BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA CƠ KHÍ CHẾ TẠO MÁY – BỘ MÔN CƠ ĐIỆN TỬ

GIÁO TRÌNH PLC S7-300
LÝ THUYẾT VÀ ỨNG DỤNG

(DÀNH CHO SINH VIÊN NGÀNH CƠ ĐIỆN TỬ – TỰ ĐỘNG HÓA)

BIÊN SOẠN:
ThS. NGUYỄN XUÂN QUANG

TP.HCM, THÁNG 12 NĂM 2006

MỤC LỤC

TÀI LIỆU THAM KHẢO

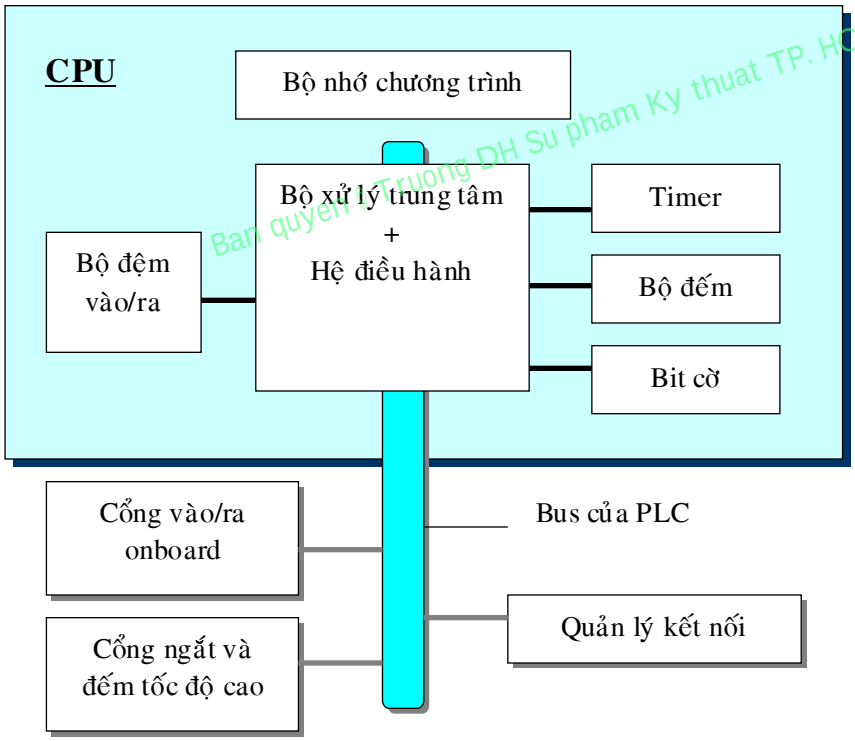
1. Nguyễn Hồng Sơn. Kỹ Thuật Truyền Số Liệu- Nhà Xuất Bản Lao Động Và Xã Hội.
2. Phan Xuân Minh & Nguyễn Doãn Phước, 1997 : Lý Thuyết Điều Khiển Mờ – Nhà Xuất Bản Khoa Học Và Kỹ Thuật.
3. Nguyễn Doãn Phước, Phan Xuân Vũ, Vũ Vân Hoà, 2000. Tự Động Hoá với SIMATIC S7-300 – Nhà Xuất Bản Khoa Học Và Kỹ Thuật
4. SIMATIC S7-300 Điều Khiển Hệ Thống (Systemhandling), 2000. Đại Học Sư Phạm Kỹ Thuật. Trung Tâm Việt Đức. Bộ Môn Điện –Điện Tử.
5. Hãng Siemens, SIMATIC's Manual.
6. http://wwww.ad.Siemens.de/

CHƯƠNG 1: GIỚI THIỆU

1.1 Giới thiệu PLC S7-300

1.1.1 Thiết bị điều khiển logic khả trình.

Thiết bị điều khiển logic khả trình (Programmable Logic Controller) là loại thiết bị thực hiện linh hoạt các thuật toán điều khiển số thông qua một ngôn ngữ lập trình, thay vì phải thực hiện thuật toán đó bằng mạch số. Như vậy, PLC là một bộ điều khiển gọn, nhẹ và dễ trao đổi thông tin với môi trường bên ngoài (với các PLC khác hoặc máy tính). Toàn bộ chương trình điều khiển được lưu trữ trong bộ nhớ của PLC dưới dạng các khối chương trình và được thực hiện theo chu kỳ của vòng quét (scan).



Hình1.1. Cấu trúc bên trong của một PLC

Để thực hiện được một chương trình điều khiển, tất nhiên PLC phải có tính năng như một máy tính, nghĩa là phải có một bộ vi xử lý (CPU), một hệ điều hành, bộ nhớ để lưu chương trình điều khiển, dữ liệu và tất nhiên phải có các cổng vào/ra để giao tiếp được với đối tượng điều khiển và để trao đổi

thông tin với môi trường xung quanh. Bên cạnh đó nhằm phục bài toán điều khiển số, PLC còn phải có thêm một số khối chức năng đặc biệt khác như bộ đếm (Counter), bộ định thời (Timer) ... và những khối hàm chuyên dùng.

Ưu điểm của bộ điều khiển lập trình được so với điều khiển nối dây:

- ✓ Tính năng mở rộng: khả năng mở rộng xử lý bằng cách thay đổi chương trình lập trình một cách dễ dàng.
- ✓ Độ tin cậy cao.
- ✓ Cách kết nối các thiết bị điều khiển đơn giản.
- ✓ Hình dáng PLC gọn nhẹ.
- ✓ Giá thành và chi phí lắp đặt thấp.
- ✓ Phù hợp với môi trường công nghiệp.

Các ứng dụng của PLC trong sản xuất và trong dân dụng:

- ✓ Điều khiển các Robot trong công nghiệp.
- ✓ Hệ thống xử lý nước sạch.
- ✓ Công nghệ thực phẩm.
- ✓ Công nghệ chế biến dầu mỏ.
- ✓ Công nghệ sản xuất vi mạch.
- ✓ Điều khiển các máy công cụ.
- ✓ Điều khiển và giám sát dây chuyền sản xuất.
- ✓ Điều khiển hệ thống đèn giao thông.
- ✓ ...

1.1.2 Các module của PLC S7-300.

Để tăng tính mềm dẻo trong các 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 module. Số các module được sử dụng nhiều hay ít tuỳ thuộc vào từng bài toán, song tối thiểu bao giờ cũng có module chính (module CPU, module nguồn). Các module còn lại là những module truyền nhận tín hiệu với các đối tượng điều khiển, chúng được gọi là các module mở rộng. Tất cả các module đều được gá trên một thanh Rack.

Module CPU:

Đây là loại module có chứa bộ vi xử lý, hệ điều hành, bộ nhớ, các bộ thời gian, bộ đếm, cổng truyền thông,... và có thể có các cổng vào/ra số. Các cổng vào/ra tích hợp trên CPU gọi là cổng vào ra onboard.

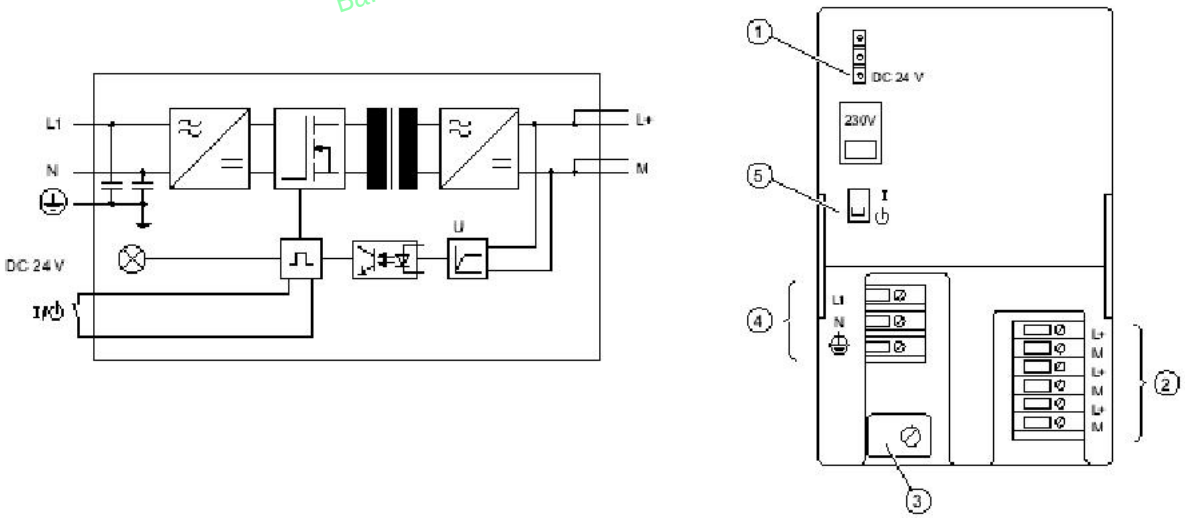
Trong họ PLC S7-300, các module CPU có nhiều loại và được đặt tên theo bộ vi xử lý bên trong như : CPU 312, CPU 314, CPU 316,.... Những module cùng một bộ vi xử lý nhưng khác nhau số cổng vào/ra onboard cũng như các khối hàm đặc biệt thì được phân biệt bằng cụm chữ cái IFM (*Intergrated Function Module*). Ví dụ như CPU 312IFM, CPU 314IFM,....

Ngoài ra, còn có loại module CPU có hai cổng truyền thông, trong đó cổng thứ hai dùng để nối mạng phân tán như mạng PROFIBUS (PROcess Field BUS). Loại này đi kèm với cụm từ DP (*Distributed Port*) trong tên gọi. Ví dụ module CPU315-DP.

Module mở rộng:

Các module mở rộng được thành 5 loại :

1) PS (*Power Supply*): module nguồn là module tạo ra nguồn có điện áp 24Vdc cấp nguồn cho các module khác. Có 3 loại: 2A, 5A và 10A.



Hình1.2. Sơ đồ khối và sơ đồ đấu dây của module nguồn PS307;2A (6ES7307-1BA00-0AB)

① Đèn chỉ thị nguồn 24Vdc

② Đômino nối dây ngõ ra điện áp 24Vdc

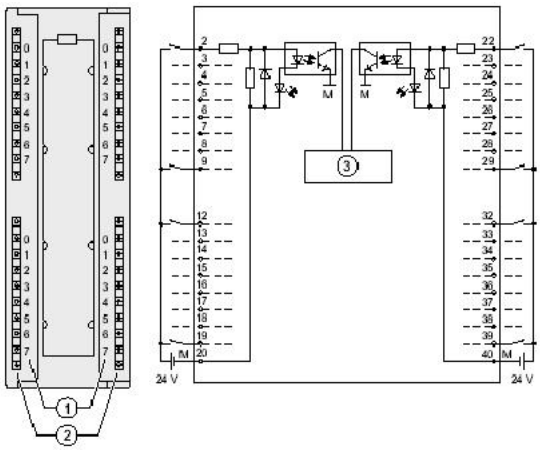
③ Cầu chì bảo vệ quá dòng

④ Đômino nối dây với điện áp 220Vac

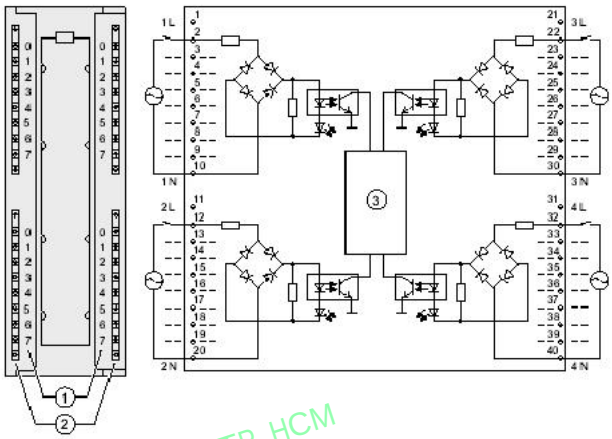
⑤ ON/OFF Switch 24Vdc

2) SM (*Signal Module*): Module mở rộng vào/ra, bao gồm :

a) DI (*Digital Input*): module mở rộng cổng vào số. Số các cổng vào số mở rộng có thể là 8, 16 hoặc 32 tuỳ thuộc vào từng loại module.



Hình 1.3. Sơ đồ đấu dây của module SM221; DI 32 x DC 24V (6ES7321-1BL00-0AA0)



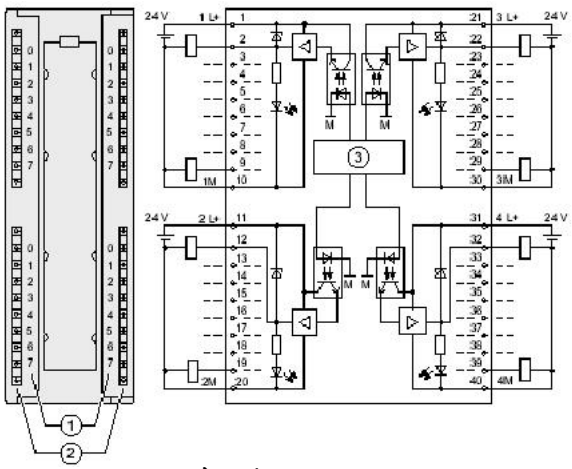
Hình 1.4. Sơ đồ đấu dây của module SM221; DI 32 x AC 120V (6ES7321-1EL00-0AA0)

① Số thứ tự các ngõ vào số trong module

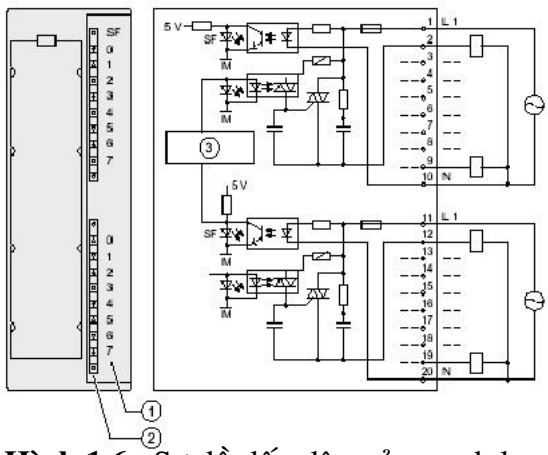
② Đèn chỉ thị mức logic

③ Bus bên trong của module

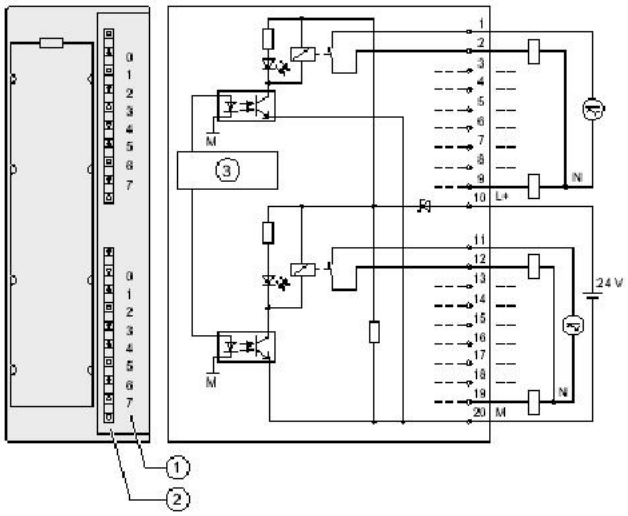
b) DO (*Digital Output*): module mở rộng cổng ra số. Số các cổng vào số mở rộng có thể là 8, 16 hoặc 32 tuỳ thuộc vào từng loại module.



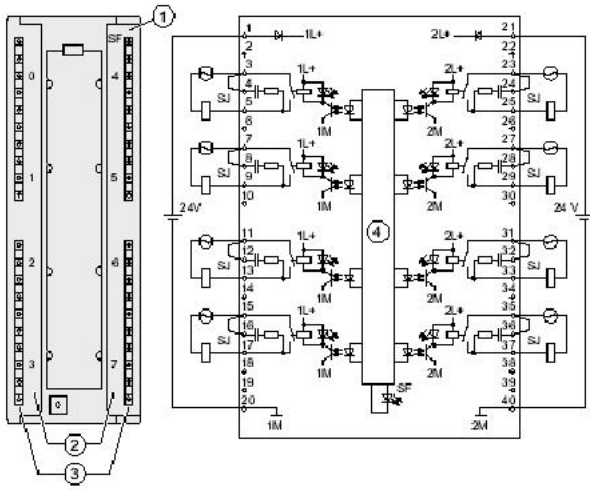
Hình 1.5.Sơ đồ đấu dây của module SM 322; DO 32 x 24 VDC/ 0.5 A; (6ES7322-1BL00-0AA0)



Hình 1.6. Sơ đồ đấu dây của module SM 322; DO 16 x AC 120/230 V/1 A; (6ES7322-1FH00-0AA0)



Hình 1.7. Sơ đồ đấu dây của module SM 322; DO 16 x Rel. AC 120/230 V; (6ES7322-1HH01-0AA0)



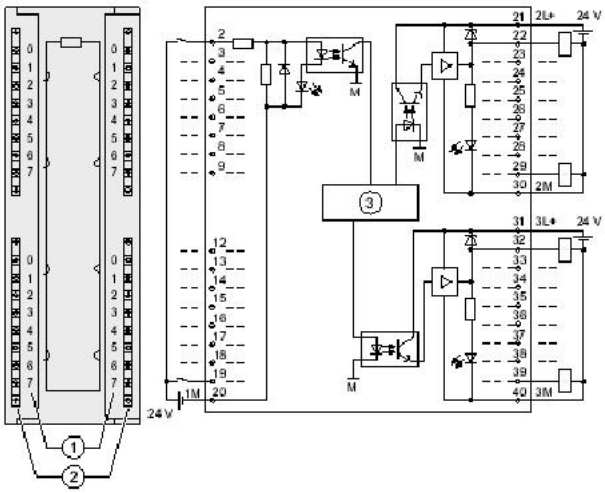
Hình 1.8. Sơ đồ đấu dây của module SM 322; DO 8 x Rel. AC 230V/5A; (6ES7322-5HF00-0AB0)

① Số thứ tự các ngõ vào số trong module

② Đèn chỉ thị mức logic

③ Bus bên trong của module

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



① Số thứ tự các ngõ vào số trong module

② Đèn chỉ thị mức logic

③ Bus bên trong của module

Hình 1.9. Sơ đồ đấu dây của module SM 323; DI 16/DO 16 x DC 24 V/0.5 A; (6ES7323-1BL00-0AA0)

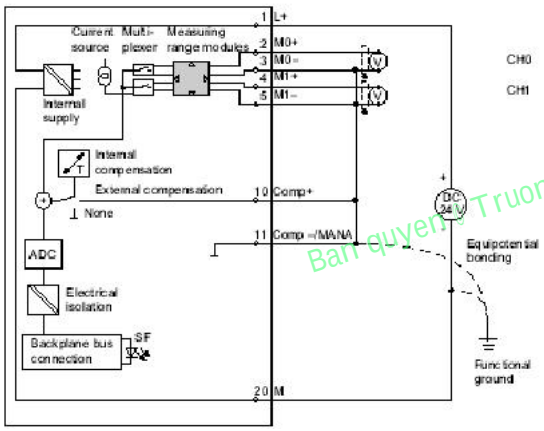
d) AI (*Analog Input*): module mở rộng cổng vào tương tự. Bản chất chúng là những bộ chuyển đổi tương tự sang số (ADC). Số các cổng vào tương tự có thể là 2, 4 hoặc 8 tuỳ từng loại module, số bit có thể là 8,10,12,14,16 tùy theo từng loại module.

Ví dụ: Module SM 331; AI 2 x 12 bit; (6ES7331-7KB02-0AB0)

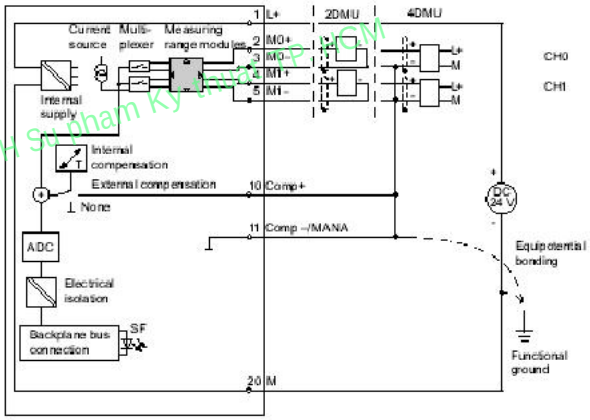
Các dạng tín hiệu đọc được

- Điện áp
- Dòng điện
- Điện trở
- Nhiệt độ

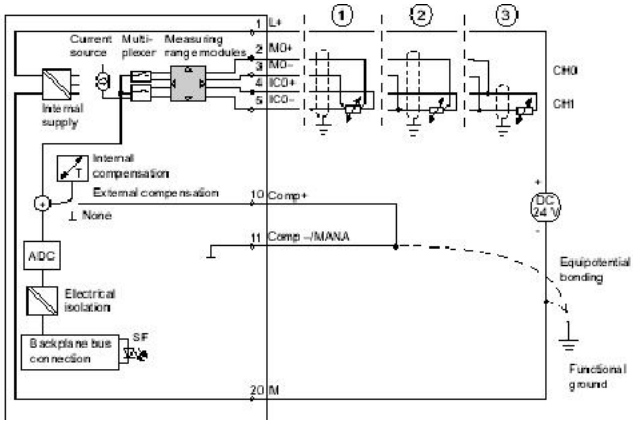
Độ phân giải 12 bit



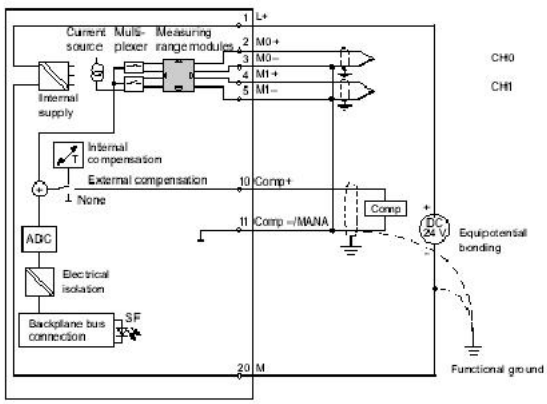
Hình 1.10. Sơ đồ đấu dây của module Khi tín hiệu vào là điện áp



Hình 1.11. Sơ đồ đấu dây của module Khi tín hiệu vào là đòng điện

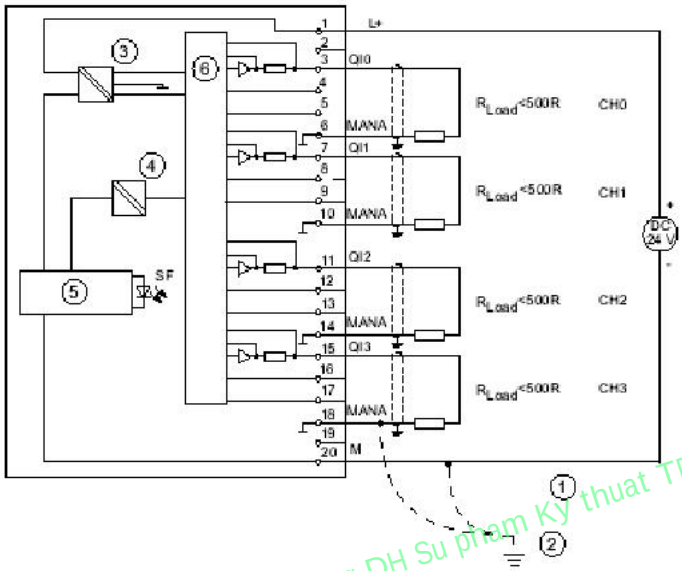


Hình 1.12. Sơ đồ đấu dây của module Khi tín hiệu vào là điện trở



Hình 1.13. Sơ đồ đấu dây của module Khi tín hiệu vào là Thermocouple

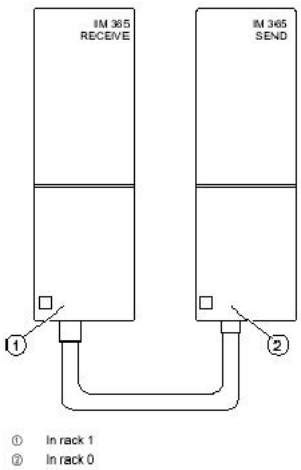
e) AO (*Analog Output*): module mở rộng cổng ra tương tự. Chúng là những bộ chuyển đổi từ số sang tương tự (DAC). Số cổng ra tương tự có thể là 2 hoặc 4 tuỳ từng loại module.



Hình 1.14. Sơ đồ đấu dây của module SM 332; AO 4 x 12 Bit; (6ES7332-5HD01-0AB0)

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

3) IM (*Interface Module*): Module kết nối.



