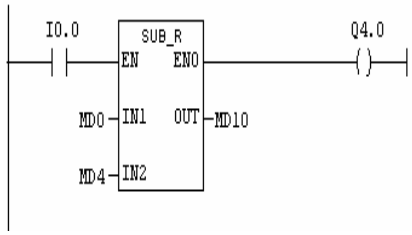
EN: BOOL IN1: REAL

IN2: REAL OUT: REAL ENO: BOOL

2/ Hàm trừ hai số thực:

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện trừ hai số thực MD0 - MD4. Kết quả được cất vào MD10.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 0 L MD 4 -R T MD 10 AN OV SAVE CLR _001: A BR = Q 4.0

Hình 4-21: Khối thực hiện chức năng trừ hai số thực.

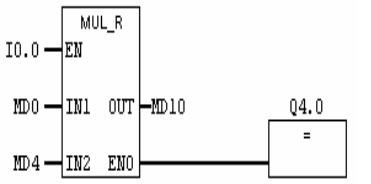
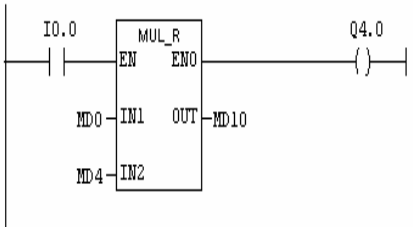
Dữ liệu vào và ra:

EN: BOOL IN1: REAL

IN2: REAL OUT: REAL ENO: BOOL

3/ Nhân hai số thực:

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện nhân hai số thực MD0 . MD4. Kết quả được cất vào MD10.

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 0 L MD 4 *R T MD 10 AN OV SAVE CLR _001: A BR = Q 4.0

Hình 4-22: Khối thực hiện chức năng nhân hai số thực.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

Dữ liệu vào và ra:

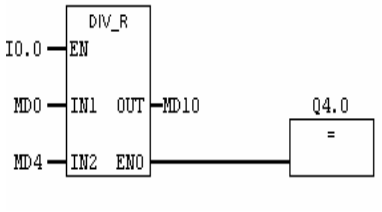
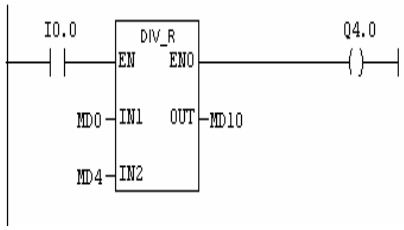
EN: BOOL IN1: REAL

IN2: REAL OUT: REAL ENO: BOOL

4/ Chia hai số thực:

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiện chia hai số thực MD0 : MD4. Kết quả được cất vào MD10.

Trong trường hợp tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 0 L MD 4 /R T MD 10 AN OV SAVE CLR _001: A BR = Q 4.0

Hình 4-23: Khối thực hiện chức năng nhân hai số thực

Dữ liệu vào và ra:

EN: BOOL IN1: REAL

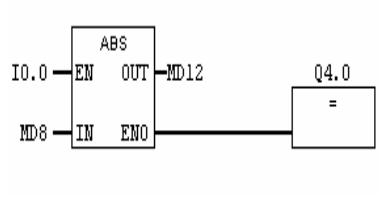
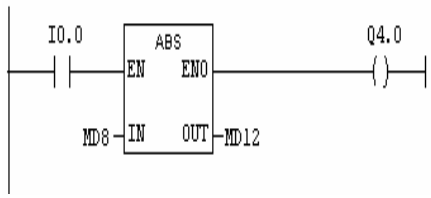
IN2: REAL OUT: REAL

ENO: BOOL

5/ Hàm lấy giá trị tuyệt đối : ABS

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiên chức năng lấy gía trị tuyệt đối của MD8 rồi cất vào MD12

Khi tín hiệu vao I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 8 ABS T MD 12 SET SAVE CLR _001: A BR = Q 4.0

Hình 4-24: Khối thực hiện chức năng lấy giá trị thuyệt đối.

Dữ liệu vào và ra:

EN: BOOL IN: REAL

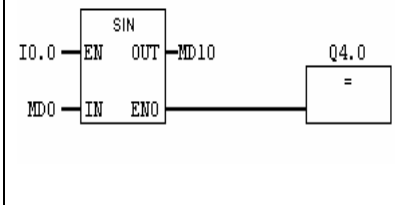
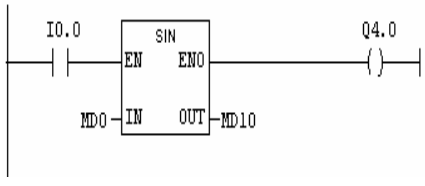
OUT: REAL ENO: BOOL

Ví dụ: MD8= -6,234 x 10-3 thi sau khi thực hiên chức năng ABS giá trị MD12 = 6,234 x 10-3.

6/ Hàm SIN, COS, TAN, ASIN, ACOS, ATAN:

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm sẽ thực hiên chức năng tinh SIN, COS, TAN, ASIN, ACOS, ATAN của MD0 rồi cất vào MD10.

Khi tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm sẽ không thực hiện chức năng.

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 0 SIN T MD 10 AN OV SAVE CLR _001: A BR = Q 4.0

Hình 4-25: Khối thực hiện chức năng tính hàm Sin.

Dữ liệu vào và ra:

EN: BOOL IN: REAL

OUT: REAL ENO: BOOL

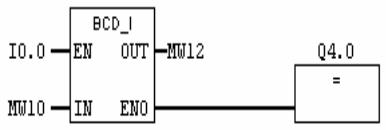
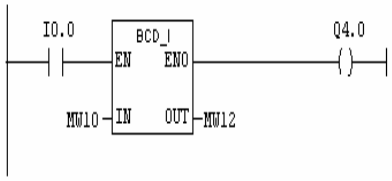
4.4.Nhóm hàm đổi kiểu dữ liêu :

Trong ngôn ngữ lập trình của S7-300 có một số kiểu dữ liệu khác nhau như:
- Số nguyên 16 bits (Interger)
- Số nguyên 32 bits (DI)
- Số nguyên dạng BCD.
- Số thực REAL
- và một số dạng dữ liệu khác .

Khi làm việc với nhiều dạng dữ liệu khác nhau cho ta vấn đề cần phải chuyển đổi chúng. Ví dụ khi đọc tín hiệu từ cổng vào tương tự ta nhận được số liệu dạng nguyên 16 bits mang giá trị tín hiệu tương tự chứ không phải bản thân giá trị đó, bởi vậy để xử lý tiếp thì cần thiết phải chuyển đổi số nguyên đó thành đúng giá trị thực, dấu phảy động của tín hiệu tương tự ở cổng. Ta có một số hàm chuyển đổi các dạng dữ liệu như sau:

4.4.1.Hàm chuyển số BCD thành số số nguyên 16 bits:

FBD	LAD	STL

		`A I 0.0`
		`JNB _001`
		`L MW 10`
		`BTI`
		`T MW 12`
		`SET`
		`SAVE`
		`CLR`
		`_001: A BR`
		`= Q 4.0`

Hình 4-26: Chuyển đổi số BCD sang dạng số nguyên 16 bits.

Dữ liệu vào và ra:

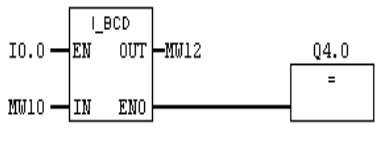
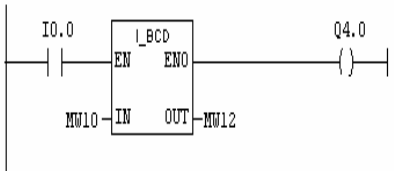
EN: BOOL IN: WORD

OUT: INT ENO: BOOL

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm thực hiên chức năng chuyển số BCD (MW10) sang số nguyên rồi cất vào MW12.

Khi tin hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm không thực hiên chức năng chuyển đổi.

4.4.2. Hàm chuyển đổi số nguyên 16 bits sang dang BCD.

FBD	LAD	STL
		A I 0.0 JNB _001 L MW 10 ITB T MW 12 AN OV SAVE CLR _001: A BR = Q 4.0

Hình 4-27: Chuyển đổi số nguyên sang số BCD.

Dữ liệu vào và ra:

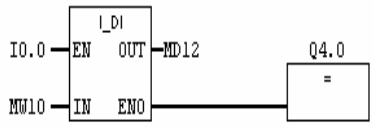
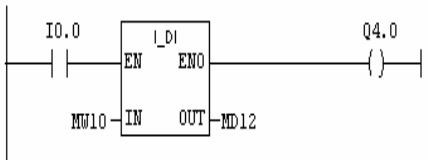
EN: BOOL IN: INT

OUT: BCD ENO: BOOL

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm thực hiên chức năng chuyển số nguyên 16 bits (MW10) sang số BCD rồi cất vào MW12.

Khi tin hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm không thực hiên chức năng chuyển đổi.

4.4.3. Hàm chuyển đổi số nguyên 16 bits sang số nguyên 32 bits:

FBD	LAD	STL
		A I 0.0 JNB _001 L MW 10 ITD T MD 12 SET SAVE CLR _001: A BR = Q 4.0

Hình 4-28: Chuyển đổi số nguyên 16 bits sang số nhuyên 32 bits.