Hình 1.15. Sơ đồ đấu dây của module IM 365; (6ES7365-0BA01-0AA0)

Đây là loại module dùng để kết nối từng nhóm các module mở rộng thành một khối và được quản lý bởi một module CPU. Thông thường các module mở rộng được gá liền nhau trên một thanh rack. Mỗi thanh rack chỉ có thể gá được nhiều nhất 8 module mở rộng (không kể module CPU và module nguồn). Một module CPU có thể làm việc nhiều nhất với 4 thanh rack và các rack này phải được nối với nhau bằng module IM.

4) FM (*Function Module*): Module có chức năng điều khiển riêng như: module điều khiển động cơ bước, module điều kiển động cơ servo, module PID,...

5) CP (*Communication Processor*): Module truyền trông giữa PLC với PLC hay giữa PLC với PC.

1.2 Tổ chức bộ nhớ CPU.

➢ Vùng nhớ chức các thanh ghi: ACCU1, ACCU2, AR1, AR2,...

➢ Load memory: là vùng nhớ chứa chương trình ứng dụng (do người sử dụng viết) bao gồm tất cả các khối chương trình ứng dụng OB, FC, FB, các khối chương trình trong thư viện hệ thống được sử dụng (SFC, SFB) và các khối dữ liệu DB. Vùng nhớ này được tạo bởi một phần bộ nhớ RAM của CPU và EEPROM (nếu có EEPROM). Khi thực hiện động tác xoá bộ nhớ (MRES) toàn bộ các khối chương trình và khối dữ liệu nằm trong RAM sẽ bị xoá. Cũng như vậy, khi chương trình hay khối dữ liệu được đổ (down load) từ thiết bị lập trình (PG, máy tính) vào module CPU, chúng sẽ được ghi lên phần RAM của vùng nhớ Load memory.

➢ Work memory: là vùng nhớ chứa các khối DB đang được mở, khối chương trình (OB, FC, FB, SFC, hoặc SFB) đang được CPU thực hiện và phần bộ nhớ cấp phát cho những tham số hình thức để các khối chương trình này trao đổi tham trị với hệ điều hành và với các khối chương trình khác (local block). Tại một thời điểm nhất định vùng Work memory chỉ chứa một khối chương trình. Sau khi khối chương trình đó được thực hiện xong thì hệ điều hành sẽ xoá khỏi Work memory và nạp vào đó khối chương trình kế tiếp đến lượt được thực hiện.

➢ System memory: là vùng nhớ chứa các bộ đệm vào/ra số (Q, I), các biến cờ (M), thanh ghi C-Word, PV, T-bit của timer, thanh ghi C-Word, PV, C-bit của Couter. Việc truy cập, sửa lỗi dữ liệu những ô nhớ này được phân chia hoặc bởi hệ điều hành của CPU hoặc do chương trình ứng dụng.

Có thể thấy rằng trong các vùng nhớ được trình bày ở trên không có vùng nhớ nào được dùng làm bộ đệm cho các cổng vào/ra tương tự. Nói cách khác

các cổng vào/ra tương tự không có bộ đệm và như vậy mỗi lệnh truy nhập module tương tự (đọc hoặc gửi giá trị) đều có tác dụng trực tiếp tới các cổng vật lý của module.

Bảng1.1. vùng địa chỉ và tầm địa chỉ

Tên gọi	Kích thước truy cập	Kích thước tối đa (tuỳ thuộc vào CPU)
Process input image (I) Bộ đệm vào số	I IB IW ID	0.0 ÷ 127.7 0 ÷ 127 0 ÷126 0 ÷ 124
Process output image (Q) Bộ đệm ra số	Q QB QW ID	0.0 ÷ 127.7 0 ÷ 127 0 ÷ 126 0 ÷ 124
Bit memory (M) Vùng nhớ cờ	M MB MW MD	0.0 ÷ 255.7 0 ÷ 255 0 ÷ 254 0 ÷ 252
Timer (T)	T0 ÷ T255	
Counter (C)	C0 ÷ C255	
Data block (DB) Khối dữ liệu share	DBX DBB DBW DBD	0.0 ÷ 65535.7 0 ÷ 65535 0 ÷ 65534 0 ÷ 65532
Data block (DI)	DIX	0.0 ÷ 65535.7

Khối dữ liệu instance	DIB DIW DID	0 ÷ 65535 0 ÷ 65534 0 ÷ 65532
Local block (L) Miền nhớ địa phương cho các tham số hình thức	L LB LW LD	0.0 ÷ 65535.7 0 ÷ 65535 0 ÷ 65534 0 ÷ 65532
Peripheral input (PI)	PIB PIW PID	0 ÷ 65535 0 ÷ 65534 0 ÷ 65532
Peripheral output (PQ)	PQB PQW PQD	0 ÷ 65535 0 ÷ 65534 0 ÷ 65532

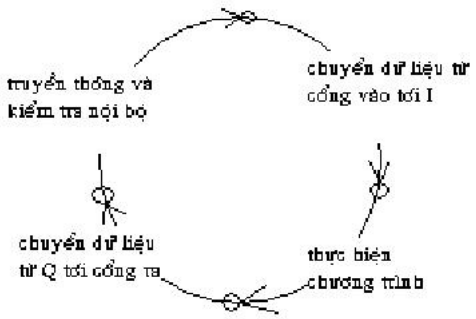
Trừ phần bộ nhớ EEPROM thuộc vùng Load memory và một phần RAM tự nuôi đặc biệt (non-volatile) dùng để lưu giữ tham số cấu hình trạm PLC như địa chỉ trạm (MPI address), tên các module mở rộng, tất cả các phần bộ nhớ còn lại ở chế độ mặc định không có khả năng tự nhớ (non-retentive). Khi mất nguồn nuôi hoặc khi thực hiện công việc xoá bộ nhớ (MRES), toàn bộ nội dung của phần bộ nhớ non-retentive sẽ bị mất.

1.3 Vòng quét chương trình của PLC.

PLC thực hiện chương trình theo chu trình lặp. Mỗi vòng lặp được gọi là vòng quét (scan). Mỗi vòng quét được bắt đầu bằng giai đoạn chuyển dữ liệu từ các cổng vào số tới vùng bộ đệm ảo I, tiếp theo là giai thực hiện chương trình. Trong từng vòng quét, chương trình được thực hiện từ lệnh đầu tiên đến lệnh kết thúc của khối OB1 (Block end). 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 truyền thông nội bộ và kiểm tra lỗi.

Thời gian cần thiết để PLC thực hiện được một vòng quét gọi là thời gian vòng quét (Scan time). Thời gian vòng quét không cố định, tức là không phải vòng quét nào cũng được thực hiện lâu, có vòng quét được thực hiện nhanh tuỳ

thuộc vào số lệnh trong chương trình được thực hiện, vào khối dữ liệu được truyền thông... trong vòng quét đó.



Hình 1.16. Vòng quét CPU

Như vậy giữa việc đọc dữ liệu từ đối tượng để xử lý, tính toán và việc gửi tín hiệu điều khiển tới đối tượng có một khoảng thời gian trễ đúng bằng thời gian vòng quét. Nói cách khác, thời gian vòng quét quyết định tính thời gian thực của chương trình điều khiển trong PLC. Thời gian 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 như 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 được thực hiện tại mọi điểm trong vòng quét chứ không bị gò ép là phải ở trong giai đoạn thực hiện chương trình. Chẳng hạn nếu 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 khối chương trình tương ứng với khối tín hiệu báo ngắt đó. Với hình thức xử lý tín hiệu 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.

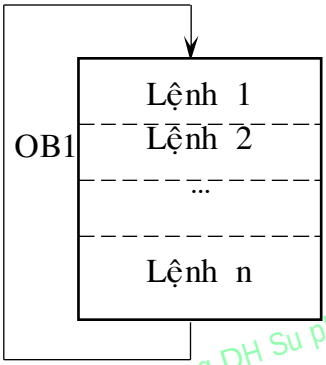
Tại thời điểm thực hiện lệnh vào/ra, thông thường lệnh không làm việc trực tiếp với cổng vào/ra mà chỉ thông qua bộ đệm ảo của cổng trong vùng nhớ tham số. Việc truyền thông giữa bộ đệm ảo với ngoại vi trong các giai đoạn 1 và 3 do hệ điều hành CPU quản lý. Ở một số module CPU, khi gặp lệnh vào/ra ngay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

1.4. Cấu trúc chương trình.

Chương trình cho S7-300 được lưu trong bộ nhớ của PLC ở vùng dành riêng cho chương trình. Ta có thể được lập trình với hai dạng cấu trúc khác nhau:

1.4.1. Lập trình tuyến tính

Toàn bộ chương trình điều khiển nằm trong một khối trong bộ nhớ. Loại lập trình cấu trúc chỉ thích hợp cho những bài toán tự động nhỏ, không phức tạp.



Hình 1.17. Vòng quét PLC

Khối được chọn là khối OB1, là khối mà PLC luôn luôn quét và thực hiện các lệnh trong nó thường xuyên, từ lệnh đầu tiên đến lệnh cuối cùng và quay lại lệnh đầu tiên:

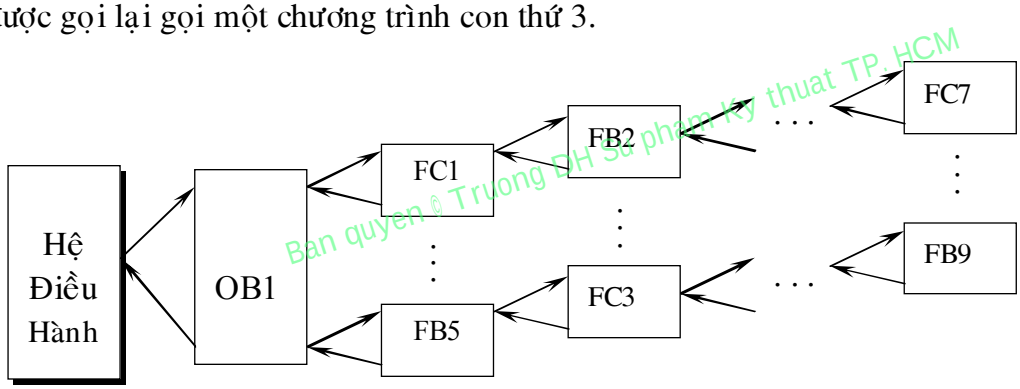
1.4.2 Lập trình cấu trúc

Chương trình được chia thành những phần nhỏ với từng nhiệm vụ riêng biệt và các phần này nằm trong những khối chương trình khác nhau. Loại lập trình có cấu trúc phù hợp với những bài toán điều khiển nhiều nhiệm vụ và phức tạp. Các khối cơ bản :

- Khối OB (Organization Block): khối tổ chức và quản lý chương trình điều khiển. Có nhiều loại khối OB với những chức năng khác nhau. Chúng được phân biệt với nhau bằng số nguyên theo sau nhóm ký tự OB, ví dụ như OB1, OB35, OB80...
- Khối FC (Program Block): khối chương trình với những chức năng riêng biệt giống như một chương trình con hay một hàm (chương trình co có biến hình thức). Một chương trình ứng dụng có thể có nhiều khối FC và các khối FC này được phân biệt với nhau bằng số nguyên theo sau nhóm ký tự FC, chẳng hạn như FC1, FC2, ...

- Khối FB (Function Block): là khối FC đặt biệt có khả năng trao đổi một lượng dữ liệu lớn với các khối chương trình khác. Các dữ liệu này phải được tổ chức thành khối dữ liệu riêng được gọi là Data Block. Một chương trình ứng dụng có thể có nhiều khối FB và các khối FB này được phân biệt với nhau bằng số nguyên theo sau nhóm ký tự FB. Chẳng hạn như FB1, FB2, ...
- Khối DB (Data Block): khối dữ liệu cần thiết để thực hiện chương trình. Các tham số của khối do người sử dụng tự đặt. Một chương trình ứng dụng có thể có nhiều khối DB và các khối DB này được phân biệt với nhau bằng số nguyên theo sau nhóm ký tự DB. Chẳng hạn như DB1, DB2, ...

Chương trình trong các khối được liên kết với nhau bằng các lệnh gọi khối và chuyển khối. Các chương trình con được phép gọi lồng nhau, tức từ một chương trình con này gọi một chương trình con khác và từ chương trình con được gọi lại gọi một chương trình con thứ 3.



Hình 1.18. Lập trình có cấu trúc

1.4.3 Các khối OB đặc biệt.

1) OB10 (*Tinme of Day Interrupt*): Chương trình trong khối OB10 sẽ được thực hiện khi giá trị thời gian của đồng hồ thời gian thực nằm trong một khoảng thời gian đã được quy định. Việc quy định khoảng thời gian hay số lần gọi OB10 được thực hiện nhờ chương trình hệ thống SFC28 hay trong bảng tham số của module CPU nhờ phần mềm STEP 7.
2) OB20 (*Time Relay 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 đặt thời gian trễ.
3) OB35 (*Cyclic Interrupt*): Chương trình trong khối OB35 sẽ được thực hiện cách đều nhau một khoảng thời gian cố định. Mặc định, khoảng thời gian này là 100ms, nhưng ta có thể thay đổi nhờ STEP 7.

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 onboard đặc biệt, hoặc thông qua các module SM, CP, FM.

5) OB80 (*Cycle Time Fault*): Chương trình trong khối OB80 sẽ được thực hiện khi thời gian vòng quét (scan time) vượt quá 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. Thời gian quét mặc định là 150ms.

6) OB81 (*Power Supply Fault*): Chương trình trong khối OB81 sẽ được thực hiện khi thấy có xuất hiện lỗi về bộ nguồn nuôi.

7) OB82 (*Diagnostic Interrupt*): Chương trình trong khối OB82 sẽ được thực hiện có sự cố từ các module mở rộng vào/ra. Các module này phải là các module có khả năng tự kiểm tra mình (diagnostic cabilities).

8) OB87 (*Communication Fault*): Chương trình trong khối OB87 sẽ được thực hiện có xuất hiện lỗi trong truyền thông.

9) OB100 (*Start Up Information*): Chương trình trong khối OB100 sẽ được thực hiện một lần khi CPU chuyển từ trạng thái STOP sang RUN.

10) OB101 (*Cold Start Up Information-chỉ với S7-400*): Chương trình trong khối OB101 sẽ được thực hiện một lần khi công tắt nguồn chuyển từ trạng thái OFF sang ON.

11) OB121 (*Synchronous Error*): Chương trình trong khối OB121 sẽ được thực hiện khi CPU phát hiện thấy lỗi logic trong chương trình đổi sai kiểu dữ liệu hay lỗi truy nhập khối DB, FC, FB không có trong bộ nhớ.

12) OB122 (*Synchronous Error*): Chương trình trong khối OB122 sẽ được thực hiện khi có lỗi truy nhập module trong chương trình.

1.5 Ngôn ngữ lập trình.

PLC S7-300 có ba ngôn ngữ lập trình cơ bản sau:

➢ Ngôn ngữ lập trình liệt kê lệnh STL (Statement List). Đây là dạng ngôn ngữ lập trình thông thường của máy tính. Một chương trình được hoàn chỉnh bởi sự ghép nối của nhiều câu lệnh theo một thuật toán nhất định, mỗi lệnh chiếm một hàng và có cấu trúc chung "tên lệnh" + "toán hạng".

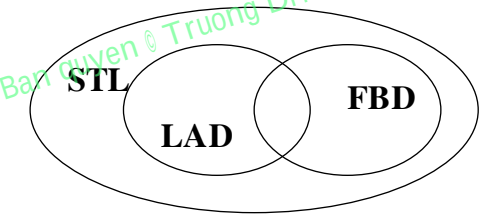
➢ Ngôn ngữ lập trình LAD (Ladder Logic). Đây là dạng ngôn ngữ đồ hoạ, thích hợp với những người lập trình quen với việc thiết kế mạch điều khiển logic.

➢ Ngôn ngữ lập trình FBD (Function Block Diagram). Đây cũng là dạng ngôn ngữ đồ hoạ, thích hợp cho những người quen thiết kế mạch điều khiển số.

➢ Ngôn ngữ lập trình Graph. Đây cũng là dạng ngôn ngữ đồ hoạ, thích hợp cho những hệ thống tuần tự.

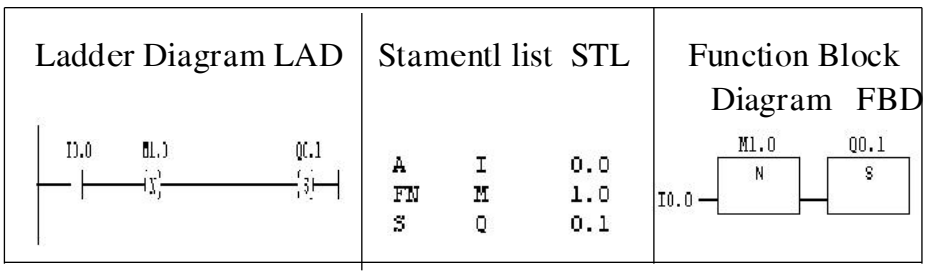
➢ Ngôn ngữ lập trình SCL. Đây cũng là dạng ngôn ngữ cấp cao, thích hợp cho những người biết viết ngôn ngữ cấp cao.

Trong PLC có nhiều ngôn ngữ lập trình nhằm phục vụ cho các đối tượng sử dụng khác nhau. Tuy nhiên một chương trình viết trên ngôn ngữ LAD hay FBD có thể chuyển sang dạng STL, nhưng ngược lại thì không. Và trong STL có nhiều lệnh mà LAD hoặc FBD không có. Đây cũng là thế mạnh của ngôn ngữ STL:



Hình 1.19. STL là ngôn ngữ mạnh nhất

Ví dụ:



CHƯƠNG 2: *NGÔN NGỮ LẬP TRÌNH STL*

2.1. Cấu trúc lệnh

Như đã biết, cấu trúc của một lệnh STL có dạng

"Tên lệnh" + "Toán hạng"

Ví dụ:

Nhãn : L PIW274 // Đọc nội dung cổng vào của module Analog

tên lệnh toán hạng

Trong đó toán hạnhg có thể là *một dữ liệu* hoặc *một địa chỉ ô nhớ*.

2.1.1 Toán hạng là dữ liệu

- Dữ liệu logic TRUE (1) và (0) có độ dài 1 bit.

Ví dụ

CALL FC1
In_Bit_1 = TRUE //Giá trị logic 1 được gán cho biến hình thức In_Bit_1
In_Bit_2 = FALSE // Giá trị logic được gán cho biến hình thức In_Bit_2
Ret_val = MW0 //Giá trị trả về.

- Dữ liệu số nhị phân.

Ví dụ

L 2#110011 //Nạp số nhị phân 110011 vào thanh ghi ACCU1

- Dữ liệu là số Hexadecimal x có độ dài 1 byte (B#16#x), 1 từ (W#16#x) hoặc 1 từ kép (DW#16#x).

Ví dụ

L B#16#1E //Nạp số 1E vào byte thấp của thanh ghi ACCU1
L W#16#3A //Nạp số 3A2 vào 2 byte thấp của thanh ghi ACCU1
L DW#16#D3A2E //Nạp số D3A2E vào thanh ghi ACCU1

- Dữ liệu là số nguyên x với độ dài 2 bytes cho biến kiểu INT.

Ví dụ

L 930

L -1025

- Dữ liệu là số nguyên x với độ dài 4 bytes dạng L#x cho biến kiểu DINT.

Ví dụ

L L#930

L L#-2047

- Dữ liệu là số thực x cho biến kiểu REAL.

Ví dụ

L 1.234567e+13

L 930.0

- Dữ liệu thời gian cho biến kiểu S5T dạng giờ _phút_giây_mili giây.

Ví dụ

L S5T#2h_1m_0s_5ms

- Dữ liệu thời gian cho biến kiểu TOD dạng giờ:phút:giây.

Ví dụ

L TOD#5:45:00

- DATE: Biểu diễn giá trị thời gian tính theo năm/tháng/ngày.

Ví dụ

L DATE#1999 – 12 – 8.

- C: Biểu diễn giá trị số đếm đặt trước cho bộ đếm.

Ví dụ

L C#20

- P: Dữ liệu biểu diễn địa chỉ của một bit ô nhớ.

Ví dụ

L P#Q0.0

- Dữ liệu “kí tự”.

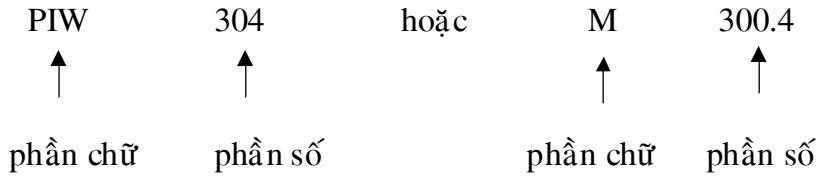
Ví dụ

L ‘ABCD’

L ‘E’

2.1.2 Toán hạng là địa chỉ.

Địa chỉ ô nhớ trong S7_300 gồm hai phần: phần chữ và phần số. Ví dụ:



a. Phần chữ chỉ vị trí và kích thước của ô nhớ.

Chúng có thể là:

- M: Chỉ ô nhớ trong miền các biến cờ có kích thước là 1 bit.
- MB: Chỉ ô nhớ trong miền các biến cờ có kích thước là 1 byte(8 bits).
- MW: Chỉ ô nhớ trong miền các biến cờ có kích thước là 2 bytes (16 bits).
- MD: Chỉ ô nhớ trong miền các biến cờ có kích thước là 4 bytes(32 bits).
- I: Chỉ ô nhô có kích thước 1 bit trong miền bộ đệm cổng vào số.
- IB: Chỉ ô nhớ có kích thước là một byte trong miền bộ đệm cổng vào số.
- IW: Chỉ ô nhớ có kích thước là một từ trong miền bộ đệm cổng vào số.
- ID: Chỉ ô nhớ có kích thước là hai từ trong miền bộ đệm cổng vào số.
- Q: Chỉ ô nhớ có kích thước 1 bit trong miền bộ đệm cổng ra số.
- QB: Chỉ ô nhớ có kích thước là một byte trong miền bộ đệm cổng raso.
- QW: Chỉ ô nhớ có kích thước là một từ trong miền bộ đệm cổng ra số.
- QD: Chỉ ô nhớ có kích thước là hai từ trong miền bộ đệm cổng ra số.
- PIB: Chỉ ô nhớ có kích thước 1byte thuộc vùng *peripheral input*. Thường là địa chỉ cổng vào của các module tương tự *(I/O external input)*.
- PIW: Chỉ ô nhớ có kích thước 1 từ (2byte) thuộc vùng *peripheral input*. Thường là địa chỉ cổng vào của các module tương tự (*I/O external input*).
- PID: Chỉ ô nhớ có kích thước 2 từ (4bytes) thuộc vùng *peripheral input*. Thường là địa chỉ cổng vào của các module tương tự (*I/O external input*).
- PQB: Chỉ ô nhớ có kích thước 1 byte thuộc vùng *peripheral output*. Thường là địa chỉ cổng ra của các module tương tự (*I/O external input*).
- PQW: Chỉ ô nhớ có kích thước 1 từ (2bytes) thuộc vùng *peripheral output*. Thường là địa chỉ cổng ra/vào của các module tương tự (*I/O external input*).

- PQD: Chỉ ô nhớ có kích thước 2 từ (4bytes) thuộc vùng *peripheral output*. Thường là địa chỉ cổng ra vào của các module tương tự (*I/O external input*).
- DBX: Chỉ ô nhớ có kích thước 1 bit trong khối dữ liệu DB được mở bằng lệnh *OPN DB(open data block)*
- DBB: Chỉ ô nhớ có kích thước 1 byte trong khối dữ liệu DB được mở bằng lệnh *OPN DB(open data block)*
- DBW: Chỉ ô nhớ có kích thước 1 từ trong khối dữ liệu DB được mở bằng lệnh *OPN DB(open data block)*
- DBD: Chỉ ô nhớ có kích thước 2 từ trong khối dữ liệu DB được mở bằng lệnh *OPN DB(open data block)*
- DBx.DBX: Chỉ trực tiếp ô nhớ có kích thước 1 bit trong khối dữ liệu DBx, với x là chỉ số của khối DB.Ví dụ:*DB5.DBX 1.6*
- DBx.DBB: Chỉ trực tiếp ô nhớ có kích thước 1 byte trong khối dữ liệu DBx, với x là chỉ số của khối DB.Ví dụ:*DB5.DBB 1.*
- DBx.DBW: Chỉ trực tiếp ô nhớ có kích thước 1 từ trong khối dữ liệu DBx, với x là chỉ số của khối DB.Ví dụ:*DB5.DBW 1.*
- DBx.DBD: Chỉ trực tiếp ô nhớ có kích thước 2 từ trong khối dữ liệu DBx, với x là chỉ số của khối DB.Ví dụ:*DB5.DBD 1.*
- DIX: Chỉ ô nhớ có kích thước 1 bit trong khối dữ liệu DB được mở bằng lệnh *OPN DI(Open distance data block)*
- DIB:Chỉ ô nhớ có kích thước 1 byte trong khối dữ liệu DB được mở bằng lệnh *OPN DI(Open distance data block)*
- DBW:Chỉ ô nhớ có kích thước 1 từ trong khối dữ liệu DB được mở bằng lệnh *OPN DI(Open distance data block)*
- DBD:Chỉ ô nhớ có kích thước 2 từ trong khối dữ liệu DB được mở bằng lệnh *OPN DI(Open distance data block)*
- L: Chỉ ô nhớ có kích thước 1 bit trong miền dữ liệu địa phương *(local block)* của các khối chương trình OB,FC,FB.
- LB: Chỉ ô nhớ có kích thước 1 byte trong miền dữ liệu địa phương (*local block*) của các khối chương trình OB,FC,FB.
- LW: Chỉ ô nhớ có kích thước 1 từ trong miền dữ liệu địa phương (*local block)* của các khối chương trình OB,FC,FB.
- LD: Chỉ ô nhớ có kích thước 2 từ trong miền dữ liệu địa phương *(local block)* của các khối chương trình OB,FC,FB.