Dữ liệu vào và ra:

EN: BOOL IN: INT

OUT: DINT ENO: BOOL

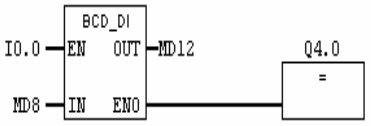
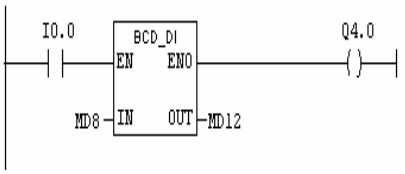
Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm thực hiên chức năng chuyển số nguyên 16 bits (MW10) sang số nguyên 32 bits rồi cất vào MW12.

Khi tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm không thực hiên chức năng chuyển đổi.

4.4.4.Chuyển đổi số BCD sang số nguyên 32 bits:

Khi tín hiệu vào I0.0 = 1 đầu ra Q4.0 = 1 và hàm thực hiên chức năng chuyển số BCD (MW10) sang số nguyên 32 bits rồi cất vào MW12.

Khi tín hiệu vào I0.0 = 0 đầu ra Q4.0 = 0 và hàm không thực hiện chức năng chuyển đổi.

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 8 BTD T MD 12 SET SAVE CLR _001: A BR = Q 4.0

Hình 4-29: Chuyển số BCD sang số nguyên 32 bits

-Kiểu dữ liệu vào/ra:

EN: BOOL IN: DWORD

OUT: DINT ENO: BOOL.

4.4.5.Hàm đảo giá tri các bits .

1/Với số nguyên có độ dài 16 bits:

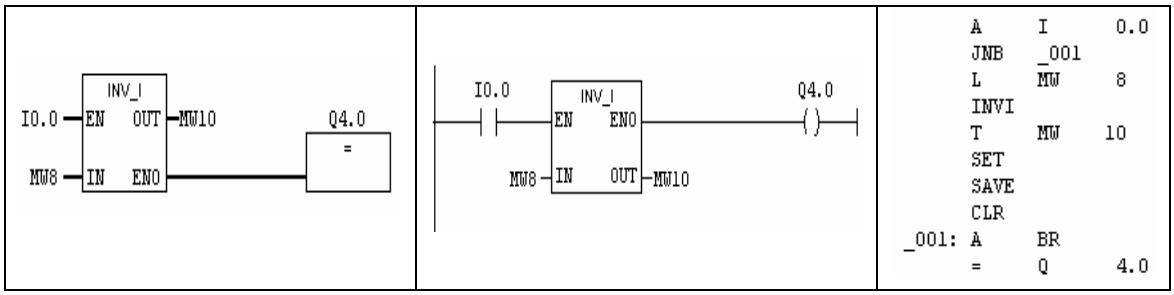
-Nguyên lý hoạt động: Hàm sẽ thực hiện chức năng chuyển đổi giá trị các bits trong MW8 rồi cất vào MW10 khi tín hiệu I0.0 =1. Đồng thời giá trị Q4.0 = 1. Khi I0.0 = 0, giá trị Q4.0 = 0

-Kiểu dữ liệu vào/ra:

EN: BOOL IN: INT

OUT: INT ENO: BOOL

FBD	LAD	STL



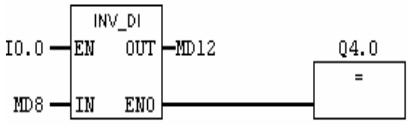
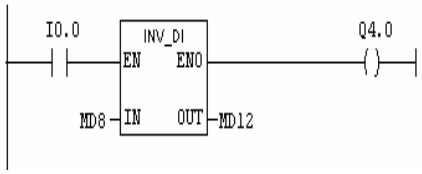
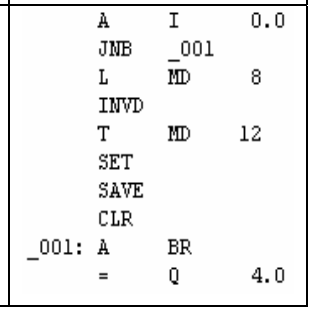
Hình 4-30: Hàm thực hiện chức năng đảo giá trị các bits

-Ví dụ:

Trước khi thực hiện MW8 = 01000001 10000001

Sau khi thực hiện MW10 = 10111110 01111110

2/ Với số nguyên có độ dài 32 bits.

FBD	LAD	STL
		

Hình 4-31: Hàm thực hiện chức năng đảo giá trị các bits.

-Nguyên lý hoạt động: Hàm sẽ thực hiện chức năng chuyển đổi giá trị các bits trong MD8 rồi cất vào MD12 khi tín hiệu I0.0 =1. Đồng thời giá trị Q4.0 = 1 . Khi I0.0 = 0, giá trị Q4.0 = 0

-Kiểu dữ liệu vào/ra:

EN: BOOL IN: DINT

OUT: DINT ENO: BOOL

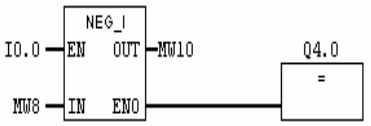
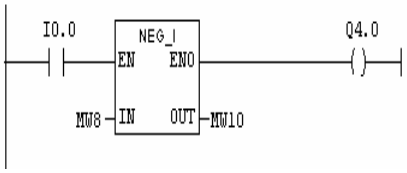
- Ví dụ:

Trước khi thực hiện: MD8 = F0FF FFF0

Sau khi thực hiện : MD12 = 0F00 000F

4.4.6.Các hàm đổi dấu :

Hàm sẽ thực hiện chức năng đổi dấu dữ liệu vào . Các hàm đổi dấu như đổi dấu số thực độ dài 16bits (I), 32 bits (DI) hay số nguyên (R).

FBD	LAD	STL
		A I 0.0 JNB _001 L MW 8 NEGI T MW 10 AN OV SAVE CLR _001: A BR = Q 4.0

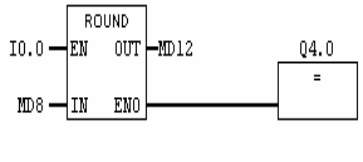
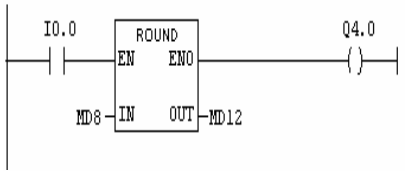
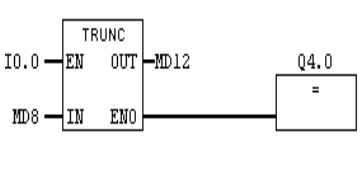
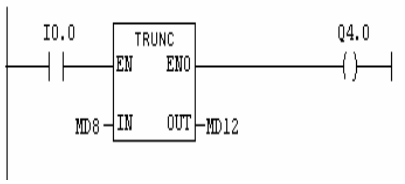
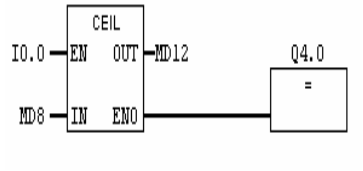
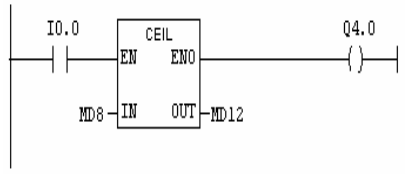
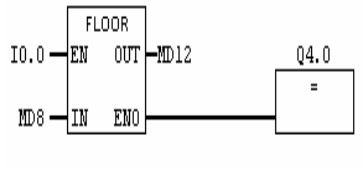
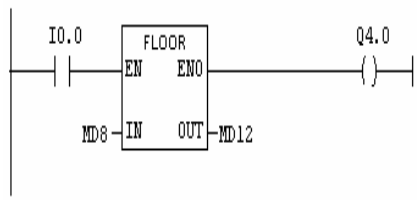
Hình 4-32:

Dạng dữ liệu vào:

	NEG_I	NEG_DI	NEG_R
EN	BOOL	BOOL	BOOL
IN	INT	DI	REAL
OUT	INT	DI	REAL
ENO	BOOL	BOOL	BOOL

Ví dụ: Trước khi thực hiện MW8 = +10, sau khi thực hiện MW10 = -10.

4.4.7.Các hàm thực hiên chức năng làm tròn (đổi kiểu dữ liêu):

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 8 RND T MD 12 AN OV SAVE CLR _001: A BR = Q 4.0
		A I 0.0 JNB _001 L MD 8 TRUNC T MD 12 AN OV SAVE CLR _001: A BR = Q 4.0
		A I 0.0 JNB _001 L MD 8 RND+ T MD 12 AN OV SAVE CLR _001: A BR = Q 4.0
		A I 0.0 JNB _001 L MD 8 RND- T MD 12 AN OV SAVE CLR _001: A BR = Q 4.0

Hình 4-33:

-Hàm ROUND : (chuyển số thực thành số nguyên gần nhất) thực hiện làm tròn như sau: nếu phần lẻ < 0,5 thì làm tròn xuống. Nếu phần lẻ > 0,5 thì làm tròn lên.

Ví dụ: 1,2 -> 1 ; 1,6 -> 2.

-1,2 -> -1 ; -1,6 -> -2.

-Hàm TRUNC: (lấy phần nguyên cắt bỏ phần lẻ) thực hiện làm tròn xuống giá trị tròn nhỏ

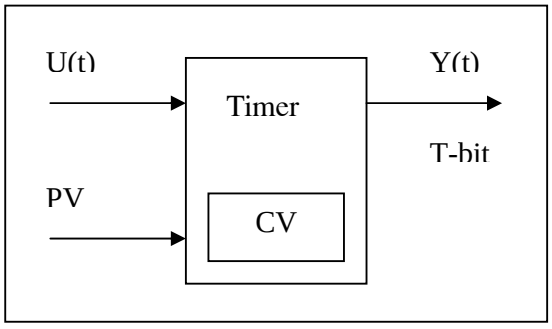
ví dụ: dữ liệu vào từ 1,1 đến 1,9 -> 1.