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

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

  I 1.3 // Chỉ bit thứ 3 trong byte 1 của miền nhớ bộ đệm cổng vào số

  M 101.5 // Chỉ bit thứ 5 trong byte 101 của miền các biến cờ M.

  Q 4.5 // Chỉ bit thứ 5

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

Ví dụ

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

DBW 18 // Chỉ ô nhớ có kích thước 1 từ gồm 2 bytes 18 và 19 trong khối DB đã được mở bằng lệnh OPN DB

DB2.DBW 15 // Chỉ ô nhớ có kích thước 2 bytes 15 và 16 trong khối dữ liệu DB2.

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

2.1.3 Thanh ghi trạng thái

Khi thực hiên lệnh, CPU sẽ ghi nhận lại trạng thái của phép tính trung gian cũng như của kết quả vào một thanh ghi đặc biệt 16 bits, được gọi là thanh ghi trạng thái (*Status Word*). Mặc dù thanh ghi trạng thái này có độ dài 16 bits nhưng chỉ sử dụng 9 bits với cấu trúc như sau:

8	7	6	5	4	3	2	1	0
BR	CC1	CC0	OV	OS	OR	STA	RLO	FC

• FC *(First check)*: Khi phải thực hiện một dãy các lệnh logic liên tiếp nhau gồm các phép tính $\wedge,\vee$ và nghịch đảo, bit FC có giá trị bằng 1. Nói cách khác, FC =0 khi dãy lệnh logic tiếp điểm vừa được kết thúc.

Ví dụ:

A I0.2 //FC = 1

AN I0.3 //FC = 1

= Q4.0 //FC = 0

• RLO *(Result of logic operation)*: Kết quả tức thời của phép tính logic vừa được thực hiện. Ví dụ lệnh

A I0.3

- Nếu trước khi thực hiện bit FC = 0 thì có tác dụng chuyển nội dung của cổng vào số I 0.3 vào bit trạng thái RLO.
- Nếu trước khi thực hiện bit FC = 1 thì có tác dụng thực hiện phép tính $\wedge$ giữa RLO và giá trị logic cổng vào I 0.3. Kết quả của phép tính được ghi lại vào bit trạng thái RLO.

• STA *(Status bit)* : Bit trạng thái này luôn có giá trị logic của tiếp điểm được chỉ định trong lệnh. Ví dụ cả hai lệnh:

A I 0.3

AN I0.3

đều gán cho bit STA cùng một giá trị là nội dung của cổng vào số I 0.3.

• OR: Ghi lại giá trị của phép tính logic $\wedge$ cuối cùng được thực hiện để phụ giúp cho việc thực hiện phép toán $\vee$ sau đó. Điều này là cần thiết vì trong một biểu thức hàm hai trị, phép tính $\wedge$ bao giờ cũng phải được thực hiện trước các phép tính $\vee$.

• OS (*Store overflow bit*): Ghi lại giá trị bit bị tràn ra ngoài mảng ô nhớ.

• OV (*Overflow bit*): Bit báo kết quả phép tính bị tràn ra ngoái mảng ô nhớ.

• CC0 và CC1(*Condition code):* Hai bit báo trạng thái của kết quả phép tính với số nguyên, số thực, phép dịch chuyển hoặc phép tính logic trong ACCU

• BR(*Binary result bit*): Bit trạng thái cho phép liên kết hai loại ngôn ngữ lập trình STL. Chẳng hạn cho phép người sử dụng có thể viết một khối chương trình FB hoặc FC trên ngôn ngữ STL nhưng gọi và sử dụng chúng trong một chương trình khác viết trên LAD. Để tạo ra được mối liên kết đó, ta cần phải kết thúc chương trình trong FB, FC bằng lệnh ghi:

- 1 vào BR, nếu chương trình chạy không có lỗi .
- 0 vào BR, nếu chương trình chạy có lỗi.

Khi sử dụng các khối hàm đặc biệt của hệ thống (SFC hoặc SFB), trạng thái làm việc của chương trình cũng được thông báo ra ngoài qua bit trạng thái BR như sau

- 1 nếu SFC hay SFB thực hiện không có lỗi.
- 0 nếu có lỗi khi thực hiện SFC hay SFB.

Chú ý: *Một chương trình viết trên STL (tuỳ thuộc vào từng người lập trình) có thể bao gồm nhiều Network. Mỗi một Network chứa một công đoạn cụ thể. Ở mỗi đầu Network, thanh ghi trạng thái nhận giá trị 0, chỉ sau lệnh đầu tiên của Network, các bit trạng thái mới thay đổi theo kết quả phép tính.*

Network 1 } Đoạn chương trình 1

Network 2 } Đoạn chương trình 2

Network 3

} Đoạn chương trình 3

2.2 CÁC LỆNH CƠ BẢN

2.2.1 Nhóm lệnh logic

Bao gồm các lệnh sau

- A And
- AN And Not
- O Or
- ON Or Not
- X ExOr
- XN ExOr Not

a. Lệnh gán

Cú pháp = <toán hạng>

Toán hạng là địa chỉ I, Q, M, L, D.

Lệnh gán giá trị logic của RLO tới ô nhớ có địa chỉ được chỉ thị trong toán hạng. Lệnh tác động vào thanh ghi trạng thái *(Status word)* như sau (Kí hiệu – chỉ nội dung bit không bị thay đổi, x là bị thay đổi theo lệnh):

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

Ví dụ: Thực hiện Q4.0 = I0.3

Network 1

```
A     I0.3   //Đọc nội dung của I0.3 vào RLO
=     Q4.0   //Đưa kết quả ra cổng Q4.0
```

b. Lệnh thực hiện phép tính AND

Cú pháp A <toán hạng>

Toán hạng là dữ liệu kiểu BOOL hoặc địa chỉ I, Q, M, L, D, T, C.

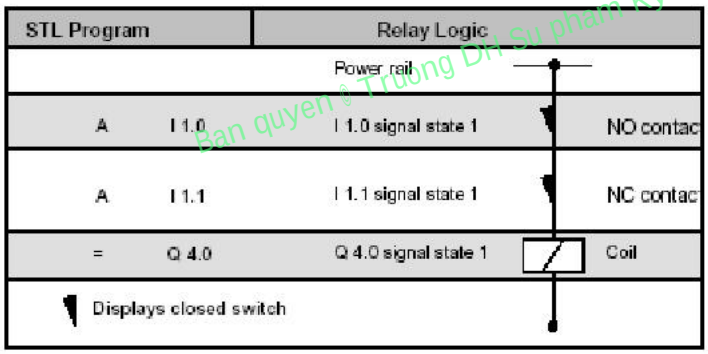
Nếu FC = 0 lệnh sẽ gán giá trị logic của toán hạng vào RLO. Ngược lại khi FC = 1 nó sẽ thực hiện phép tính AND giữa RLO với toán hạng và ghi lại kết quả vào RLO.
Lệnh tác động vào thanh ghi trạng thái (*Status word*) như sau (kí hiệu – chỉ nội dung bit không bị thay đổi, x là bị thay đổi theo lệnh):

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

Ví dụ1: Thực hiện Q4.0 = I0.3 AND I0.4 (mắc nối tiếp hai công tắc)

```
Network 1
A       I0.3        //Đọc nội dung của I0.3 vào RLO
A       I0.4        //Kết hợp AND với nội dung cổng I0.4
=       Q4.0        //Đưa kết quả ra cổng Q4.0
```

Ví dụ 2:



c. Lệnh thực hiện phép tính AND với giá trị nghịch đảo

Cú pháp AN <toán hạng>

Toán hạng là dữ liệu kiểu BOOL hoặc địa chỉ I, Q, M, L, D, T, C.
Nếu FC = 0 lệnh sẽ gán giá trị logic nghịch đảo của toán hạng vào RLO. Ngược lại khi FC = 1 nó sẽ thực hiện phép tính AND giữa RLO với giá trị nghịch đảo của toán hạng và ghi lại kết quả vào RLO.

Lệnh tác động vào thanh ghi trạng thái (*Status word*) như sau (kí hiệu – chỉ nội dung bit không bị thay đổi, x là bị thay đổi theo lệnh):

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

Ví dụ1: Thực hiện Q4.0 = I0.3 AND NOT (I0.4) (mắc nối tiếp hai công tắc)

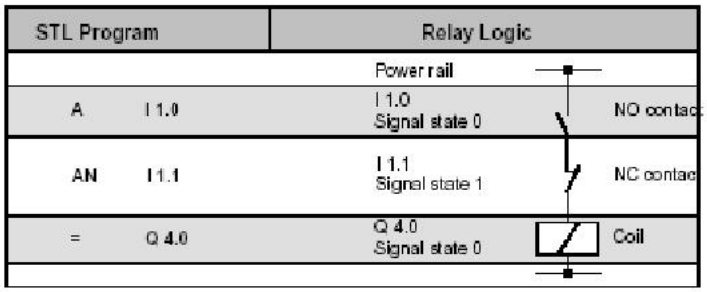
Network 1

A I0..3 // Đọc nội dung của I0.3 vào RLO

AN I0.4 //Kết hợp AND với đảo nội dung cổng I0.4

= Q4.0 //Đưa kết quả ra cổng Q4.0

Ví dụ 2



d. Lệnh OR

Cú pháp O <Toán hạng >

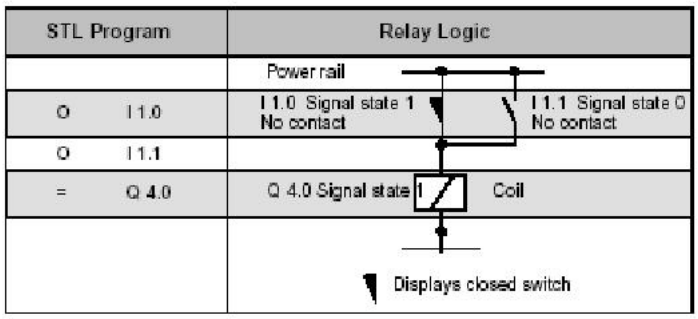
Toán hạng là dữ liệu kiểu BOOL hoặc địa chỉ I, Q, M, L, D, T, C.

Nếu FC = 0 lệnh sẽ gán giá trị logic của toán hạng vào RLO. Ngược lại khi FC = 1 nó sẽ thực hiện phép tính OR giữa RLO với toán hạng và ghi lại kết quả vào RLO.

Lệnh tác động vào thanh ghi trạng thái (*Status word*) như sau (kí hiệu – chỉ nội dung bit không bị thay đổi, x là bị thay đổi theo lệnh):

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

Ví dụ



e. Lệnh OR NOT

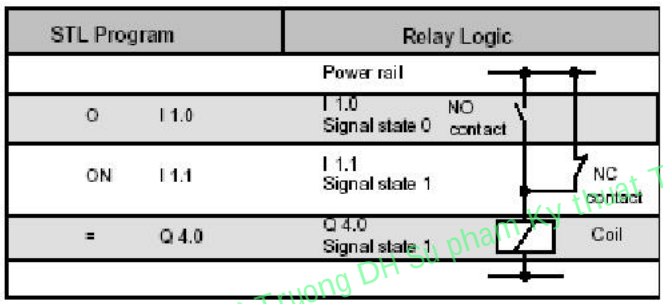
Cú pháp ***ON <Toán hạng >***

Toán hạng là dữ liệu kiểu BOOL hoặc địa chỉ I, Q, M, L, D, T, C.

Nếu FC = 0 lệnh sẽ gán giá trị logic của toán hạng vào RLO. Ngược lại khi FC = 1 nó sẽ thực hiện phép tính OR giữa RLO với NOT toán hạng và ghi lại kết quả vào RLO.

Lệnh tác động vào thanh ghi trạng thái (*Status word*) như sau (kí hiệu – chỉ nội dung bit không bị thay đổi, x là bị thay đổi theo lệnh):

Ví dụ



f. Lệnh thực hiện phép tính AND với một biểu thức

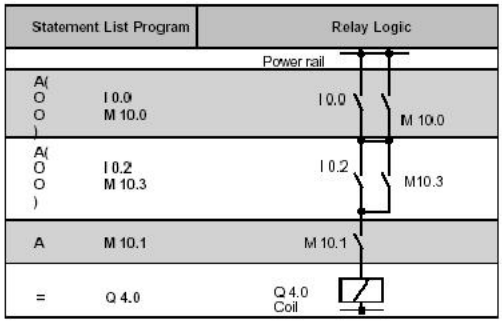
Cú pháp ***A(***

Nếu FC = 0 lệnh sẽ gán giá trị logic của biểu thức trong dấu ngoặc sau nó vào RLO. Ngược lại khi FC = 1 nó sẽ thực hiện phép tính AND giữa RLO với giá trị logic của biểu trong dấu ngoặc sau nó và ghi lại kết quả vào RLO.

Lệnh tác động vào thanh ghi trạng thái (*Status word*) như sau (kí hiệu – chỉ nội dung bit không bị thay đổi, x là bị thay đổi theo lệnh):

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	1	-	0

Ví du Q0.4 = ((I0.0 OR M10.0) AND (I0.2 OR M10.3)) AND (M10.1)



g. Lệnh thực hiện phép tính AND với giá trị nghịch đảo của một biểu thức

Cú pháp AN(

Nếu FC = 0 lệnh sẽ gán giá trị logic của biểu thức trong dấu ngoặc sau nó vào RLO. Ngược lại khi FC = 1 nó sẽ thực hiện phép tính AND giữa RLO với giá trị nghịch đảo logic của biểu trong dấu ngoặc sau nó và ghi lại kết quả vào RLO.
Lệnh tác động vào thanh ghi trạng thái (*Status word*) như sau (kí hiệu – chỉ nội dung bit không bị thay đổi, x là bị thay đổi theo lệnh):

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	1	-	0

h. Lệnh thực hiện phép tính OR với gia 1trị một biểu thức

Cú pháp O(

Nếu FC = 0 lệnh sẽ gán giá trị logic của biểu thức trong dấu ngoặc sau nó vào RLO. Ngược lại khi FC = 1 nó sẽ thực hiện phép tính OR giữa RLO với giá trị logic của biểu trong dấu ngoặc sau nó và ghi lại kết quả vào RLO.
Lệnh tác động vào thanh ghi trạng thái (*Status word*) như sau (kí hiệu – chỉ nội dung bit không bị thay đổi, x là bị thay đổi theo lệnh):

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	1	-	0

i. Lệnh thực hiện phép tính OR với nghịch đảo giá trị một biểu thức

Cú pháp ON(

Nếu FC = 0 lệnh sẽ gán giá trị logic của biểu thức trong dấu ngoặc sau nó vào RLO. Ngược lại khi FC = 1 nó sẽ thực hiện phép tính OR giữa RLO với giá trị nghịch đảo logic của biểu trong dấu ngoặc sau nó và ghi lại kết quả vào RLO.
Lệnh tác động vào thanh ghi trạng thái (*Status word*) như sau (kí hiệu – chỉ nội dung bit không bị thay đổi, x là bị thay đổi theo lệnh):

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	1	-	0

j. Lệnh ghi giá trị logic 1 vào RLO

Cú pháp SET

Lệnh không có toán hạng và có tác dụng ghi 1 vào RLO
Lệnh tác động vào thanh ghi trạng thái *(Status word)* như sau:

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

k. Lệnh gán có điều kiện giá trị logic 1 vào ô nhớ
Cú pháp S <toán hạng>
Toán hạng là địa chỉ bit I, Q, M, L, D.
Nếu RLO = 1, lệnh sẽ ghi giá trị 1 vào ô nhớ có địa chỉ cho trong toán hạng.
Lệnh tác động vào thanh ghi trạng thái (Status word) như sau:

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

l. Lệnh gán có điều kiện giá trị logic 0 vào ô nhớ

Cú pháp R <toán hạng>

Toán hạng là địa chỉ bit I, Q, M, L, D.
Nếu RLO = 1, lệnh sẽ ghi giá trị 0 vào ô nhớ có địa chỉ cho trong toán hạng.
Lệnh tác động vào thanh ghi trạng thái (Status word) như sau:

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

m. Lệnh phát hiện sườn lên

Cú pháp FP <toán hạng>

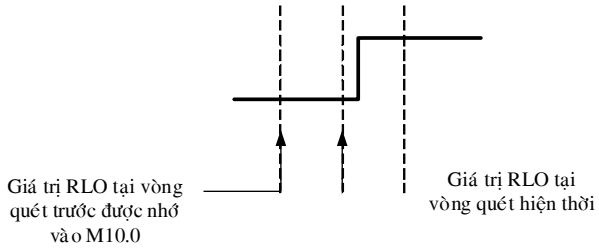
Toán hạng là địa chỉ bit I, Q, M, L, D và được sử dụng như một biến cờ để ghi nhận lại giá trị của RLO tại vị trí này trong chương trình, nhưng của vòng quét trước. Tại mỗi vòng lệnh sẽ kiểm tra: nếu biến cờ (toán hạng) có giá trị 0 và

RLO có giá trị 1 thì sẽ ghi 1 vào RLO, các trường hợp khác thì ghi 0, đồng thời chuyển nội dung của RLO vào lại biến cờ. Như vậy RLO sẽ có giá trị 1 trong một vòng quét khi có sườn lên trong RLO. Ví dụ: Lệnh phát hiện sườn lên.

A *I0.0*
FP *M10.0*
= *Q4.5*

*S*ẽ tương đương với đoạn chương trình sau

A *I0.0*
AN *M10.0*
= *Q4.5*
A *I0.0*
= *M10.0*



Hình 2.1. Hình mô tả lệnh FP

Lệnh tác động vào thanh ghi trạng thái *(Status word)* như sau:

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

n. Lệnh phát hiện sườn xuống

Cú pháp **FN** **<toán hạng>**

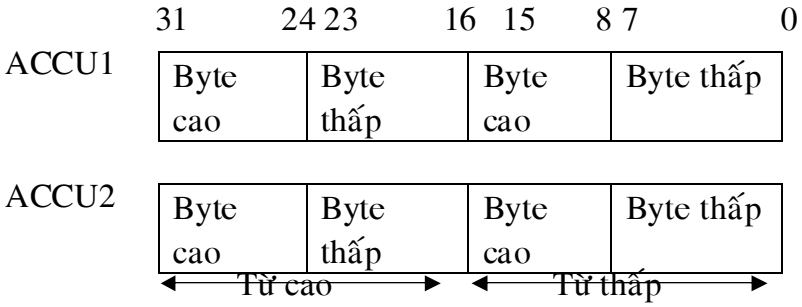
Toán hạng là địa chỉ bit I, Q, M, L, D và được sử dụng như một biến cờ để ghi nhận lại giá trị của RLO tại vị trí này trong chương trình, nhưng của vòng quét trước. Tại mỗi vòng lệnh sẽ kiểm tra: nếu biến cờ (toán hạng) có giá trị 1 và RLO có giá trị 0 thì sẽ ghi 1 vào RLO, các trường hợp khác thì ghi 0, đồng thời chuyển nội dung của RLO vào lại biến cờ. Như vậy RLO sẽ có giá trị 1 trong một vòng quét khi có sườn xuống trong RLO.Lệnh tác động vào thanh ghi trạng thái (*Status word)* như sau:

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

2.2.2 Lệnh đọc, ghi trong ACCU

Các CPU của S7_300 thường có hai thanh ghi *Accunulator* (ACCU) kí hiệu là ACCU1 và ACCU2. Hai thanh ghi ACCU có cùng kích thước 32 bits (1 từ kép). Mọi phép tính toán trên số thực, số nguyên, các phép tính logic với mảng

nhiều bits... đều được thực hiện trên hai thanh ghi này. Chúng có cấu trúc như sau:



a. Lệnh đọc vào ACCU

Cú pháp L <toán hạng>

Toán hạng là dữ liệu (số nguyên, thực, nhị phân) hoặc địa chỉ. Nếu là địa chỉ thì

- Byte IB, QB, PIB, MB, LB, DBB, DIB trong khoảng 0 – 65535
- Từ IW, QW, PIW, MW, LW, DBW, DIW trong khoảng 0 – 65534
- Từ kép ID, QD, PID, MD, LD, DBD, DID trong khoảng từ 0 – 65534

Nếu là dữ liệu thì các dạng dữ liệu hợp lệ của toán hạng cho trong bảng sau

Bảng 2.1: Các dạng dữ liệu hợp lệ của toán hạng

Dữ liệu	Ví dụ	Giải thích
± ...	L +5	Ghi 5 vào từ thấp của ACCU1
B#(...,...)	L B#(1,8)	Ghi 1 vào byte cao của từ thấp và 8 vào byte thấp của từ thấp trong ACCU
L# ...	L L#5	Ghi 5 vào ACCU1(số nguyê 32 bits)
16# ...	L B#16#2E L W#A2EB L DW#2C1E_A2EB	Dữ liệu dạng cơ số 16
2# ...	L 2#11001101	Dữ liệu dạng cơ số 2
'...'	L 'AB' L 'ABCD'	Dữ liệu dạng kí tự
C# ...	L C#1000	Dữ liệu là giá trị đặt trước cho bộ đếm
S5TIME# ...	L S5TIME#2S	Dữ liệu là giá trị đặt trước cho Timer (PV)
P# ...	L P#M10.2	Dữ liệu là địa chỉ ô nhớ(dùng cho con trỏ)
D# ...	L D#2000-6-20	Dữ liệu là giá trị về ngày/tháng/năm(16bits)
T# ...	L T#0H_1M_10S	Dữ liệu về thời gian giờ/phút/giây(32bits)

Lệnh L có tác dụng chuyển dữ liệu hoặc nội dung của ô nhớ có địa chỉ là toán hạng vào thanh ghi ACCU1. Nội dung cũ của ACCU1 được chuyển vào ACCU2. Trong trường hợp giá trị chuyển vào có kích thước nhỏ hơn từ kép thì chúng sẽ được ghi vào theo thứ tự byte thấp của từ thấp, byte cao của từ thấp, byte thấp của từ cao, byte cao của từ cao. Những bit còn trống trong ACCU1 được ghi 0.

Ví dụ1

L IB0

sẽ chuyển nội dung của IB0 vào ACCU1 như sau

	31 24	23 16	15 8	7 0
ACCU1	0	0	0	IB 0

Ví dụ 2

L MW20 //sẽ chuyển nội dung của MW20 gồm 2 bytes MB20, MB21 vào ACCU1 theo thứ tự

31 24 23 16 15 8 7 0

ACCU1	0	0	MB20	MB21

Lệnh không sửa đổi thanh ghi trạng thái (Status word).

Ví dụ 3

```
L     100  // ACCU1 =100
L     200  // ACCU1 =200, ACCU2=100
```

b. Lệnh chuyển nội dung của ACCU1 tới ô nhớ.

Cú pháp T <toán hạng>

Toán hạng là đại chỉ:
- Byte IB, QB, PIB, MB, LB, DBB, DIB trong khoảng 0 – 65535
- Từ IW, QW, PIW, MW, LW, DBW, DIW trong khoảng 0 – 65534
- Từ kép ID, QD, PID, MD, LD, DBD, DID trong khoảng từ 0 - 65534

Lệnh chuyển nội dung của ACCU1 vào ô nhớ có địa chỉ là toán hạng. Lệnh không thay đổi nội dung của ACCU2.Trong trường hợp ô nhớ có kích thước nhỏ hơn từ kép thì nội dung của ACCU1 được chuyển ra theo thứ tự byte thấp của từ thấp, byte cao của từ thấp, byte thấp của từ cao, byte cao của từ cao.

Ví dụ

```
T     QB0
```

sẽ chỉ chuyển nội dung của byte thấp của từ thấp trong ACCU1 vào IB0 và lệnh

```
T     MW20
```

sẽ chỉ chuyển byte cao của từ thấp vào MW20, byte thấp của từ thấp vào MW21.

Lệnh không sửa đổi thanh ghi trạng thái (Status word).