-Hàm CEIL: thực hiện làm tròn lên.
ví dụ: dữ liệu vào từ 1,1 đến 1,9 -> 2.
-Hàm FLOOR: thực hiện làm tròn xuống.
ví dụ: +1,7 -> 1 ; - 1,7 -> -2

4.5.Bộ thời gian:

4.5.1 Nguyên lý làm việc chung của bô Timer.

Bộ thời gian Timer là bộ tạo thời gian trễ T mong muốn giữa tín hiệu logic đầu vào X(t) và đầu ra Y(t)



Hình 4-34: Sơ đồ khối bộ thời gian.

S7-300 có 5 bộ thời gian Timer khác nhau. Tất cả 5 loại Timer này cùng bắt đầu tạo thời gian trễ tín hiệu kể từ thời điểm có sườn lên của tín hiệu đầu vào , tức là khi có tín hiệu đầu vào U(t) chuyển trạng thái từ logic "0" lên logic"1", được gọi là thời điểm Timer được kích.

Thời gian trễ T mong muốn được khai báo với Timer bằng giá trị 16 bits bao gồm hai thành phần :

- Độ phân giải với đơn vị là mS. Timer của S7 có 4 loại phân giải khác nhau là 10ms, 100ms, 1s và 10s.
- Một số nguyên BCD trong khoảng từ 0 đến 999 được gọi là PV (Preset Value- giá trị đặt trước).

Như vậy thời gian trễ T mong muốn sẽ được tính như sau :

T= Độ phân giải x PV.

Ngay tại thời điểm kích Timer, giá trị PV được chuyển vào thanh ghi 16 bits của Timer T-Word (gọi là thanh ghi CV- Curren value- giá trị tức thời). Timer sẽ ghi nhớ khoảng thời gian trôi qua kể từ khi kích bằng cách giảm dần một cách tương ứng nội dung thanh ghi CV. Nếu nội dung thanh ghi CV trở về bằng 0 thì Timer đã đạt được thời gian mong muốn T và điều này được báo ra ngoài bằng cách thay đổi trạng thái tín hiệu đầu ra Y(t). Việc thông báo ra ngoài bằng cách đổi trạng thái tín hiệu dầu ra Y(t) như thế nào còn phụ thuộc vào loại Timer được sử dụng.

Bên cạnh sườn lên của tín hiệu đầu vào U(t), Timer còn có thể kích bằng sườn lên của tín hiệu kích chủ động có tên là tín hiệu ENABLE nếu như tại thời điểm có sườn lên của tín hiệu ENABLE, tín hiệu đầu vào U(t) có gic là "1".

Từng loại Timer được đánh số từ 0 đến 255 (tuỳ thuộc vào từng loại CPU). Một Timer được đặt tên là Tx, trong đó x là số hiệu của Timer (0<=x<=255). Ký hiệu Tx cũng đồng thời là tín hiệu hình thức của thanh ghi CV (T-Word) và đầu ra T-bits của Timer đó . Tuy chúng có cùng địa chỉ hình thức , nhưng T-Word và T-bits vẫn được phân biệt với nhau nhờ kiểu lệnh sử dụng toán hạng Tx. Khi dùng làm việc với từ Tx được hiểu là T-Word còn khi làm việc với điểm thi Tx được hiểu là T-bit.

Để xóa tức thời trạng thái của T-word và T-bit người ta sử dụng một tín hiệu reset Timer . Tại thời điểm sườn lên của tín hiệu này giá trị T-Word và T-bit đồng thời có giá trị bằng 0 tức là thanh ghi tức thời CV được đặt về 0 và tín hiệu đầu ra cũng có trạng thái Logic là "0". Trong thời gian tín hiệu Reset có giá trị logic là "1" Timer sẽ không làm việc.

4.5.2. Khai báo sử dụng:

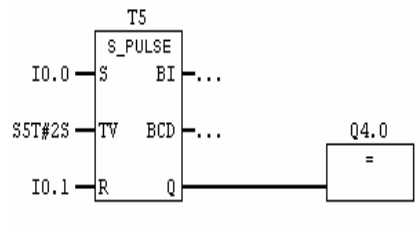
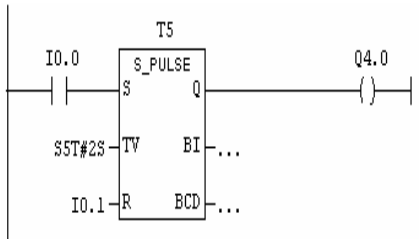
Các tín hiệu điều khiển cho một bộ Timer phải được khai báo bao gồm các bước sau:

- Khai báo tín hiệu ENABLE nếu muốn sử dụng tín hiệu chủ động kích.
- Khai báo tín hiệu đầu vào U(t).
- Khai báo thời gian trễ mong muốn TW.
- Khai báo loại Timer được sử dụng (SP, SE, SD, SS, SF).
- Khai báo tín hiệu xoá Timer nếu muốn sử dụng chế độ Reset chủ động.
  Trong các bước trên thì bước 1 và 5 có thể bỏ qua .
- Dạng dữ liệu vào / ra của bộ Timer:

S : BOOL	BI (DUAL): WORD
TW: S5TIME	BCD (DEZ) : WORD
R : BOOL	Q : BOOL

1. Bộ thời gian SP :

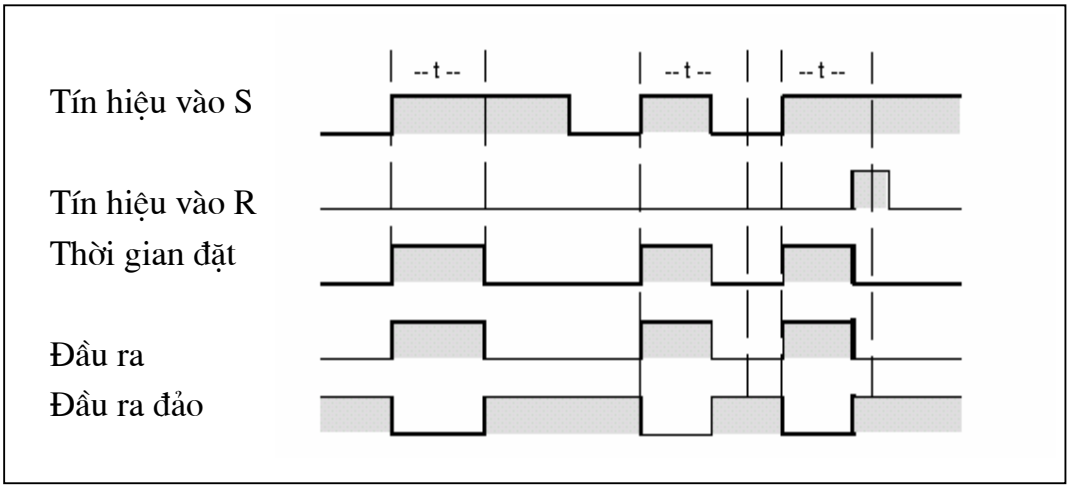
-Sơ đồ khối:

FBD	LAD	STL
		A I 0.0 L S5T#2S SP T 5 A I 0.1 R T 5 NOP 0 NOP 0 A T 5 = Q 4.0

Hình 4-35: Bộ thời gian SP.

-Nguyên lý làm việc:

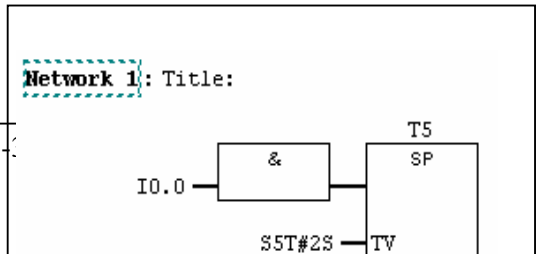
Tại thời điểm sườn lên của tín hiệu vào SET thời gian sẽ đượ tính đồng thời giá trị Logic ở đầu ra là "1". Khi thời gian đặt kết thúc giá trị đầu ra cũng trở về 0.



Hình 4-36: Giản đồ thời gian của bộ tạo trễ kiểu SP.

Khi có tín hiệu RESET (R) thời gian tính lập tức trở về 0 và tín hiệu đầu ra cũng giá trị là "0".