CHƯƠNG 3
NGÔN NGỮ GRAPH VÀ ỨNG DỤNG

Khi lập trình cho PLC sử dụng khối FB thì chúng ta có thể sử dụng ngôn ngữ Graph. Ngôn ngữ này rất thuận lợi trong những hệ thống điều khiển tuần tự. Lưu ý trong khi cài đặt phần mềm Step7 ta phải chọn cài đặt ngôn ngữ này.

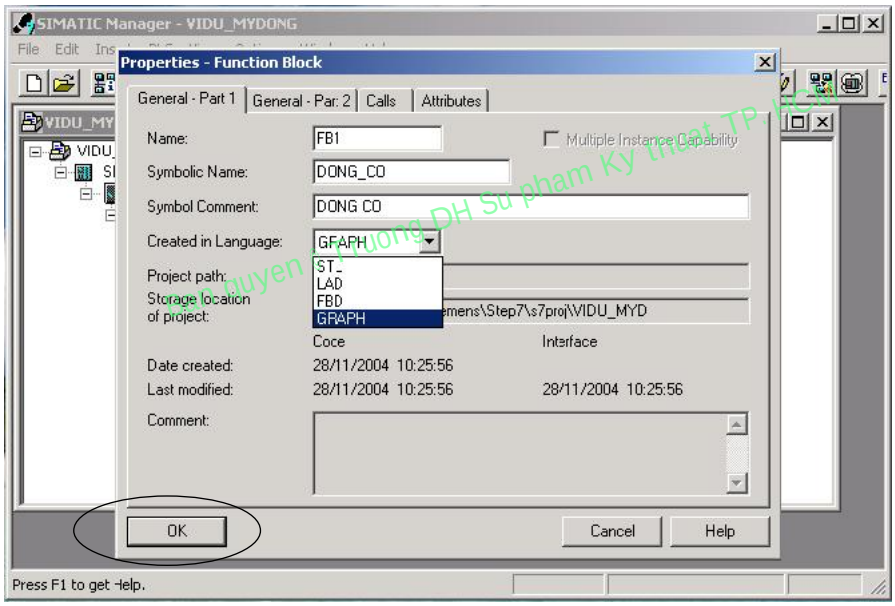
3.1 Tạo một khối FB dưới dạng ngôn ngữ Graph

3.1.1. Tạo một khối FB Graph

Bước 1: Double click vào folder **Blocks.**

Bước 2: Chọn trên thanh Menu: **Insert > S7 Block > Function Block**.

Bước 3: Một hộp thoại " Properties" xuất hiện. Chọn ngôn ngữ lập trình là Graph



Hình 3.1. Chọn ngôn ngữ Graph khi lập trình trên khối FB

Rồi chọn OK. Như vậy kết quả là một khối FB1 được tạo ra trong folder Blocks

3.1.2. Viết chương trình theo kiểu tuần tự

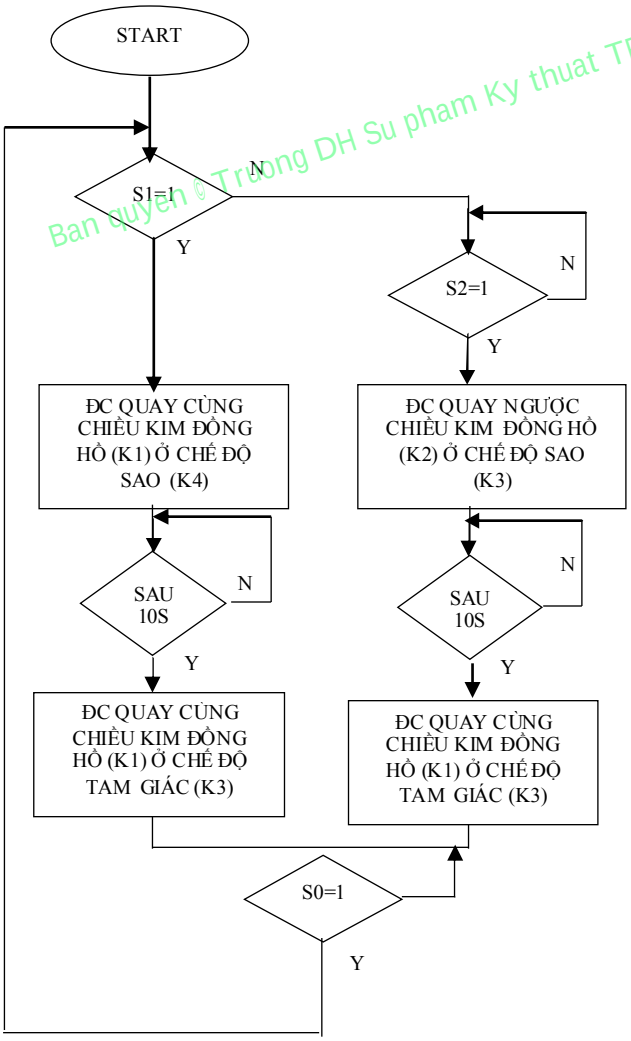
Để tiện theo dõi xét ví dụ điều khiển khởi động SAO/TAMGIAC của động cơ 3 pha như sau

Động cơ không đồng bộ 3 pha rô to lồng sóc phải được vận hành cả 2 chiều quay. Để khắc phục được dòng khởi động lớn, động cơ phải được khởi động với chế độ kết nối sao - tam giác

YÊU CẦU

Khi nhấn nút S1 thì động cơ chạy và quay cùng chiều kim đồng hồ, và động cơ sẽ quay theo chiều ngược lại nếu nhấn nút S2. Công tắc tơ chính K1 cho chiều quay cùng chiều kim đồng hồ và K2 cho chiều ngược lại, kích hoạt công tắc tơ chế độ sao là K4 và một timer. Sau một khoảng thời gian khởi động gần 5 s, động cơ tự động ngắt chế độ chạy sao. Công tác tơ chính K1 vẫn còn được kích hoạt và ngắt sự kết nối với chế độ chạy sao – công tắc tơ K4 trước khi chuyển sang kết nối với chế độ tam giác – công tắc tơ K3. Chiều quay của động cơ chỉ được thay đổi khi động cơ đã được tắt trước đó. Động cơ chỉ có thể được tắt khi nhấn nút S0, độc lập với trạng thái hoạt động . Trạng thái ON của động cơ phải được hiển thị qua đèn H1 và H2 tuỳ thuộc chiều quay của động cơ. Khi động cơ quá tải nó sẽ được tự động tắt qua rơ le Q1(S5).

Hoạt động Sao/tamgiac của động cơ 3 pha được trình bày theo lưu đồ giải thuật sau



Hình 3.2. Lưu đồng giải thuật của khởi động Sao/tamgiac

Bảng địc chỉ vào ra

Ngõ vào		Ngõ ra	
THIẾT BỊ NGOÀI	ĐỊA CHỈ	THIẾT BỊ NGOÀI	ĐỊA CHỈ
S0	I0.0	Q1	Q01
S1	I0.1	Q2	Q0.2
S2	I0.2	Q3	Q0.3
S5	I0.5	Q4	Q0.4

Trình tự lập trình như sau

Sau khi bắt đầu làm việc với S7 Graph bằng cách double click vào khối FB1 thì hệ thống được chèn vào một STEP đầu tiên và một TRANSITION đầu tiên.

Có 2 phương pháp để tạo cấu trúc Sequencer.

Phương pháp 1: Ở chế độ "Direct": **Insert > Direct**

Phương pháp 2: Ở chế độ "Drap-and-Drop": **Insert > Drap-and-Drop**

Sau đây chỉ trình bày cách viết theo phương pháp 1

- Bước 1: Chọn transition 1 và nhấp chuột vào biểu tượng một lần

 insert step + transition,

Kết quả tạo ra một step 2. Tại step này động cơ thực hiện chế độ quay cùng chiều kim đồng hồ, và mạch được kết nối dạng **SAO**.

- Bước 2: Chọn step 2 và chọn biểu tượng

 open alternative branch.

Điều này sẽ mở ra một nhánh xen vào cho chế độ động cơ cũng quay cùng chiều kim đồng hồ nhưng mạch được kết nối dạng **TAM GIÁC**. Nhánh này bắt đầu với transition 3 (T3)

- Bước 3: Tiếp tục với con chuột đang ở tại vị trí T3, nhấp chuột chọn biểu tượng

 insert step + transition,

Và sẽ được chèn vào step 3 cùng với transition 4

- Bước 4: Chọn step 1 và chọn biểu tượng

 open alternative branch.

Điều này sẽ mở ra một nhánh xen vào cho chế độ động cơ ngược chiều kim đồng hồ. Nhánh này bắt đầu với transition 5 (T5)

➢ Bước 5: Tương tự như nhánh chính ứng với chế độ quay của động cơ là cùng chiều kim đồng hồ.Vẫn để con chuột tại transition 5 và nhấp chuột vào biểu tượng sau một lần

 insert step + transition,

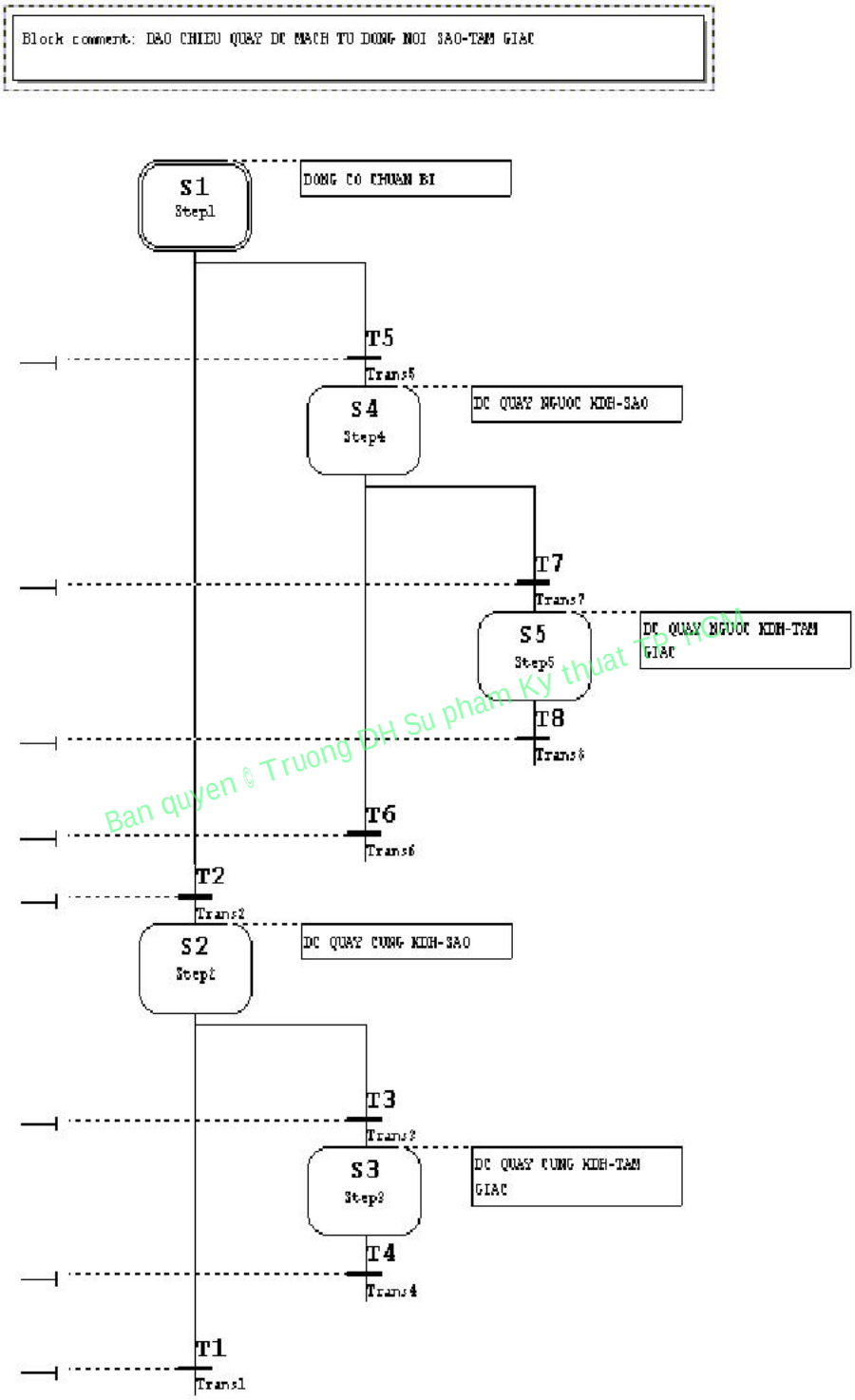
Kết quả tạo ra một step 4 và transition 6. Tại step này động cơ thực hiện chế độ quay ngược chiều kim đồng hồ, và mạch được kết nối dạng **SAO**.

tiếp tục thực hiện giống như nhánh chinh ta được mạch như sau:

➢ Bước 6: Và bấy giờ ta hoàn thành cấu trúc của một Sequencer bằng cách đầu tiên ta chọn transition 1 (T1) rồi nhấp chuột chọn biểu tượng

 insert jump

và rồi chọn step1 hoặc gõ vào số “1”



Hình 3.2. Tạo nhánh trong Graph

3.2. Viết chương trình các ACTION cho các step

Cũng có 2 phương pháp để viiết chương trình các action cho các step và các transition: **Direct** và **Drap-and-Drop**

Sau đây sẽ sử dụng phương pháp **Drap-and-Drop** : **Insert > Drap-and-Drop**
Bước 1: Chọn trên thanh menu **Insert > Action**
Kết quả là: Trên con chuột sẽ xuất hiện biểu tượng sau

Bước 2: chèn dòng action rỗng bằng cách nhấp chuột vào ô action
Bước 3: Enter vào các action
Một action bao gồm 1 lệnh và 1 địa chỉ. Trong ngôn ngữ Graph có 4 lệnh hay sử dụng :

S Set ngõ ra
R Reset ngõ ra
D Delay 1 khoảng thời gian (xem thêm S4 hình 3)
C Đếm sự kiện

S1 CU	C20	→ Sử dụng Counter C20 đếm số hành động của S1
S1 CR	C20	→ Reset Counter C20

3.3. Viết chương trình các TRANSITION

Có các hàm logic " Công tắc thường mở", "Công tắc thường đóng", " Hàm so sánh" được sử dụng cho các điều kiện-CONDITION trong các transition. Viết chương trình cho các transition như sau:
Bước 1: Chọn View >LAD

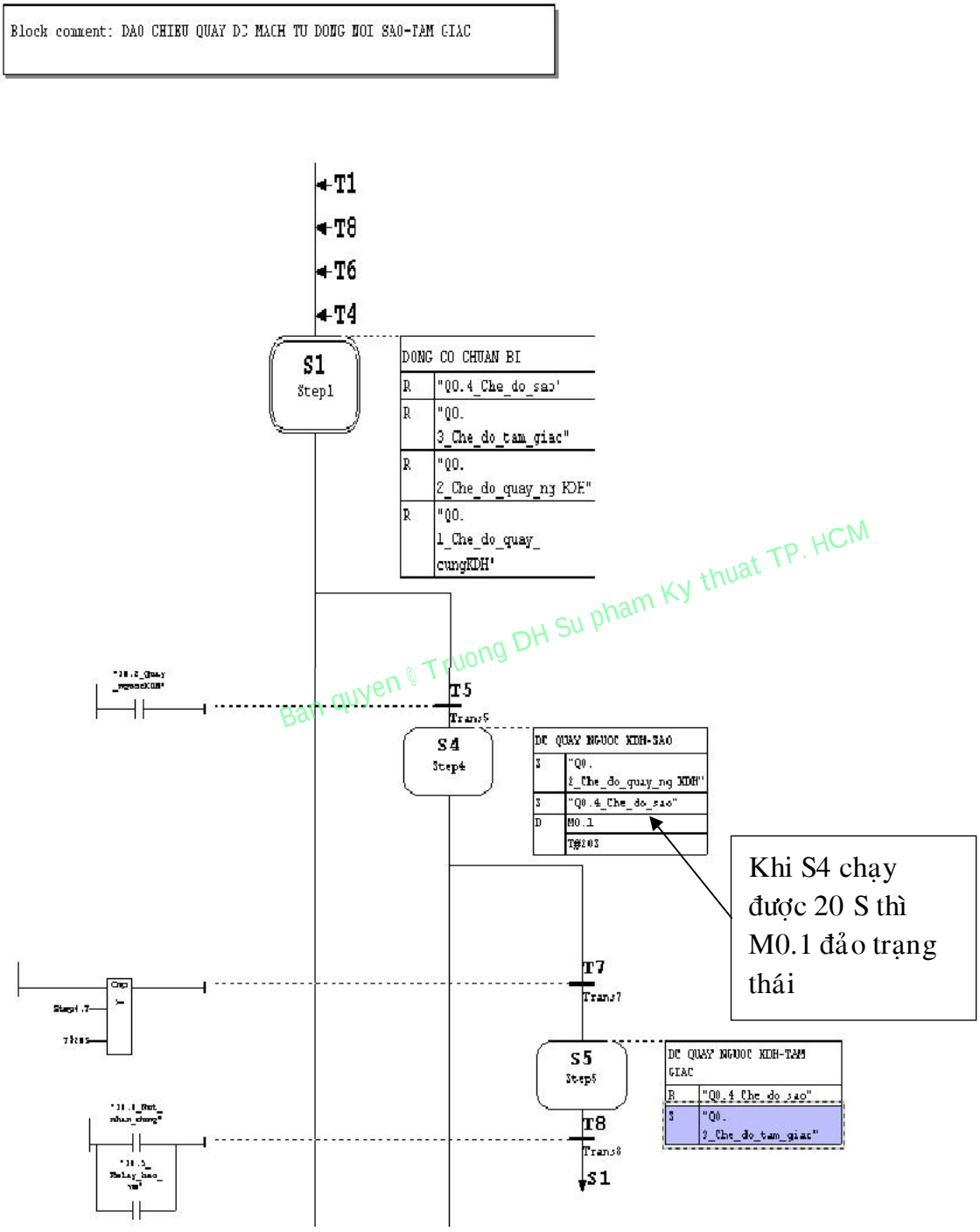
 Chèn vào công tắc thường mở

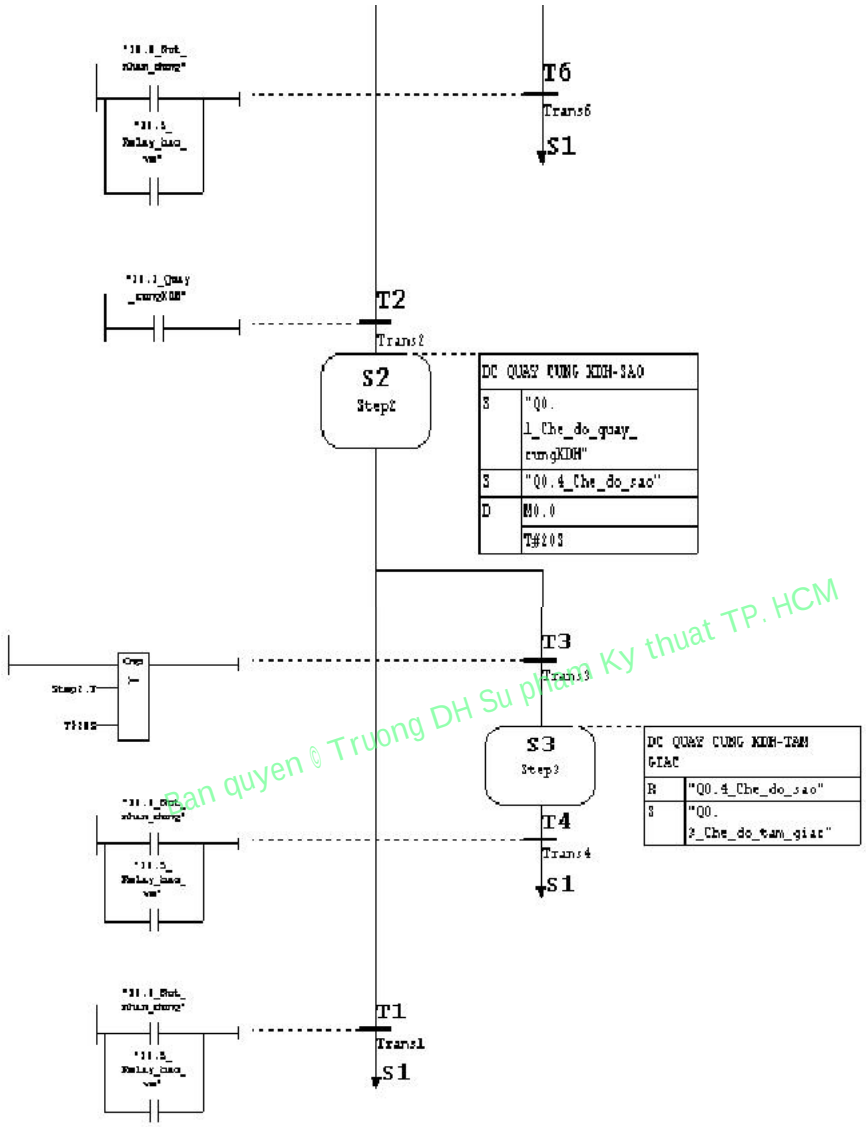
 Chèn vào công tắc thường đóng

 Chèn vào phép so sánh

Bước 2: Sau khi nhấp chọn và chèn vào đúng vị trí có thể thoát ra bất cứ lúc nào bằng cách nhấn phím ESC

Bước 3: Enter địa chỉ vào. Nhấp chuột vào vùng yêu cầu "??.?"
Rồi gõ vào đó địa chỉ hoặc kí hiệu của địa chỉ (Ví dụ I0.0 hoặc I0.0_Nut_nhan_dung)
Bước 4:





Hình 3.3. Chương trình khởi động sao/tamgiac dùng ngôn ngữ Graph

3.4. Lưu và đóng chương trình lại

Khi lưu chương trình lại, thì phần mềm tự động được kiểm tra (compile)
Bước 1: Chọn trên thanh menu **File > Save**
Kết quả là: một hộp thoại "Select Instance DB" được mở ra với thông số mặc định là DBx (với x trùng với x của khối FBx ví dụ nếu FB1 thì DB1)
Bước 2: Đồng ý với mặc định này bằng cách nhấp chọn **"OK"**
Kết quả là: Khối dữ liệu "**DB**-Data block" tự động được tạo ra trong folder "Blocks"
Bước 3: Đóng chương trình lại bằng cách chọn **File > Close**

3.5 Gọi chương trình từ trong khối FB1 vào khối OB1

Chương trình điều khiển động cơ được gọi vào trong khối OB1. Chúng ta có thể tạo khối OB1 viết dưới dạng LAD, FBD, STL, hoặc SCL (Ở đây khối OB1 được tạo ra dưới dạng LAD. Chương trình của khối OB1 được biểu diễn như sơ đồ sau. Làm trình tự các bước như sau:

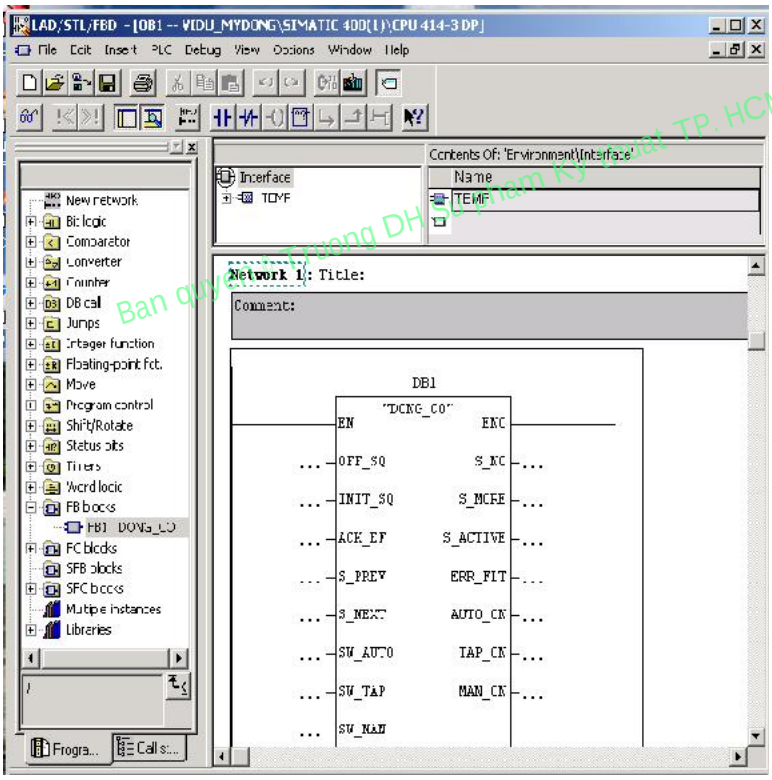
Bước 1: Mở folder "Blocks" trong S7 program trong cửa sổ SIMATIC Manager

Bước 2: Double- click vào khối OB1

Bước 3: Chọn ngôn ngữ viết dạng LAD bằng cách View > LAD

Bước 4: Mở Overviews ra nếu chư có sẵn bằng cách View > Overwiews
Rồi nhấp chọn FB, rồi double-click vào FB1

Bước 5: Gõ tên của khối dữ liệu "DB1" vào phía trên



Hình 3.4. Gọi khối FB trong OB1

Bước 6: Lưu và đóng khối OB1 lại bằng cách **File > Save** rồi **File >Close**

3.6. Download chương trình xuống CPU và kiểm tra tuần tự chương trình

3.6.1. Download chương trình xuống CPU

Để cho phép download chương trình xuống CPU, ta phải download tất cả các khối (**DB1, FB1, OB1, FC70/71, FC72 và/hoặc FC73**) xuống CPU theo các bước sau:

Bước 1: Mở cửa sổ chính SIMATIC Manager và chọn folder " Blocks".
Bước 2: Chọn menu PLC> Download

3.6.2 Kiểm tra chương trình

Để kiểm tra chương trình , đòi hỏi cần phải kết nối với với CPU
Bước 1: Mở cửa sổ chính SIMATIC Manager.
Bước 2: Mở Sequencer bằng cách double-click vào khối FB1.
Bước 3: Chọn menu Debug > Monitor.