-Trường hợp không sử dụng các tín hiệu đầu vào SET(S), RESET (R), BI và BCD ta sử dụng khối Timer SI sau:

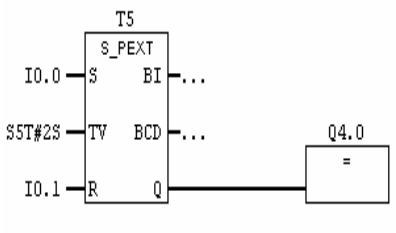
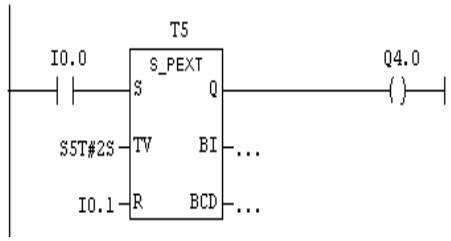


Tín hiệu đầu vào I0.0 chính là tín hiệu kích.
S5T#2s là thời gian đặt 2s

Tín hiệu ra của bộ thời gian tác động tới đầu ra Q4.0

Hình 4-37: Ví dụ khai báo một bộ thời gian SP

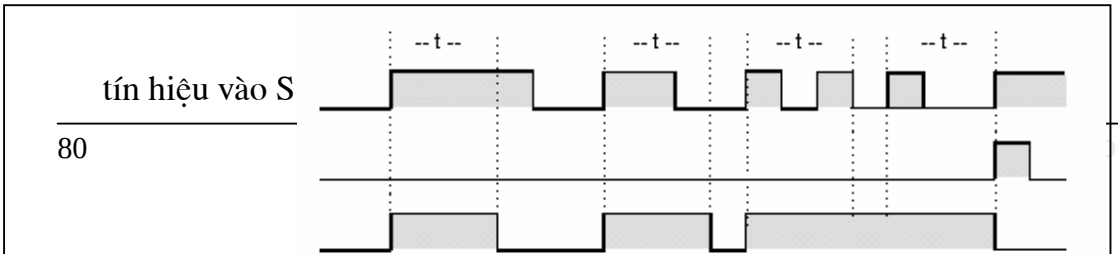
2. *Bộ thời gian SE.*

FBD	LAD	STL
		A I 0.0 L S5T#2S SE T 5 A I 0.1 R T 5 NOP 0 NOP 0 A T 5 = Q 4.0

Hình 4-38: Khối hàm thời gian SE

-Nguyên lý làm việc:

Tại thời điểm sườn lên của tín hiệu vào SET cuối cùng bộ thời gian được thiết lập và thời gian sẽ được tính đồng thời giá trị Logic ở đầu ra là "1". Kết thúc thời gian đặt tín hiệu đầu ra sẽ trở về 0.



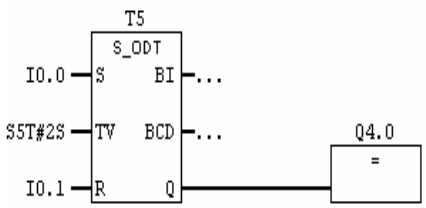
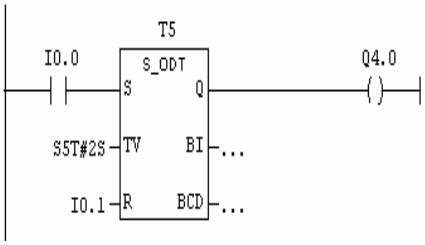
Tín hiệu vào R

Thời gian đặt
Đầu ra
Đầu ra đảo

Hình 4-39: Giản đồ thời gian khối SE

Khi có tín hiệu RESET (R) thời gian tính lập tức trở về 0 và tín hiệu đầu ra cũng giá trị là "0".

3. Bộ thời gian SD.

FBD	LAD	STL
		A I 0.0 L S5T#2S SD T 5 A I 0.1 R T 5 NOP 0 NOP 0 A T 5 = Q 4.0

Hình 4-40: Sơ đồ khối hàm SD.

-Nguyên lý làm việc:

Tại thời điểm sườn lên của tín hiệu vào SET bộ thời gian được thiết lập và thời gian sẽ được tính. Kết thúc thời gian đặt tín hiệu đầu ra sẽ có giá trị là "1". Khi tín hiệu đầu vào kích S là "0" đầu ra cũng lập tức trở về "0" nghĩa là tín hiệu đầu ra sẽ không được duy trì hi tín hiệu kích có giá trị là "0".

Tín hiệu vào R

Thời gian đặt
Đầu ra

Đầu ra đảo

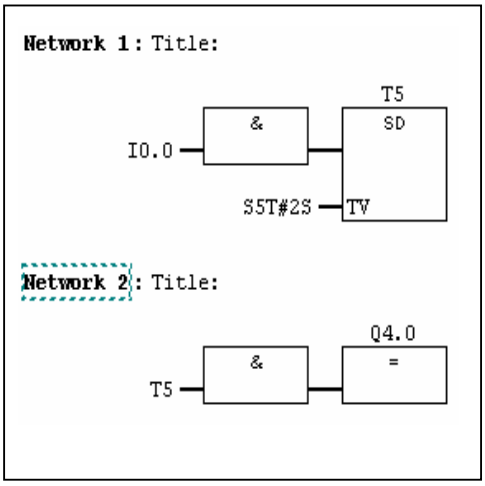
Hình 4-41: Giản đồ thời gian SD.

Khi có tín hiệu RESET (R) thời gian tính lập tức trở về "0" và tín hiệu đầu ra cũng giá trị là "0".

-Trường hợp không sử dụng các tín hiệu đầu vào SET(S), RESET (R), BI và BCD ta sử dụng khối Timer SE sau:

Tín hiệu đầu vào I0.0 chính là tín hiệu kích.
S5T#2s là thời gian đặt 2s

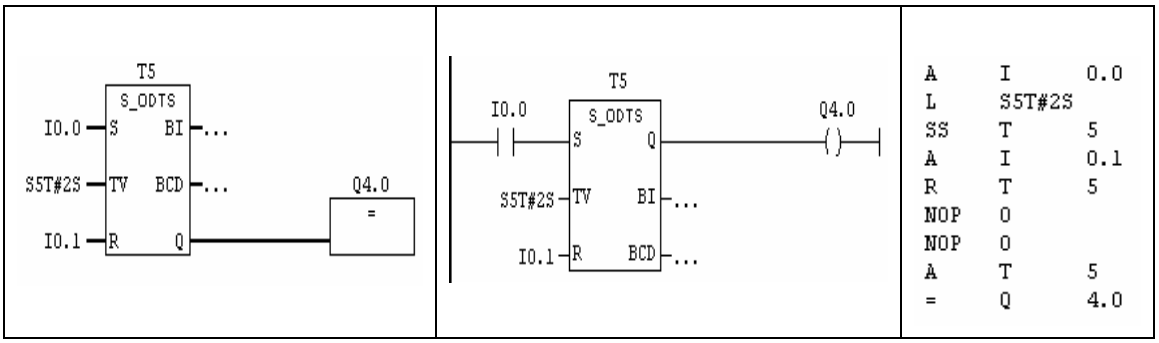
Tín hiệu ra của bộ thời gian tác động tới đầu ra Q4.0.



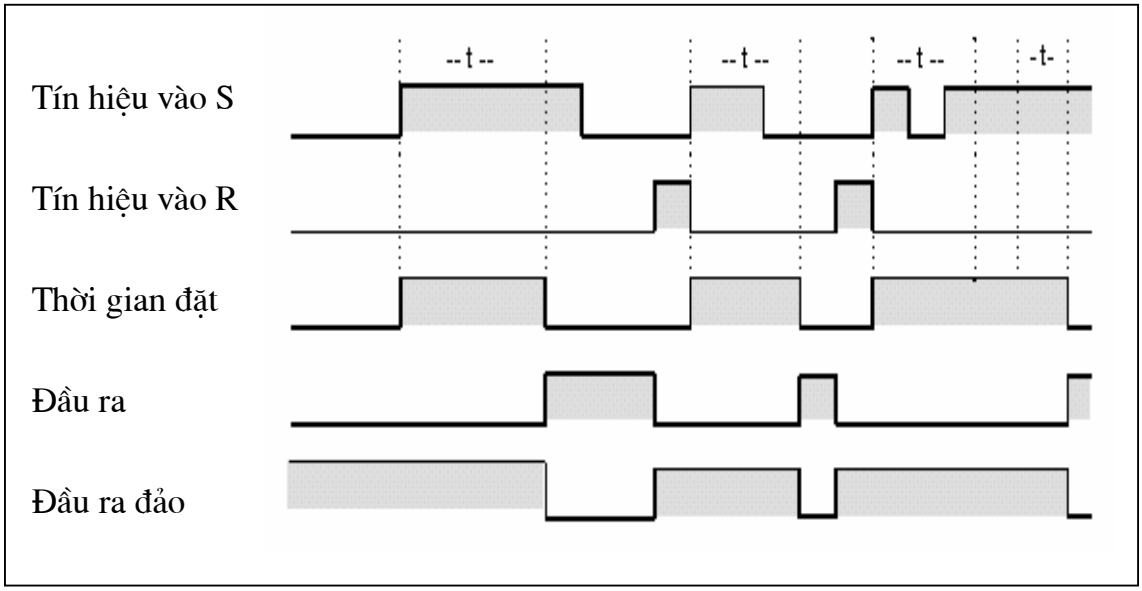
Hình 4-42: ví dụ sử dụng khối hàm SD.

4. *Bộ thời gian SS:*

FBD	LAD	STL



Hình 4-43: Khai báo bộ thời gian SS .



Hình 4-44: Giản đồ thời gian hàm SS.

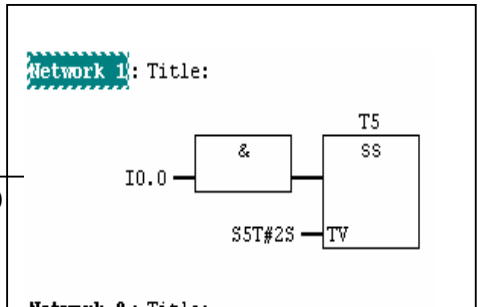
-Nguyên lý làm việc:

Tại thời điểm sườn lên của tín hiệu vào SET bộ thời gian được thiết lập và thời gian sẽ được tính. Kết thúc thời gian đặt tín hiệu đầu ra sẽ có giá trị 1 giá trị này vẫn duy trì ngay cả khi tín hiệu đầu vào kích S có giá trị là 0. Khi có tín hiệu RESET (R) thời gian tính lập tức trở về 0 và tín hiệu đầu ra cũng giá trị là "0".