Kết quả là: Trạng trái chương trình được hiển thị (Step đầu tiên được kích hoạt). Step nào hoạt động được hiển thị màu xanh

Chương 4
PHẦN MỀM STEP7

4.1 Sơ lược về phần mềm STEP7.

STEP 7 là một phần mềm dùng để phục vụ cho việc đặt cấu hình và lập trình cho các bộ điều khiển lập trình được (PLC_Programmable Logic Controller). Đây là bộ phần mềm do hãng Siemens thiết kế, bao gồm các version cơ bản sau :

- STEP 7 Micro/Dos và STEP 7 Micro/Win dành cho các ứng dụng chuẩn, đơn giản trên SIMATIC S7-200.
- STEP 7 Mini dành cho các ứng dụng chuẩn, đơn giản trên SIMATIC S7-300 và SIMATIC C7-620.
- STEP 7 dành cho các ứng dụng trên SIMATIC S7-300/S7-400, SIMATIC M7-300/M7-400 và SIMATIC C7 với các chức năng rộng hơn:
  - Có khả năng gán các thông số cho các module hàm và các bộ xử lý truyền thông.
  - Có thể hoạt động ở chế độ nhiều máy tính.
  - Truyền thông dữ liệu toàn cục.
  - Truyền dữ liệu theo sự kiện sử dụng các khối hàm truyền thông (communication function blocks).
  - Đặt cấu hình kết nối.

4.1.1 Cài đặt Step7.

- Yêu cầu phần cứng:
  - Hệ điều hành : Windows 95, Windows 98 hay Windows NT.
  - Phần cứng :
    - Bộ xử lý 80486 hay cao hơn.
    - RAM: ít nhất là 32Mbytes.
    - Màn hình, chuột, bàn phím có hỗ trợ Win 95/98/NT.
- Cài đặt STEP 7:
  - Cho đĩa STEP 7 vào ổ đĩa CD-ROM.
  - Chạy chương trình setup trên đĩa, cũng giống như việc cài đặt các phần mềm khác. Tuy nhiên việc cài đặt STEP 7 có vài điểm khác biệt so với các phần mềm khác:

- Khai báo số hiệu sản phẩm: số hiệu sản phẩm luôn đi kèm theo đĩa. Do đó khi quá trình cài đặt yêu cầu số hiệu sản phẩm, bạn phải điền đầy đủ các thông tin vào các mục yêu cầu.

- Đăng ký bản quyền (AuthorsW): bản quyền của STEP 7 do Simens cung cấp thường được chứa trong đĩa mềm riêng (màu đỏ). Ta có thể đăng ký bản quyền ngay trong quá trình cài đặt hay sau khi cài đặt phần mềm xong bạn chạy chương trình AuthorsW.exe có trong danh sách của SIMATIC.

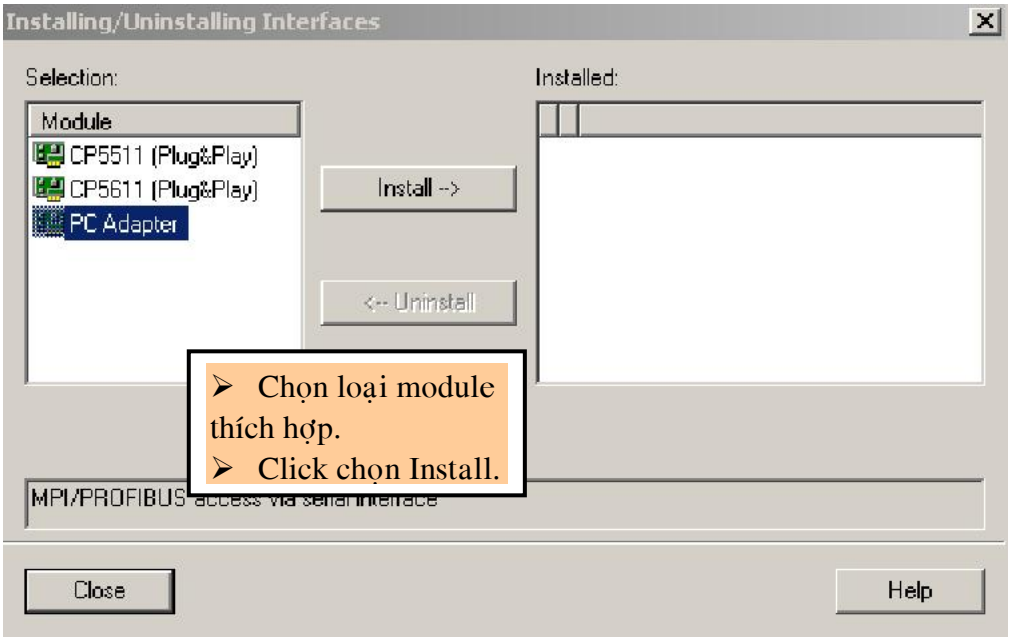
4.1.2 Các công việc khi làm việc với phần mềm STEP 7.

- Lập kế hoạch cho bộ điều khiển.
- Thiết kế cấu trúc chương trình.
- Khởi động STEP 7.
- Tạo cấu trúc project.
- Đặt cấu hình cho trạm.
- Đặt cấu hình mạng và các kết nối truyền thông.
- Định nghĩa các ký hiệu.
- Tạo chương trình.
- Đối với S7: tạo và đánh giá các dữ liệu tham chiếu.
- Đặt cấu hình các thông điệp.
- Đặt cấu hình các biến điều khiển.
- Download chương trình xuống bộ điều khiển.
- Kiểm tra chương trình.
- Quan sát hoạt động và chẩn đoán lỗi.

4.1.3. Set giao diện PG/PC.

Với việc thiết lập này, giúp bạn thiết lập kiểu kết nối giao tiếp giữa thiết bị lập trình (PC) và bộ điều khiển logic khả trình (PLC).

- Khi Set PG/PC Interfaces lần đầu tiên, ta phải cài đặt module giao tiếp như sau:



Hình 4.1. Set giao diện PG/PC

➢ Trong hộp thoại Set PG/PC Interfaces ta chọn loại card phù hợp chuẩn giao tiếp hệ thống mạng và click vào nút Properties...

➢ Hộp thoại Properties - PC Adapter hiện ra, ta thiết lập các thông số giao tiếp cần thiết như: địa chỉ, tốc độ truyền,...

4.2 CÁCH TẠO 1 CHƯƠNG TRÌNH ỨNG DỤNG VỚI STEP7

4.2.1. Các bước soạn thảo 1 Project

Cách xây dựng cấu hình phần cứng cho trạm PLC.

Ví dụ xây dựng cấu hình phần cứng cho 1 trạm như sau:

- Phần cứng của trạm gồm một thanh ray

RACK-300 : thanh RAIL

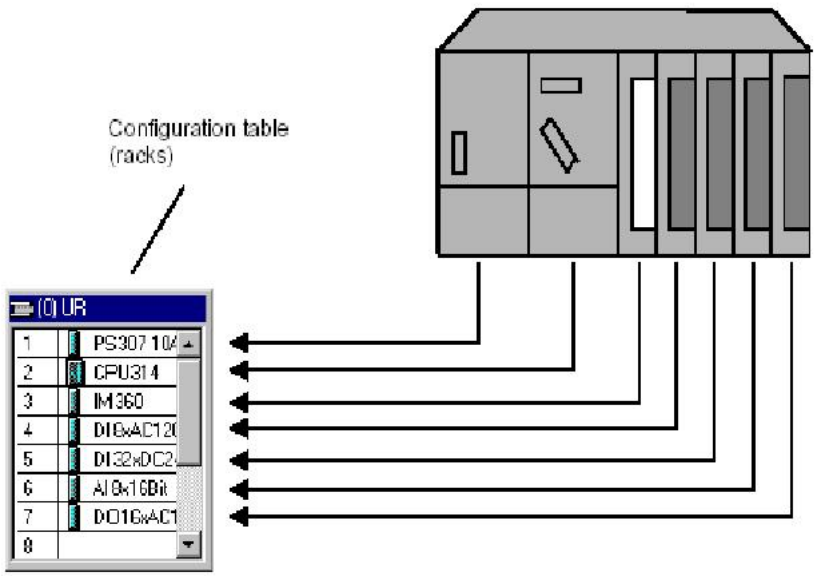
- Trên thanh này có gắn các môđun : nguồn PS , CPU, DI/DO, AI, AO trên các SLOT Trong đó :

- SLOT 1: có gắn môđun nguồn “PS 307 5A với mã số : 6ES7 307-1EA00-0AA0

- SLOT 2 và SLOT 3 : Có gắn môđun CPU –300 "CPU 315-2 DP với mã số : 6ES7 315- 2AFO3-OABO- V1.2 " môđun này để truyền dữ liệu từ S7-300 bằng đường truyền mạng MPI và PROFIBUS với tốc độ 1.5 MB
- SLOT 4 : Môđun tín hiệu ngõ vào/ ra digital DI8 /DO8 x24V/0.5A với mã số : 6ES7 323-1BH00-0AA0
- SLOT 5 : Môđun tín hiệu ngõ vào analog AI 2x12bit với mã số : 6ES7 331-7KB02-0AB0
- SLOT 6 : Môđun tín hiệu ngõ ra analog A0 2x12bit với mã số : 6ES7 332-5HB01-0AB0

Chú ý: ta không thể đặt các thành phần ở cửa sổ bên phải vào cửa sổ bên trái một cách tuỳ tiện không theo một thứ tự. Thường thì các thành phần được đặt vào các Slot ở cửa sổ bên trái theo thứ tự như sau:

- Slot 1: chỉ sử dụng đặt modul nguồn.
- Slot 2: chỉ sử dụng đặt modul CPU.
- Slot 3: thông thường để rỗng.
- Slot 4 tới Slot 11: dùng cho các module truyền thông xử lý(modul xuất, modul nhập, modul vào ra tương tự...).

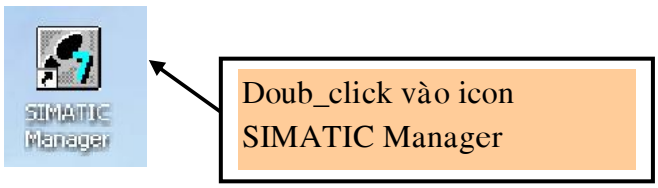


Hình 4.2. Thứ tự sắp xếp của các Slot trên một Rack

4.2.2. Thiết lập phần cứng cho trạm

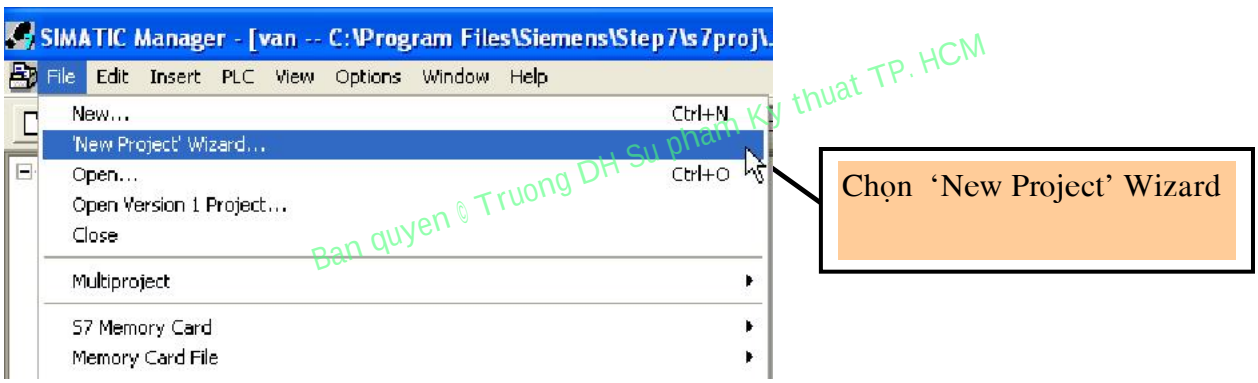
1. Khởi động phần mềm SIMATIC Manager

Start -> SIMATIC Manager hoặc **doub_ click** vào biểu tượng :



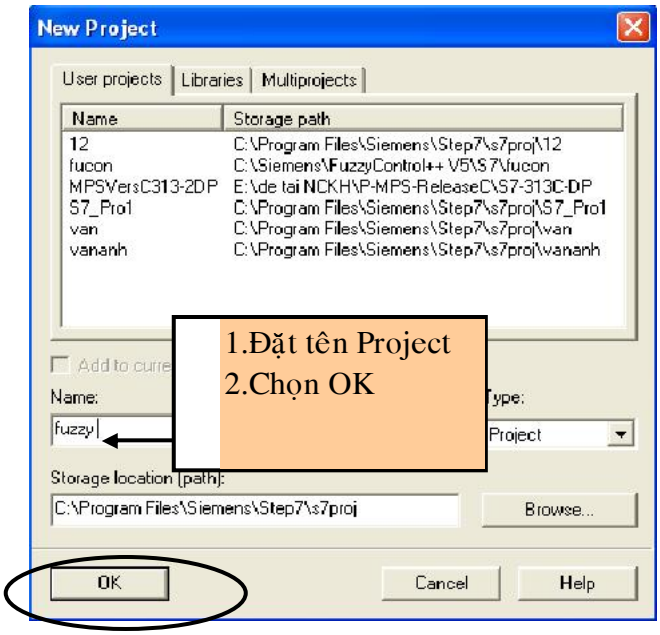
Hình 4.2. *Biểu tượng của SIMATIC Manager*

2. Tạo 1 Project mới : **File ->'New Project' Wizard**



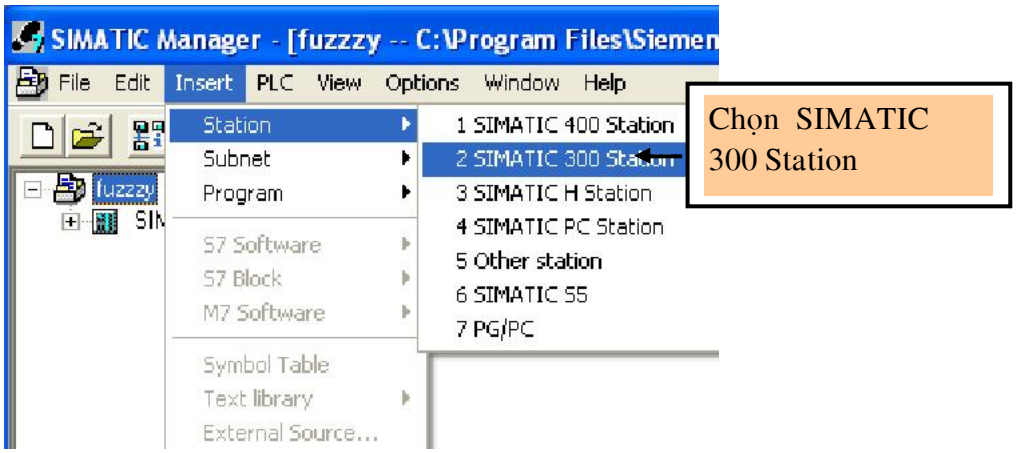
Hình 4.3. *Tạo 1 Project mới*

3. Đặt tên cho Project :



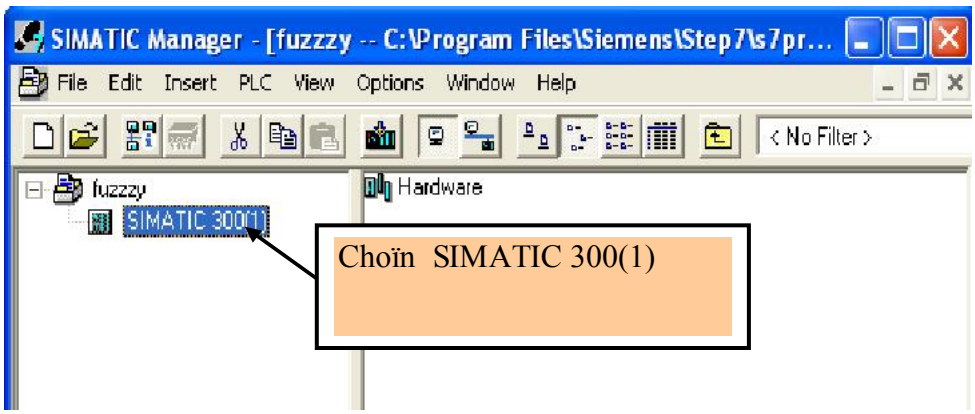
Hình 4.4. *Đặt tên cho Project*

4. Chèn SIMATIC 300 Station : Insert -> station -> SIMATIC 300 Station



Hình 4.5. *Chèn 1 trạm SIMATIC mới*

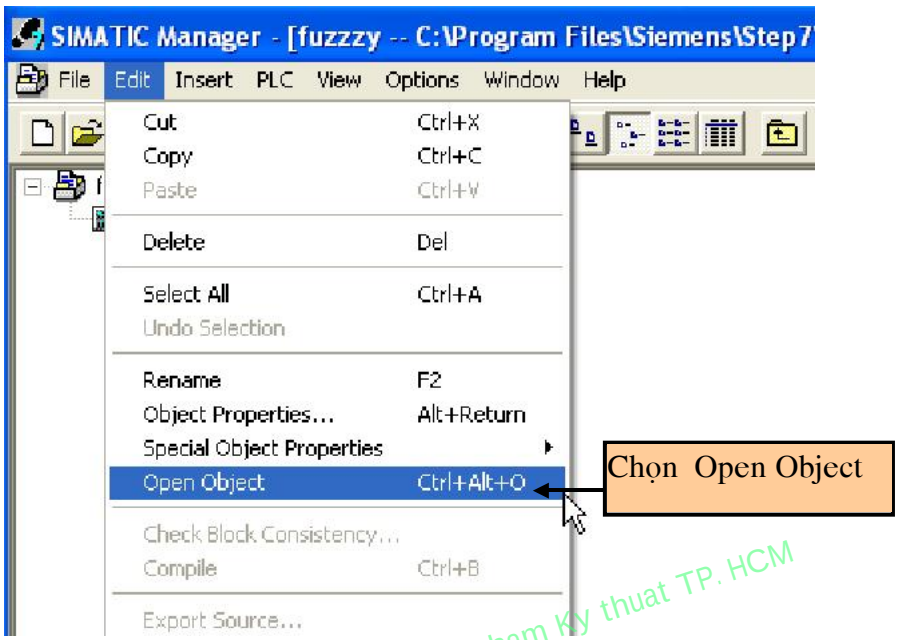
5. Chọn SIMATIC 300 Station(1)



Hình 4.6: *Chọn trạm SIMATIC vừa tạo*

6. Mở cấu hình phần cứng :

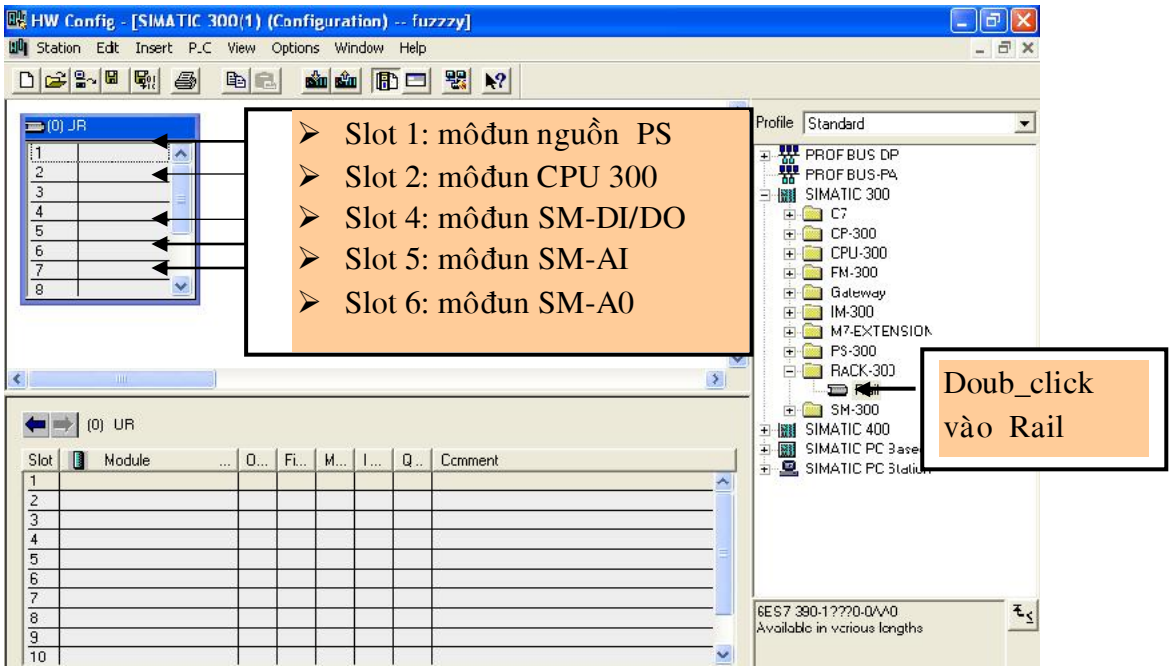
Click doub vào **Hardware** hoặc vào **Edit -> Open Object**



Hình 4.7: *Mở cấu hình phần cứng*

7. Lấy thanh rail ở slot 0 nằm dưới dấu + của RACK :

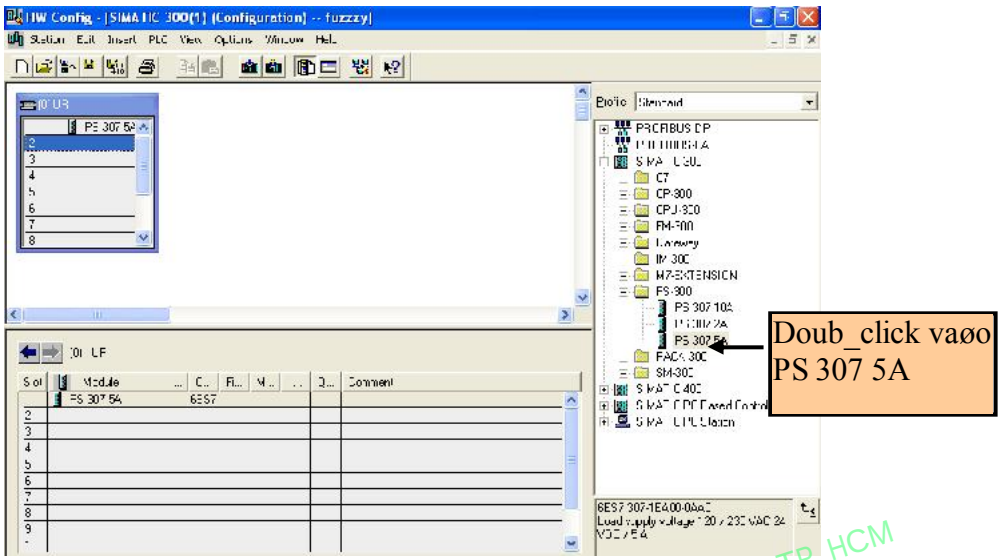
SIMATIC 300 -> RACK-300 -> doub- click Rail



Hình 4.8: *Vị trí của mỗi Slot*

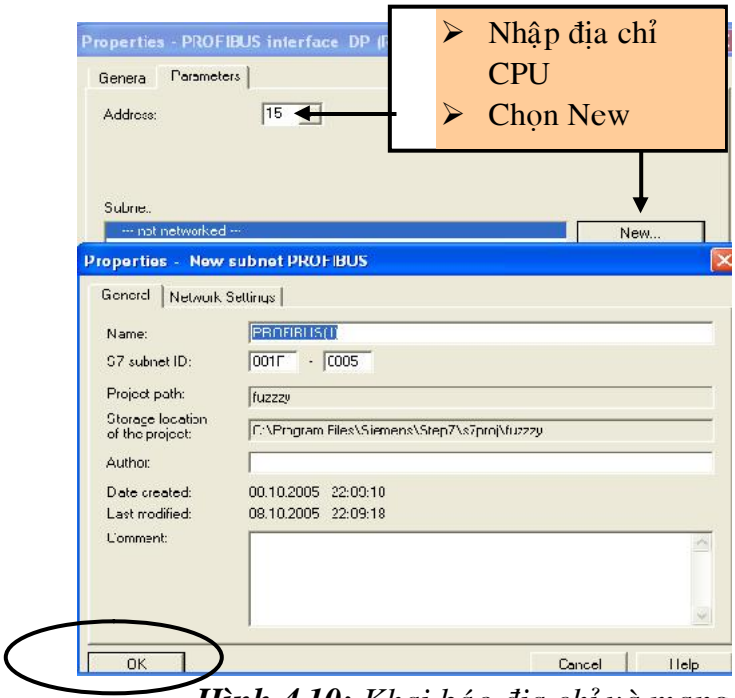
8. **Click vào SLOT 1**

chọn môđun nguồn "PS 307 5A với mã số : 6ES7 307-1EA00-0AA0, bằng cách doub_click vào PS 307 5A



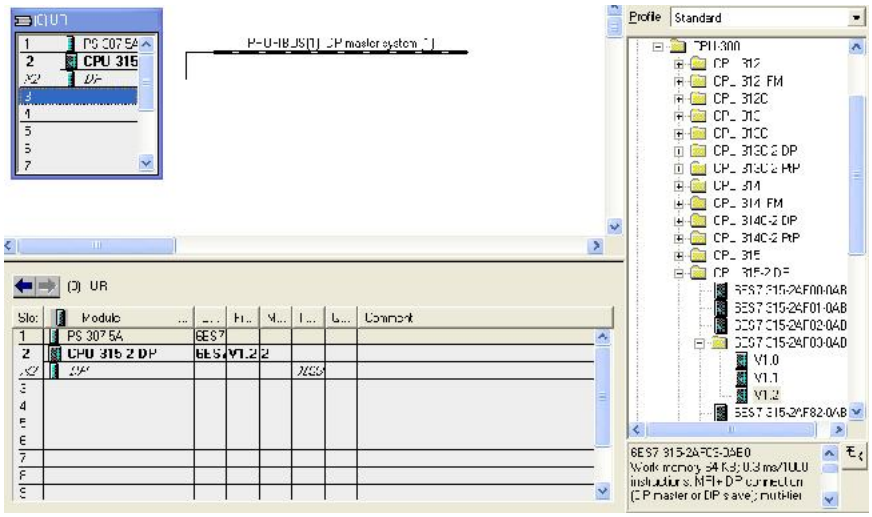
Hình 4.9. *Khai báo địa chỉ nguồn*

9. **Tương tự như bước 8 click vào Slot 2** chọn môđun CPU –300 "CPU 315-2 DP với mã số : 6ES7 315- 2AFO3-OABO bằng cách doub_click vào **V1.2** lúc đó sẽ xuất hiện hộp thoại "Properties" nhập địa chỉ DP của CPU là 15 và nhấn phím **NEW** sau đó chọn OK



Hình 4.10: *Khai báo địa chỉ và mạng kết nối Profibus*

Kết quả sẽ tạo ra : hệ thống chủ (1) "PROFIBUS(1) : DP"



Hình 4.11: *Khai báo CPU*

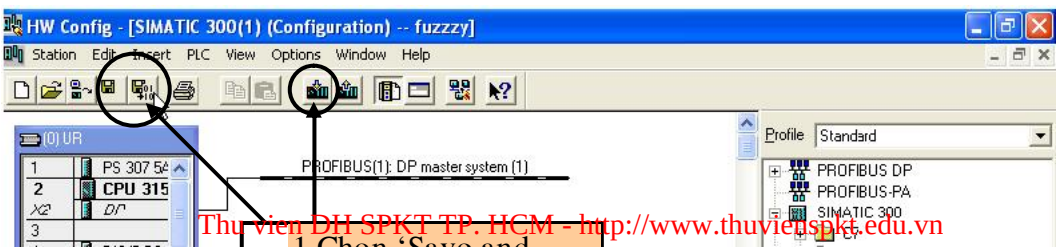
10. **Click vào slot 4** : từ SM-300 chọn môđun tín hiệu ngõ vào/ ra digital DI8 /DO8 x24V/0.5A với mã số : 6ES7 323-1BH00-0AA0 bằng cách doub _click vào DI8 /DO8 x24V/0.5A

11. Click vào slot 5 từ SM-300 chọn môđun tín hiệu ngõ vào analog AI 2x12bit với mã số : 6ES7 331-7KB02-0AB0 bằng cách doub _click vào AI 2x12bit

12. Click vào slot 6 từ SM-300 chọn môđun tín hiệu ngõ ra analog A0 2x12bit với mã số : 6ES7 332-5HB01-0AB0 bằng cách doub _click vào A0 2x12bit

13. Sau khi thiết lập phần cứng xong ta tiến hành lưu và kiểm tra bằng cách chọn menu **Station > Save and Compile**

14. Download cấu hình phần cứng xuống dưới CPU của PLC bằng cách chọn menu **PLC -> Download**



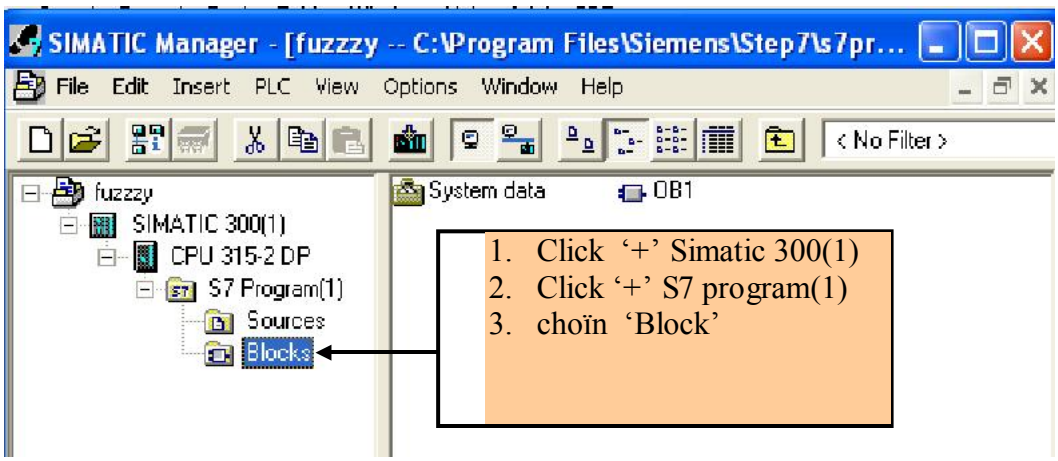
Hình 4.12. Save và download cấu hình phần cứng

4.2.3 Soạn thảo chương trình cho các khối logic

Sau khi khai báo xong cấu hình cứng cho một trạm PLC và quay trở về cửa sổ chính của step7 ta sẽ thấy Step7 trong thư mục SIMATIC 300(1) bây giờ có thêm các thư mục con CPU315-2DP, S7 Program(1), Sources, Blocks

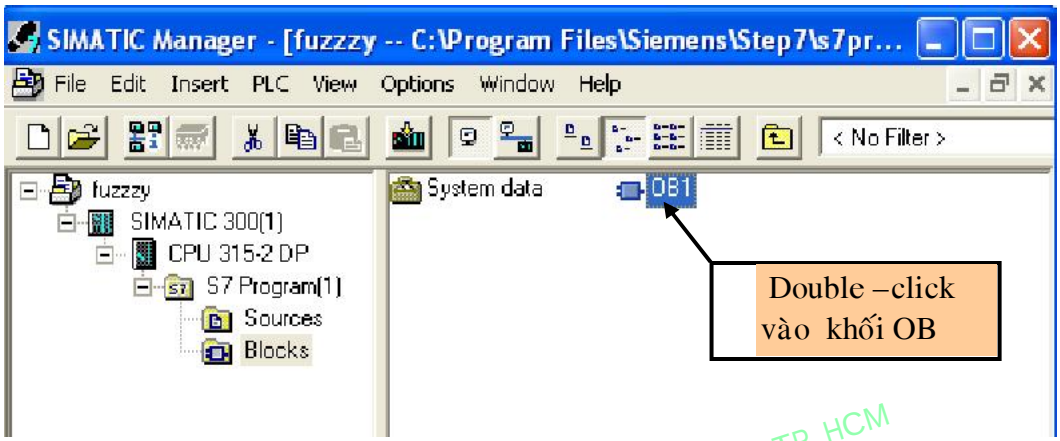
Mở cửa sổ SIMATIC manager lên và chọn **'Block'**

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



Hình 4.13. Chọn khối Blocks

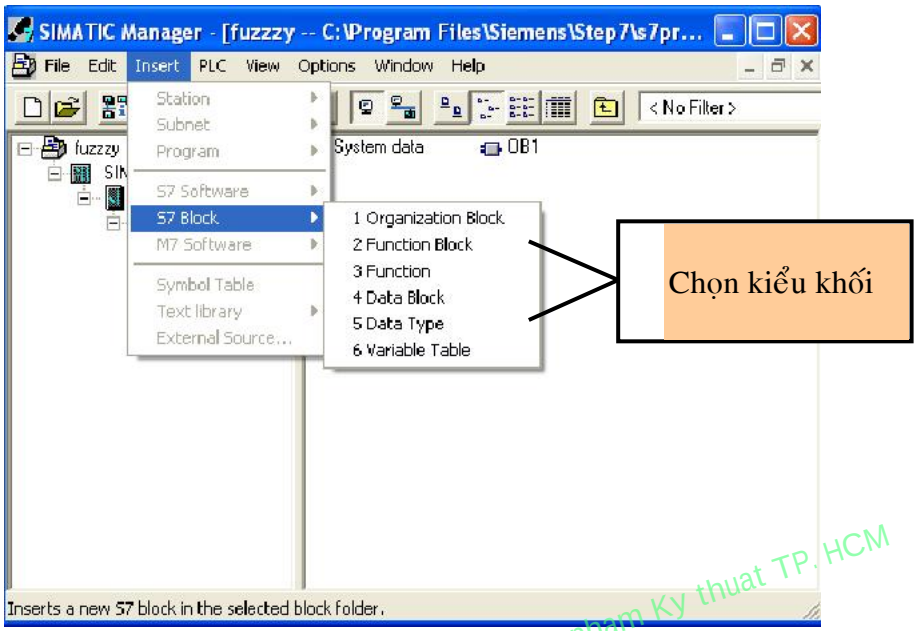
Muốn soạn thảo chương trình cho khối OB1 ta doud-click vào biểu tượng OB1 bên nửa cửa sổ bên phải



Hình 4.14: Lập trình trên khối OB1

Để khai báo và soạn thảo chương trình cho các khối OB khác hoặc cho các khối FC , FB hay DB, ta có thể tạo một khối mới ngay trực tiếp từ chương trình soạn thảo bằng cách chèn thêm khối mới đó trước từ cửa sổ chính của step7 bằng phím

Insert -> S7 Block -> chọn kiểu khối -> chọn số khối -> nhấn OK



Hình 4.15. Chèn thêm khối điều khiển

CHƯƠNG 5

BỘ HIỆU CHỈNH PID, CÁC HÀM XỬ LÝ TÍN HIỆU TƯƠNG TỰ VÀ ỨNG DỤNG

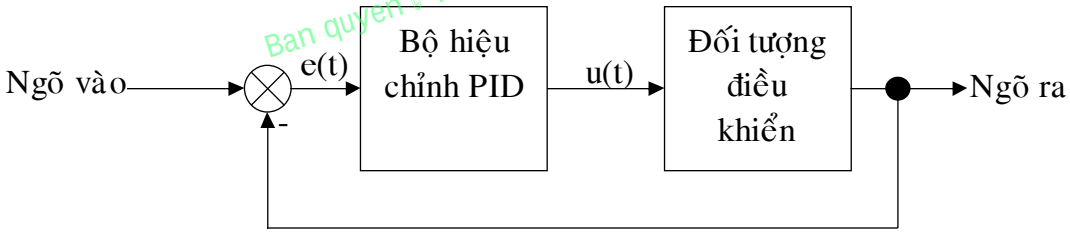
5.1 Giới thiệu.

Nhiều năm trước đây bộ điều khiển PID được coi là bộ điều khiển lý tưởng đối với các đối tượng có mô hình liên tục. Bộ PID thực sự là bộ điều khiển động mà việc thay đổi các tham số của bộ điều khiển có khả năng làm thay đổi đặc tính động và tĩnh của hệ thống điều khiển tự động.

Bộ điều khiển PID thực chất là thiết bị điều khiển thực hiện luật điều khiển được mô tả bằng phương trình sau:

$$u(t) = k_p e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau + T_D e(t)$$

trong đó e(t) là tín hiệu vào, u(t) là tín hiệu ra của bộ điều khiển, k_p là hệ số khuếch đại của luật điều khiển tỷ lệ , T_I hằng số thời gian tích phân và T_D là hằng số thời gian vi phân .



Hình 5.1. Điều khiển với bộ điều khiển PID

Với bộ điều khiển PID, người sử dụng dễ dàng tích hợp các luật điều khiển khác như luật điều khiển tỉ lệ (luật P), điều khiển tỉ lệ - tích phân (luật PI), điều khiển tỉ lệ -vi phân (luật PD). Bộ điều khiển PID luôn là một phần tử không thể thay thế được trong các quá trình tự động khống chế nhiệt độ, mức, tốc độ...

Một trong những ứng dụng của bộ điều khiển PID trong điều khiển thích nghi và điều khiển mờ là thường xuyên phải chỉnh định lại các tham số của nó cho phù hợp với sự thay đổi không biết trước của đối tượng cũng như của môi trường nhằm đảm bảo được các chỉ tiêu chất lượng đã đề ra trong hệ thống. Nếu như ta đã tự động hoá được công việc thay đổi tham số này thì bộ điều khiển PID đó sẽ là một bộ điều khiển bền vững với mọi tác động của nhiễu nội cũng như nhiễu ngoại lên hệ thống.

Cũng chính vì vậy mà các thiết bị điều khiển quá trình như DCS Disbuted Control system, PLC Programmeble Logic Control,PCS Process Control System của các hãng sản xuất thiết bị tự động trên thế giới không thể thiếu được module điều khiển PID hoặc cứng hoặc mềm.

Để sử dụng tốt các module này, người thiết kế phải nắm được các phương pháp chọn luật điều khiển và các tham số cho bộ điều khiển.

Trong phần mềm Step 7 có nhiều khối FB để hổ trợ việc viết chương trình điều khiển thiết bị sử dụng luật hiệu chỉnh PID, như FB40,FB41, FB58, FB59..

5.2 Môđun mềm FB58

5.2.1 Giới thiệu

Sơ đồ khối FB58

Block Diagram

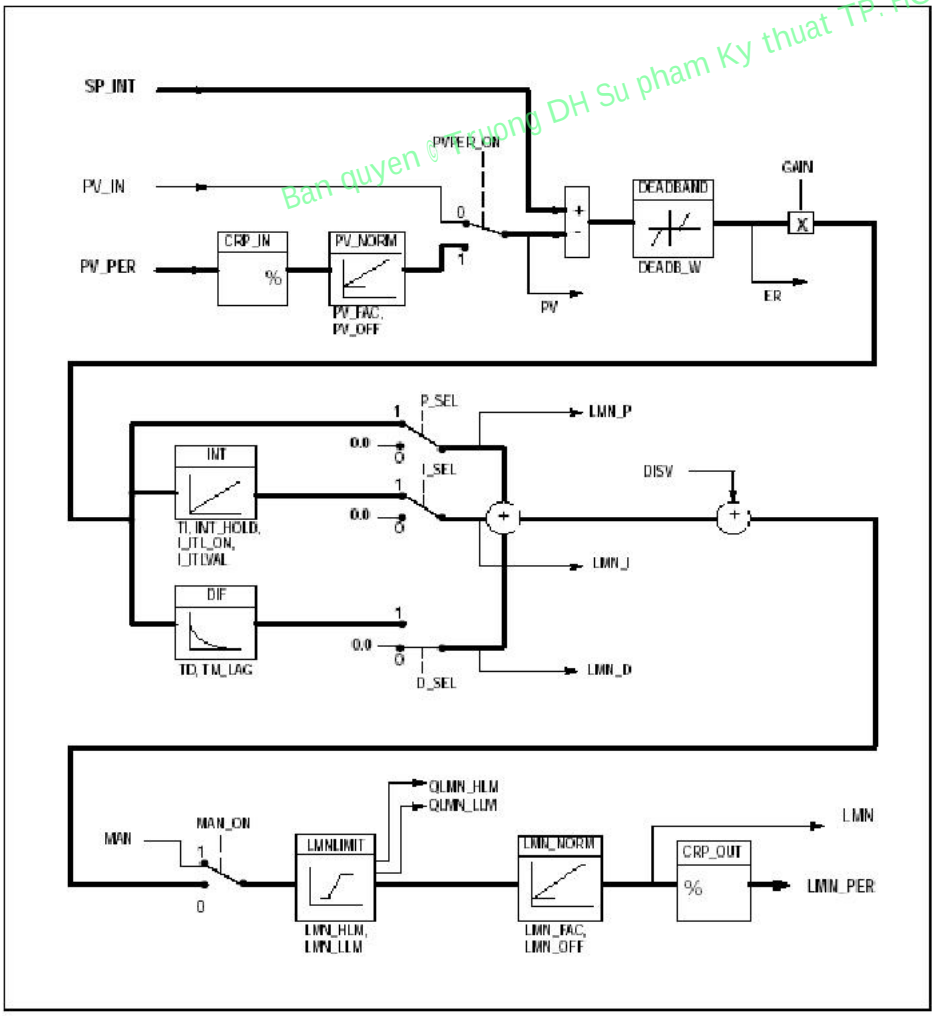


Figure 3-1 Block Diagram of CONT_C

Hình 5.2. Sơ đồ khối của khối FB58

```
CALL  "TCONT_CP" , DB58
 PV_IN    :=
 PV_PER   :=
 DISV     :=
 INT_HPOS:=
 INT_HNEG:=
 SELECT   :=
 PV       :=
 LMN      :=
 LMN_PER  :=
 QPULSE   :=
 QLMN_HLM:=
 QLMN_LLM:=
 QC_ACT   :=
 CYCLE    :=
 CYCLE_P  :=
 SP_INT   :=
 MAN      :=
 COM_RST  :=
 MAN_ON   :=
```


Hình 5.3. Các câu lệnh của FB58

Điểm Setpoint

Điểm Setpoint đặt ở ngõ vào SP- INT ở dạng số thực như một đại lượng vật lí hoặc tỉ lệ phần trăm. Điểm setpoint và giá trị xử lí thường tạo thành sai số phải có cùng một đơn vị

Sự lựa chọn giá trị xử lí (*PVPER_ON)*

Tuỳ thuộc vào PVER ON, giá trị có thể có được từ thiết bị ngoại vi hoặc ở dạng số thực

PVER-ON xử lí giá trị ngõ vào:

- TRUE: Giá trị xử lí được đo thông qua thiết bị ngoại vi Analog (PIWxxx) tại ngõ vào PV PER.
- FALSE: Giá tri xử lí có được ở dạng số thực đặt tại ngõ vào PV-IN.

Sự chuyển đổi giá trị xử lí bằng hàm CRP-IN (PER-MOD)

Hàm CRP_IN chuyển giá trị ngoại vi sang dạng số thực tuỳ thuộc vào sự lựa chọn PER_MODE

PER_MODE	Output of CRP_IN	Analog Input Type	Unit
0	PV_PER * 0.1	Thermoelements; PT100/NI100; standard	°C; °F
1	PV_PER * 0.01	PT100/NI100; climate;	°C; °F
2	PV_PER * 100/27648	Voltage/current	%

Việc tiêu chuẩn hoá giá trị xử lí PV_NORM (PF_FAC, PV_OFFS)

Hàm *PV_NORM* tính toán giá trị ngõ ra của hàm CRP_IN như sau:

"*Output of PV_NORM*" = "*Output of CPR_IN*" * *PV_FAC* + *PV_OFFS*

Nó được dùng với ý định:

PV_FAC:như hệ số của giá trị xử lí.

PV_OFFS:sự offset của giá trị xử lí.

Sự tiêu chuẩn hoá nhiệt độ sang tỉ lệ phần trăm: điểm setpoint ở dạng %, ta phải chuyển giá trị nhiệt độ được đo sang tỉ lệ %.

Sự tiêu chuẩn hoá tỉ lệ % sang nhiệt độ: điểm setpoint ở dạng nhiệt độ ta phải chuyển điện áp/dòng điện.

Việc tính toán các thông số:

- *PV_FAC = range of PV_NORM/range of CRP_IN;*
- *PV_OFFS = LL(PV_NORM) - PV_FAC * LL(CRP_IN);*

Với:

range: dải,vùng,miền

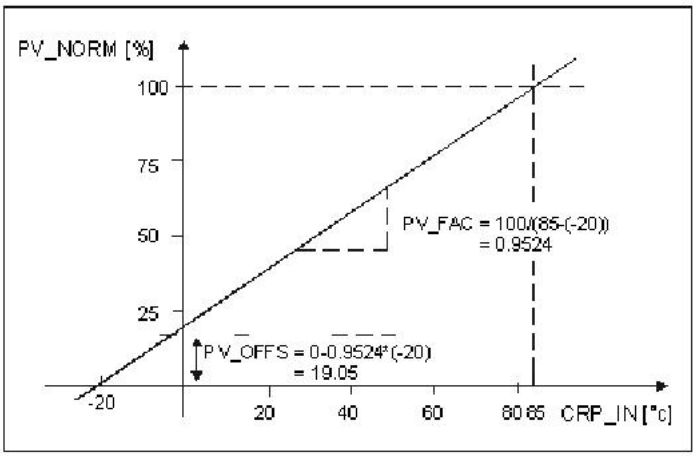
LL : giới hạn dưới

Với giá trị mặc định (PV_FAC = 1.0 và PV_OFFS = 0.0) thì sự tiêu chuẩn hoá sẽ không được thích hợp thì kết quả giá trị xử lí là ngõ ra tại PV.

Ví dụ việc tiêu chuẩn hoá giá trị xử lí

Nếu đặt giá trị setpoint là tỉ lệ % và bạn có miền nhiệt độ là (-20 ÷+85)^{o}C thì bạn phải tiêu chuẩn hoá dải nhiệt độ thành tỉ lệ %.

Sơ đồ dưới đây trình bày một ví dụ về việc chuyển miền nhiệt độ (-20÷+85)^{o}C sang tỉ lệ từ 0 ÷100%



Hình 5.4. Ví dụ việc tiêu chuẩn hoá giá trị xử lí

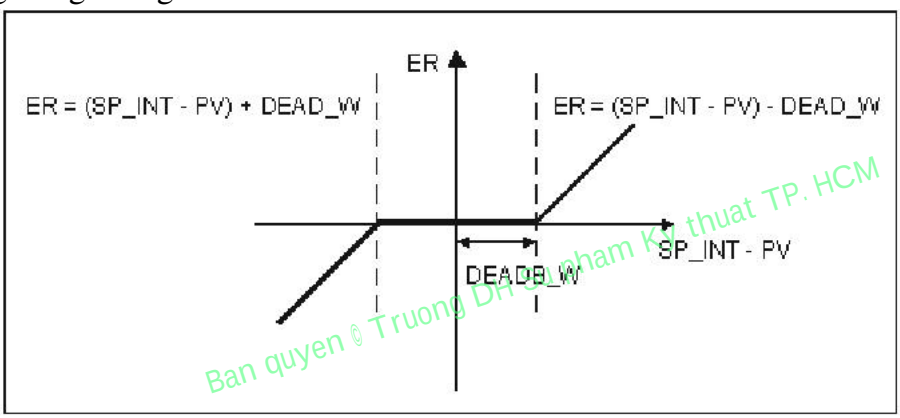
Hình thức sai số

Sự khác biệt giữa điểm setpoint và giá trị xử lí trước khi bị đưa vào miền chết sẽ tạo ra sai số. Điểm setpoint và giá trị xử lí phải có cùng kiểu đơn vị(% hoặc đại lượng vật lí)

Miền chết (Deadb_W)

Với mỗi một giá trị, Deaband sẽ đưa ra một khoảng sai số

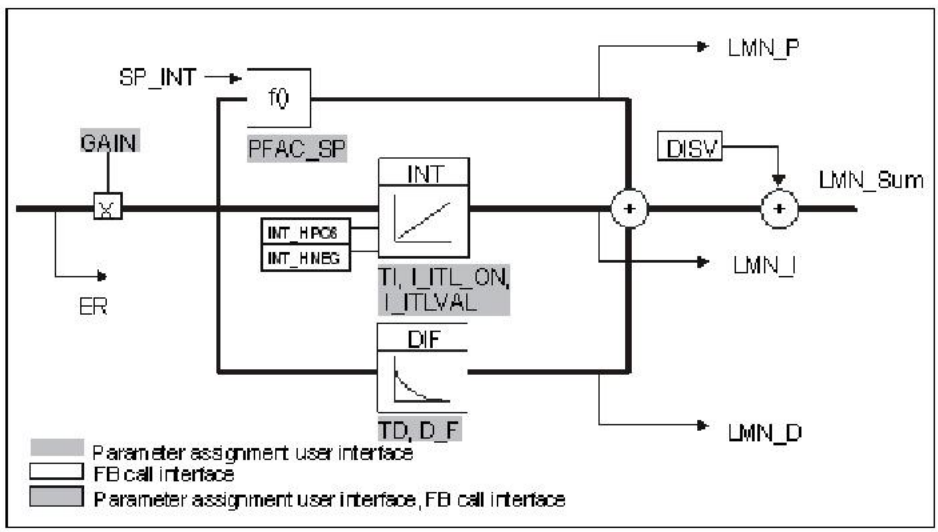
Nếu DEADB_W = 0 thì Deaband sẽ bị giảm sự kích hoạt. Sai số được đặc trưng bằng thông số ER



Hình 5.5. Minh hoạ về miền chết

Thuật toán PID (GAIN, TI, TD, D_F)

Sơ đồ dưới đây là sơ đồ khối của thuật toán PID



Hình 5.6. Sơ đồ khối của thật toán PID

Thuật toán PID hoạt động như một thuật toán về sự định vị. Các cơ cấu tỉ lệ,tích phân,vi phân(DIF) được kết nối song song và có thể được kích hoạt hoặc không kích hoạt một cách riêng lẻ. Điều này cho phép bộ P, bộ PI, bộ PID được cấu hình.