-Trường hợp không sử dụng các tín hiệu đầu vào SET(S), RESET (R), BI và BCD ta sử dụng khối Timer SS sau:

Tín hiệu đầu vào I0.0 chính là tín hiệu kích.

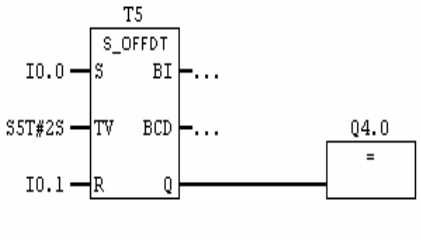
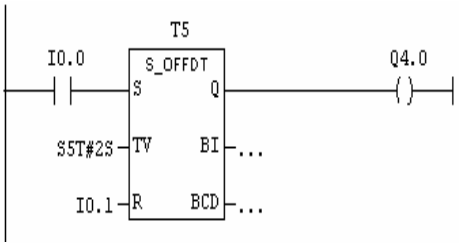
S5T#2s là thời gian đặt 2s



Tín hiệu ra của bộ thời gian tác
động tới đầu ra Q4.0

Hình 4-45: Ví dụ sử dụng khối hàm SS

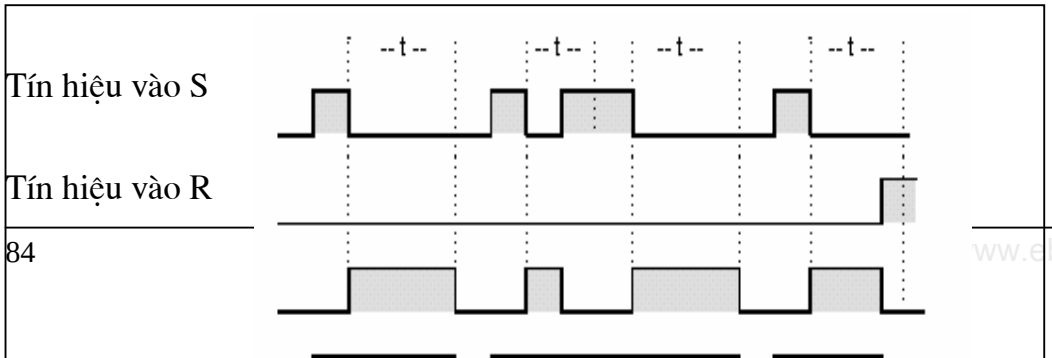
5. *Bộ thời gian SA:*

FBD	LAD	STL
		A I 0.0 L S5T#2S SF T 5 A I 0.1 R T 5 NOP 0 NOP 0 A T 5 = Q 4.0

Hình 4-46: Sơ đồ khối.

-Nguyên lý làm việc:

Tại thời điểm sườn lên của tín hiệu vào SET bộ thời gian đươc thiết lập. Tín hiệu đầu ra có giá trị là 1. Nhưng thời gian sẽ đươc tính ở thời điểm sườn xuống cuối cùng của tín hiệu đầu vao SET(S). Kết thúc thời gian đặt tín hiệu đầu ra sẽ trở về 0.



Thời gian đặt

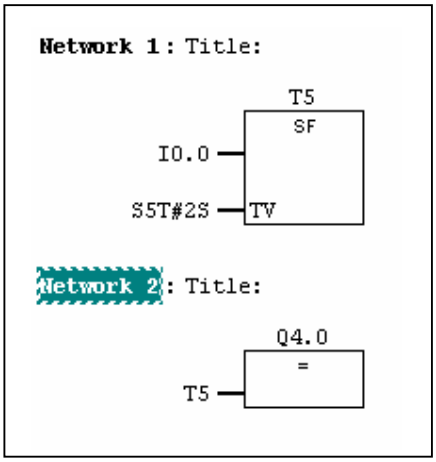
Đầu ra

Đầu ra đảo

Hình 4-47: Giản đồ thời gian.

Khi có tín hiệu RESET (R) thời gian tính lập tức trở về 0 và tín hiệu đầu ra cũng giá trị là "0".

-Trường hợp không sử dụng các tín hiệu đầu vào SET(S), RESET (R), BI và BCD ta sử dụng khối Timer SF sau:



Tín hiệu I0.0 là tín hiệu kích
Thời gian đặt S5T#2s là 2s

Hàm thời gian sẽ tác động tới đầu ra Q4.0

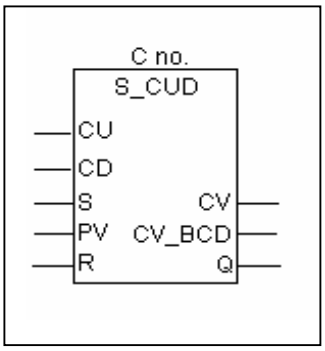
Hình 4-48: Sử dụng hàm SF.

4.6 Bộ đếm COUNTER:

4.6.1.Nguyên lý làm việc:

Counter thực hiện chức năng đếm tại các sườn lên của các xung đầu vào. S7-300 có tối đa là 256 bộ đếm phụ thuộc vào từng loại CPU, ký hiệu bởi Cx. Trong đó x là số nguyên trong khoảng từ 0 đến 255. Trong S7-300 có 3 loại bộ đếm thường sử dụng nhất đó là : Bộ đếm tiến lùi (CUD), bộ đếm tiến (CU)và bộ đếm lùi (CD).

Một bộ đếm tổng quát có thể được mô tả như sau: trong đó:



CU : BOOL là tín hiệu đếm tiến
CD : BOOL là tín hiệu đếm lùi
S : BOOL là tín hiệu đặt
PV : WORD là giá trị đặt trước
R : BOOL là tín hiệu xoá
CV : WORD Là giá trị đếm ở hệ đếm 16
CV_BCD: WORD là giá trị đếm ở hệ đếm BCD
Q : BOOL Là tín hiệu ra .

Hình 4-49: sơ đồ khối bộ đếm Counter

Quá trình làm việc của bộ đếm được mô tả như sau:
Số sườn xung đếm được, được ghi vào thanh ghi 2 Byte của bộ đếm, gọi là thanh ghi C-Word. Nội dung của thanh ghi C-Word được gọi là giá trị đếm tức thời của bộ đếm và ký hiệu bằng CV và CV_BCD. Bộ đếm báo trạng thái của C-Word ra ngoài C-bit qua chân Q của nó. Nếu CV <> 0 , C-bit có giá trị "1". Ngược lại khi CV = 0, C- bit nhận giá trị 0. CV luôn là giá trị không âm. Bộ đếm sẽ không đếm lùi khi CV = 0.

Đối với Counter, giá trị đặt trước PV chỉ được chuyển vào C-Word tại thời điểm xuất hiện sườn lên của tín hiệu đặt tới chân S.

Bộ đếm sẽ được xoá tức thời bằng tín hiệu xoá R (Reset). Khi bộ đếm được xóa cả C-Word và C- bit đều nhận giá trị 0.

4.6.2.Khai báo sử dụng:

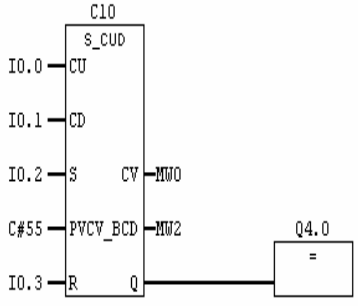
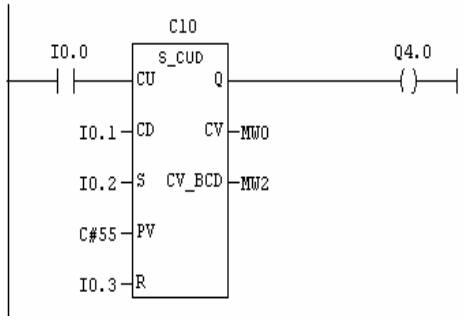
Việc khai báo sử dụng một Counter bao gồm các bước sau:

- Khai báo tín hiệu Enable nếu muốn sử dụng tín hiệu chủ động kích đếm (S): dạng dữ liệu BOOL
- Khai báo tín hiệu đầu vào đếm tiến CU : dạng dữ liệu BOOL
- Khai báo tín hiệu đầu vào đếm lùi CD : dạng dữ liệu BOOL
- Khai báo giá trị đặt trước PV: dạng dữ liệu WORD
- Khai báo tín hiệu xoá: dạng dữ liệu BOOL
- Khai báo tín hiệu ra CV nếu muốn lấy giá trị đếm tức thời ở hệ 16. dạng dữ liệu WORD
- Khai báo tín hiệu ra CV-BCD nếu muốn lấy giá trị đếm tức thời ở hệ BCD dạng dữ liệu WORD
- Khai báo đầu ra Q nếu muốn lấy tín hiệu tác động của bộ đếm. dạng dữ liệu BOOL

Trong đó cần chú ý các tín hiệu sau bắt buộc phải khai báo: Tên của bộ đếm cần sử dụng, tín hiệu kích đếm CU hoặc CD.

1. <u>Bộ đếm tiến lùi:</u>

-Sơ đồ khối :

FBD	LAD	STL
		A I 0.0 CU C 10 A I 0.1 CD C 10 A I 0.2 L C#55 S C 10 A I 0.3 R C 10 L C 10 T MW 0 LC C 10 T MW 2 A C 10 = Q 4.0

Hình 4-50: Sơ đồ khối bộ đếm tiến lùi.

-Nguyên lý hoạt động:

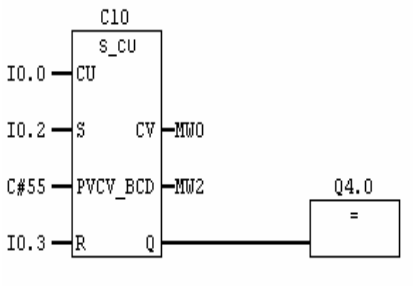
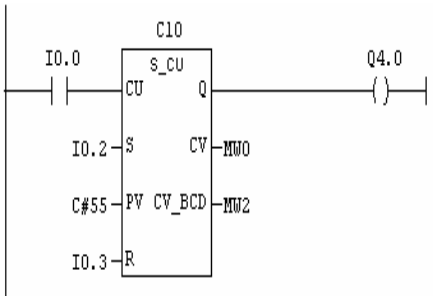
Khi tín hiệu I0.2 chuyển từ 0 lên 1bộ đếm được đặt giá trị là 55. Giá trị đầu ra Q4.0 =1 .

Bộ đếm sẽ thực hiên đếm tiến tại các sườn lên của tín hiệu tại chân CU khi tín hiệu I0.0 chuyển giá trị từ "0" lên "1"

Bộ đếm sẽ đếm lùi tại các sườn lên của tín hiệu tại chân I0.1 khi tín hiệu chuyển từ "0" lên "1"

Giá trị của bộ đếm sẽ trở về 0 khi có tín hiệu tai sườn lên của chân R (I0.3)

2. <u>Bộ đếm tiến : CU</u>

FBD	LAD	STL
		A I 0.0 CU C 10 BLD 101 A I 0.2 L C#55 S C 10 A I 0.3 R C 10 L C 10 T MW 0 LC C 10 T MW 2 A C 10 = Q 4.0

Hình 4-51: sơ đồ khối bộ đếm tiến.

-Nguyên lý hoạt động:

Khi tín hiệu I0.2 chuyển từ "0" lên "1" bộ đếm được đặt giá trị là 55. Giá trị đầu ra Q4.0 =1 .

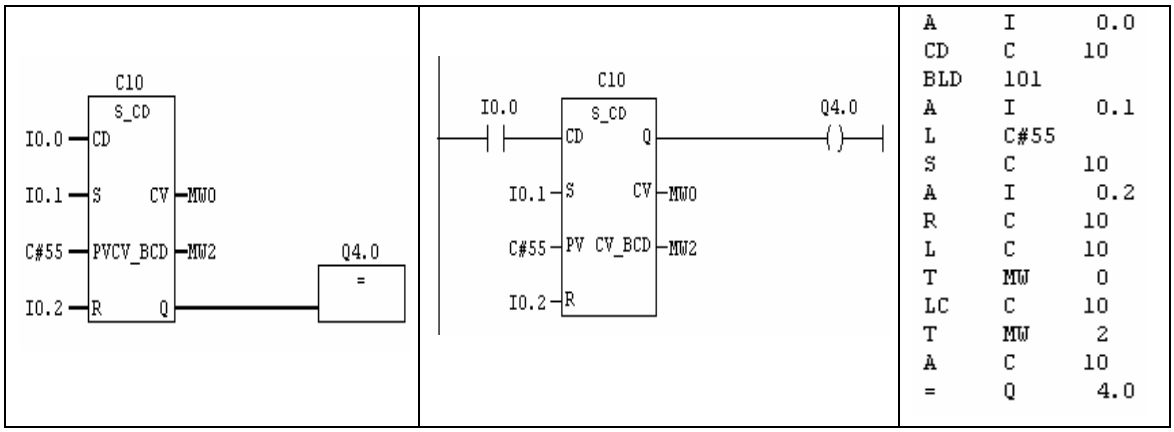
Bộ đếm sẽ thực hiên đếm tiến tại các sườn lên của tín hiệu tại chân CU khi tín hiệu I0.0 chuyển giá trị từ "0" lên "1"

Giá trị của bộ đếm sẽ trở về 0 khi có tín hiệu tai sườn lên của chân R (I0.3)

Bộ đếm sẽ chỉ đếm đến giá trị <= 999.

3. <u>Bộ đếm lùi: CD</u>

FBD	LAD	STL



Hình 4-52: Sơ đồ khối bộ đếm lùi.

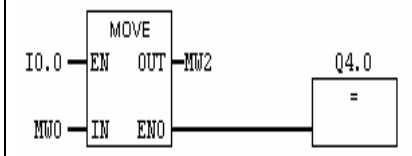
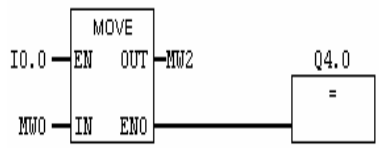
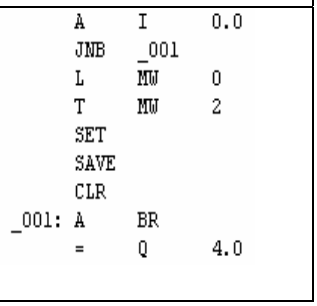
-Nguyên lý hoạt động:

Khi tín hiệu I0.2 chuyển từ "0" lên "1" bộ đếm được đặt giá trị là 55. Giá trị đầu ra Q4.0 =1 .

Bộ đếm sẽ thực hiên đếm lùi tại các sườn lên của tín hiệu tại chân CD khi tín hiệu I0.0 chuyển giá trị từ "0" lên"1"

Giá trị của bộ đếm sẽ trở về 0 khi có tín hiệu tai sườn lên của chân R (I0.3). Bộ đếm sẽ chỉ đếm đến giá trị >= 0.

4.7.Khối chuyển dữ liêu:

FBD	LAD	STL
		

Hình 4-53: Sơ đồ khối MOV

-Nguyên lý hoạt động:

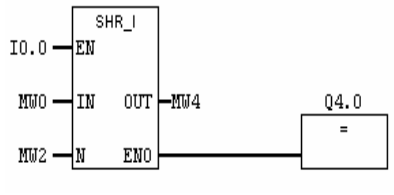
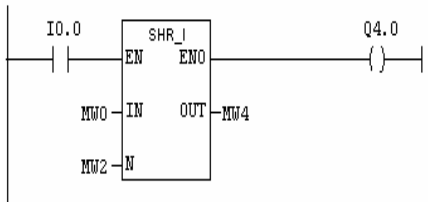
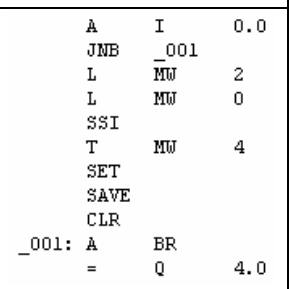
Khi có tín hệu kích I0.0 khối Copy được thiết lập , tín hiệu đầu ra ENO là Q4.0 =1. Đồng thời số liệu ở đầu vào IN là MW0 được Copy sang đầu ra OUT là MW2.

Khi tín hiệu kích I0.0 = 0 tín hiệu đầu ra Q4.0 = 0.

Trong trường hợp muốn thay đổi số liệu trong bộ nhớ (tức là thay đổi giá trị trong MW2) ta có thể không cần sử dụng tín hiệu kích I0.0.

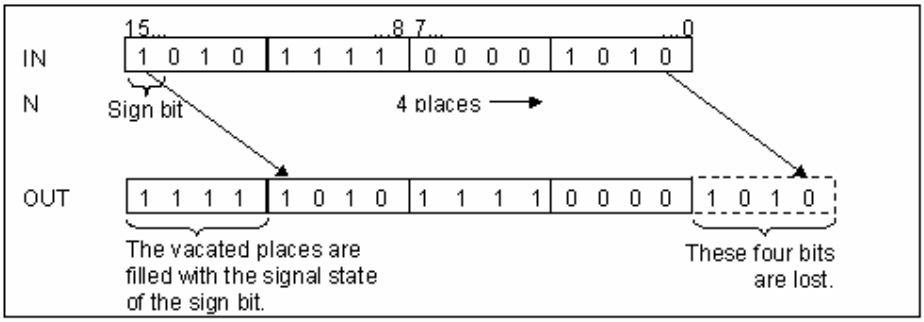
4.8.Các bộ ghi dịch và quay số liệu trên thanh ghi:

1. ***Dịch phải số nguyên 16 bits:***

FBD	LAD	STL
		


```
      A     I     0.0
      JNB   _001
      L     MW    2
      L     MW    0
      SSI
      T     MW    4
      SET
      SAVE
      CLR
_001: A     BR
      =     Q     4.0
```

Hình 4-54: Sơ đồ khối dịch phải.



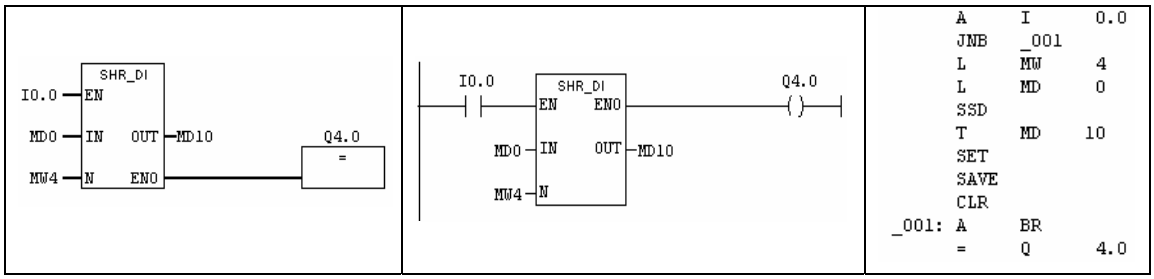
Hình 4-55: Nguyên lý hoạt động.

Khi tín hiệu kích I0.0 = 1 Khối sẽ thực hiện chức năng dịch chuyển sang phải số liệu trong thanh ghi. Đồng thời tín hiệu ra tại ENO là Q4.0 có giá trị là 1.
Số liệu đưa vào tại IN là MW0
Số bit sẽ dịch chuyển là MW2 (tại chân N).
Kết quả sau khi dịch được cất vào MW4.
Trên sơ đồ cho ta thấy kết quả của bộ dịch phải 4 bit.

2. Dịch phải số nguyên 32 bits:

-Sơ đồ khối:

FBD	LAD	STL



Hình 4-56: Khối dịch phải.

Khi tín hiệu kích I0.0 = 1. Khối sẽ thực hiện chức năng dich chuyển sang phải số liệu trong thanh ghi. Đồng thời tín hiệu ra tại ENO là Q4.0 có giá trị là 1.

Số liệu đưa vào tại IN là MD0

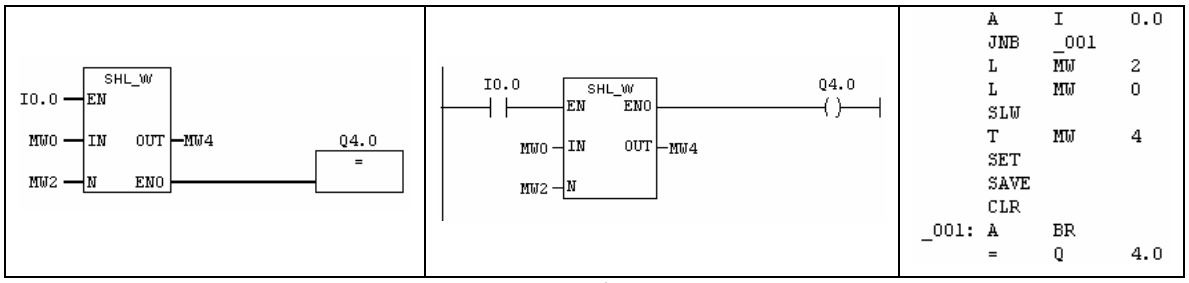
Số bit sẽ dịch chuyển là MW2 (tại chân N). Kết quả sau khi dịch được cất vào MW4.

Trên sơ đồ cho ta thấy kết quả của bộ dịch phải 4 bit.

3. ***Dich trái 16 bit:***

-Sơ đồ khối:

FBD	LAD	STL



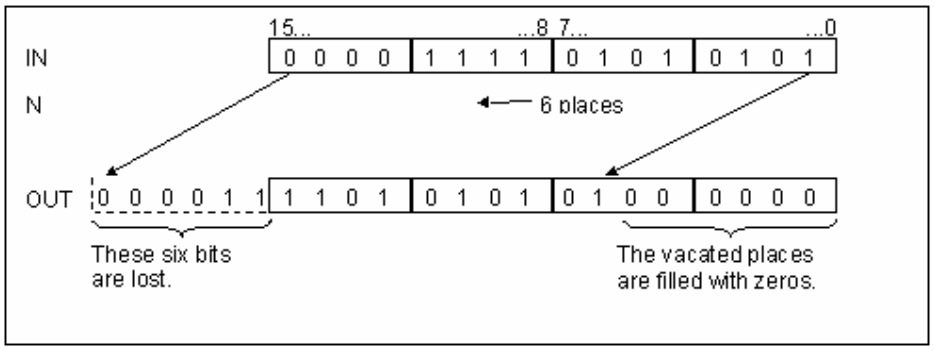
Hình 4-57: Khối dịch trái.

-Nguyên lý hoạt động:

Khi có tín hiệu kích I0.0 = 1 tín hiệu ra Q4.0 được thiết lập và có giá trị 1. Dữ liệu ở đầu vào MW0 được dịch sang trái với số bit được đặt tại chân N (MW2).

Kết quả sau khi dịch được ghi vào MW4.

-Giản đồ thời gian:



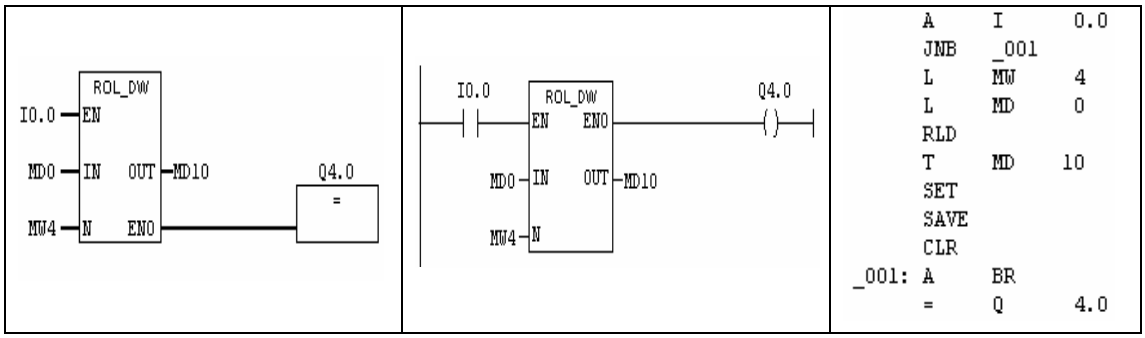
Hình 4-58: Giản đồ thời gian bộ dịch trái 6 vị trí.

Chú ý: Trong trường hợp cần dich trái một số 32 bits ta chỉ cần khai báo dữ liệu ở đầu vào IN dưới dạng MD ví dụ: MD0 và kết quả đầu ra cũng sẽ được lưu giữ ở MD Ví dụ: MD4

4. ***Quay trái số 32 bits:***

-Sơ đồ khối:

FBD	LAD	STL

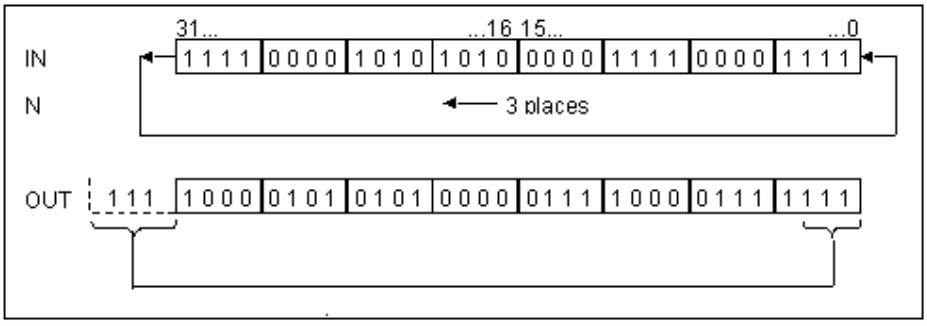


Hình 4-59: Sơ đồ khối quay trái.

-Nguyên lý hoạt động:

Khi có tín hiệu kích I0.0 = 1 tín hiệu ra Q4.0 được thiết lập và có giá trị 1. Dữ liệu ở đầu vào MD0 được quay sang tráI với số bit được đặt tại chân N (MW4).

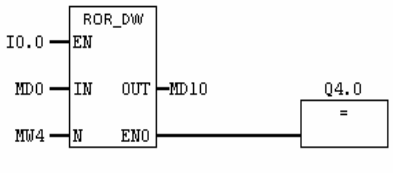
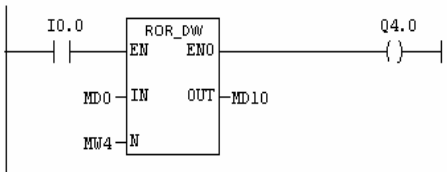
Kết quả sau khi dịch được ghi vào MD10.



Hình 4-60: Giản đồ thời gian.

5. Quay phải số 32 bits:

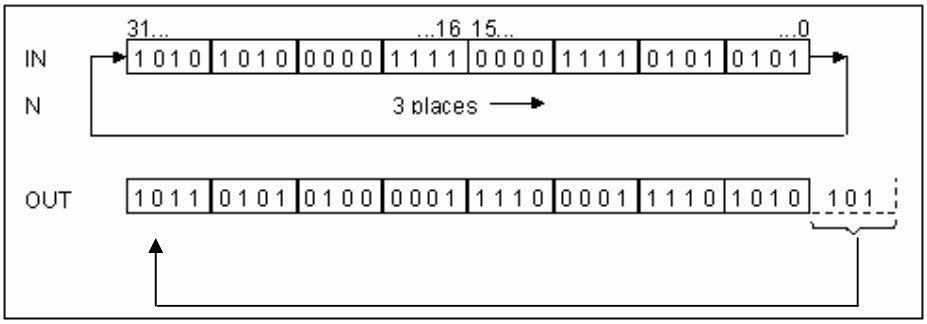
-Sơ đồ khối:

FBD	LAD	STL
		A I 0.0 JNB _001 L MW 4 L MD 0 RRD T MD 10 SET SAVE CLR _001: A BR = Q 4.0

Hình 4-61: sơ đồ khối bộ quay phải.

-Nguyên lý hoạt động:

Khi có tín hiệu kích I0.0 = 1 tín hiệu ra Q4.0 được thiết lập và có giá trị 1. Dữ liệu ở đầu vào MD0 được quay phải với số bit được đặt tại chân N (MW4). Kết quả sau khi dịch được ghi vào MD10.

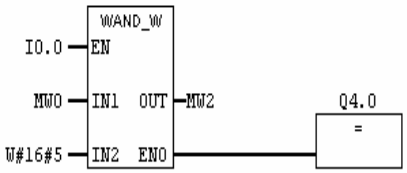
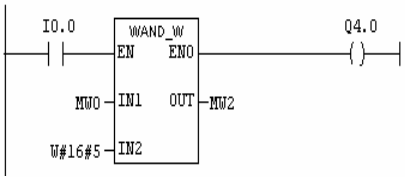


Hình 4-62: Giản đồ thời gian của bộ dịch phải 3 vị trí số 32 bits.

4.9.Các hàm Logic thực hiện trên thanh ghi :

1. Hàm AND hai số có độ dài là 16 bits.

-Sơ đồ khối:

FBD	LAD	STL
		A I 0.0 JNB _001 L MW 0 L W#16#5 AW T MW 2 SET SAVE CLR _001: A BR = Q 4.0

Hình 4-63: sử dụng khối AND 16 bits

-Nguyênlý hoạt động:

Hàm sẽ thực hiện chức năng nhân hai số nhị phân tai đầu vào IN1 và đầu vào IN2 kết quả được cất ở OUT (MW2) khi có tín hiệu kích tại chân EN (I0.0 =1).

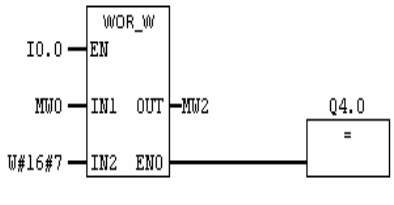
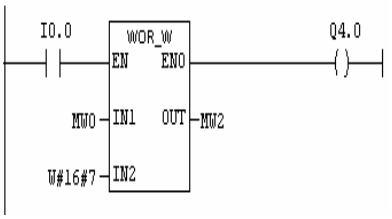
Tín hiệu ở đầu ra ENO (Q4.0 = 1) khi hàm thực hiện chức năng.

-Ví dụ:

IN1	=	0101010101010101	Số thứ nhất
IN2	=	0100000000001111	Số thứ hai
OUT	=	0100000000000101	Kết quả

2. ***Hàm OR hai số có độ dài là 16 bits:***

-Sơ đồ khối :

FBD	LAD	STL
		A I 0.0 JNB _001 L MW 0 L W#16#7 OW T MW 2 SET SAVE CLR _001: A BR = Q 4.0

Hình 4-64: Sử dụng khối OR 16 bits.

-Nguyên lý hoạt động:

Hàm sẽ thực hiện chức năng OR hai số nhị phân tai đầu vào IN1 và đầu vào IN2 kết quả được cất ở OUT (MW2) khi có tín hiệu kích tại chân EN (I0.0 = 1).

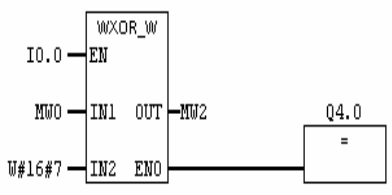
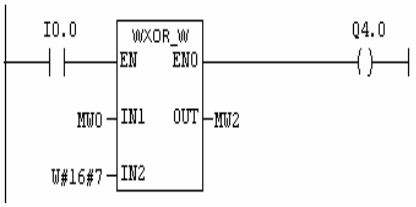
Tín hiệu ở đầu ra ENO (Q4.0 = 1) khi hàm thực hiện chức năng.

-Ví dụ:

IN1	=	0101010101010101	Số thứ nhất
IN2	=	0000000000001111	Số thứ Hai
OUT	=	0101010101011111	Kết quả

3.Hàm XOR hai số có đô dài 16 bits:

-Sơ đồ khối:

FBD	LAD	STL
		A I 0.0 JNB _001 L MW 0 L W#16#7 XOW T MW 2 SET SAVE CLR _001: A BR = Q 4.0

Hình 4-65: sơ đồ khối XOR 16 bits.

-Nguyênlý hoạt động:

Hàm sẽ thực hiện chức năng XOR hai số nhị phân tai đầu vào IN1 và đầu vào IN2 kết quả được cất ở OUT khi có tín hiệu kích tại chân EN.

Tín hiệu ở đầu ra ENO khi hàm thực hiện chức năng.

-Ví dụ:

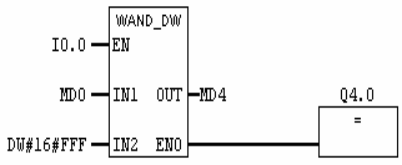
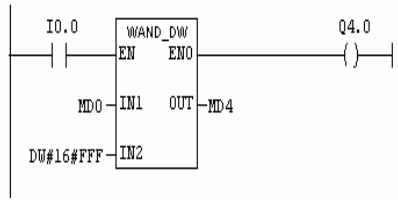
IN1 = 0101010101010101 Số thứ nhất

IN2 = 0000000000001111 Số thứ Hai

OUT = 0101010101011010 Kết quả

4.Hàm AND hai từ kép:

-Sơ đồ khối:

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 0 L DW#16#FFF AD T MD 4 SET SAVE CLR _001: A BR = Q 4.0

Hình 4-66: Sử dụng khối AND hai từ kép .

EN(I0.0): BOOL - tín hiệu kích
IN1: DWORD - Vào 1
IN2: DWORD - vào2
OUT: DWORD - Ra
ENO: BOOL - Tín hiệu ra của khối.

-Nguyên lý hoạt động:

Hàm sẽ thực hiện chức năng AND hai số nhị phân tại đầu vào IN1 và đầu vào IN2 kết quả được cất ở OUT khi có tín hiệu kích tại chân EN.

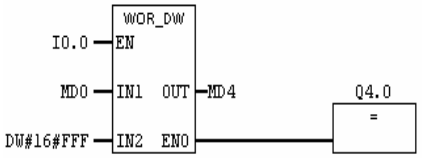
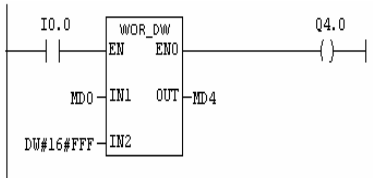
Tín hiệu ở đầu ra ENO khi hàm thực hiện chức năng.

-Ví dụ:

IN1	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
OUT	=	0000000000000000	0000010101010101

5.Hàm OR hai từ kép:

-Sơ đồ khối:

FBD	LAD	STL
		A I 0.0 JNB _001 L MD 0 L DW#16#FFF OD T MD 4 SET SAVE CLR _001: A BR = Q 4.0

Hình 4-67: Sử dụng khối OR hai từ kép.

EN(I0.0): BOOL - tín hiệu kích
IN1: DWORD - Vào 1
IN2: DWORD - vào2
OUT: DWORD - Ra
ENO: BOOL - Tín hiệu ra của khối.

-Nguyên lý hoạt động:

Hàm sẽ thực hiện chức năng OR hai số có độ dài 2 từ tại đầu vào IN1 và đầu vào IN2 kết quả được cất ở OUT khi có tín hiệu kích tại chân EN.

Tín hiệu ở đầu ra ENO khi hàm thực hiện chức năng.

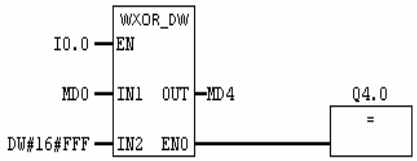
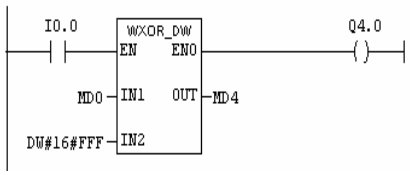
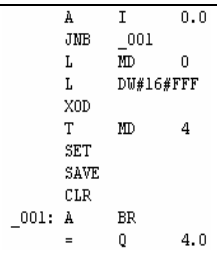
-Ví dụ:

IN1	=	0101010101010101	0101010101010101

IN2	=	0000000000000000	0000111111111111
OUT	=	0101010101010101	0101111111111111

6.Hàm XOR hai từ kép :

-sơ đồ nguyên lý:

FBD	LAD	STL
		

Hình 4-68: Sử dụng khối XOR hai từ kép.

EN(I0.0): BOOL - tín hiệu kích
IN1: DWORD - Vào 1
IN2: DWORD - vào2
OUT: DWORD - Ra
ENO: BOOL - Tín hiệu ra của khối.

-Nguyên lý hoạt động:

Hàm sẽ thực hiện chức năng XOR hai số có độ dài 2 từ tại đầu vào IN1 và đầu vào IN2 kết quả được cất ở OUT khi có tín hiệu kích tại chân EN.
Tín hiệu ở đầu ra ENO khi hàm thực hiện chức năng.

-Ví dụ:

IN1	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
OUT	=	0101010101010101	0101101010101010

Tài liệu tham khảo

1.Tự động hoá với Simatic S7-200. Nhà xuất bản nông nghiệp,1997- Doãn Minh Phước, Phan Xuân Minh.

2.S5-95U và phần mềm Step5. Giáo trình giảng dạy của trung tâm đào tạo Simens tự động hoá trường ĐHBK Hà nội, 1997- Doãn Minh Phước, Phan Xuân Minh.

3.SPS-Grundkurs, Volgel Buchverlag- Juergen Kaftan.

4.Speicherprogrammierte Steuerungen Aufgaben mit Loesungen, Europa-Fachbuchreihe.

5.Tự động hoá với Simatic S7-300. Nhà xuất bản khoa học và kỹ thuật, 2000-Doãn Minh Phước, Phan Xuân Minh, Vũ Văn Hà .