Sự điều chỉnh của người điều khiển sẽ hỗ trợ bộ PI,PID. Bộ điều khiển nghịch được thi hành khi sử dụng một bộ GAIN (*cooling controller*).

Nếu set giá trị TI,TD đến giá trị 0.0 thì bạn sẽ thu được một bộ điều khiển P tại điểm hoạt động

$$LMN\_Sum(t) = GAIN * ER(0)(1 + \frac{1}{TI} * t + D\_F * e^{\frac{-t}{TD/D\_F}})$$

Từng bước đáp ứng trong miền thời gian là:

LMN_Sum(t):là biến tổng trong chế độ tự động của bộ điều khiển

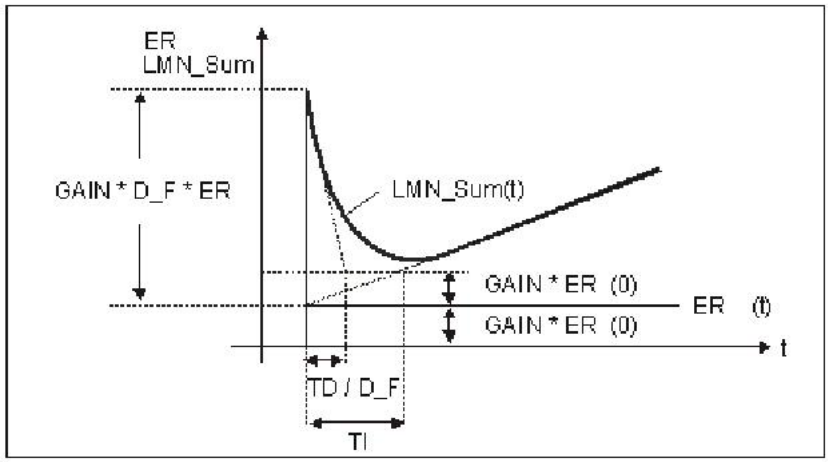
ER (0) : là sự thay đổi từng bước của sai số đã được chuẩn hoá

GAIN : độ lợi của bộ điều khiển

TI : thời gian tích phân

TD :thời gian vi phân

D_ :hệ số vi phân



Hình 5.7. Minh hoạ các thông số của thuật toán PID

Bộ tích phân (TI, I_ITL_ON, I_ITLVAL)

Trong điều khiển tay nó được điều chỉnh như sau:

LMN_I = LMN - LMN_P - DISV.

Nếu biến vận hành *(manipulated variable)* bị giới hạn thì cơ cấu I sẽ bị ngưng hoạt động. Nếu sai số đưa cơ cấu I trở về trong khoảng cho phép của biến

vận hành thì cơ cấu I có thể được thay đổi bằng cách: cơ cấu I của bộ điều khiển có thể được kích hoạt bởi TI = 0.

Sự hoạt động của cơ cấu P sẽ bị yếu đi khi có sự thay đổi của điểm setpoint.

Sự hoạt động của cơ cấu P sẽ bị yếu đi khi có sự thay đổi của điểm setpoint (PFAC_SP)

Để ngăn chặn sự quá tầm(vượt quá giới hạn),bạn có thể làm giảm sự hoạt động của cơ cấu P bằng cách sử dụng hệ số tỉ lệ đối với sự thay đổi của điểm setpoint, thông số PFAC_SP. Sử dụng PFAC_SP, bạn có thể chọn các giá trị liên tiếp từ 0.0 ÷1.0 để quyết định hiệu quả của cơ cấu P khi điểm setpoint thay đổi.

- PFAC_SP=1.0: cơ cấu P bị ảnh hưởng nếu điểm setpoint bị thay đổi.
- PFAC_SP=0.0: cơ cấu P hoàn toàn không bị thay đổi nếu điểm setpoint bị thay đổi.

Việc giảm ảnh hưởng của cơ cấu P sẽ đạt được bằng sự cân bằng thêm cơ cấu I

Các nhân tố của cơ cấu vi phân (TD, D_F)

Cơ cấu D của bộ điều khiển sẽ bị giảm kích hoạt với TD = 0.

Nếu cơ cấu D được kích hoạt ,theo mối quan hệ giữa các thông số ta có:

*TD = 0.5 * CYCLE * D_F*

Cài đặt thông số của bộ điều khiển P và PD với điểm đang hoạt động

Trong giao diện của người sử dụng, cơ cấu I không được kích hoạt (TI = 0.0) và cơ cấu (TD = 0.0) cũng vậy. Sau đó cài đặt các thông số:

I_ITL_ON = TRUE

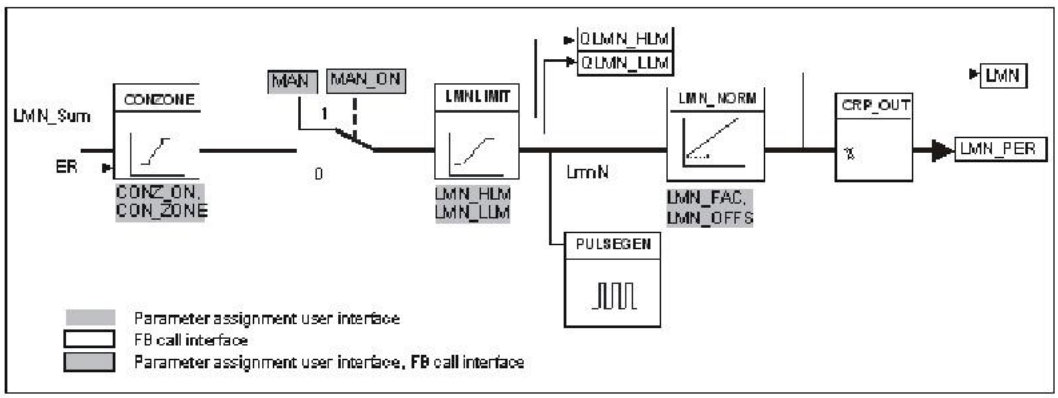
I_ITLVAL = điểm đang hoạt động.

Điều khiển hồi tiếp (DISV)

Biến hồi tiếp có thể được thêm vào trong ngõ vào DISV

Việc tính toán biến vận hành

Sơ đồ khối dưới đây là sơ đồ về sự tính toán biến vận hành



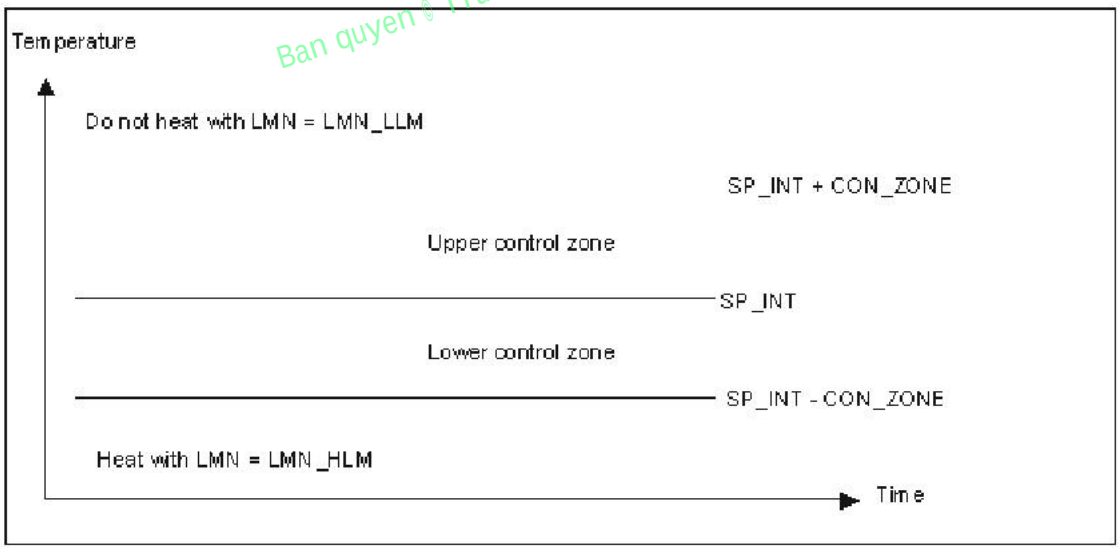
Hình 5.8. Sơ đồ khối tính toán biến vận hành

Vùng điều khiển (CONZ_ON, CON_ZONE)

Nếu CONZ_ON = TRUE, bộ điều khiển sẽ hoạt động trong vùng điều khiển. Điều này có nghĩa là bộ điều khiển hoạt động theo thuật toán sau:

- Nếu PV vượt quá giá trị SP_INT gần cận trên của CON_ZONE, thì giá trị LMN_LLM là giá trị ngõ ra như biến vận hành (điều khiển theo chu trình kín)
- Nếu PV nằm dưới giá trị SP_INT gần cận dưới của CON_ZONE,thì giá trị LMN_HLM là giá trị ngõ ra như biến vận hành(điều khiển theo chu trình kín)
- Nếu PV nằm trong khoảng vùng điều khiển (CON_ZONE) biến vận hành sẽ giữ giá trị của nó lại từ LMN_Sum của thuật toán.

Chú ý: Sự chuyển đổi từ điều khiển kín sang điều khiển tự động theo chu trình kín thì sự điều khiển sẽ tính toán 1 khoảng trễ khoảng 20% của vùng điều khiển.



Hình 5.9. Sơ đồ vùng điều khiển (CONZ_ON, CON_ZONE)

Trước khi kích hoạt vùng điều khiển tay, phải chắc chắn rằng vùng điều khiển không quá hẹp, nếu vùng điều khiển quá nhỏ, sự dao động sẽ xuất hiện trong biến vận hành và biến xử lí.

Thuận lợi của vùng điều khiển

Khi giá trị xử lí thuộc vùng điều khiển , cơ cấu D tạo ra một sự giảm cực nhanh của biến vận hành. Điều này có nghĩa là vùng điều khiển chỉ hữu ích khi cơ cấu D được kích hoạt. Không có vùng điều khiển, về cơ bản cơ cấu P sẽ làm giảm biến vận hành. Vùng điều khiển sẽ có tác động tới sự ổn định nhanh hơn mà không có sự vượt quá giới hạn hoặc sai lệch dưới(thấp hơn trị số danh nghĩa).

Xử lí giá trị bằng tay (MAN_ON, MAN)

Bạn có thể bật công tắc điều khiển tay hoặc tự động . Trong điều khiển tay biến vận hành được điều chỉnh đến một giá trị theo hướng dẫn .

Cơ cấu tích phân (INT) được set đến giá trị *LMN - LMN_P – DISV* và cơ cấu vi phân (DIF) được set đến giá tri 0 và được đồng bộ hoá bên trong. Do đó chuyển sang chế độ tự động sẽ ít bị va chạm hơn.

Chú ý: trong khi điều khiển thông số MAN_ON không có ảnh hưởng gì.

Sự giới hạn của biến vận hành LMNLIMIT (LMN_HLM, LMN_LLM)

Giá trị của biến vận hành được giới hạn đến 2 giá trị giới hạn LMN_HLM và LMN_LLM bởi hàm LMNLIMIT. Nếu sự giới hạn này đạt được, điều này được chỉ định bởi bit thông tin QLMN_HLM và QLMN_LLM. Nếu biến vận hành bị giới hạn thì cơ cấu sẽ bị ngưng hoạt động. Nếu sai số đưa cơ cấu I về đúng vùng biến vận hành thì cơ cấu I sẽ được phục hồi.

Tay đổi sự giới hạn của biến kết quả

Nếu miền biến vận hành bị giảm và giá trị mới không được giới hạn của biến vận hành nằm ngoài khoảng giới hạn, thì cơ cấu I và giá trị của biến vận hành sẽ bị thay đổi.

Việc tiêu chuẩn hoá biến vận hành (LMN_FAC, LMN_OFFS)

Hàm LMN_NORM chuẩn hoá biến vận hành theo công thức sau:

*LMN = LmnN * LMN_FAC + LMN_OFFS*

Nó được dùng với ý định:

LMN_FAC: như hệ số của giá trị xử lí.

LMN_OFFS: sự offset của giá trị xử lí.

Giá trị biến vận hành cũng có khả năng được định dạng từ bên ngoài.

Hàm CRP_OUT chuyển số thực sang giá trị ngoại vi theo công thức sau:

LMN_PER = LMN * 27648/100

Với giá trị mặc định (LMN_FAC = 1.0 và LMN_OFFS = 0.0) thì sự chuẩn hoá sẽ không được thích hợp. Lúc này kết quả của biến vận hành là ngõ ra tại LMN.

Ghi nhận và chuyển tải các thông số của bộ điều khiển

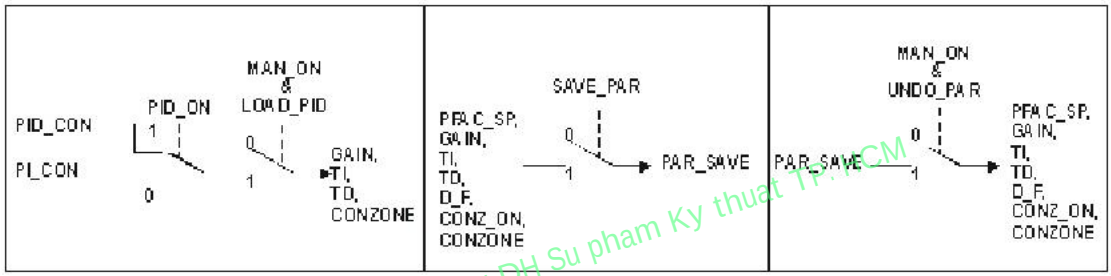
- Việc ghi nhận các thông số của bộ điều khiển SAVE_PAR

Nếu việc cài đặt các thông số hiện hành được dùng, bạn có thể ghi nhận chúng vào một cấu trúc đặc biệt trong hàm *FB 58"TCONT_CP*" trước khi tạo ra một sự thay đổi .Nếu bạn điều chỉnh bộ điều khiển, việc các thông số ghi nhận được viết đè lên thay giá trị trước khi chuyển đổi.

PFAC_SP, GAIN, TI, TD, D_F, CONZ_ON và CONZONE được chuyển sang cấu trúc PAR_SAVE.

- Việc tải các thông số đã được ghi nhận của bộ điều khiển UNDO_PAR

Hàm này được sử dụng để kích hoạt thông số được cài đặt cuối cùng của bộ điều khiển mà bạn đã ghi nhận để phục hồi bộ điều khiển (chỉ trong điều khiển tay)



Hình 5.10. Sơ đồ khối của việc ghi nhận và chuyển tải các thông số của bộ điều khiển

Việc chuyển đổi các thông số giữa bộ PI và PID LOAD_PID (PID_ON)

Theo quá trình điều chỉnh các thông số PID và PI sẽ được lưu vào trong cấu trúc PI_CON và PID_CON. Tuỳ vào PID_ON,bạn có thể sử dụng LOAD_PID trong điều khiển tay đối với các thông số PI hoặc PID để tạo ra các thông số của bộ điều khiển

PID parameter PID_ON = TRUE		PI parameter PID_ON = FALSE	
• GAIN	= PID_CON.GAIN	• GAIN	= PI_CON.GAIN
• TI	= PID_CON.TI	• TI	= PI_CON.TI
• TD	= PID_CON.TD		

Chu ý:

Các thông số của bộ điều khiển chỉ được ghi trở lại vào bộ điều khiển với UNDO_PAR hoặc LOAD_PID khi độ lợi của bộ điều khiển khác 0.

LOAD_PID copy các thông số nếu độ lợi GAIN <> 0 (các thông số của một trong hai bộ PI và PID)

D_F, PFAC_SP có thể được set đến giá trị mặc định bằng cách điều chỉnh. Những giá trị này sau đó có thể được xác định bởi người điều chỉnh. LOAD_PID không thay đổi các thông số này.

Với LOAD_PID vùng điều khiển luôn được tính toán lại.

(CON_ZONE = 250/GAIN) ngay khi CONZ_ON = FALSE được set.

Quá trình chạy (Tuning) của FB 58 "TCONT_CP"

❖ **Giới thiệu**

Với việc điều khiển việc chạy tự điều chỉnh của "TCONT_CP", bộ điều khiển PI/PID cập nhật tự động các thông số của bộ điều khiển. Có hai phương thức chạy Tuning:

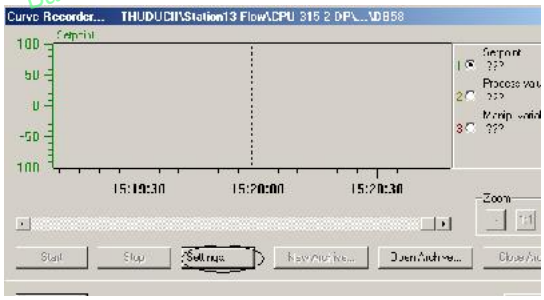
- Chạy tự điều chỉnh bằng sự tiến gần tới điểm hoạt động với sự thay đổi từng bước của điểm đặt.
- Chạy tự điều chỉnh điểm hoạt động bằng việc đặt một bit bắt đầu.

Cả hai cách trên, quá trình xử lí được kích thích bởi có thể lựa chọn biến gán thay đổi. Sau khi chỉ ra điểm uốn, các thông số bộ điều khiển có giá trị và bộ điều khiển được khởi hoạt động tự động và tiếp tục điều khiển với những thông số này.

Bạn có thể điều khiển việc chạy điều chỉnh bằng việc sử dụng các thông số được thiết lập trên giao diện chương trình thiết kế.

Các bước tiến hành:

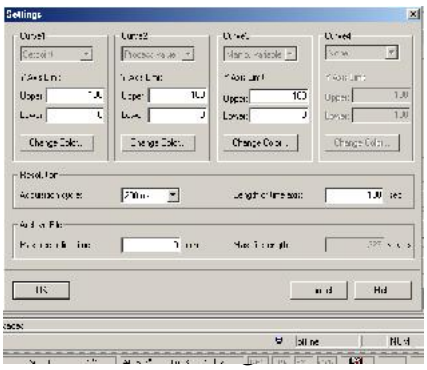
- *Start -> Simatic Manager -> Project -> Block -> DB58 -> Option -> Curve Recorder*



Hình 5.11. Màn hình cập nhật đồ thị

- Thiết lập các thông số vào sau khi vào *Curve Recorder* nhấp vào *Setting*

(cài đặt thông số)



Hình 5.12. Màn hình cài đặt thông số

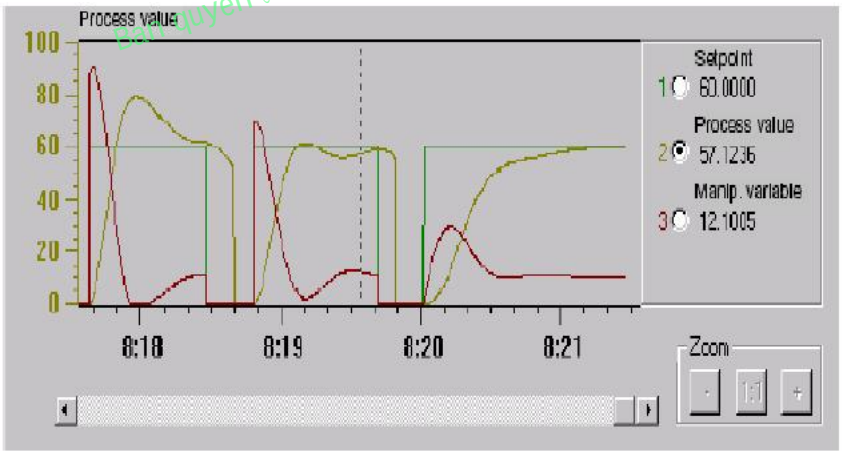
- Sau khi cài đặt thông số xong ta vào *Data Block* -> *Open Oline* -> *Option* *Controller Tuning*

❖ **Kết quả**

Các thông số điều khiển được cập nhật trong quá trình FB58 "TCONT_CP" chạy tự điều chỉnh.

Nhân tố để làm giảm thông số tác động P là PFAC_SP = 0.8

- Độ lợi của bộ điều khiển GAIN
- Thời gian bộ tích phân: TI
- Thời gian bộ vi phân: TD
- Nhân tố bộ vi phân: D_F = 5.0
- Vùng điều khiển Control zone on/off: CONZ_ON
- Độ rộng vùng Control zone: CON_ZONE



Hình 5.13. Đồ thị kết quả của quá trình cập nhật thông số

Ví dụ về làm giảm đáp ứng điều khiển với PFAC_SP

Thông số quá trình:

GAIN = 6

T1 = 50s

T2 = 5s

Thông số bộ điều khiển:

GAIN = 6
TI = 19.6s

Bảng 5.2: Chú thích kết quả của đồ thị

Thời gian thử	Hệ số SP	Chú thích	Độ vọt lố
8:18	1.0	Cơ cấu P không có sự hồi tiếp	32%
8:19	0.8	Cơ cấu P có sự hồi tiếp 20%, sự đáp ứng điều khiển ở điều kiện tốt nhất	2%
8:20	0.0	Cơ cấu P hồi tiếp hoàn toàn, sự chấn động giảm mạnh.	-

5.2.2 Các thông số của FB58

ĐỊA CHỈ	THÔNG SỐ	IN/OUT	KIỂU DỮ LIỆU	VÙNG GT	GT ĐẦU	MÔ TẢ
0.0	PV_IN	INPUT	REAL	Tuỳ cảm biến sử dụng	0.0	PROCESS VARIABLE IN. Giá trị khởi tạo có thể đặt ở đầu vào"*process variable on*" hoặc từ biến quá trình được biểu diễn dưới dạng số thực dấu phảy động
4.0	PV_PER	INPUT	INT		0	PROCESS VARIABLE PERIPHERY Biến quá trình được nối với CPU thông qua cổng vào tương tự.
6.0	DISV	INPUT	REAL		0.0	DISTURBANCE VARIABLE Đối với điều khiển thuận biến nhiễu sẽ

						được nối ở ngõ vào disturbance variable
10.0	INT_HPOS	INPUT	BOOL		FALSE	INTEGRAL ACTION HOLD IN POSITIVE DIRECTION . Ngõ ra của cơ cấu I có thể được chỉ ra trong một phạm vi trực tiếp. Để đạt được điều này, ngõvào INT_HPOS phải được set đến giá trị TRUE . Trong từng đợt điều khiển, INT_HPOS của bộ điều khiển đầu tiên được kết nối đến ngõ QLMN_HLM của bộ điều khiển thứ 2
10.1	INT_HNEG	INPUT	BOOL		FALSE	INTEGRAL ACTION HOLD IN NEGATIVE DIRECTION. Ngõ ra của cơ cấu I có thể được chỉ ra trong một hướng phủ định. Để đạt được điều này ,ngõ vào INT_HPOS phải được set đến giá trị TRUE . Trong từng đợt điều khiển, INT_HPOS của bộ điều khiển đầu tiên được kết nối đến ngõ QLMN_LLM của bộ điều khiển thứ 2
12.0	SELECT	INPUT	INT	0ĐẾN 3	0	SELECTION OF CALL PID AND PULSE GENERATOR.

						Nếu máy phát sung được kích hoạt, có một vài cách để gọi thuật toán PID và bộ phát sung. SELECT =0: Bộ điều khiển sẽ được gọi trong một chu kì nhanh ở mức độ gián đoạn, thuật toá PID và bộ phát sung sẽ được xử lí. SELECT =1:Bộ điều khiển sẽ được gọi trong khối OB1 và chỉ thuật toán PID được xử lí. SELECT = 2: bộ điều khiển sẽ được gọi trong một chu kì nhanh ở mức độ gián đoạn và chỉ bộ phát sung được xử lí. SELECT =3: Bộ điều khiển sẽ được gọi trong một chu kì chậm ở mức độ gián đoạn và chỉ thuật toán PID được xử lí.
14.0	PV	OUTPUT	REAL	Tuỳ cảm biến sử dụng	0.0	PROCESS VARIABLE Tín hiệu quá trình được xuất qua cổng ra “process variable”
18.0	LMN	OUTPUT	REAL		0.0	MANIPULATED VARIABLE Giá trị ra được thiết lập bằng tay thông qua cổng ra” manipulated variable”
22.0	LMN_PER	OUTPU	INT		0	MANIPULATED

		T				VARIABLE PERIPHERY. Giá trị đầu ra thiết lập bằng tay theo kiểu biểu diễn phù hợp với các cổng vào/ra tương tự được chọn qua ngõ ra"*manipulated variable periphery*"
24.0	QPULSE	OUTPUT	BOOL		FALSE	OUTPUT PULSE SIGNAL. Khi có giá trị xung được kích hoạt thì sẽ có tín hiệu ra tại ngõ ra "QPULSE"
24.1	QLMN HLM	OUTPUT	BOOL		FALSE	HIGH LIMIT OF MANIPULATED VARIABLE REACHED. Cổng ra" *hight limit of manipulated variable reached* " thông báo giá trị của biến quá trình vượt quá giá trị giới hạn
24.2	QLMN_LLM	OUTPUT	BOOL		FALSE	LOW LIMIT OF MANIPULATED VARIABLE REACHED Cổng ra" low limit of manipulated variable reached " thông báo giá tị của biến quá trình nhỏ hơn giá trị giới hạn
24.3	QC_ACT	OUTPUT	BOOL		TRUE	NEXT CYCLE, THE CONTINUOUS CONTROLLER IS

						WORKING. Thông số này sẽ được chỉ rõ khi có hay không trạng thái điều khiển tiếp theo sẽ được thi hành tại lần gọi tiếp theo (chỉ thích hợp khi SELECT có giá trị 1 hoặc 0)
26.0	CYCLE	INPUT/ OUTPUT	REAL	≥0.001 s	0.1s	SAMPLE TIME OF CONTINUOUS CONTROLLER[s]. Tại đây sẽ set thời gian mẫu đối với thuật toán PID. Bộ điều chỉnh trong pha thứ nhất và đưa vào CYCLE. Thời gian lấy mẫu là khoảng thời gian không đổi giữa các lần khối được cập nhật.
30.0	CYCLE_P	INPUT/ OUTPUT	REAL	≥0.001 s	0.02s	SAMPLE TIME OF PULSE GENERATOR[s]. Tại ngõ vào này, bạn đưa vào thời gian mẫu đối với máy phát xung. FB 58 “TCONT_CP” tính toán thời gian mẫu trong pha thứ nhất và đưa nó vào trong CYCLE_P.
34.0	SP_INT	INPUT/ OUTPUT	REAL	vùng giá trị của giá trị xử lí	0.0	INTERNAL SETPOINT Đầu vào “internal setpoint” được sử dụng để thiết lập tín hiệu chủ

						đạo.
38.0	MAN	INPUT/ OUTPUT	REAL		0.0	MANUAL VALUE. Cổng vào" manual value" được sử dụng để đặt giá trị bằng các hàm giao diện
42.0	COM_RST	INPUT/ OUTPUT	BOOL		FALSE	COMPLETE RESTART Khối có chức năng khởi tạo lại hệ thống hoàn toàn khi đầu vào "*complete restart*" được thiết lập giá trị logic TRUE
42.1	MAN_ON	INPUT/ OUTPUT	BOOL		TRUE	MANUAL OPERATION ON. Khi đầu vào "*manual operation on*" có giá trị logic TRUE mạch vòng điều khiển sẽ bị ngắt, các giá trị sẽ được thiết lập bằng tay.
90.0	PVPER_ON	INPUT	BOOL		FALSE	PROCES VARIABLE PERIPHERY ON Nếu bạn muốn giá trị xử lí được đọc thông qua thiết bị I/O, ngõ vào PV_PER phải được kết nối đến I/O và giá trị xử lí ngoại vi phải được set lên (TRUE).
186.5	LOAD_PID	INPUT/ OUTPUT	BOOL		FALSE	LOAD OPTIMIZED PI/PID PARAMETERS
186.6	PID_ON	INPUT/ OUTPUT	BOOL		FALSE	PID MODE ON PID controller: PID_ON = TRUE

						PI controller: PID _ON = FALSE

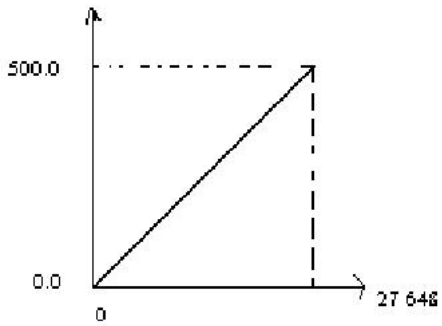
5.3 HÀM FC105, FC106

Để thuật lợi trong quá trình xử lý các tín hiệu tương tự phần mềm Step 7 có sẵn hàm thư viện FC105,FC106

5.3.1 Hàm FC105_Định tỉ lệ gía trị ngõ vào Analog

```
CALL  "SCALE"
 IN      :=PIW272
 HI_LIM :=5.000000e+002
 LO_LIM :=0.000000e+000
 BIPOLAR:=M0.0
 RET_VAL:=MW102
 OUT     :=MD104
```

Hình 5.14. Các câu lệnh của hàm FC105



- Ví dụ: Mức đầy trong bồn được đo bằng lít .Bộ chuyển đổi đo được chọn 500 lít thì tương ứng với một giá trị đo là 10V.
- Tỉ lệ: Module Analog chuyển đổi giá trị analog 10V thành số nguyên 27 648. Giá trị này bây giờ đã được chuyển đổi thành đại lượng vật lí lít. Quá trình này người ta gọi là định tỉ lệ giá trị Analog.
- Chương trình:Việc định tỉ lệ giá trị analog được thực hiệ trong khối chuẩn FC 105. Khối FC 105 nằm trong thư viện "*Standard Library*"trong chương trình S7"*TI-S7 Converting Block*" của phần mềm Step 7.
- IN : Giá trị Analog tại ngõ vào IN có thể được đọc trực tiếp từ module hoặc đọc qua ngõ giao tiếp dữ liệu trong dạng INTEGR
- LO_LIM,HI_LIM: Các giới hạn chuyển đổi các đại lương vật lí sẽ được đặt trước ở các ngõ vào LO_LIM (giới hạn dưới) và HI_LIM (giới hạn trên). Trong thí dụ trên thì giới hạn chuyển đổi từ 0 đến 500 lít.

- OUT: Giá trị tỉ lệ (đại lượng vật lí) thì được lưu trữ như là một số thực tại ngõ

ra OUT.

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

Hằng số K_1, K_2 sẽ được set dựa trên giá trị ngõ vào là BIPPOLAR hay UNBIPPOLAR

- BIPPOLAR: Ngõ vào BIPPOLAR xác định liệu giá trị âm có được chuyển đổi

hay không.

BIPPOLAR: Giá trị ngõ vào là số nguyên được thừa nhận giữa -27648 và +27648,do đó K_1 là -27648, K_2 là +27648.

UNBIPPOLAR: Giá trị ngõ vào là số nguyên được thừa nhận giữa 0 và +27648,do đó K_1 là0, K_2 là +27648.

Trong ví dụ trên, bit nhớ 0.0 có tín hiệu "0" và vì thế báo hiệu giá trị ngõ vào là một cực.

- RET_VAL: Ngõ ra RET_VAL có giá trị 0 nếu sự hoạt động không có sự cố.

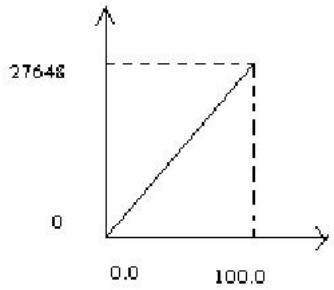
Nếu giá trị tại ngõ vào lớn hơn K_2 , ngõ ra OUT được kiểm soát bởi HI_LIM và lỗi xuất hiện, nếu giá trị tại ngõ vào nhỏ hơn K_1 , ngõ ra OUT được kiểm soát bởi LO_LIM và lỗi xuất hiện. Lúc này ENO sẽ được set giá trị 0, và RET_VALsẽ có giá trị W#16#0008.

5.3.2 Hàm FC106_Không chia tỉ lệ số thực cho ngõ ra Analog

```
CALL  "UNSCALE"
 IN      :=MD34
 HI_LIM  :=1.000000e+002
 LO_LIM  :=0.000000e+000
 BIPOLAR:=FALSE
 RET_VAL:=MW104
 OUT     :=MW30
```

Hình 5.15. Các câu lệnh của hàm FC106

Ví dụ: Chương trình tính toán giá trị Analog trong phạm vi từ 0 đến 100.0%. Giá trị này được chuyển tới ngõ ra nhờ một module ngõ ra Analog.



- Không chia tỉ lệ: Khối chuẩn FC106 được sử dụng cho việc không chia tỉ lệ (sự biến đổi của một số thực từ 0 đến 100.0% thành một số nguyên 16 bits từ 0 đến 27648)
- OUT:Giá trị Analog không chia tỉ lệ tại ngõ ra out có thể được tryền đi dưới dạng một số nguyên 16 bits đến ngõ giao tiếp dữ liệu hoặc trực tiếp đến ngoại vi.

Chương trình: Khối FC106 nằm trong thư viện "*Standard Library*" trong chương trình S7 "*TI-S7 Converting Block*" của phần mềm Step7.

5.4 Ví dụ ứng dụng điều khiển mức nước trong bồn

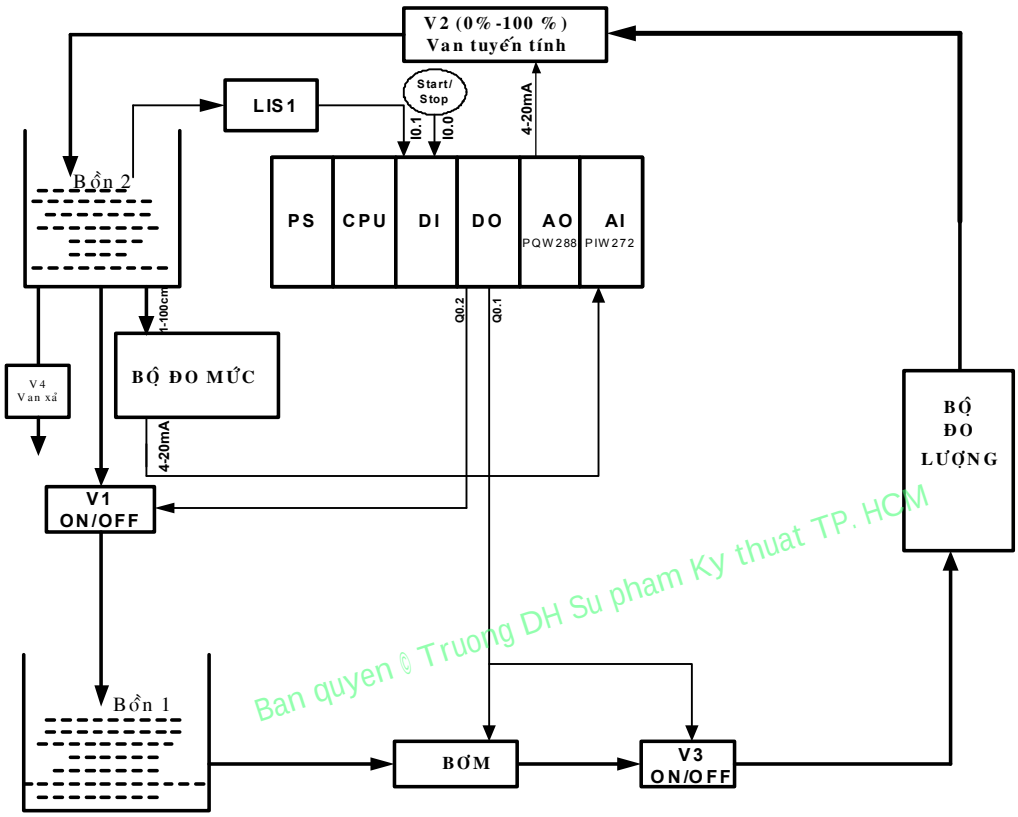
5.4.1 Nguyên Lý hoạt động :

Nước được bơm từ bình chứa 1 lên bình 2 bằng bơm ly tâm ,bơm ly tâm hoạt động theo hai chế độ có thể điều khiển bằng tay hoặc điều khiển từ PLC bằng núm chuyển hai chế độ AUTO và MAN, bơm hoạt động ở hiệu điện thế xoay chiều 220-240 VAC ,công suất là 30 W, lưu lượng nước khoảng 20 lít/phút, nó được tác động bởi 2 rơle(250 VAC/5A). Dòng nước qua bơm chia làm 2 phần, 1 lượng sẽ qua van solenoid và 1 lượng được hồi về bể 1 để bảo đảm an toàn cho bơm khi van solenoid đóng lại. Van solenoid cũng có thể được điều khiển bằng tay hoặc bằng PLC dùng đề đóng mở tức thời dòng nước lên bồn chứa 1 khi điều khiển. Áp suất hoạt động từ 0-10 bar, cấp dòng 24 VDC, dòng có thể được cấp từ rơle hoặc trực tiếp từ PLC.

Dòng tiếp tục qua bộ phận hiển thị lưu lượng dòng cơ khí dùng để điều chỉnh lưu lượng sao cho vừa phải ổn định giữa 2 luồng nước phân nhánh từ bơm. Khi qua bộ chuyển đổi lưu lượng bộ phận này sẽ hiển thị giá trị lưu lượng dòng chảy chính xác và truyền thông số này về cho PLC xử lý dưới dạng dòng điện từ 0/4 – 20 mA tuỳ theo giá trị lớn nhỏ của lưu lượng nước.

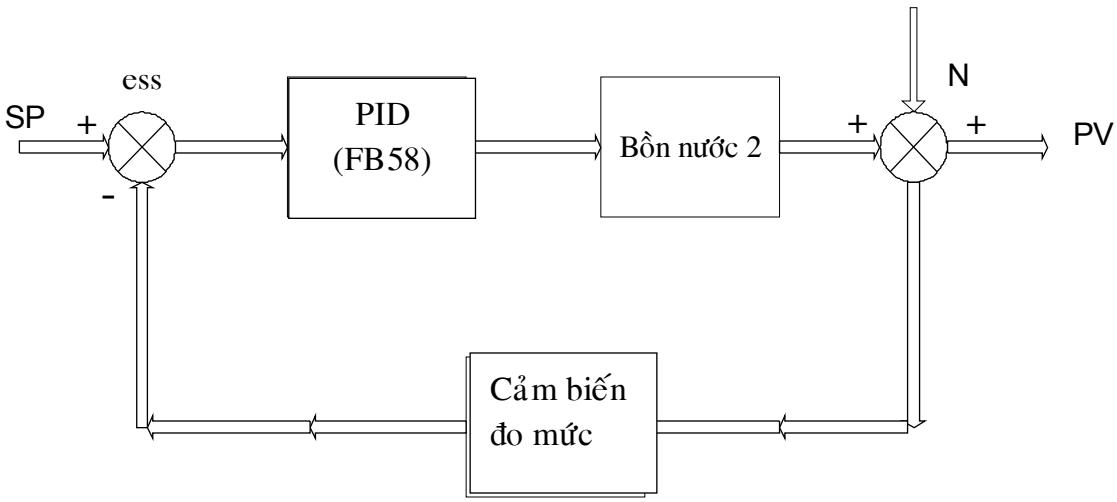
PLC nhận 2 ngõ vào analog là bộ chuyển đổi lưu lượng và cảm biến, tuỳ thuộc vào chương trình phần mềm điều khiển được viết sẵn mà PLC sẽ điều khiển proportional

vale(vale điều khiển tỷ lệ) để đóng mở góp mở của van . Do đó lưu lượng nước sẽ được điều khiển 1 cách tỷ lệ với dòng mà PLC xuất ra từ ngõ ra AO.



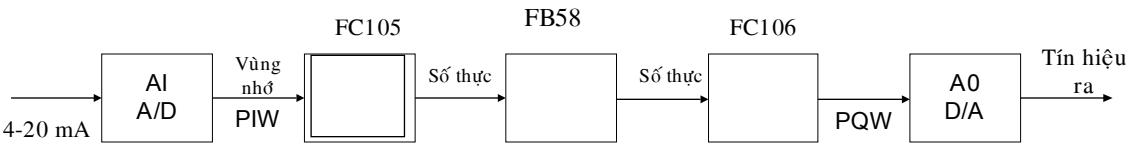
Hình 5.16. Sơ đồ khối điều khiển mức nước.

5.4.2 Sơ đồ khối của hệ thống tự động

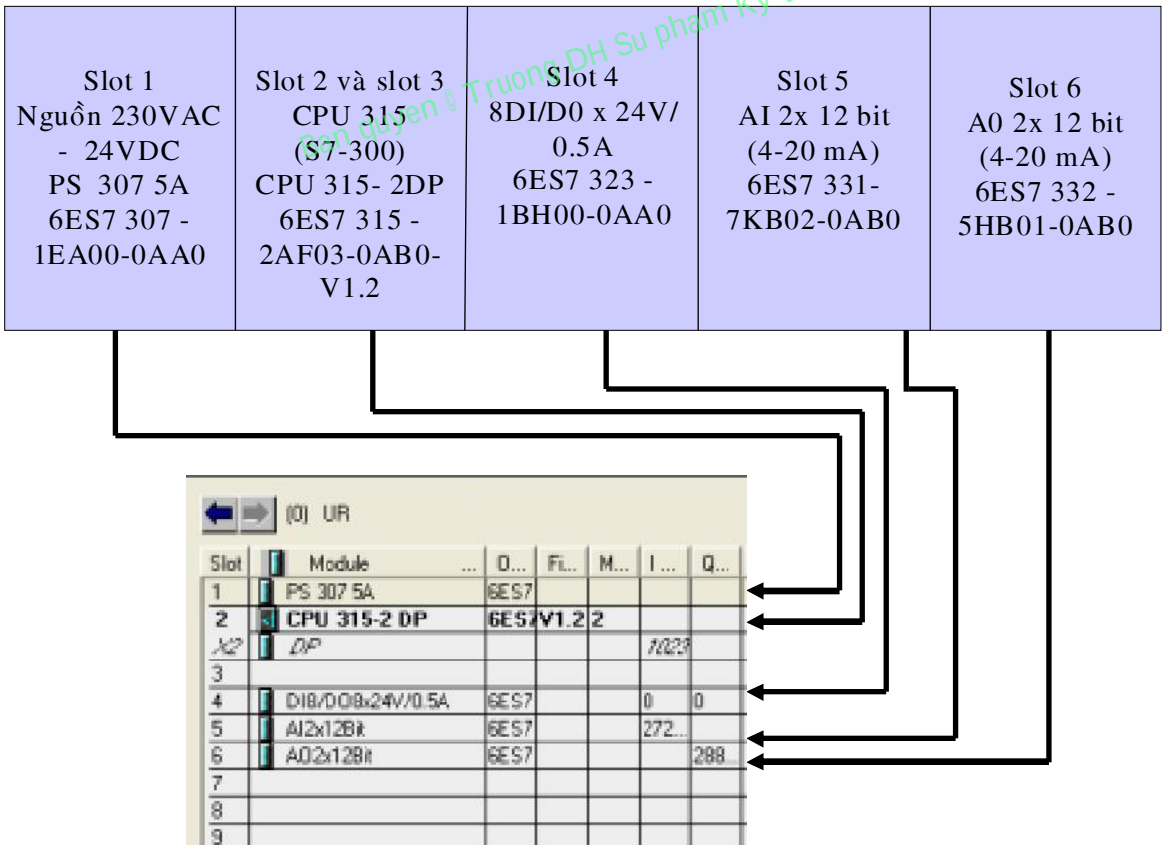


Hình 5.17. Sơ đồ khối của hệ thống tự động

PID control: được điều khiển thông qua cục PLC
Bồn nước 2: là đối tượng điều khiển
Cảm biến đo : cảm biến đo mức nước dựa vào áp lực đè lên bề mặt từ 0 – 0.1 bar để đưa ra dòng và áp tương ứng từ 4-20mA
Tín hiệu nhiễu e_{ss} do tác động của van xả
Trong đó : SP : là giá trị tự nhập vào (được nhập vào từ bên ngoài).
PV : là giá trị hiện tại có trong bồn nước (giá trị được đọc từ vùng nhớ vào analog)
Ta lấy giá trị hiện tại PV so với giá trị đưa vào SP ta được e_{ss} = SP-PV
Sơ đồ điều khiển



5.4.3. Khai Báo Thông Số Phần Cứng:



Hình 5.18. Khai báo phần cứng cho trạm PLC

Bảng địa chỉ vào ra

Địa Chỉ	Tên thiết bị
I0.0	S1 Nút Start/Stop
I0.1	LIS1 Cảm biến phát hiện ngưỡng trên
Q0.0	Đèn báo trạng thái Start/ Stop
Q0.1	Van V3 và Bơm
Q0.2	Van V1
AI 0	LIS1 Cảm biến đo mức 4...20mA
AI 1	FIS1 Cảm biến đo lưu lượng 4...20mA
AO 0	V2 Van tuyến tính

Phần mềm điều khiển

	Symbol	Address	Data type	Comment
1	COMPLETE RESTART	OB 100	OB 100	Complete Restart
2	Curr_Flow	MD 90	REAL	
3	Curr_Level	MD 16	REAL	
4	CYCL_EXC	OB 1	OB 1	Cycle Execution
5	db58	DB 58	FB 58	
6	Flow_Transmit	PIW 274	INT	
7	Level_SP	MD 25	REAL	
8	Level_Switch	I 0.1	BOOL	
9	Level_Transmitter	PIW 272	INT	4-20ma
10	Op_Per_V2	MD 29	REAL	
11	Proportional_Valve	PQW 288	INT	
12	Pump_Valve3	Q 0.1	BOOL	
13	SCALE	FC 105	FC 105	Scaling Values
14	Start/Stop push button	I 0.0	BOOL	
15	Start/Stop push lamp	Q 0.0	BOOL	
16	TCONT_CP	FB 58	FB 58	temperature PID controller with pulse generato
17	tt	Q 1.0	BOOL	
18	UNSCALE	FC 106	FC 106	Unscaling Values
19	V1	Q 0.2	BOOL	
20	VAT_1	VAT 1		
21				

OB1:

Network 1

A "Start/Stop push button"

FN M 0.0

S "Start/Stop push lamp"

S "Pump_Valve3"

```
Network 2
A     "Level_Switch"
      =     "V1"

Network3
// Lay gia tri hien thoi cua level

      CALL  "SCALE"
       IN     :=MW10
       HI_LIM :=1.000000e+002
       LO_LIM :=0.000000e+000
       BIPOLAR:=FALSE
       RET_VAL:=MW12
       OUT    :="Curr_Level"

      L     "Curr_Level"
      T     "db58".PV_IN

// Lay SP cua PID Controller


      L     "Level_SP"
      T     "db58".SP_INT

      CALL  "TCONT_CP" , "db58"
       PV_IN   :=
       PV_PER  :=
       DISV    :=
       INT_HPOS:=
       INT_HNEG:=
       SELECT  :=1
       PV      :=
       LMN     :=
       LMN_PER :=
       QPULSE  :=
       QLMN_HLM:=
```

```
      QLMN_LLM:=
      QC_ACT  :=
      CYCLE   :=
      CYCLE_P :=
      SP_INT  :=
      MAN     :=
      COM_RST :=
      MAN_ON  :=

     L    "db58".LMN_PER
     T    "Proportional_Valve"



// Lay % do mo vua Van V2

     L    "db58".LMN
     T    "Op_Per_V2"

Network 4

   A    "db58".PID_ON
   S    M      0.2
   S    "db58".LOAD_PID

Network 5

A     "Start/Stop push lamp"
A     "Start/Stop push button"
FN    M      0.1
 R    "Start/Stop push lamp"
 R    "Pump_Valve3"


OB100:

Network1:
```

SET

R "db58".MAN_ON

R "db58".PVPER_ON

DB58: