LỜI NÓI ĐẦU

Trong các hệ thống sản xuất, trong các thiết bị tự động và bán tự động, hệ thống điều khiển đóng vai trò điều phối toàn bộ các hoạt động của máy móc thiết bị. Các hệ thống máy móc và thiết bị sản xuất thường rất phức tạp, có rất nhiều đại lượng vật lý phải điều khiển để có thể hoạt động đồng bộ hoặc theo một trình tự công nghệ nhất định nhằm tạo ra một sản phẩm mong muốn. Nhờ sự phát triển nhanh chóng của kỹ thuật điện tử, các thiết bị điều khiển logic khả lập trình PLC (Programmable Logic Controller) đã xuất hiện vào năm 1969 thay thế các hệ thống điều khiển rơ le. Càng ngày PLC càng trở nên hoàn thiện và đa năng. Các PLC ngày nay không những có khả năng thay thể hoàn toàn các thiết bị điều khiển lo gíc cổ điển, mà còn có khả năng thay thế các thiêt bị điều khiển tương tự. Ngày nay chúng ta có thể thấy PLC trong hầu hết ứng dụng công nghiệp. Các PLC có thể được kết nối với các máy tính để truyền, thu thập và lưu trữ số liệu bao gồm cả quá trình điều khiển bằng thống kê, quá trình đảm bảo chất lượng, chẩn đoán sự cố trực tuyến, thay đổi chương trình điều khiển từ xa. Ngoài ra PLC còn được dùng trong hệ thống quản lý năng lượng nhằm giảm giá thành và cải thiện môi trường điều khiển trong các các hệ thống phục vụ sản xuất, trong các dịch vụ và các văn phòng công sở. Với sự hỗ trợ của máy tính cá nhân PC đã nâng cao đáng kể tính năng và khả năng sử dụng của PLC trong điều khiển máy và quá trình sản xuất. Các PC giá thành không cao có thể sử dụng như các thiêt bị lập trình và là giao diện giữa người vận hành và hệ thống điêu khiển. Nhờ sự phát triển của các phần mềm đồ hoạ cho máy tính cá nhân PC, các PLC cũng được trang bị các giao diện đồ hoạ để có thể mô phỏng hoặc hiện thị các hoạt động của từng bộ phận trong hệ thống điêu khiển. Điều này có ý nghĩa đặc biệt quan trọng đối với các máy CNC, vì nó tạo cho ta khả năng mô phỏng trước quá trình gia công, nhằm tránh các sự cố do lập trình sai. Máy tính cá nhân PC và PLC đều được sử dụng rộng rãi trong các hệ thống điều khiển sản xuất và cả trong các hệ thống dịch vụ.

Tài liệu “Lập trình PLC” với nội dung từ lý thuyết cơ bản về điều khiển học và điều khiển logic khả trình đến các ứng dụng lập trình tiêu biểu giúp người học có thể tự lập trình một ứng dụng điều khiển trực tiếp trên PLC cũng như trên máy tính PC và nạp chương trình để thực hiện trong PLC tương ứng.

CHƯƠNG 1: LÝ THUYẾT VỀ LOGIC HAI TRẠNG THÁI

1.1. Những khái niệm cơ bản

1.1.1. Khái niệm về logic hai trạng thái

Trong cuộc sống các sự vật và hiện tượng thường biểu diễn ở hai trạng thái đối lập, thông qua hai trạng thái đối lập rõ rệt, con người nhận thức được sự vật và hiện tượng một cách nhanh chóng bằng cách phân biệt hai trạng thái đó. Chẳng hạn như ta nói nước sạch và bẩn, giá cả đắt và rẻ, nước sôi và không sôi, học sinh học giỏi và dốt, kết quả tốt và xấu...

Trong kỹ thuật, đặc biệt là kỹ thuật điện và điều khiển, ta thường có khái niệm về hai trạng thái: đóng và cắt như đóng điện và cắt điện, đóng máy và ngừng máy...

Trong toán học, để lượng hoá hai trạng thái đối lập của sự vật và hiện tượng người ta dùng hai giá trị: 0 và 1. Giá trị 0 hàm ý đặc trưng cho một trang thái của sự vật hoặc hiện tượng, giá trị 1 đặc trưng cho trạng thái đối lập của sự vật và hiện tượng đó. Ta gọi các giá trị 0 hoặc 1 đó là các giá trị logic.

Các nhà bác học đã xây dựng các cơ sở toán học để tính toán các hàm và các biến chỉ lấy hai giá trị 0 và 1 này, hàm và biến đó được gọi là hàm và biến logic, cơ sở toán học để tính toán hàm và biến logic gọi là đại số logic. Đại số logic cũng có tên là đại số Boole vì lấy tên nhà toán học có công đầu trong việc xây dựng nên công cụ đại số này.

Đại số logic là công cụ toán học để phân tích và tổng hợp các hệ thống thiết bị và mạch số. Nó nghiên cứu các mối quan hệ giữa các biến số trạng thái logic. Kết quả nghiên cứu thể hiện là một hàm trạng thái cũng chỉ nhận hai giá trị 0 hoặc 1.

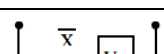
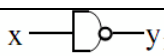
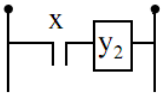
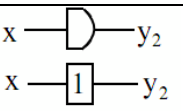
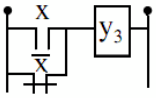
1.1.2. Các hàm logic cơ bản

Một hàm $y = f(x_1,x_2,...,x_n)$ với các biến $x_1, x_2, ... x_n$ chỉ nhận hai giá trị: 0 hoặc 1 và hàm y cũng chỉ nhận hai giá trị: 0 hoặc 1 thì gọi là hàm logic.

Hàm logic một biến: y = f (x)

Với biến x sẽ nhận hai giá trị: 0 hoặc 1, nên hàm y có 4 khả năng hay thường gọi là 4 hàm y_0, y_1, y_2, y_3. Các khả năng và các ký hiệu mạch rơle và điện tử của hàm một biến như trong bảng 1.1

Bảng 1.1

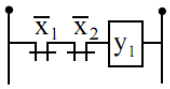
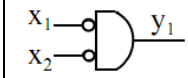
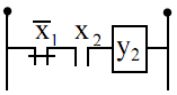
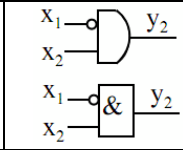
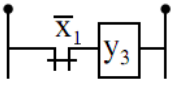
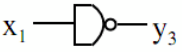
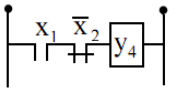
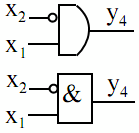
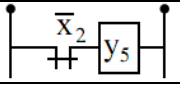
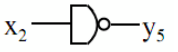
Tên hàm	Bảng chân lý			Thuật toán logic	Ký hiệu sơ đồ		Ghi chú
	x	0	1		Kiểu rơle	Kiểu khối điện tử	
Hàm không	y_0	0	0	$y_0 = 0$ $y_0 = x.\bar{x}$			
Hàm đảo	y_1	1	0	$y_1 = \bar{x}$			
Hàm lặp (YES)	y_2	0	1	$y_2 = x$			
Hàm đơn vị	y_3	1	1	$y_3 = 1$ $y_3 = x + \bar{x}$			

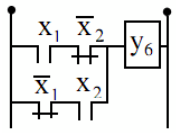
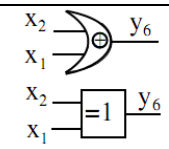
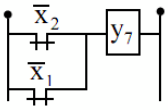
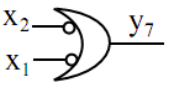
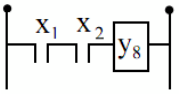
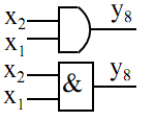
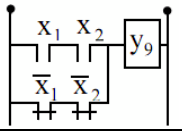
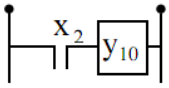
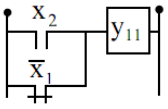
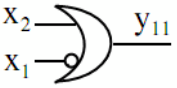
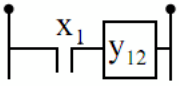
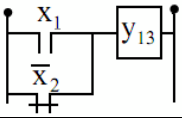
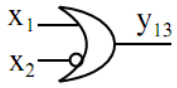
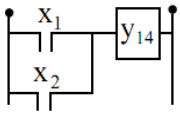
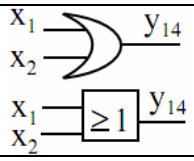
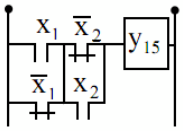
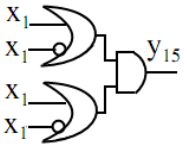
Trong các hàm trên hai hàm y_0 và y_3 luôn có giá trị không đổi nên ít được quan tâm, thường chỉ xét hai hàm y_1 và y_2.

Hàm logic hai biến $y = f(x_1,x_2)$

Với hai biến logic x_1, x_2, mỗi biến nhận hai giá trị 0 và 1, như vậy có 16 tổ hợp logic tạo thành 16 hàm. Các hàm này được thể hiện trên bảng 1.2

Bảng 1.2

Tên hàm	Bảng chân lý					Thuật toán logic	Ký hiệu sơ đồ		Ghi chú
	x_1 x_2	1 1	1 0	0 1	0 0		Kiểu rơle	Kiểu khối điện tử	
Hàm không	y_0	0	0	0	0	$y_0 = x_1\bar{x}_1 + x_2\bar{x}_2$			Hàm luôn bằng 0
Hàm Piec	y_1	0	0	0	1	$y_1 = \bar{x}_1\bar{x}_2 + \overline{x_1 + x_2}$			
Hàm cấm x_1 INHIBIT x_1	y_2	0	0	1	0	$y_2 = \bar{x}_1x_2$			
Hàm đảo x_1	y_3	0	0	1	1	$y_3 = \bar{x}_1$			
Hàm cấm x_2 INHIBIT x_2	y_4	0	1	0	0	$y_4 = x_1\bar{x}_2$			
Hàm đảo x2	y_5	0	1	0	1	$y_5 = \bar{x}_2$			

Hàm hoặc loại trừ XOR	y_6	0	1	1	0	$y_6 = x_1\bar{x}_2 + \bar{x}_1x_2$			Cộng module
Hàm Cheffer	y_7	0	1	1	1	$y_7 = \bar{x}_1 + \bar{x}_2 + \overline{x_1x_2}$			
Hàm và AND	y_8	1	0	0	0	$y_8 = x_1x_2$			
Hàm cùng dấu	y_9	1	0	0	1	$y_9 = x_1x_2 + \bar{x}_1\bar{x}_2$			
Hàm lặp x_2	y_{10}	1	0	1	0	$y_{10} = x_2$			Chỉ phụ thuộc x2
Hàm kéo theo x_2	y_{11}	1	0	1	1	$y_{11} = \bar{x}_1 + x_2$			
Hàm lặp x_1	y_{12}	1	1	0	0	$y_{12} = x_1$			Chỉ phụ thuộc x1
Hàm kéo theo x_1	y_{13}	1	1	0	1	$y_{13} = x_1 + \bar{x}_2$			
Hàm hoặc OR	y_{14}	1	1	1	0	$y_{14} = x_1 + x_2$			
Hàm đơn vị	y_{15}	1	1	1	1	$y_{15} = (x_1 + \bar{x}_1)(x_2 + \bar{x}_2)$			Hàm luôn bằng 1

Ta nhận thấy rằng, các hàm đối xứng nhau qua trục nằm giữa y_7 và y_8, nghĩa là $y_0 = y_{15}$, $y_1 = y_{14}$ ...

Hàm logic n biến $y = f(x_1, x_2,..., x_n)$

Với hàm logic n biến, mỗi biến nhận một trong hai giá trị 0 hoặc 1 nên ta có 2^n tổ hợp biến, mỗi tổ hợp biến lại nhận hai giá trị 0 hoặc 1, do vậy số hàm logic tổng là 2^{2^n}. Ta thấy với 1 biến có 4 khả năng tạo hàm, với 2 biến có 16 khả năng tạo hàm, với 3 biến có 256 khả năng tạo hàm. Như vậy khi số biến tăng thì số hàm có khả năng tạo thành rất lớn.

Trong tất cả các hàm được tạo thành ta đặc biệt chú ý đến hai loại hàm là hàm tổng chuẩn và hàm tích chuẩn. Hàm tổng chuẩn là hàm chứa tổng các tích

mà mỗi tích có đủ tất cả các biến của hàm. Hàm tích chuẩn là hàm chứa tích các tổng mà mỗi tổng đều có đủ tất cả các biến của hàm.

1.1.3. Các phép tính cơ bản

Người ta xây dựng ba phép tính cơ bản giữa các biến logic đó là:

1. Phép phủ định (đảo): ký hiệu bằng dấu "-" phía trên ký hiệu của biến.

2. Phép cộng (tuyển): ký hiệu bằng dấu "+" (song song)

3. Phép nhân (hội): ký hiệu bằng dấu "." (nối tiếp)

1.1.4. Tính chất và một số hệ thức cơ bản

a. Các tính chất

Tính chất của đại số logic được thể hiện ở bốn luật cơ bản là: luật hoán vị, luật kết hợp, luật phân phối và luật nghịch đảo.

+ Luật hoán vị:

$$x_1 + x_2 = x_2 + x_1$$

$$x_1.x_2 = x_2.x_1$$

+ Luật kết hợp:

$$x_1 + x_2 + x_3 = (x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)$$

$$x_1.x_2.x_3 = (x_1.x_2).x_3 = x_1.(x_2.x_3)$$

+ Luật phân phối:

$$(x_1 + x_2).x_3 = x_1.x_3 + x_2.x_3$$

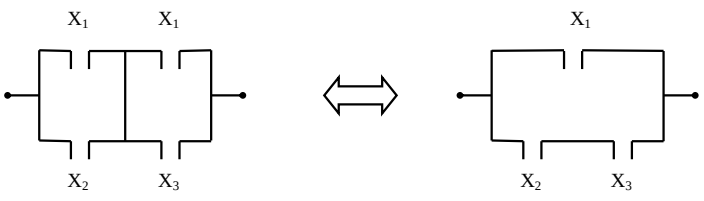
$$x_1 + x_2.x_3 = (x_1 + x_2).(x_1 + x_3)$$

Ta có thể minh hoạ để kiểm chứng tính đúng đắn của luật phân phối bằng cách lập bảng 1.3

Bảng 1.3

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
$(x_1 + x_2).(x_1 + x_3)$	0	0	0	1	1	1	1	1
$x_1 + x_2.x_3$	0	0	0	1	1	1	1	1

Luật phân phối được thể hiện qua sơ đồ rơle hình 1.1:



Hình 1.1

+ Luật nghịch đảo:

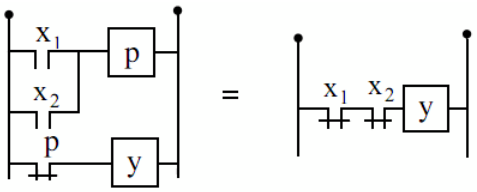
$$\overline{x_1.x_2} = \overline{x_1} + \overline{x_2}$$
$$\overline{x_1 + x_2} = \overline{x_1}.\overline{x_2}$$

Ta cũng minh hoạ tính đúng đắn của luật nghịch đảo bằng cách thành lập bảng 1.4:

Bảng 1.4

x_1	x_2	$\overline{x}_1$	$\overline{x}_2$	$\overline{x_1 + x_2}$	$\overline{x}_1\overline{x}_2$	$\overline{x}_1 + \overline{x}_2$	$\overline{x_1x_2}$
0	0	1	1	1	1	1	1
0	1	1	0	0	0	1	1
1	0	0	1	0	0	1	1
1	1	0	0	0	0	0	0

Luật nghịch đảo được thể hiện qua mạch rơle như trên hình 1.2:



Hình 1.2

Luật nghịch đảo tổng quát được thể hiện bằng định lý De Morgan:

$$\overline{x_1.x_2.x_3...} = \overline{x}_1 + \overline{x}_2 + \overline{x}_3 + ...$$
$$\overline{x_1 + x_2 + x_3 + ...} = \overline{x}_1.\overline{x}_2.\overline{x}_3....$$

b. Các hệ thức cơ bản

Một số hệ thức cơ bản thường dùng trong đại số logic được cho ở bảng:

Bảng 1.5

1	$x + 0 = x$	10	$x_1.x_2 = x_2.x_1$
2	$x.1 = x$	11	$x_1 + x_1.x_2 = x_1$
3	$x.0 = 0$	12	$x_1(x_1 + x_2) = x_1$
4	$x + 1 = 1$	13	$x_1.x_2 + x_1.\overline{x}_2 = x_1$
5	$x + x = x$	14	$(x_1 + x_2)(x_1 + \overline{x}_2) = x_1$
6	$x.x = x$	15	$x_1 + x_2 + x_3 = (x_1 + x_2) + x_3$
7	$x + \overline{x} = 1$	16	$x_1.x_2.x_3 = (x_1.x_2).x_3$
8	$x.\overline{x} = 0$	17	$\overline{(x_1 + x_2)} = \overline{x}_1.\overline{x}_2$

9	$x_1 + x_2 = x_2 + x_1$	18	$\overline{x_1.x_2} = \overline{x}_1 + \overline{x}_2$

1.2. Các phương pháp biểu diễn hàm logic

Có thể biểu diễn hàm logic theo bốn cách là: biểu diễn bằng bảng trạng thái, biểu diễn bằng phương pháp hình học, biểu diễn bằng biểu thức đại số, biểu diễn bằng bảng Karnaugh (bìa Canô).

1.2.1. Phương pháp biểu diễn bằng bảng trạng thái:

Ở phương pháp này các giá trị của hàm được trình bày trong một bảng. Nừu hàm có n biến thì bảng có n +1 cột (n cột cho biến và 1 cột cho hàm) và 2n hàng tương ứng với 2n tổ hợp của biến. Bảng này thường gọi là bảng trạng thái hay bảng chân lý.

Ví dụ: một hàm 3 biến $y = f(x_1, x_2, x_3)$ với giá trị của hàm đã cho trước được biểu diễn thành bảng 1.6:

Bảng 1.6

TT tổ hợp biến	x_1	x_2	x_3	y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Ưu điểm của phương pháp biểu diễn bằng bảng là dễ nhìn, ít nhầm lẫn. Nhược điểm là cồng kềnh, đặc biệt khi số biến lớn.

1.2.2. Phương pháp biểu diễn hình học

Với phương pháp hình học hàm n biến được biểu diễn trong không gian n chiều, tổ hợp biến được biểu diễn thành một điểm trong không gian. Phương pháp này rất phức tạp khi số biến lớn nên thường ít dùng.

1.2.3. Phương pháp biểu diễn bằng biểu thức đại số

Người ta chứng minh được rằng, một hàm logic n biến bất kỳ bao giờ cũng có thể biểu diễn thành các hàm tổng chuẩn đầy đủ và tích chuẩn đầy đủ.

Cách viết hàm dưới dạng tổng chuẩn đầy đủ

- Hàm tổng chuẩn đầy đủ chỉ quan tâm đến tổ hợp biến mà hàm có giá trị bằng 1. Số lần hàm bằng 1 sẽ chính là số tích của các tổ hợp biến.

- Trong mỗi tích, các biến có giá trị bằng 1 được giữ nguyên, còn các biến có giá trị bằng 0 thì được lấy giá trị đảo; nghĩa là nếu x_i =1 thì trong biểu thức tích sẽ được viết là x_i , còn nếu x_i = 0 thì trong biểu thức tích được viết là $\bar{x}_i$. Các tích này còn gọi là các mintec và ký hiệu là m.

- Hàm tổng chuẩn đầy đủ sẽ là tổng của các tích đó.

Ví dụ: Với hàm ba biến ở bảng 1.6, ta có hàm ở dạng tổng chuẩn đầy đủ:

$$f = \bar{x}_1.\bar{x}_2.\bar{x}_3 + \bar{x}_1.x_2.\bar{x}_3 + \bar{x}_1.x_2.x_3 + x_1.x_2.\bar{x}_3 = m_0 + m_2 + m_3 + m_6$$

Cách viết hàm dưới dạng tích chuẩn đầy đủ

- Hàm tích chuẩn đầy đủ chỉ quan tâm đến tổ hợp biến mà hàm có giá trị bằng 0. Số lần hàm bằng không sẽ chính là số tổng của các tổ hợp biến.

- Trong mỗi tổng các biến có giá trị 0 được giữ nguyên, còn các biến có giá trị 1 được lấy đảo; nghĩa là nếu x_i = 0 thì trong biểu thức tổng sẽ được viết là x_i , còn nếu x_i =1 thì trong biểu thức tổng được viết bằng $\bar{x}_i$. Các tổng cơ bản còn được gọi tên là các Maxtec ký hiệu M.

- Hàm tích chuẩn đầu đủ sẽ là tích của các tổng đó.

Ví dụ: Với hàm ba biến ở bảng 1.6, ta có hàm ở dạng tích chuẩn đầy đủ:

$$f = (x_1 + x_2 + \bar{x}_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + x_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$$
$$= M_1 + M_4 + M_5 + M_7$$

1.2.4. Phương pháp biểu diễn bằng bảng Karnaugh (bìa canô)

Nguyên tắc xây dựng bảng Karnaugh:

- Để biểu diễn hàm logic n biến cần thành lập một bảng có 2n ô, mỗi ô tương ứng với một tổ hợp biến. Đánh số thứ tự các ô trong bảng tương ứng với thứ tự các tổ hợp biến.

- Các ô cạnh nhau hoặc đối xứng nhau chỉ cho phép khác nhau về giá trị của 1 biến.

- Trong các ô ghi giá trị của hàm tương ứng với giá trị tổ hợp biến.

Ví dụ 1: bảng Karnaugh cho hàm ba biến ở bảng 1.6 như bảng 1.7 sau:

x_1 \ x_2, x_3	*00*	*01*	*11*	*10*
0	*0* 1	*1*	*3* 1	*2* *1*
1	*4*	*5*	*7*	*6* 1

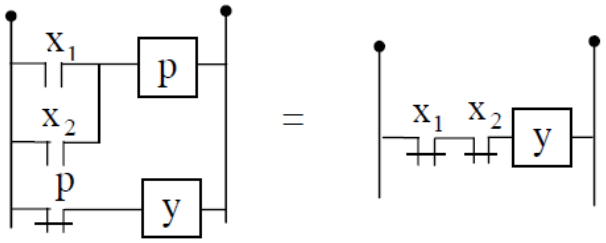
Ví dụ 2: bảng Karnaugh cho hàm bốn biến như bảng 1.8 sau:

x_1, x_2 \ x_3, x_4	*00*	*01*	*11*	*10*
00	*0* 1	*1*	*3* 1	*2* *1*
01	*4*	*5*	*7*	*6* 1
11	*12* 1	*13*	*15* 1	*14*
10	*8*	*9* 1	*11*	*10*

1.3. Các phương pháp tối thiểu hoá hàm logic

Trong quá trình phân tích và tổng hợp mạch logic, ta phải quan tâm đến vấn đề tối thiểu hoá hàm logic. Bởi vì, cùng một giá trị hàm logic có thể có nhiều hàm khác nhau, nhiều cách biểu diễn khác nhau nhưng chỉ tồn tại một cách biểu diễn gọn nhất, tối ưu về số biến và số số hạng hay thừa số được gọi là dạng tối thiểu. Việc tối thiểu hoá hàm logic là đưa chúng từ một dạng bất kỳ về dạng tối thiểu. Tối thiểu hoá hàm logic mang ý nghĩa kinh tế và kỹ thuật lớn, đặc biệt khi tổng hợp các mạch logic phức tạp. Khi chọn được một sơ đồ tối giản ta sẽ có số biến cũng như các kết nối tối giản, giảm được chi phí vật tư cũng như giảm đáng kể xác suất hỏng hóc do số phần tử nhiều.

Ví dụ: Hai sơ đồ hình 1.3 đều có chức năng như nhau, nhưng sơ đồ a số tiếp điểm cần là 3, đồng thời cần thêm 1 rơle trung gian P, sơ đồ b chỉ cần 2 tiếp điểm, không cần rơle trung gian.



Hình 1.3

Thực chất việc tối thiểu hoá hàm logic là tìm dạng biểu diễn đại số đơn giản nhất của hàm và thường có hai nhóm phương pháp là:

- Phương pháp biến đổi đại số

- Phương pháp dùng thuật toán.

1.3.1. Phương pháp tối thiểu hoá hàm logic bằng biến đổi đại số

ở phương pháp này ta phải dựa vào các tính chất và các hệ thức cơ bản của đại số logic để thực hiện tối giản các hàm logic. Nhưng do tính trực quan của phương pháp nên nhiều khi kết quả đưa ra vẫn không khẳng định rõ được là đã tối thiểu hay chưa. Như vậy, đây không phải là phương pháp chặt chẽ cho quá trình tối thiểu hoá.

Ví dụ: cho hàm

$$\begin{aligned} f &= \overline{x}_1 x_2 + x_1 x_2 + x_1 \overline{x}_2 \\ &= \overline{x}_1 x_2 + x_1 x_2 + x_1 x_2 + x_1 \overline{x}_2 \\ &= x_2(\overline{x}_1 + x_1) + x_1(x_2 + \overline{x}_2) = x_1 + x_2 \end{aligned}$$

1.3.2. Phương pháp tối thiểu hoá hàm logic dùng bảng Karnaugh

Đây là phương pháp thông dụng và đơn giản nhất, nhưng chỉ tiến hành được với hệ có số biến $n \leq 6$. ở phương pháp này cần quan sát và xử lý trực tiếp trên bảng Karnaugh.

Qui tắc của phương pháp là: nếu có 2n ô có giá trị 1 nằm kề nhau hợp thành một khối vuông hay chữ nhật thì có thể thay 2n ô này bằng một ô lớn với số lượng biến giảm đi n lần. Như vậy, bản chất của phương pháp là tìm các ô kề nhau chứa giá trị 1 (các ô có giá trị hàm không xác định cũng gán cho giá trị 1) sao cho lập thành hình vuông hay chữ nhật càng lớn càng tốt. Các biến nằm trong khu vực này bị loại bỏ là các biến có giá trị biến đổi, các biến được dùng là các biến có giá trị không biến đổi (chỉ là 0 hoặc 1).

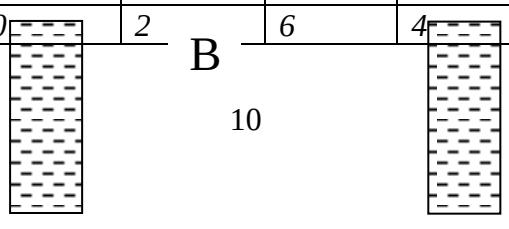
Qui tắc này áp dụng theo thứ tự giảm dần độ lớn các ô, sao cho cuối cùng toàn bộ các ô chưa giá trị 1 đều được bao phủ. Cũng có thể tiến hành tối thiểu theo giá trị 0 của hàm nếu số lượng của nó ít hơn nhiều so với giá trị 1, lúc bấy giờ hàm là hàm phủ định.

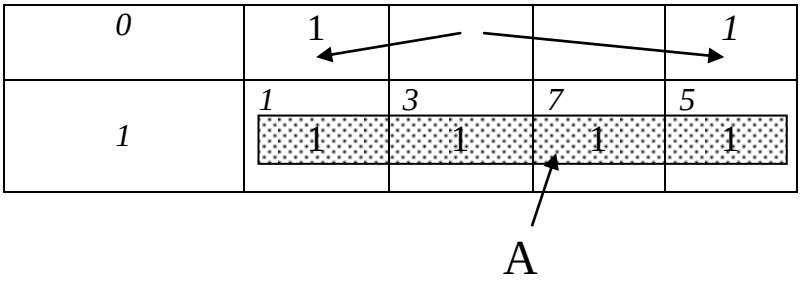
Ví dụ: Tối thiểu hàm

$$f = \overline{x}.\overline{y}.\overline{z} + \overline{x}.\overline{y}.z + x.\overline{y}.\overline{z} + \overline{x}.y.z + x.\overline{y}.z + x.y.z = m_0 + m_1 + m_3 + m_4 + m_5 + m_7$$

+ Lập bảng Karnaugh được như bảng 1.9, có 3 biến với 6 mintec

x, y / z	*00*	*01*	*11*	*10*
	0	*2*	*6*	*4*





+ Tìm nhóm các ô (hình chữ nhật) chứa các ô có giá trị bằng 1, ta được hai nhóm, nhóm A và nhóm B.

+ Loại bớt các biến ở các nhóm:

Nhóm A có biến z =1 không đổi vậy nó được giữ lại còn hai biến x và y thay đổi theo từng cột do vậy mintec mới A chỉ còn biến z: A = z.

Nhóm B có biến x và z thay đổi, còn biến y không đổi vậy mintec mới B chỉ còn biến y : B = y .

Kết quả tối thiểu hoá là: f = A + B = z + y

1.4. Các hệ mạch logic

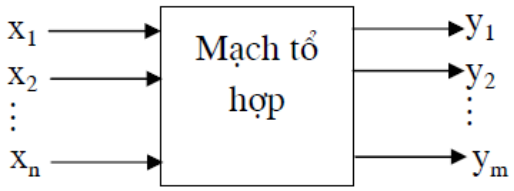
Các phép toán và định lý của đại số Boole giúp cho thao tác các biểu thức logic. Trong kỹ thuật thực tế là bằng cách nối cổng logic của các mạch logic với nhau (theo kết cấu đã tối giản nếu có). Để thực hiện một bài toán điều khiển phức tạp, số mạch logic sẽ phụ thuộc vào số lượng đầu vào và cách giải quyết bằng loại mạch logic nào, sử dụng các phép toán hay định lý nào. Đây là một bài toán tối ưu nhiều khi có không chỉ một lời giải. Tuỳ theo loại mạch logic mà việc giải các bài toán có những phương pháp khác nhau. Về cơ bản các mạch logic được chia làm hai loại:

+ Mạch logic tổ hợp

+ Mạch logic trình tự

1.4.1. Mạch logic tổ hợp

Mạch logic tổ hợp là mạch mà đầu ra tại bất kỳ thời điểm nào chỉ phụ thuộc tổ hợp các trạng thái của đầu vào ở thời điểm đó. Như vậy, mạch không có phần tử nhớ. Theo quan điểm điều khiển thì mạch tổ hợp là mạch hở, hệ không có phản hồi, nghĩa là trạng thái đóng mở của các phần tử trong mạch hoàn toàn không bị ảnh hưởng của trạng thái tín hiệu đầu ra. Sơ đồ mạch logic tổ hợp như hình 1.4



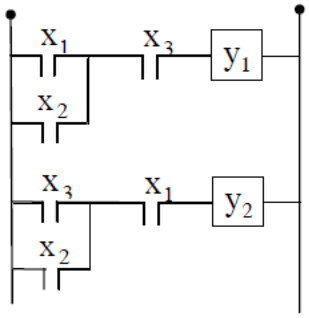
Hình 1.4

Với mạch logic tổ hợp tồn tại hai loại bài toán là bài toán phân tích và bài toán tổng hợp.

+ Bài toán phân tích có nhiệm vụ là từ mạch tổ hợp đã có, mô tả hoạt động và viết các hàm logic của các đầu ra theo các biến đầu vào và nếu cần có thể xét tới việc tối thiểu hoá mạch.

+ Bài toán tổng hợp thực chất là thiết kế mạch tổ hợp. Nhiệm vụ chính là thiết kế được mạch tổ hợp thoả mãn yêu cầu kỹ thuật nhưng mạch phải tối giản. Bài toán tổng hợp là bài toán phức tạp, vì ngoài các yêu cầu về chức năng logic, việc tổng hợp mạch còn phụ thuộc vào việc sử dụng các phần tử, chẳng hạn như phần tử là loại: rơle - công tắc tơ, loại phần tử khí nén hay loại phần tử là bán dẫn vi mạch... Với mỗi loại phần tử logic được sử dụng thì ngoài nguyên lý chung về mạch logic còn đòi hỏi phải bổ sung những nguyên tắc riêng lúc tổng hợp và thiết kế hệ thống.

Ví dụ: về mạch logic tổ hợp như hình 1.5

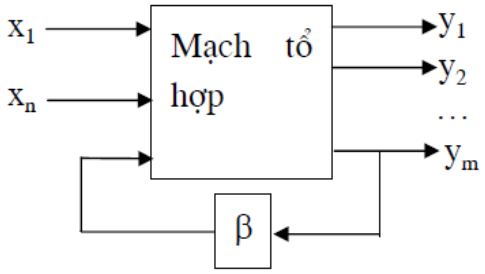


Hình 1.5

1.4.2. Mạch logic trình tự

Mạch trình tự hay còn gọi là mạch dãy (sequential circuits) là mạch trong đó trạng thái của tín hiệu ra không những phụ thuộc tín hiệu vào mà còn phụ thuộc cả trình tự tác động của tín hiệu vào, nghĩa là có nhớ các trạng thái. Như vậy, về mặt thiết bị thì ở mạch trình tự không những chỉ có các phần tử đóng mở mà còn có cả các phần tử nhớ.

Sơ đồ nguyên lý mạch logic trình tự như hình 1.6



Hình 1.6

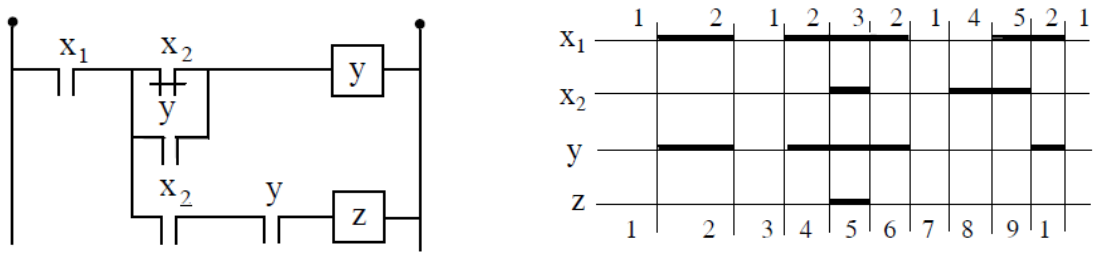
Xét mạch logic trình tự như hình 1.7. Ta xét hoạt động của mạch khi thay đổi trạng thái đóng mở của x_1 và x_2. Biểu đồ hình 1.7b mô tả hoạt động của mạch, trong biểu đồ các nét đậm biểu hiện tín hiệu có giá trị 1, còn nét mảnh biểu hiện tín hiệu có giá trị 0.

Từ biểu đồ hình 1.7b ta thấy, trạng thái z =1 chỉ đạt được khi thao tác theo trình tự x_1 =1, tiếp theo x_2 =1. Nếu cho x_2 =1 trước, sau đó cho x_1 =1 thì cả y và z đều không thể bằng 1.

Để mô tả mạch trình tự ta có thể dùng bảng chuyển trạng thái, dùng đồ hình trạng thái Mealy, đồ hình trạng thái Moore hoặc dùng phương pháp lưu đồ.

Trong đó phương pháp lưu đồ có dạng trực quan hơn. Từ lưu đồ thuật toán ta dễ dàng chuyển sang dạng đồ hình trạng thái Mealy hoặc đồ hình trạng thái Moore. và từ đó có thể thiết kế được mạch trình tự.

Với mạch logic trình tự ta cũng có bài toán phân tích và bài toán tổng hợp.



a) b)

Hình 1.7

1.5. Grafcet - để mô tả mạch trình tự trong công nghiệp

1.5.1. Hoạt động của thiết bị công nghiệp theo logic trình tự

Trong dây truyền sản xuất công nghiệp, các thiết bị máy móc thường hoạt động theo một trình tự logic chặt chẽ nhằm đảm bảo chất lượng sản phẩm và an toàn cho người và thiết bị.

Một quá trình công nghệ nào đó cũng có thể có ba hình thức điều khiển hoạt động sau:

+ Điều khiển hoàn toàn tự động, lúc này chỉ cần sự chỉ huy chung của nhân viên vận hành hệ thống.

+ Điều khiển bán tự động, quá trình làm việc có liên quan trực tiếp đến các thao tác liên tục của con người giữa các chuỗi hoạt động tự động.

+ Điều khiển bằng tay, tất cả hoạt động của hệ đều do con người thao tác.

Trong quá trình làm việc để đảm bảo an toàn, tin cậy và linh hoạt, hệ điều khiển cần có sự chuyển đổi dễ dàng từ điều khiểu bằng tay sang tự động và ngược lại, vì như vậy hệ điều khiển mới đáp ứng đúng các yêu cầu thực tế.

Trong quá trình làm việc sự không bình thường trong hoạt động của dây truyền có rất nhiều loại, khi thiết kế ta phải cố gắng mô tả chúng một cách đầy đủ nhất. Trong số các hoạt động không bình thường của chương trình điều khiển một dây truyền tự động, người ta thường phân biệt ra các loại sau:

+ Hư hỏng một bộ phận trong cấu trúc điều khiển. Lúc này cần phải xử lý riêng phần chương trình có chỗ hư hỏng, đồng thời phải lưu tâm cho dây truyền hoạt động lúc có hư hỏng và sẵn sàng chấp nhận lại điều khiển khi hư hỏng được sửa chữa xong.

+ Hư hỏng trong cấu trúc trình tự điều khiển.

+ Hư hỏng bộ phận chấp hành (như hư hỏng thiết bị chấp hành, hư hỏng cảm biến, hư hỏng các bộ phân thao tác...)

Khi thiết kế hệ thống phải tính đến các phường thức làm việc khác nhau để đảm bảo an toàn và xử lý kịp thời các hư hỏng trong hệ thống, phải luôn có phương án can thiệp trực tiếp của người vận hành đến việc dừng máy khẩn cấp, xử lý tắc nghẽn vật liệu và các hiện tượng nguy hiểm khác. Grafcel là công cụ rất hữu ích để thiết kế và thực hiện đầy đủ các yêu cầu của hệ tự động cho các quá trình công nghệ kể trên.

1.5.2. Định nghĩa Grafcet

Grafcet là từ viết tắt của tiếng Pháp "Graphe fonctionnel de commande étape transition" (chuỗi chức năng điều khiển giai đoạn - chuyển tiếp), do hai cơ quan AFCET (Liên hợp Pháp về tin học, kinh tế và kỹ thuật) và ADEPA (tổ chức nhà nước về phát triển nền sản xuất tự động hoá) hợp tác soạn thảo tháng

11/1982 được đăng ký ở tổ chức tiêu chuẩn hoá Pháp. Như vậy, mạng grafcet đã được tiêu chuẩn hoá và được công nhận là một ngôn ngữ thích hợp cho việc mô tả hoạt động dãy của quá trình tự động hoá trong sản xuất.

Mạng grafcet là một đồ hình chức năng cho phép mô tả các trạng thái làm việc của hệ thống và biểu diễn quá trình điều khiển với các trạng thái và sự chuyển đổi từ trạng thái này sang trạng thái khác, đó là một đồ hình định hướng được xác định bởi các phần tử là: tập các trạng thái, tập các điều kiện chuyển trạng thái.

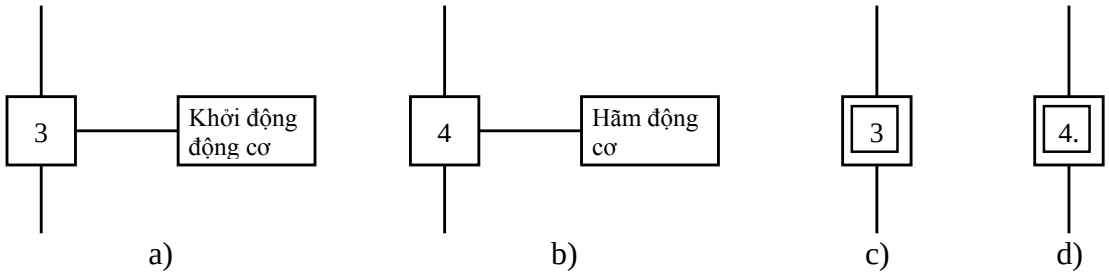
Mạng grafcet mô tả thành chuỗi các giai đoạn trong chu trình sản xuất.

Mạng grafcet cho một quá trình sản xuất luôn luôn là một đồ hình khép kín từ trạng thái đầu đến trạng thái cuối và từ trạng thái cuối về trạng thái đầu.

1.5.3. Một số ký hiệu trong grafcet

- Một trạng thái (giai đoạn) được biểu diễn bằng một hình vuông có đánh số thứ tự chỉ trạng thái. Gắn liền với biểu tượng trạng thái là một hình chữ nhật bên cạnh, trong hình chữ nhật này có ghi các tác động của trạng thái đó hình 1.8a và b.

Một trạng thái có thể tương ứng với một hoặc nhiều hành động của quá trình sản xuất.



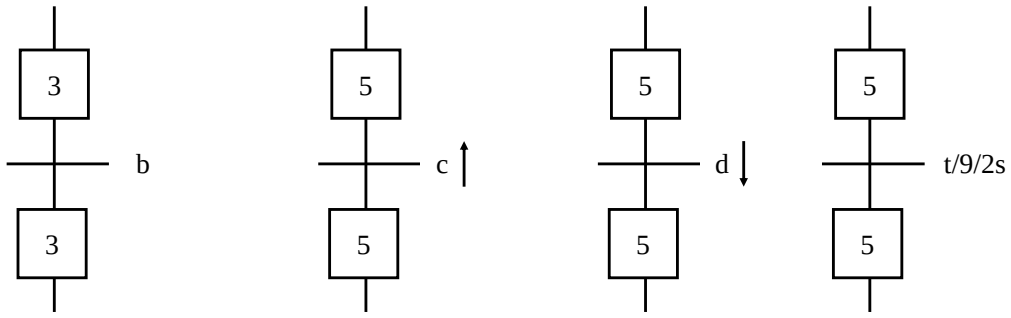
Hình 1.8

- Trạng thái khởi động được thể hiện bằng 2 hình vuông lồng vào nhau, thứ tự thường là 1 hình 1.8c.

- Trạng thái hoạt động (tích cực) có thêm dấu "." ở trong hình vuông trạng thái hình 1.8d.

- Việc chuyển tiếp từ trạng thái này sang trạng thái khác chỉ có thể được thực hiện khi các điều kiện chuyển tiếp được thoả mãn. Chẳng hạn, việc chuyển tiếp giữa các trạng thái 3 và 4 hình 1.9a được thực hiện khi tác động lên biến b, còn chuyển tiếp giữa trạng thái 5 và 6 được thực hiện ở sườn tăng của biến c hình 1.9b, ở hình 1.9c là tác động ở sườn giảm của biến d. Chuyển tiếp giữa

trạng thái 9 và 10 hình 1.9d sẽ xảy ra sau 2s kể từ khi có tác động cuối cùng của trạng thái 9 được thực hiện.



Hình 1.9

- Ký hiệu phân nhánh như hình 1.10. ở sơ đồ phân nhánh lại tồn tại hai loại là sơ đồ rẽ nhánh và sơ đồ song song.

Sơ đồ rẽ nhánh là phần sơ đồ có hai điều kiện liên hệ giữa ba trạng thái như hình 1.10a và b.

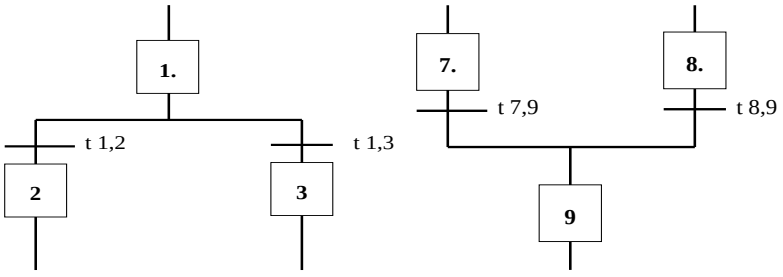
Sơ đồ song song là sơ đồ chỉ có một điều kiện liên hệ giữa 3 trạng thái như hình 1.10c và d.

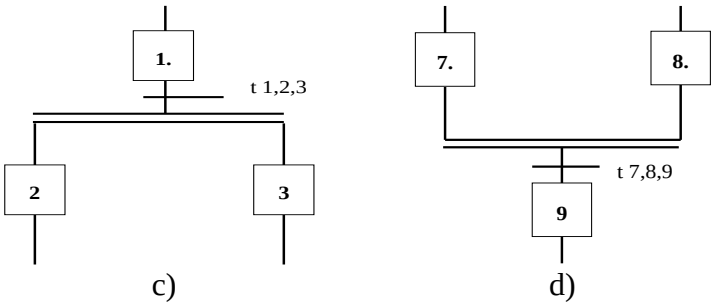
Ở hình 1.10a , khi trạng thái 1 đang hoạt động, nếu chuyển tiếp t12 thoả mãn thì trạng thái 2 hoạt động; nếu chuyển tiếp t13 thoả mãn thì trạng thái 3 hoạt động.

Ở hình 1.10b nếu trạng thái 7 đang hoạt động và có t79 thì trạng thái 9 hoạt động, nếu trạng thái 8 đang hoạt động và có t89 thì trạng thái 9 hoạt động.

Ở hình 1.10c nếu trạng thái 1 đang hoạt động và có t123 thì trạng thái 2 và 3 đồng thời hoạt động.

Ở hình 1.10d nếu trạng thái 7 và 8 đang cùng hoạt động và có t789 thì trạng thái 9 hoạt động.

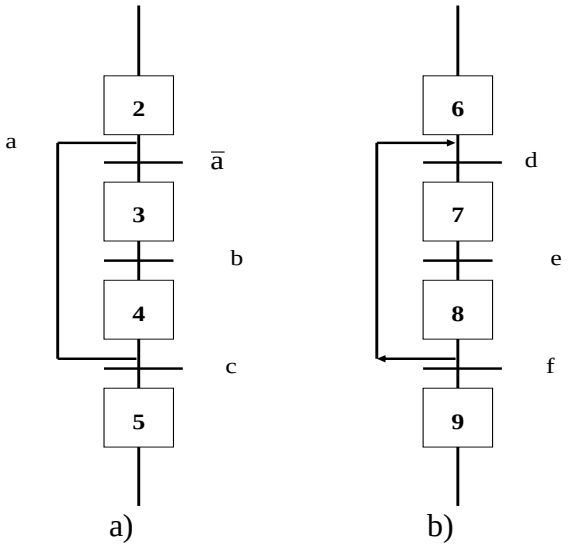




Hình 1.10

- Ký hiệu bước nhảy như hình 1.11.

Hình 1.11a biểu diễn grafcet cho phép thực hiện bước nhảy, khi trạng thái 2 đang hoạt động nếu có điều kiện a thì quá trình sẽ chuyển hoạt động từ trạng thái 2 sang trạng thái 5 bỏ qua các trạng thái trung gian 3 và 4, nếu điều kiện a không được thoả mãn thì quá trình chuyển tiếp theo trình tự 2, 3, 4, 5.



Hình 1.11

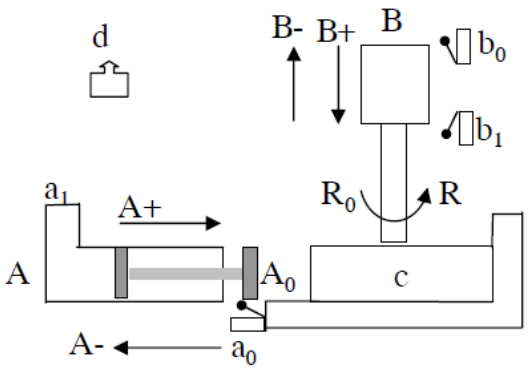
Hình 1.11b, khi trạng thái 8 đang hoạt động nếu thoả mãn điều kiện f thì quá trình chuyển sang trạng thái 9, nếu không thoả mãn điều kiện 8 thì quá trình quay lại trạng 7.

1.5.4. Cách xây dựng mạng grafcet

Để xây dựng mạng grafcet cho một quá trình nào đó thì trước tiên ta phải mô tả mọi hành vi tự động bao gồm các giai đoạn và các điều kiện chuyển tiếp, sau đó lựa chọn các dẫn động và các cảm biến rồi mô tả chúng bằng các ký hiệu, sau đó kết nối chúng lại theo cách mô tả của

grafcet.

Ví dụ: để kẹp chặt chi tiết c và khoan trên đó một lỗ (hình 1.12) thì trước tiên người điều khiển ấn nút khởi động d để khởi động chu trình công nghệ tự động, quá trình bắt đầu từ giai đoạn 1:



Hình 1.12

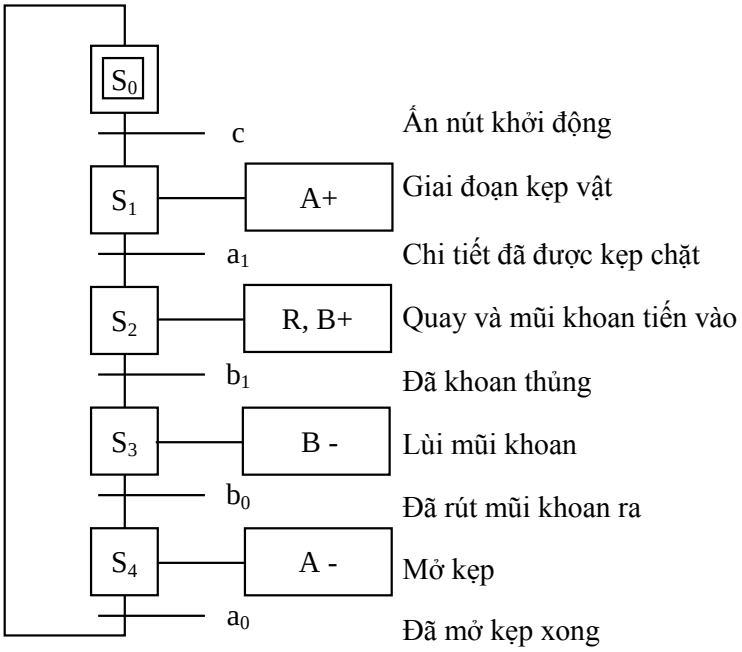
+ Giai đoạn 1: S_1 píttông A chuyển động theo chiều A+ để kẹp chặt chi tiết c. Khi lực kẹp đạt yêu cầu được xác định bởi cảm biến áp suất a_1 thì chuyển sang giai đoạn 2.

+ Giai đoạn 2: S_2 đầu khoan B đi xuống theo chiều B+ và mũi khoan quay theo chiều R, khi khoan đủ sâu, xác định bằng nút b_1 thì kết thúc giai đoạn 2, chuyển sang giai đoạn 3.

+ Giai đoạn 3: S_3 mũi khoan đi lên theo chiều B- và ngừng quay. Khi mũi khoan lên đủ cao, xác định bằng b_0 thì khoan dừng và chuyển sang giai đoạn 4.

+ Giai đoạn 4: S_4 píttông A trở về theo chiều A - nới lỏng chi tiết, vị trí trở về được xác định bởi a_0, khi đó píttông ngừng chuyển động, kết thúc một chu kỳ gia công.

Ta có sơ đồ grafcet như hình 1.13



Hình 1.13

1.5.5. Phân tích mạng grafcet

a. Qui tắc vượt qua, chuyển tiếp

- Một trạng thái trước chỉ chuyển tiếp sang trạng thái sau khi nó đang hoạt động (tích cực) và có đủ điều kiện chuyển tiếp.

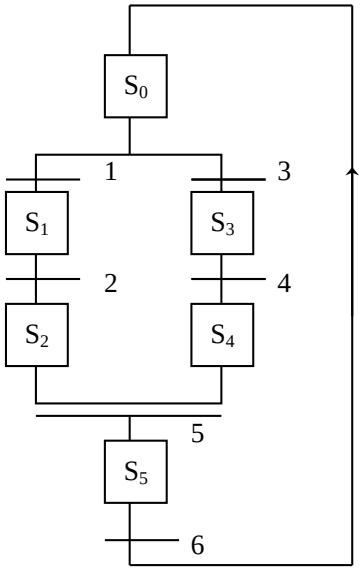
- Khi quá trình đã chuyển tiếp sang trạng thái sau thì giai đoạn sau hoạt động (tích cực) và sẽ khử bỏ hoạt động của trạng thái trước đó (giai đoạn trước hết tích cực).

Với các điều kiện hoạt động như trên thì có nhiều khi sơ đồ không hoạt động được hoặc hoạt động không tốt. Người ta gọi:

+ Sơ đồ không hoạt động được là sơ đồ có nhánh chết. (Sơ đồ có nhánh chế có thể vẫn hoạt động nếu như không đi vào nhánh chết).

+ Sơ đồ không sạch là sơ đồ mà tại một vị trí nào đó được phát lệnh hai lần.

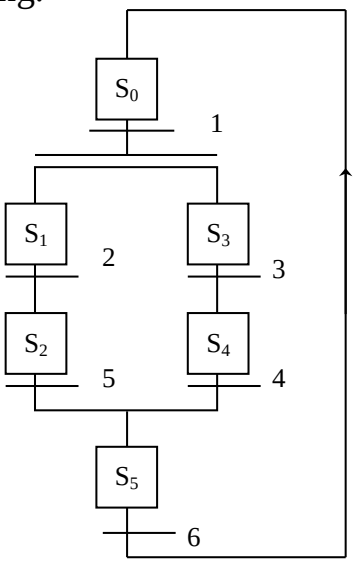
Ví dụ 1: Sơ đồ hình 1.14 là sơ đồ có nhánh chết.



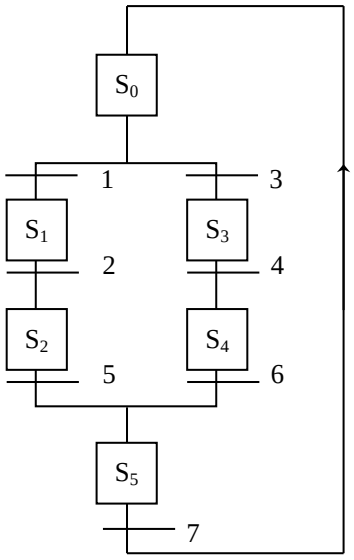
Hình 1.14

Sơ đồ này không thể làm việc được do S_2 và S_4 không thể cùng tích cực vì giả sử hệ đang ở trạng thái ban đầu S_0 nếu có điều kiện 3 thì S_0 hết tích cực và chuyển sang S_3 tích cực. Sau đó nếu có điều kiện 4 thì S_3 hết tích cực và S_4 tích cực. Nếu lúc này có điều kiện 1 thì S_1 cũng không thể tích cực được vì S_0 đã hết tích cực. Do đó không bao giờ S_2 tích cực được nữa mà để S_5 tích cực thì phải có S_2 và S_4 tích cực kèm điều kiện 5 như vậy hệ sẽ nằm im ở vị trí S_4.

Muốn sơ đồ trên làm việc được ta phải chuyển mạch rẽ nhánh thành mạch song song.



Hình 1.15



Hình 1.16

Ví dụ 2: Sơ đồ hình 1.15 là sơ đồ không sạch. Mạng đang ở trạng thái ban đầu nếu có điều kiện 1 thì sẽ chuyển trạng thái cho cả S_1 và S_3 tích cực. Nếu có

điều kiện 3 rồi 4 thì sẽ chuyển cho S_5 tích cực. Khi chưa có điều kiện 6 mà lại có điều kiện 2 rồi 5 trước thì S_5 lại chuyển tích cực lần nữa. Tức là có hai lần lệnh cho S_5 tích cực, vậy là sơ đồ không sạch.

Ví dụ 3: Sơ đồ hình 1.16 là sơ đồ sạch. ở sơ đồ này nếu đã có S_3 tích cực (điều kiện 3) thì nếu có điều kiện 1 cũng không có nghĩa vì S_0 đã hết tích cực. Như vậy, mạch đã rẽ sang nhánh 2, nếu lần lượt có các điều kiện 4 và 6 thì S_5 sẽ tích cực sau đó nếu có điều kiện 7 thì hệ lại trở về trạng thái ban đầu.

b. Phân tích mạng grafcet

Như phân tích ở trên thì nhiều khi mạng grafcet không hoạt động được hoặc hoạt động không tốt. Nhưng đối với các mạng không hoạt động được hoặc hoạt động không tốt vẫn có thể làm việc được nếu như không đi vào nhánh chết.

Trong thực tế sản xuất một hệ thống có thể đang hoạt động rất tốt, nhưng nếu vì lý do nào đó mà hệ thống phải thay đổi chế độ làm việc (do sự cố từng phần hoặc do thay đổi công nghệ...) thì có thể hệ thống sẽ không hoạt động được nếu đó là nhánh chết.

Với cách phân tích sơ đồ như trên thì khó đánh giá được các mạng có độ phức tạp lớn. Do đó ta phải xét một cách phân tích mạng grafcet là dùng phương pháp giản đồ điểm.

Để thành lập giản đồ điểm ta đi theo các bước sau:

+ Vẽ một ô đầu tiên cho giản đồ điểm, ghi số 0. Xuất phát từ giai đoạn đầu trên grafcet được coi là đang tích cực, giai đoạn này đang có dấu ".", khi có một điều kiện được thực hiện, sẽ có các giai đoạn mới được tích cực thì:

- Đánh dấu "." vào các giai đoạn vừa được tích cực trên grafcet.

- Xoá dấu "." ở giai đoạn hết tích cực trên grafcet.

- Tạo một ô mới trên giản đồ điểm sau điều kiện vừa thực hiện.

- Ghi hết các giai đoạn tích cực của hệ (có dấu ".") vào ô mới vừa tạo.

+ Từ các ô đã thành lập khi một điều kiện nào đó lại được thực hiện thì các giai đoạn tích cực lại được chuyển đổi, ta lại lặp lại bốn bước nhỏ trên.

+ Quá trình cứ như vậy tiếp tục, ta có thể vẽ hoàn thiện được giản đồ điểm (sơ đồ tạo thành mạch liên tục, sau khi kết thúc lại trở về điểm xuất phát) hoặc không vẽ hoàn thiện được. Nhìn vào giản đồ điểm ta sẽ có các kết luận sau:

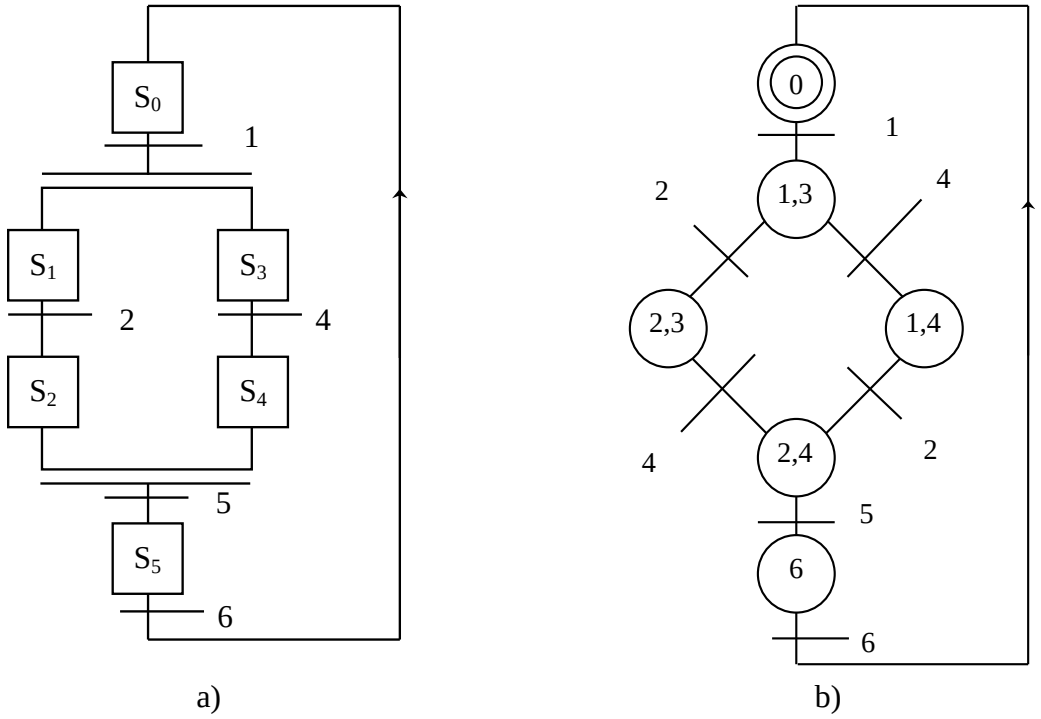
- Nếu trong quá trình vẽ đến giai đoạn nào đó không thể vẽ tiếp được nữa (không hoàn thiện sơ đồ) thì sơ đồ đó là sơ đồ có nhánh chết, ví dụ 2.

- Nếu vẽ được hết mà ở vị trí nào đó có các điểm làm việc cùng tên thì là sơ đồ không sạch ví dụ 3.

- Nếu vẽ được hết và không có vị trí nào có các điểm làm việc cùng tên thì là sơ đồ làm việc tốt, sơ đồ sạch ví dụ 1.

Ví dụ 1: Vẽ giản đồ điểm cho sơ đồ sạch hình 1.17a.

Ở thời điểm đầu, hệ đang ở giai đoạn S_0 (có dấu "."), khi điều kiện 1 được thực hiện thì cả S_1 và S_3 cùng chuyển sang tích cực, đánh dấu "." vào S_1 và S_3, xoá dấu "." ở S_0. Vậy, sau điều kiện 1 ta tạo ô mới và trong ô này ta ghi hai trạng thái tích cực là 1,3. Nếu các điều kiện khác không diễn ra thì mạch vẫn ở trạng thái 1 và 3.



a) b)

Hình 1.17

Khi hệ đang ở 1,3 nếu điều kiện 4 được thực hiện thì giai đoạn 4 tích cực (thêm dấu "."), giai đoạn 3 hết tích cực (mất dấu "."). Vậy sau điều kiện 4 tạo ô mới (nối với ô 1,3), ô này ghi hai trạng thái tích cực còn lại trên grafcet là 1,4.

Khi hệ đang ở 1,3 nếu điều kiện 2 được thực hiện thì giai đoạn 2 tích cực (thêm dấu "."), giai đoạn 1 hết tích cực (mất dấu "."). Vậy sau điều kiện 2 tạo ô mới (nối với ô 1,3), ô này ghi hai trạng thái tích cực còn lại trên grafcet là 2,3.

Khi hệ đang ở 1,4 hoặc 2,3 nếu có điều kiện 5 thì quá trình vẫn không chuyển tiếp vì để chuyển giai đoạn 5 phải có S_2 và S_4 cùng tích cực kết hợp điều kiện 5.

Khi hệ đang ở 1,4 nếu điều kiện 2 được thực hiện thì giai đoạn 2 tích cực (thêm dấu "."), giai đoạn 1 hết tích cực (mất dấu "."). Vậy sau điều kiện 2 tạo ô mới (nối với ô 1,4), ô này ghi hai trạng thái tích cực còn lại trên grafcet là 2,4.

Khi hệ đang ở 2,3 nếu điều kiện 4 được thực hiện thì giai đoạn 4 tích cực (thêm dấu "."), giai đoạn 3 hết tích cực (mất dấu "."). Vậy sau điều kiện 4 tạo ô mới (nối với ô 2,3), ô này ghi hai trạng thái tích cực còn lại trên grafcet là 2,4.

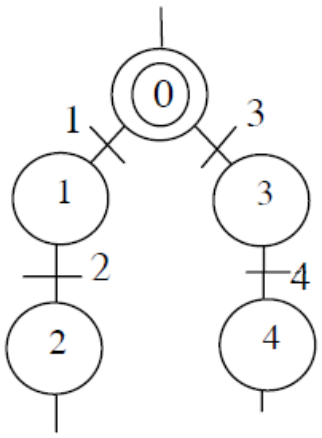
Khi hệ đang ở 2,4 nếu điều kiện 5 được thực hiện thì giai đoạn 5 tích cực (thêm dấu "."), giai đoạn 2 và 4 hết tích cực (mất dấu "."). Vậy sau điều kiện 5 tạo ô mới (nối với ô 2,4), ô này ghi trạng thái tích cực còn lại trên grafcet là 5.

Khi hệ đang ở 5 nếu điều kiện 6 được thực hiện thì giai đoạn 0 tích cực (thêm dấu "."), giai đoạn 5 hết tích cực (mất dấu "."), hệ trở về trạng thái ban đầu.

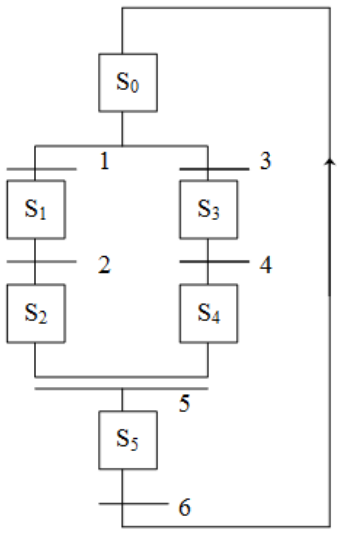
Từ giản đồ điểm ta thấy không có ô nào có 2 điểm làm việc cùng tên và vẽ được cả sơ đồ, vậy đó là sơ đồ sạch.

Ví dụ 2: Vẽ giản đồ điểm cho sơ đồ có nhánh chết hình 1.14

Giản đồ điểm như hình 1.18.



Hình 1.18

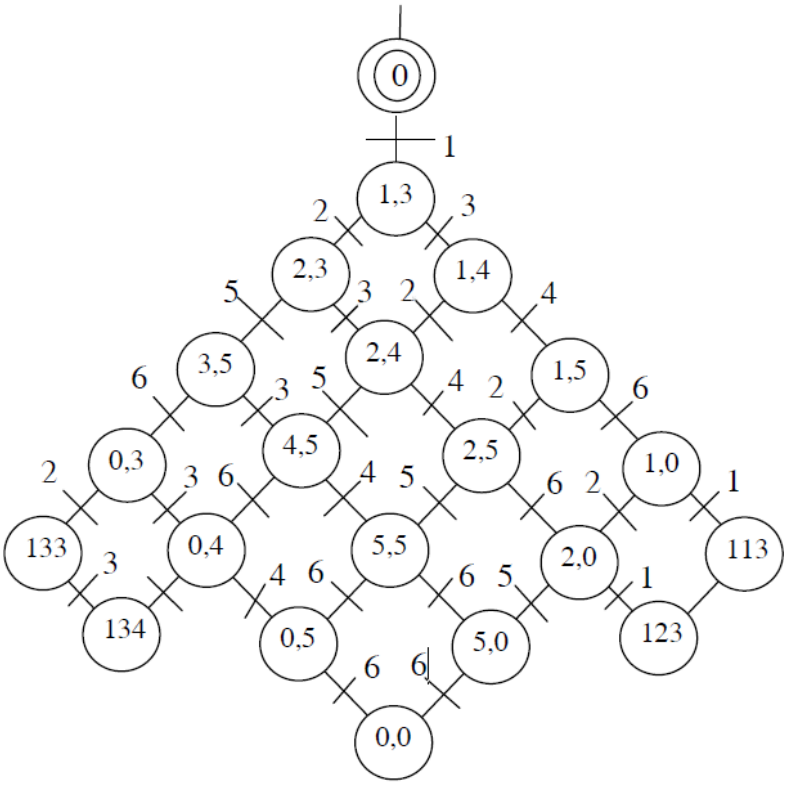


Hình 1.14

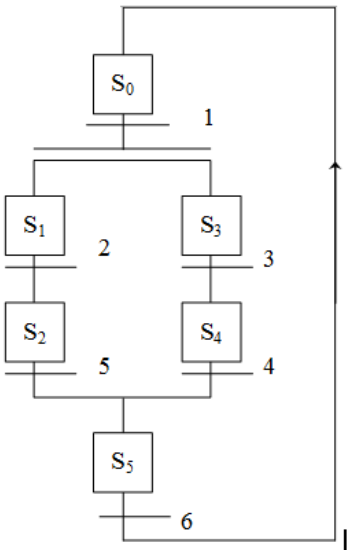
Trong trường hợp này ta không thể vẽ tiếp được nữa vì để $_{S5}$ tích cực phải có cả S_2 và S_4 cùng tích cực cùng điều kiện 5. Nhưng không có ô nào có 2,4.

Ví dụ 3: Vẽ giản đồ điểm cho sơ đồ không sạch hình 1.15

Cách tiến hành vẽ giản đồ điểm như trên, giản đồ điểm như hình 1.19. Từ giản đồ điểm ta thấy có nhiều điểm có 2 điểm làm việc trùng nhau (cùng tên), vậy đó là sơ đồ không sạch. ở giản đồ điểm hình 1.19 có thể tiếp tục vẽ giản đồ sẽ mở rộng.



Hình 1.19



Hình 1.15

Chú ý: Để hệ thống làm việc tốt thì trong mạng grafcet ở một phần mạch nàp đó bắt buộc phải có:

+ Khi mở ra là song song thì kết thúc phải là song song.

+ Khi mở ra là rẽ nhánh thì kết thúc phải là rẽ nhánh.

CHƯƠNG 2: ỨNG DỤNG MẠCH LOGIC TRONG ĐIỀU KHIỂN

2.1. Các thiết bị điều khiển

2.1.1. Các nguyên tắc điều khiển

Quá trình làm việc của động cơ điện để truyền động một máy sản xuất thường gồm các giai đoạn: khởi động, làm việc và điều chỉnh tốc độ, dừng và có thể có cả giai đoạn đảo chiều. Ta xét động cơ là một thiết bị động lực, quá trình làm việc và đặc biệt là quá trình khởi động, hãm thường có dòng điện lớn, tự thân động cơ điện vừa là thiết bị chấp hành nhưng cũng vừa là đối tượng điều khiển phức tạp.

Về nguyên lý khống chế truyền động điện, để khởi động và hãm động cơ với dòng điện được hạn chế trong giới hạn cho phép, ta thường dùng ba nguyên tắc khống chế tự động sau:

- *Nguyên tắc thời gian*: Việc đóng cắt để thay đổi tốc độ động cơ dựa theo nguyên tắc thời gian, nghĩa là sau những khoảng thời gian xác định sẽ có tín hiệu điều khiển để thay đổi tốc độ động cơ. Phần tử cảm biến và khống chế cơ bản ở đây là rơle thời gian.

- *Nguyên tắc tốc độ*: Việc đóng cắt để thay đổi tốc độ động cơ dựa vào nguyên lý xác định tốc độ tức thời của động cơ. Phần tử cảm biến và khống chế cơ bản ở đây là rơle tốc độ.

- *Nguyên tắc dòng điện*: Ta biết tốc độ động cơ do mômen động cơ xác định, mà mômen lại phụ thuộc vào dòng điện chạy qua động cơ, do vậy có thể đo dòng điện để khống chế quá trình thay đổi tốc độ động cơ điện. Phần tử cảm biến và khống chế cơ bản ở đây là rơle dòng điện.

Mỗi nguyên tắc điều khiển đều có ưu nhược điểm riêng, tùy từng trường hợp cụ thể mà chọn các phương pháp cho phù hợp.

2.1.2. Các thiết bị điều khiển

Để điều khiển sự làm việc của các thiết bị cần phải có các thiết bị điều khiển. Để đóng cắt không thường xuyên ta thường dùng áptômát. Trong áptômát hệ thống tiếp điểm có bộ phân dập hồ quang và các bộ phân tự động cắt mạch để bảo vệ quá tải và ngắn mạch. Bộ phận cắt mạch điện bằng tác động điện từ theo kiểu dòng điện cực đại. Khi dòng điện vượt quá trị số cho phép chúng sẽ cắt mạch điện để bảo vệ ngắn mạch, ngoài ra còn có rơle nhiệt bảo vệ quá tải.

Phần tử cơ bản của rơle nhiệt là bản lưỡng kim gồm hai miếng kim loại có độ dãn nở nhiệt khác nhau dán lại với nhau. Khi bản lưỡng kim bị đốt nóng (thường là bằng dòng điện cần bảo vệ) sẽ bị biến dạng (cong), độ biến dạng tới ngưỡng thì sẽ tác động vào các bộ phận khác để cắt mạch điện.

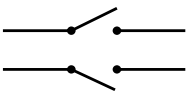
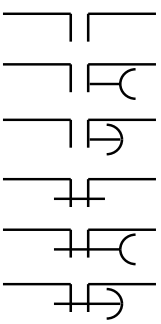
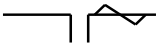
Các rơle điện từ, công tắc tơ tác dụng nhờ lực hút điện từ. Cấu tạo của rơle điện từ thường gồm các bộ phân chính sau: cuộn hút; mạch từ tĩnh làm bằng vật liệu sắt từ; phần động còn gọi là phần ứng và hệ thống các tiếp điểm.

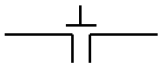
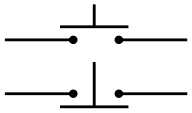
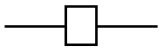
Mạch từ của rơle có dòng điện một chiều chạy qua làm bằng thép khối, còn mạch từ của rơle xoay chiều làm bằng lá thép kỹ thuật điện.

Để chống rung vì lực hút của nam châm điện có dạng xung trên mặt cực người ta đặt vòng ngắn mạch. Sức điện động cảm ứng trong vòng ngắn mạch sẽ tạo ra dòng điện và làm cho từ thông qua vòng ngắn mạch lệch pha với từ thông chính, nhờ đó lực hút phần ứng không bị gián đoạn, các tiếp điểm luôn được tiếp xúc tốt.

Tuỳ theo nguyên lý tác động người ta chế tạo nhiều loại thiết bị điều khiển khác nhau như rơle dòng điện, rơle điện áp, rơle thời gian....

Hệ thống tiếp điểm có cấu tạo khác nhau và thường mạ bạc hay thiếc để đảm bảo tiếp xúc tốt. Các thiết bị đóng cắt mạch động lực có dòng điện lớn, hệ thống tiếp điểm chính có bộ phận dập hồ quang, ngoài ra còn có các tiếp điểm phụ để đóng cắt cho mạch điều khiển. Tuỳ theo trạng thái tiếp điểm người ta chia ra các loại tiếp điểm khác nhau. Một số ký hiệu thường gặp như bảng 2.1.

TT	Tên gọi	Ký hiệu
1	Tiếp điểm cầu dao, máy cắt, áptômát Thường mở Thường đóng	
2	Tiếp điểm công tắctơ, khởi động từ, rơle Thường mở Thường mở khi mở có thời gian Thường mở khi đóng có thời gian Thường đóng Thường đóng khi mở có thời gian Thường đóng khi đóng có thời gian	
3	Tiếp điểm có bộ phận dập hồ quang	

4	Tiếp điểm có bộ phận trả lại vị trí ban đầu bằng tay	
5	Nút ấn thường mở Nút ấn thường đóng	
6	Cuộn dây rơle, công tắc tơ, khởi động từ	
7	Phần tử nhiệt của rơle nhiệt	

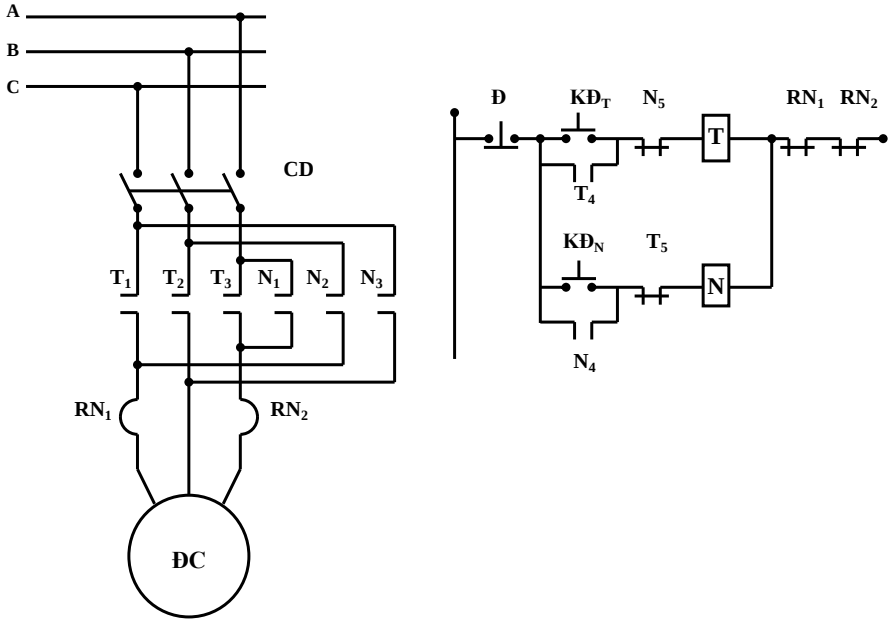
2.2. Các sơ đồ khống chế động cơ rôto lồng sóc

Tuỳ theo công suất và yêu cầu công nghệ mà động cơ không đồng bộ rôto lồng sóc có thể được nối trực tiếp vào lưới điện, dùng đổi nối sao-tam giác, qua điện kháng, qua biến áp tự ngẫu, ngày nay thường dùng các bộ khởi động mềm để khởi động động cơ.

2.2.1. Mạch khống chế đơn giản

Với động cơ công suất nhỏ ta có thể đóng trực tiếp vào lưới điện. Nếu động cơ chỉ quay theo một chiều thì mạch đóng cắt có thể dùng cầu dao, áptômát với thiết bị đóng cắt này có nhược điểm là khi đang làm việc nếu mất điện, thì khi có điện trở lại động có thể tự khởi động. Để tránh điều đó ta dùng khởi động từ đơn để đóng cắt cho động cơ.

Xét sơ đồ đóng cắt có đảo chiều dùng khởi động từ kép như hình 2.1



Hình 2.1: Sơ đồ đóng cắt có đảo chiều

Cầu dao CD trên mạch động lực là cầu dao cách ly (cầu dao này chủ yếu để đóng cắt không tải, để cách ly khi sửa chữa). Các tiếp điểm T_1, T_2, T_3 để đóng động cơ chạy thuận, các tiếp điểm N_1, N_2, N_3 để đóng động cơ chạy ngược (đảo thứ tự hai trong ba pha lưới điện). Các tiếp điểm T_5 và N_5 là các khoá liên động về điện để khống chế các chế độ chạy thuận và ngược không thể cùng đồng thời, nếu đang chạy thuận thì T_5 mở, N không thể có điện, nếu đang chạy ngược thì N_5 mở, T không thể có điện. Ngoài các liên động về điện ở khởi động từ kép còn có liên động cơ khí, khi cuộn T đã hút thì lẫy cơ khí khoá không cho cuộn N hút nữa khi cuộn N đã hút thì lẫy cơ khí khoá không cho cuộn T hút nữa.

Trong mạch dùng hai rơle nhiệt RN_1 và RN_2 để bảo vệ quá tải cho động cơ, khi động cơ quá tải thì rơle nhiệt tác động làm các tiếp điểm của nó bên mạch điều khiển mở, các cuộn hút mất điện cắt điện động cơ.

Để khởi động động cơ chạy thuận (hoặc ngược) ta ấn nút KĐT (hoặc KĐN), cuộn hút T có điện, đóng các tiếp điểm T_1... T_3 cấp điện cho động cơ chạy theo chiều thuận, tiếp điểm T_4 đóng lại để tự duy trì.

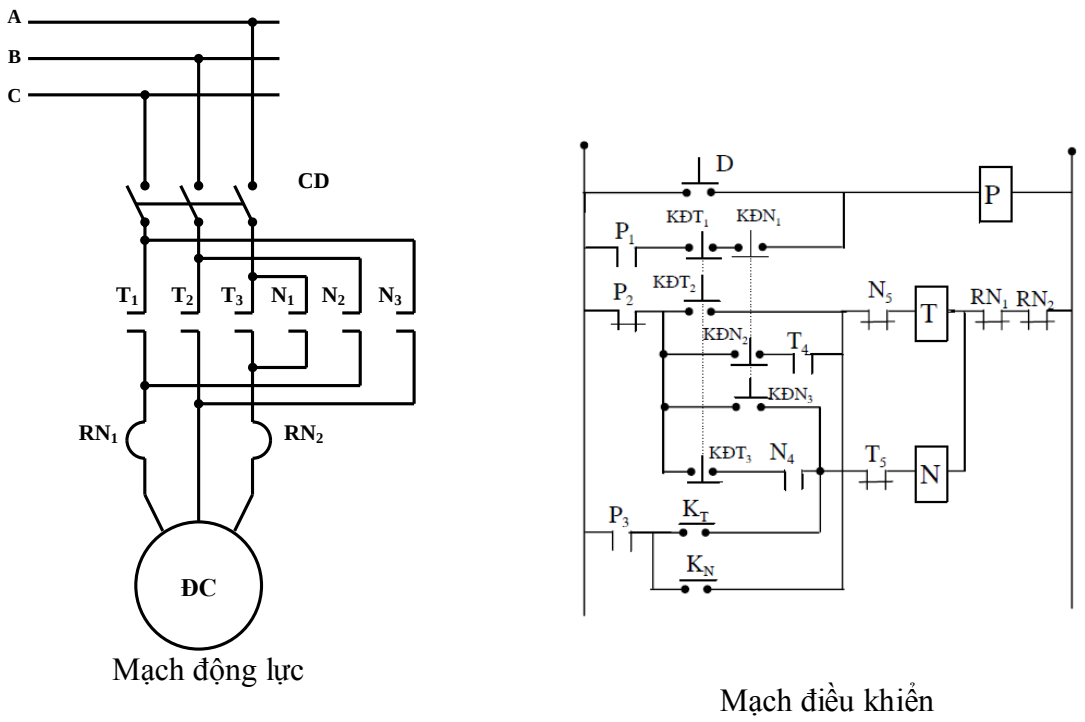
Để dừng động có ta ấn nút dừng D, các cuộn hút mất điện, cắt điện động cơ, động cơ tự dừng.

Để đảo chiều động cơ trước hết ta phải ấn nút dùng D, các cuộn hút mất điện mới ấn nút để đảo chiều.

2.2.2. Mạch khống chế đảo chiều có giám sát tốc độ.

Xét sơ đồ khống chế động cơ lồng sóc quay theo hai chiều và có hãm ngược. Hãm ngược là hãm xảy ra lúc động cơ còn đang quay theo chiều này (do quán tính), nhưng ta lại đóng điện cho động cơ quay theo chiều ngược lại mà không chờ cho động cơ dừng hẳn rồi mới đóng điện cho động cơ đảo chiều. Hãm ngược có khả năng hãm nhanh vì có thể tạo mômen hãm lớn (do sử dụng cả hai nguồn năng lượng là động năng và điện năng tạo thành năng lượng hãm), tuy vậy dòng điện hãm sẽ lớn và trong ứng dụng cụ thể phải lưu ý hạn chế dòng điện hãm này.

Sơ đồ hình 2.2 thực hiện nhiệm vụ đó. Trong sơ đồ có thêm rơle trung gian P. Hai rơle tốc độ (gắn với động cơ), rơle tốc độ thuận có tiếp điểm KT và rơle tốc độ ngược có tiếp điểm KN, các rơle này khi tốc độ cao thì các tiếp điểm rơle kín, tốc độ thấp thì tiếp điểm rơle hở.



Hình 2.2

Khi khởi động chạy thuận ta ấn nút khởi động thuận KĐT, tiếp điểm $KĐT_1$ hở, $KĐT_3$ hở ngăn không cho cuộn hút N và P có điện, tiếp điểm $KĐT_2$ kín cấp điện cho cuộn hút T, các tiếp điểm T_1... T_3 kín cấp điện cho động cơ chạy thuận, Tiếp điểm T_4 kín để tự duy trì, tiếp điểm T_5 hở cấm cuộn N có điện.

Khi đang chạy thuận cần chạy ngược ta ấn nút khởi động ngược KĐN, tiếp điểm KĐN1 hở không cho P có điện, tiếp điểm $KĐN_2$ hở cắt điện cuộn hút T làm mất điện chế độ chạy thuận, tiếp điểm $KĐN_3$ kín cấp điện cho cuộn hút N để cấp điện cho chế độ chạy ngược và tiếp điểm N_4 kín để tự duy trì.

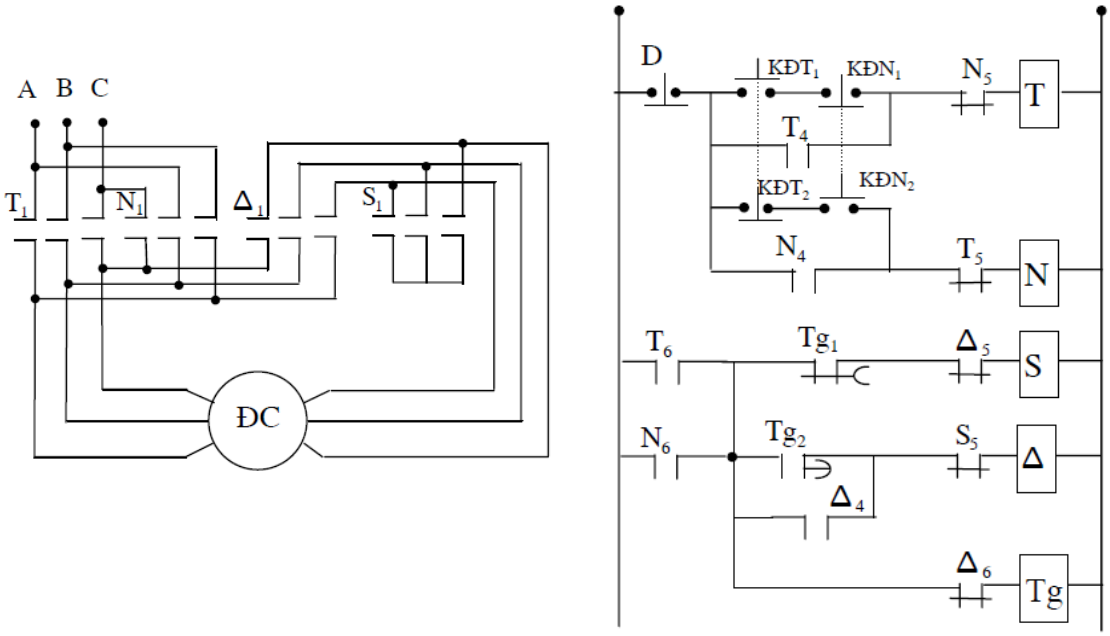
Nếu muốn dừng ta ấn nút dừng D, cấp điện cho cuộn hút P, cuộn hút P đóng tiếp điểm P_1 để tự duy trì, hở P_2 cắt đường nguồn đang cấp cho cuộn hút T hoặc N, nhưng lập tức P_3 kín cuộn hút N hoặc T lại được cấp điện, nếu khi trước động cơ đang chạy thuận (cuộn T làm việc) tốc độ đang lớn thì KT kín, cuộn N được cấp điện đóng điện cho chế độ chạy ngược làm động cơ dừng nhanh, khi tốc độ đã giảm thấp thì KT mở cắt điện cuộn hút N, động cơ dừng hẳn. Khi các rơle nhiệt tác động thì động cơ dừng tự do.

2.2.3. Khống chế động cơ lồng sóc kiểu đổi nối Y/Δ có đảo chiều

Với một số động cơ khi làm việc định mức nối Δ thì khi khởi động có thể nối hình sao làm điện áp đặt vào dây cuốn giảm 3 do đó dòng điện khởi động giảm. Sơ đồ hình 2.3 cho phép thực hiện đổi nối Y/Δ có đảo chiều.

Trong sơ đồ có khởi động từ T đóng cho chế độ chạy thuận, khởi động từ N đóng cho chế độ chạy ngược, khởi động từ S đóng điện cho chế độ khởi động hình sao, khởi động từ Δ đóng điện cho chế độ chạy tam giác. Rơle thời gian Tg để duy trì thời gian, có hai tiếp điểm Tg_1 là tiếp điểm thường kín mở chậm thời gian Δt_1, Tg_2 là tiếp điểm thường mở đóng chậm thời gian Δt_2 với $\Delta t_1 > \Delta t_2$.

Khi cần khởi động thuận ta ấn nút khởi động thuận KĐT, tiếp điểm $KĐT_2$ ngăn không cho cuộn N có điện, tiếp điểm $KĐT_1$ kín đóng điện cho cuộn thuận T, đóng các tiếp điểm T_1...T_3 đưa điện áp thuận vào động cơ, T_4 để tự duy trì, T_5 ngăn không cho N có điện, T_6 cấp điện cho rơle thời gian Tg, đồng thời cấp điện ngay cho cuộn hút S, đóng động cơ khởi động kiểu nối sao, tiếp điểm S_5 mở chưa cho cuộn Δ. Khi Tg có điện thì sau thời gian ngắn Δt_2 thì Tg_2 đóng chuẩn bị cấp điện cho cuộn hút Δ. Sau khoảng thời gian duy trì Δt_1 thì tiếp điểm Tg_1 mở ra cuộn hút S mất điện cắt chế độ khởi động sao của động cơ, tiếp điểm S_5 kín cấp điện cho cuộn hút Δ, đưa động cơ vào làm việc ở chế độ nối tam giác và tự duy trì bằng tiếp điểm Δ_4.



Hình 2.3

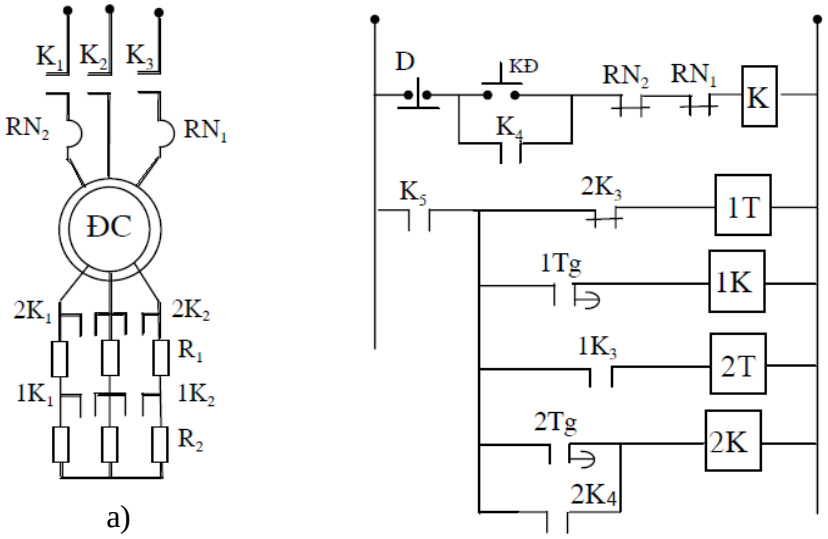
Khi cần đảo chiều (nếu đang chạy thuận) ta ấn nút khởi động ngược KĐN, T mất điện làm T_6 mở quá trình lại khởi động theo chế độ nối sao như trên với cuộn hút N, các tiếp điểm N_1 ... N_3 đổi thứ tự hai trong ba pha (đổi pha A và B cho nhau) làm chiều quay đổi chiều. Khi muốn dứng ta ấn nút dừng D, động cơ dừng tự do.

2.3. Các sơ đồ khống chế động cơ không đồng bộ rôto dây quấn

Các biện pháp khởi động và thay đổi tốc độ như động cơ rôto lồng sóc cũng có thể áp dụng cho động cơ rôto dây quấn. Nhưng như vậy không tận dụng được ưu điểm của động cơ rôto dây quấn là khả năng thay đổi dòng khởi động cũng như thay đổi tốc độ bằng cách thay đổi điện trở phụ mắc vào mạch rôto. Do đó với động cơ rôto dây quấn để giảm dòng khi khởi động cũng như để thay đổi tốc độ động cơ người ta dùng phương pháp thay đổi điện trở phụ mắc vào mạch rôto.

2.3.1. Khởi động động cơ rôto dây quấn theo nguyên tắc thời gian

Cách này thường dùng cho hệ thống có công suất trung bình và lớn. Sơ đồ khống chế như hình 2.4.



Hình 2.4

Trong sơ đồ có 2 rơle nhiệt RN_1 và RN_2 để bảo vệ quá tải cho động cơ, hai rơle thời gian 1Tg và 2Tg với hai tiếp điểm thường mở đóng chậm để duy trì thời gian loại điện trở phụ ở mạch rôto.

Để khởi động ta ấn nút khởi động KĐ cấp điện cho cuộn hút K các tiếp điểm K_1, K_2, K_3 đóng cấp điện cho động cơ, động cơ khởi động với hai cấp điện trở phụ, tiếp điểm K_4 để tự duy trì, tiếp điểm K_5 để cấp điện cho các rơle thời gian.

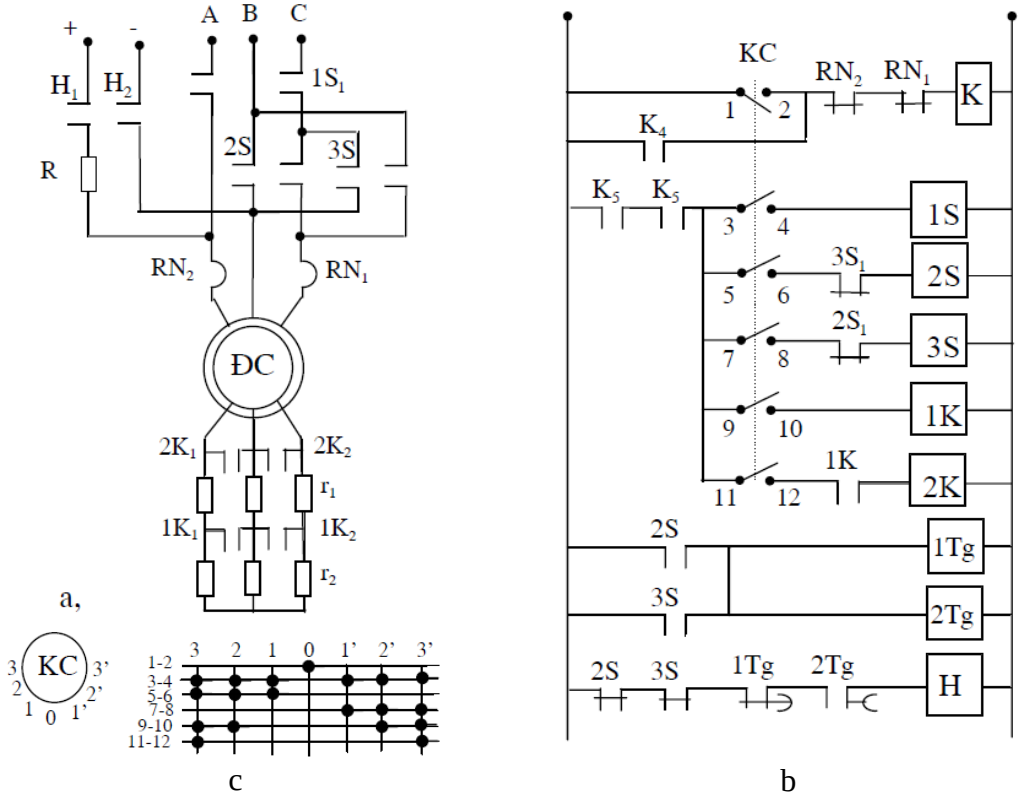
Sau khoảng thời gian chỉnh định tiếp điểm thường mở đóng chậm 1Tg đóng lại cấp điện cho 1K để loại điện trở phụ R_2 ra khỏi mạch rôto, tiếp điểm $1K_3$ đóng để cấp điện cho rơle thời gian 2Tg. Sau thời gian chỉnh định tiếp điểm thường mở đóng chậm 2Tg đóng lại cấp điện cho 2K loại nốt điện trở R_1 khỏi

mạch khởi động, động cơ làm việc trên đặc tính cơ tự nhiên. Tiếp điểm $2K_4$ để tự duy trì, $2K_5$ cắt điện các rơle thời gian.

Khi muốn dừng ấn nút dừng D, động cơ được cắt khỏi lưới và dừng tự do.

2.3.2. Thay đổi tốc độ động cơ rôto dây quấn bằng thay đổi điện trở phụ

Trong công nghiệp có nhiều máy sản xuất dùng truyền động động cơ rôto dây quấn để điều chỉnh tốc độ như cầu trục, máy cán.... và ở đây thường dùng thêm khâu hãm động năng để dừng máy. Hãm động năng là cách hãm sử dụng động năng của động cơ đang quay để tạo thành năng lượng hãm. Với động cơ rôto dây quấn, muốn hãm động năng thì khi đã cắt điện phải nối các cuộn dây stato vào điện áp một chiều để tạo thành từ thông kích thích cho động cơ tạo mômen hãm. Sơ đồ nguyên lý của hệ thống như hình 2.5.



Hình 2.5

Động cơ rôto dây quấn có thể quay theo hai chiều, theo chiều thuận nếu 1S, 2S đóng và theo chiều ngược nếu 1S, 3S đóng. Công tắc tơ H để đóng nguồn một chiều lúc hãm động năng, công tắc tơ 1K, 2K để cắt điện trở phụ trong mạch rôto làm thay đổi tốc độ động cơ khi làm việc. Khi hãm động năng toàn bộ điện trở phụ r_1 và r_2 được đưa vào mạch rôto để hạn chế dòng điện hãm, còn điện trở phụ R trong mạch một chiều để đặt giá trị mô men hãm.

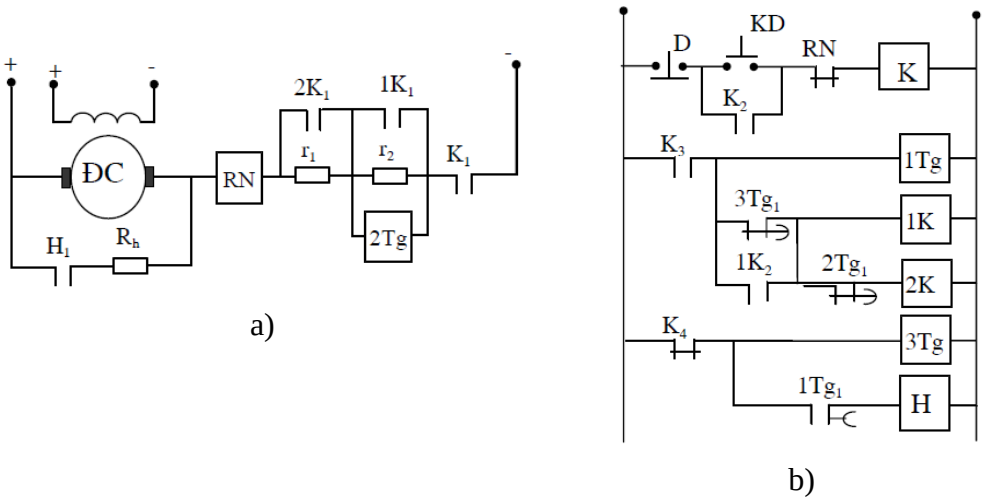
Trong hệ thống có bộ khống chế chỉ huy kiểu chuyển mạch cơ khí KC. Bộ KC có nguyên lý cấu tạo là một trụ tròn cơ khí, có thể quay hai chiều, trên trục có gắn các tiếp điểm động và kết hợp với các tiếp điểm tĩnh tạo thành các cặp tiếp điểm được đóng cắt tuỳ thuộc vào vị trí quay của trụ. Đồ thị đóng mở tiếp điểm của bộ khống chế KC được thể hiện trên hình 2.5c. Ví dụ ở vị trí 0 của bộ khống chế chỉ có tiếp điểm 1-2 đóng, tất cả các vị trí còn lại của các tiếp điểm đều cắt hoặc cặp tiếp điểm 9-10 sẽ đóng ở các vị trí 2, 3 bên trái và 2', 3' bên phải.

Hoạt động của bộ khống chế như sau: Khi đã đóng điện cấp nguồn cho hệ thống. Ban đầu bộ khống chế được đặt ở vị trí 0 công tắc tơ K có điện, các tiếp điểm K ở mạch khống chế đóng lại, chuẩn bị cho hệ thống làm việc. Nếu muốn động cơ quay theo chiều thuận thì ta quay bộ KC về phía trái, nếu muốn động cơ quay ngược thì ta quay bộ KC về phía phải. Giả thiết ta quay bộ KC về vị trí 2 phía trái, lúc này các tiếp điểm 3-4, 5-6, 9-10 của bộ KC kín, các cuộn dây công tắc tơ 1S, 2S, 1K và các rơle thời gian 1Tg, 2Tg có điện, các tiếp điểm 1S, 2S ở mạch động lực đóng lại, cuộn dây xtato được đóng vào nguồn 3 pha, tiếp điểm 1K trong mạch rôto đóng lại cắt phần điện trở phụ r_2 ra, động cơ được khởi động và làm việc với điện trở phụ r_1 trong mạch rôto, tiếp điểm 1Tg mở ra, 2Tg đóng lại chuẩn bị cho quá trình hãm động năng khi dừng. Nếu muốn dừng động cơ thì quay bộ KC về vị trí 0, các công tắc tơ 1S, 2S, 1K và các rơle thời gian 1Tg, 2Tg mất điện, động cơ được cắt khỏi nguồn điện 3 pha với toán bộ điện trở r_1, r_2 được đưa vào rôto, đồng thời tiếp điểm thường kín đóng chậm 1Tg đóng lại (đóng chậm một thời gian ngắn đảm bảo hệ đã được cắt khỏi lưới điện), tiếp điểm thường mở mở chậm 2Tg chưa mở ($\Delta t_2 > \Delta t_1$) công tắc tơ H có điện tiếp điểm H_1, H_2 đóng lại cấp nguồn một chiều cho xtato động cơ và động cơ được hãm động năng. Sau thời gian chỉnh định Δt_2 tiếp điểm thường mở mở chậm mở ra tương ứng với tốc độ động cơ đã đủ nhỏ, cuộn dây H mất điện, nguồn một chiều được cắt khỏi cuộn dây xtato, kết thúc quá trình hãm động năng. Trong thực tế, người ta yêu cầu người vận hành khi quay bộ khống chế KC qua mỗi vị trí phải dừng lại một thời gian ngắn để hệ thống làm việc an toàn cả về mặt điện và cơ.

2.4. Khống chế động cơ điện một chiều

Với động cơ điện một chiều khi khởi động cần thiết phải giảm dòng khởi động. Để giảm dòng khi khởi động có thể đưa thêm điện trở phụ vào mạch phần ứng. Ngày nay nhờ kỹ thuật điện tử và tin học phát triển người ta đã chế tạo các bộ biến đổi một chiều bằng bán dẫn công suất lớn làm nguồn trực tiếp cho động

cơ và điều khiển các bộ biến đổi này bằng mạch số logic khả trình. Các bộ biến đổi này nối trực tiếp vào động cơ, việc khống chế khởi động, hãm và điều chỉnh tốc độ đều thực hiện bằng các mạch số khả trình rất thuận tiện và linh hoạt. Tuy nhiên, một số mạch đơn giản vẫn có thể dùng sơ đồ các mạch logic như hình 2.6



Hình 2.6

Để khởi động động cơ ta ấn nút khởi động KĐ lúc đó công tắc tơ K có điện, các tiếp điểm thường mở K_1 đóng lại để cấp điện cho động cơ với 2 điện trở phụ, K_2 đóng lại để tự duy trì, K_3 đóng lại, K_4 mở ra làm rơle thời gian 3Tg mất điện, sau thời gian chỉnh định tiếp điểm thường đóng đóng chậm 3Tg1 đóng lại làm công tắc tơ 1K có điện, đóng tiếp điểm $1K_1$ loại điện trở phụ r_2 khỏi mạch động cơ và làm rơle thời gian 2Tg mất điện, sau thời gian chỉnh định tiếp điểm thường đóng đóng chậm $2Tg_1$ đóng lại cấp điện cho công tắc tơ 2K đóng tiếp điểm $2K_2$ loại r_1 ra khỏi mạch động lực quá trình khởi động kết thúc.

Để dừng động cơ ta ấn nút dừng D lúc đó công tắc tơ K mất điện, tiếp điểm K_1 ở mạch động lực mở ra cắt phần ứng động cơ khỏi nguồn điện. Đồng thời tiếp điểm K_2, K_3 mở ra làm rơle thời gian 1Tg mất điện bắt đầu tính thời gian hãm, K_4 đóng lại làm công tắc tơ H có điện đóng tiếp điểm H_1 đưa điện trở hãm Rh vào để thực hiện quá trình hãm. Sau thời gian chỉnh định tiếp điểm thường mở mở chậm $1Tg_1$ mở ra, công tắc tơ H mất điện kết thúc quá trình hãm, hệ thống khống chế và mạch động lực trở về trạng thái ban đầu chuẩn bị cho lần khởi động sau.

CHƯƠNG 3: ĐIỀU KHIỂN LOGIC CÓ LẬP TRÌNH

3.1. Mở đầu

Sự phát triển của kỹ thuật điều khiển tự động hiện đại và công nghệ điều khiển logic khả trình dựa trên cơ sở phát triển của tin học mà cụ thể là sự phát triển của kỹ thuật máy tính.

Kỹ thuật điều khiển logic khả trình PLC (Programmable Logic Control) được phát triển từ những năm 1968 -1970. Trong gia đoạn đầu các thiết bị khả trình yêu cầu người sử dụng phải có kỹ thuật điện tử, phải có trình độ cao. Ngày nay các thiết bị PLC đã phát triển mạnh mẽ và có mức độ phổ cập cao.

Thiết bị điều khiển logic lập trình được PLC là dạng thiết bị điều khiển đặc biệt dựa trên bộ vi xử lý, sử dụng bộ nhớ lập trình được để lưu trữ các lệnh và thực hiện các chức năng, chẳng hạn, cho phép tính logic, lập chuỗi, định giờ, đếm, và các thuật toán để điều khiển máy và các quá trình công nghệ. PLC được thiết kế với yêu cầu không cao về kiến thức và ngôn ngữ máy tính và không chỉ các nhà lập trình máy tính mới có thể cài đặt hoặc thay đổi chương trình. Vì vậy, các nhà thiết kế PLC phải lập trình sẵn sao cho chương trình điều khiển có thể nhập bằng cách sử dụng ngôn ngữ đơn giản (ngôn ngữ điều khiển). Thuật ngữ logic được sử dụng vì việc lập trình chủ yếu liên quan đến các hoạt động logic ví dụ nếu có các điều kiện A và B thì C làm việc... Người vận hành nhập chương trình (chuỗi lệnh) vào bộ nhớ PLC, thiết bị điều khiển PLC giám sát các tín hiệu vào, ra theo chương trình và thực hiện các quy tắc điều khiển đã được lập trình.

Các PLC tương tự máy tính, nhưng máy tính được tối ưu hoá cho các tác vụ tính toán và hiển thị, còn PLC được chuyên biệt cho các tác vụ điều khiển và môi trường công nghiệp. Vì vậy các PLC:

+ Được thiết kế bền để chịu được rung động, nhiệt, ẩm và tiếng ồn.

+ Có sẵn giao diện cho các thiết bị vào ra.

+ Được lập trình dễ dàng với ngôn ngữ điều khiển dễ hiểu, chủ yếu giải quyết các phép toán logic và chuyển mạch.

Về cơ bản chức năng của bộ điều khiển logic PLC cũng giống như chức năng của bộ điều khiển thiết kế trên cơ sở các rơle công tắc tơ hoặc trên cơ sở các khối điện tử đó là:

+ Thu thập các tín hiệu vào và các tín hiệu phản hồi từ các cảm biến.

+ Liên kết, ghép nối các tín hiệu theo yêu cầu điều khiển và thực hiện đóng mở các mạch phù hợp với công nghệ.

+ Tính toán và soạn thảo các lệnh điều khiển trên cơ sở so sánh các thông tin thu thập được.

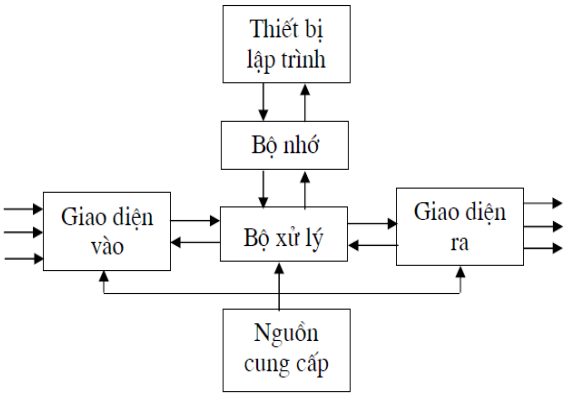
+ Phân phát các lệnh điều khiển đến các địa chỉ thích hợp.

Trong hệ thống trung tâm gia công, mọi quy trình công nghệ đều được bộ PLC điều khiển tập trung.

3.2. Các thành phần cơ bản của một bộ PLC

3.2.1. Cấu hình phần cứng

Hệ thống PLC thông dụng có năm bộ phận cơ bản gồm: bộ xử lý, bộ nhớ, bộ nguồn, giao diện vào/ra và thiết bị lập trình. Sơ đồ hệ thống như hình 3.1



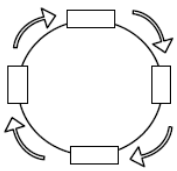
Hình 3.1

a. Bộ xử lý

Bộ xử lý còn gọi là bộ xử lý trung tâm (CPU), là linh kiện chứa bộ vi xử lý. Bộ xử lý biên dịch các tín hiệu vào và thực hiện các hoạt động điều khiển theo chương trình được lưu trong bộ nhớ của CPU, truyền các quyết định dưới dạng tín hiệu hoạt động đến các thiết bị ra.

Nguyên lý làm việc của bộ xử lý tiến hành theo từng bước tuần tự, đầu tiên các thông tin lưu trữ trong bộ nhớ chương trình được gọi lên tuần tự và được kiểm soát bởi bộ đếm chương trình. Bộ xử lý liên kết các tín hiệu và đưa kết quả ra đầu ra. Chu kỳ thời gian này gọi là thời gian quét (scan). Thời gian vòng quét phụ thuộc vào tầm vóc của bộ nhớ, vào tốc độ của CPU. Nói chung chu kỳ một vòng quét như hình 3.2

4. Chuyển dữ liệu từ bộ đệm ảo ra TB ngoại vi



1. Nhập dữ liệu từ TB ngoại vi vào bộ đệm

3. Truyền thông và kiểm tra lỗi

2. Thực hiện chương trình

Hình 3.2

Sự thao tác tuần tự của chương trình dẫn đến một thời gian trễ trong khi bộ đếm của chương trình đi qua một chu trình đầy đủ, sau đó bắt đầu lại từ đầu.

Để đánh giá thời gian trễ người ta đo thời gian quét của một chương trình dài 1Kbyte và coi đó là chỉ tiêu để so sánh các PLC. Với nhiều loại thiết bị thời gian trễ này có thể tới 20ms hoặc hơn. Nếu thời gian trễ gây trở ngại cho quá trình điều khiển thì phải dùng các biện pháp đặc biệt, chẳng hạn như lặp lại những lần gọi quan trọng trong thời gian một lần quét, hoặc là điều khiển các thông tin chuyển giao để bỏ bớt đi những lần gọi ít quan trọng khi thời gian quét dài tới mức không thể chấp nhận được. Nếu các giải pháp trên không thoả mãn thì phải dùng PLC có thời gian quét ngắn hơn.

b. Bộ nguồn

Bộ nguồn có nhiệm vụ chuyển đổi điện áp AC thành điện áp thấp cho bộ vi xử lý (thường là 5V) và cho các mạch điện trong các module còn lại (thường là 24V).

c. Thiết bị lập trình

Thiết bị lập trình được sử dụng để lập các chương trình điều khiển cần thiết sau đó được chuyển cho PLC. Thiết bị lập trình có thể là thiết bị chuyên dụng, có thể là thiết bị cầm tay gọn nhẹ, có thể là phần mềm được cài đặt trên máy tính cá nhân.

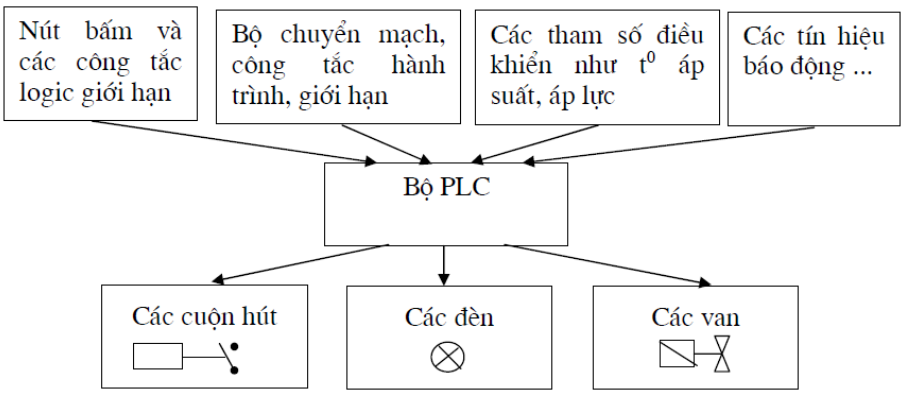
d. Bộ nhớ

Bộ nhớ là nơi lưu giữ chương trình sử dụng cho các hoạt động điều khiển. Các dạng bộ nhớ có thể là RAM, ROM, EPROM. Người ta luôn chế tạo nguồn dự phòng cho RAM để duy trì chương trình trong trường hợp mất điện nguồn, thời gian duy trì tuỳ thuộc vào từng PLC cụ thể. Bộ nhớ cũng có thể được chế tạo thành module cho phép dễ dàng thích nghi với các chức năng điều khiển có kích cỡ khác nhau, khi cần mở rộng có thể cắm thêm.

e. Giao diện vào/ra

Giao diện vào là nơi bộ xử lý nhận thông tin từ các thiết bị ngoại vi và truyền thông tin đến các thiết bị bên ngoài. Tín hiệu vào có thể từ các công tắc, các bộ cảm biến nhiệt độ, các tế bào quang điện....

Tín hiệu ra có thể cung cấp cho các cuộn dây công tắc tơ, các rơle, các van điện từ, các động cơ nhỏ... Tín hiệu vào/ra có thể là tín hiệu rời rạc, tín hiệu liên tục, tín hiệu logic... Các tín hiệu vào/ra có thể thể hiện như hình 3.3.

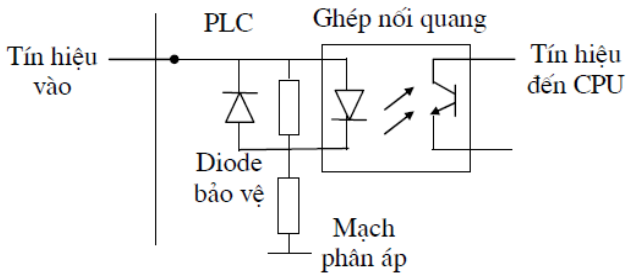


Hình 3.3

Mỗi điểm vào ra có một địa chỉ duy nhất được PLC sử dụng.

Các kênh vào/ra đã có các chức năng cách ly và điều hoá tín hiệu sao cho các bộ cảm biến và các bộ tác động có thể nối trực tiếp với chúng mà không cần thêm mạch điện khác.

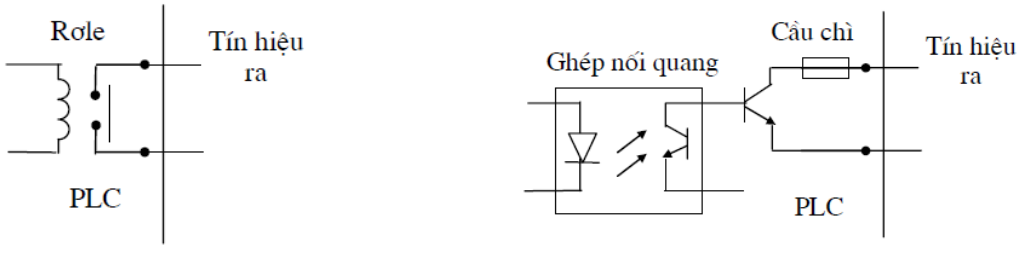
Tín hiệu vào thường được ghép cách điện (cách ly) nhờ linh kiện quang như hình 3.4.



Hình 3.4

Dải tín hiệu nhận vào cho các PLC cỡ lớn có thể là 5v, 24v, 110v, 220v. Các PLC cơ nhỏ thường chỉ nhập tín hiệu 24v.

Tín hiệu ra cũng được ghép cách ly, có thể cách ly kiểu rơle như hình 3.5a, cách ly kiểu quang như hình 3.5b.

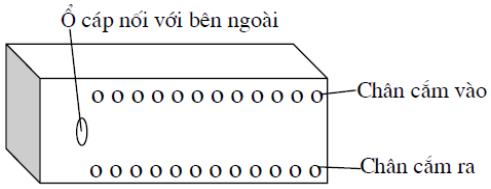


Hình 3.5

Tín hiệu ra có thể là tín hiệu chuyển mạch 24V, 100mA; 110V, 1A một chiều; thậm chí 240V, 1A xoay chiều tuỳ loại PLC. Tuy nhiên, với PLC cỡ lớn dải tín hiệu ra có thể thay đổi bằng cách lựa chọn các module ra thích hợp.

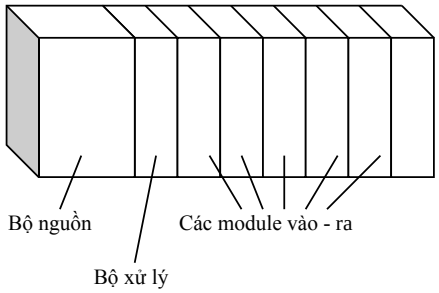
3.2.2. Cấu tạo chung của PLC

Các PLC có hai kiểu cấu tạo cơ bản là: kiểu hộp đơn và kiểu modulle nối ghép. Kiểu hộp đơn thường dùng cho các PLC cỡ nhỏ và được cung cấp dưới dạng nguyên chiếc hoàn chỉnh gồm bộ nguồn, bộ xử lý, bộ nhớ và các giao diện vào/ra. Kiểu hộp đơn thường vẫn có khả năng ghép nối được với các module ngoài để mở rộng khả năng của PLC. Kiểu hộp đơn như hình 3.6.



Hình 3.6

Kiểu module gồm các module riêng cho mỗi chức năng như module nguồn, module xử lý trung tâm, module ghép nối, module vào/ra, module mờ, module PID... các module được lắp trên các rãnh và được kết nối với nhau.



Hình 3.7

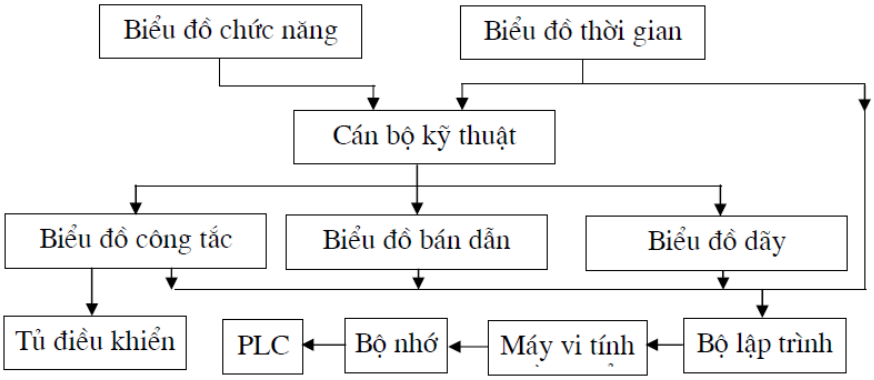
Kiểu cấu tạo này có thể được sử dụng cho các thiết bị điều khiển lập trình với mọi kích cỡ, có nhiều bộ chức năng khác nhau được gộp vào các module riêng biệt. Việc sử dụng các module tuỳ thuộc công dụng cụ thể. Kết cấu này khá linh hoạt, cho phép mở rộng số lượng đầu nối vào/ra bằng cách bổ sung các module vào/ra hoặc tăng cường bộ nhớ bằng cách tăng thêm các đơn vị nhớ.

3.3. Các vấn đề về lập trình

3.3.1. Khái niệm chung

Một PLC có thể sử dụng một cách kinh tế hay không phụ thuộc rất lớn vào thiết bị lập trình. Khi trang bị một bộ PLC thì đồng thời phải trang bị một thiết bị lập trình của cùng một hãng chế tạo. Tuy nhiên, ngày nay người ta có thể lập trình bằng phần mềm trên máy tính sau đó chuyển sang PLC bằng mạch ghép nối riêng. Sự khác nhau chính giữa bộ điều khiển khả trình PLC và công nghệ rơle hoặc bán dẫn là ở chỗ kỹ thuật nhập chương trình vào bộ điều khiển như thế nào.

Trong điều khiển rơle, bộ điều khiển được chuyển đổi một cách cơ học nhờ đấu nối dây "điều khiển cứng". Còn với PLC thì việc lập trình được thực hiện thông qua một thiết bị lập trình và một ngoại vi chương trình. Có thể chỉ ra qui trình lập trình theo giản đồ hình 3.8.



Hình 3.8

Để lập trình người ta có thể sử dụng một trong các mô hình sau đây:

+ Mô hình dãy

+ Mô hình các chức năng

+ Mô hình biểu đồ nối dây

+ Mô hình logic

Việc lựa chọn mô hình nào trong các mô hình trên cho thích hợp là tuỳ thuộc vào loại PLC và điều quan trọng là chọn được loại PLC nào cho phép giao lưu tiện lợi và tránh được chi phí không cần thiết. Đa số các thiết bị lưu hành trên thị trường hiện nay là dùng mô hình dãy hoặc biểu đồ nối dây. Những PLC hiện đại cho phép người dùng chuyển từ một phương pháp nhập này sang một phương pháp nhập khác ngay trong quá trình nhập.

Trong thực tế khi sử dụng biểu đồ nối dây thì việc lập trình có vẻ đơn giản hơn vì nó có cách thể hiện gần giống như mạch rơle công tắc tơ. Tuy nhiên, với những người đã có sẵn những hiểu biết cơ bản về ngôn ngữ lập trình thì lại cho

rằng dùng mô hình dãy dễ dàng hơn, đồng thời với các mạch cỡ lớn thì dùng mô hình dãy có nhiều ưu điểm hơn.

Mỗi nhà chế tạo đều có những thiết kế và phương thức thao tác thiết bị lập trình riêng, vì thế khi có một loại PLC mới thì phải có thời gian và cần phải được huấn luyện để làm quen với nó.

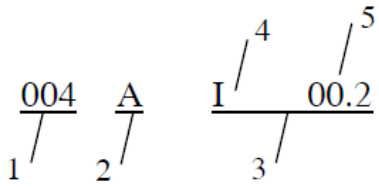
3.3.2. Các phương pháp lập trình

Từ các cách mô tả hệ tự động các nhà chế tạo PLC đã soạn thảo ra các phương pháp lập trình khác nhau. Các phương pháp lập trình đều được thiết kế đơn giản, gần với các cách mô tả được biết đến. Từ đó nói chung có ba phương pháp lập trình cơ bản là phương pháp bảng lệnh STL, phương pháp biểu đồ bậc thang LAD và phương pháp lưu đồ điều khiển CSF. Trong đó, hai phương pháp bảng lệnh STL và biểu đồ bậc thang LAD được dùng phổ biến hơn cả.

Một số ký hiệu chung

Cấu trúc lệnh:

Một lệnh thường có ba phần chính và thường viết như hình 3.9



Hình 3.9

1. Địa chỉ tương đối của lệnh (khi lập trình thiết bị lập trình tự đưa ra)

2. Phần lệnh là nội dung thao tác mà PLC phải tác động lên đối, trong lập trình LAD phần này tự thể hiện trên thanh LAD, không được ghi ra.

3. Đối tượng lệnh, là phần mà lệnh tác động theo yêu cầu điều khiển, trong đối tương lệnh lại có hai phần:

4. Loại đối tượng, có trường hợp sau loại đối tượng có dấu ":", loại đối tượng như tín hiệu vào, tín hiệu ra, cờ (rơle nội)...

5. Tham số của đối tượng lệnh để xác định cụ thể đối tượng, cách ghi tham số cũng phụ thuộc từng loại PLC khác nhau.

Ký hiệu thường có trong mỗi lệnh:

Các ký hiệu trong lệnh, qui ước cách viết với mỗi quốc gia có khác nhau, thậm chí mỗi hãng, mỗi thời chế tạo của hãng có thể có các ký hiệu riêng. Tuy nhiên, cách ghi chung nhất cho một số quốc gia là:

• Mỹ:

+ Ký hiệu đầu vào là I (In), đầu ra là Q (out tránh nhầm O là không)

+ Các lệnh viết gần đủ tiếng Anh ví dụ ra là out

+ Lệnh ra (gán) là out

+ Tham số của lệnh dùng cơ số 10

+ Phía trước đối tượng lệnh có dấu %

+ Giữa các số của tham số không có dấu chấm.

Ví dụ: AND% I09; out%Q10.

• Nhật:

+ Đầu vào ký hiệu là X, đầu ra ký hiệu là Y

+ Các lệnh hầu như được viết tắt từ tiếng anh

+ Lệnh ra (gán) là out

+ Tham số của lệnh dùng cơ số 8.

Ví dụ: A X 10; out Y 07

• Tây đức

+ Đầu vào ký hiệu là I, đầu ra ký hiệu là Q

+ Các lệnh hầu như được viết tắt từ tiếng Anh

+ Lệnh ra (gán) là =

+ Tham số của lệnh dùng cơ số 8

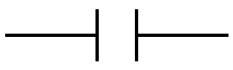
+ Giữa các số của tham số có dấu chấm để phân biệt khe và kênh

Ví dụ: A I 1.0; = Q 0.7

Ngoài các ký hiệu khá chung như trên thì mỗi hãng còn có các ký hiệu riêng, có bộ lệnh riêng. Ngay cùng một hãng ở các thời chế tạo khác nhau cũng có đặc điểm khác nhau với bộ lệnh khác nhau. Do đó, khi sử dụng PLC thì mỗi loại PLC ta phải tìm hiểu cụ thể hướng dẫn sử dụng của nó. Một số ký hiệu khác nhau với các lệnh cơ bản được thể hiện rõ trên bảng 3.1

a. Phương pháp hình thang LAD (Ladder Logic)

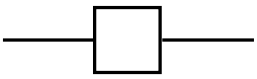
Phương pháp hình thang có dạng của biểu đồ nút bấm. Các phần tử cơ bản của phương pháp hình thang là:

+ Tiếp điểm: Thường mở 

Thương kín 

+ Cuộn dây (mô tả các rơle) 

+ Hộp (mô tả các hàm khác nhau, các lệnh đặc biệt)



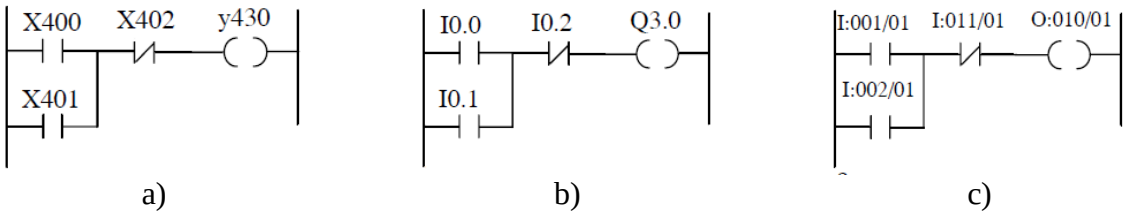
Bảng 3.1

IEC 1131-3	Misubishi	OMRON	Siemens	Teleme-canique	Spreher và Schuh	Chú thích
LD	LD	LD	A	L	STR	Khởi đầu với tiếp điểm thường mở
LDN	LDI	LD NOT	AN	LN	STR NOT	Khởi đầu với tiếp điểm thường kín
AND	AND	AND	A	A	AND	Phần tử nối tiếp có tiếp điểm mở
ANDN	ANI	AND NOT	AN	AN	AND NOT	Phần tử nối tiếp có tiếp điểm đóng
O	OR	OR	O	O	OR	Phần tử song song có tiếp điểm mở
ORN	ORI	OR NOT	ON	ON	OR NOT	Phần tử song song có tiếp điểm đóng
ST	OUT	OUT	=	=	OUT	Lấy tín hiệu ra

Mạng LAD là đường nối các phần tử thành một mạch hoàn chỉnh, theo thứ tự từ trái sang phải, từ trên xuống dưới. Quá trình quét của PLC cũng theo thứ tự này. Mỗi một nấc thang xác định một số hoạt động của quá trình điều khiển.

Một sơ đồ LAD có nhiều nấc thang. Trên mỗi phần tử của biểu đồ hình thang LAD có các tham số xác định tuỳ thuộc vào ký hiệu của từng hãng sản xuất PLC.

Ví dụ: một nấc của phương pháp hình thang như hình 3.10



a) b) c)

Hình 3.10. Phương pháp lập trình thang LAD

Hình 3.10a là kiểu ký hiệu của Misubishi (Nhật)

Hình 3.10b là kiểu ký hiệu của Siemens (Tây đức)

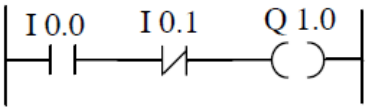
Hình 3.10c là ký hiệu của Allen Bradley

b. Phương pháp liệt kê lệnh STL (Statement List)

Phương pháp STL gần với biểu đồ logic. ở phương pháp này các lệnh được liệt kê thứ tự. Tuy nhiên, để phân biệt các đoạn chương trình người ta thường dùng các mã nhớ, mỗi mã nhớ tương ứng với một nấc thang của biểu đồ hình thang. Để khởi đầu mỗi đoạn (tương ứng như khởi đầu một nấc thang) ta sử dụng các lệnh khởi đầu như LD, L, A, O... (bảng 3.1). Kết thúc mỗi đoạn thường là lệnh gán cho đầu ra, đầu ra có thể là đầu ra cho thiết bị ngoại vi có thể là đầu ra cho các rơle nội.

Ví dụ:

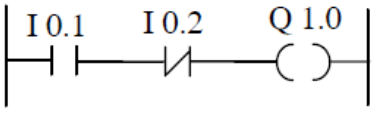
Một đoạn STL của PLC S5 (Siemens)



Hình 3.11

0	A	I 0.0
1	A	I 0.1
2	=	Q 1.0

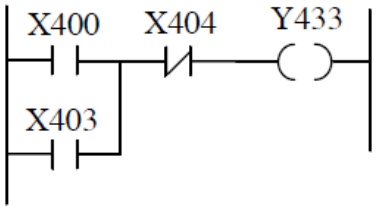
Một đoạn STL của PLC S7-200 (Siemens)



Hình 3.12

0	LD	I 0.1
1	A	I 0.2
3	=	Q 1.0

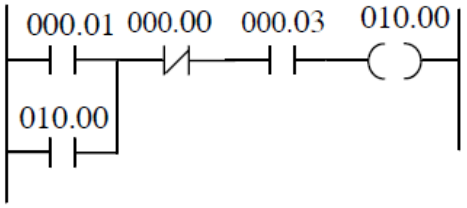
Một đoạn STL của PLC MELSEC F1 (Nhật)



Hình 3.13

0	LD	X 400
1	O	X 403
2	ANI	X 404
3	OUT	Y 433

Một đoạn STL của CPM1A (OMRON)

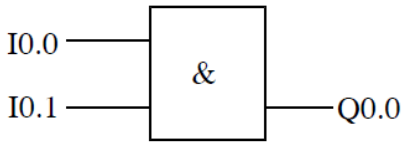


Hình 3.14

0	LD	000.01
1	OR	010.00
2	AND NOT	000.00
3	AND	000.03
4	OUT	010.00

c. Phương pháp lưu đồ điều khiển CSF (Control System Flow)

Phương pháp lưu đồ điều khiển CSF trình bày các phép toán logic với các ký hiệu đồ hoạ đã được tiêu chuẩn hoá như hình 3.15. Phương pháp lưu đồ điều khiển thích hợp với người đã quen với phép tính điều khiển bằng đại số Booole.



Hình 3.15

3.3.3. Các rơle nội

Trong các loại PLC có nhiều thuật ngữ dùng để chỉ các linh kiện loại này, ví dụ: rơle phụ, bộ vạch dấu, cờ hiệu, lưu trữ bít, bit nhớ...

Đây là linh kiện cung cấp các chức năng đặc biệt gắn liền với PLC và được dùng phổ biết trong lập trình. Rơle nội này tương tự như các rơle trung gian trong sơ đồ rơle công tắc tơ.

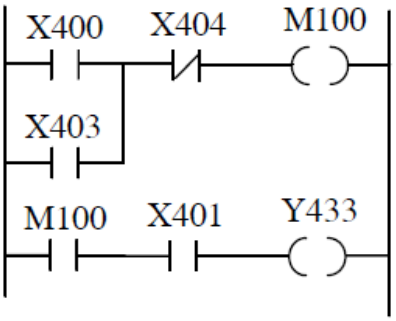
Rơle nội cũng được coi là các đầu ra để nhận các lệnh gán đầu ra, nhưng thực chất đầu ra này không đưa ra ngoài (không phải thiết bị ngoại vi) mà chỉ nằm nội tại trong PLC. PLC nhỏ có thể có tới hàng trăm rơle nội, các rơle nội đều được nuôi bằng nguồn dự phòng khi mất điện.

Một số ký hiệu các rơle nội:

Hãng	Tên gọi	Ký hiệu	Ví dụ
Misubishi	Rơle phụ hoặc bộ đánh dấu	M	M100; M101
Siemens	Cờ hiệu	F	F0.0; F0.1
Sprecher và Schuh	Cuộn dây	C	C001; C002
Telemecanique	Bit	B	B0; B1
Toshiba	Rơle nội	R	R000; R001
Bradley	Lưu trữ bit	B	B3/001; B3/002

Ví dụ: sử dụng rơle nội (của Misibishi)

0	LD	X 400
1	OR	X 403
2	ANI	X 404
3	OUT	M 100
4	LD	M 100
5	AND	X 401
6	OUT	Y 433



Hình 3.16

3.3.4. Các rơle thời gian

Trong các hệ thống điều khiển luôn luôn phải sử dụng rơle thời gian để duy trì thời gian cho quá trình điều khiển. Trong các PLC người ta cũng gắn các rơle thời gian vào trong đó. Tuy nhiên, thời gian ở đây được xác định nhờ đồng hồ trong CPU. Các rơle thời gian cũng có các tên gọi khác nhau nhưng thường gọi nhất là bộ thời gian (Time).

Các nhà sản xuất PLC không thống nhất về cách lập trình cho các rơle thời gian này. Mỗi loại PLC (thậm chí trong cùng hãng) cũng có các ký hiệu và cách lập trình rất khác nhau cho rơle thời gian. Số lượng rơle thời gian trong mỗi PLC cũng rất khác nhau.

Điểm chung nhất đối với các rơle thời gian là các hãng đều coi rơle thời gian là các đầu ra nội, do đó rơle thời gian là đầu ra của nấc thang, hay của một đoạn chương trình.

3.3.5. Các bộ đếm

Bộ đếm cho phép đếm tần suất xuất hiện tín hiệu vào. Bộ đếm có thể được dùng trong trường hợp đếm các sản phẩm di chuyển trên băng chuyền và số sản phẩm xác định cần chuyển vào thùng. Bộ đếm có thể đếm số vòng quay của trục, hoặc số người đi qua cửa. Các bộ đếm này được cài đặt sẵn trong PLC.

Có hai loại bộ đếm là bộ đếm tiến và bộ đếm lùi. Các nhà sản xuất PLC cũng sử dụng các bộ đếm theo những cách có khác nhau. Tuy nhiên, cũng như các bộ thời gian, bộ đếm cũng được coi là đầu ra của PLC và đây cũng là đầu ra nội, để xuất tín hiệu ra ngoài phải qua đầu ra ngoại vi (có chân nối ra ngoài PLC).

3.4. Đánh giá ưu nhược điểm của PLC

Trước đây, bộ PLC thường rất đắt, khả năng hoạt động bị hạn chế và qui trình lập trình phức tạp. Vì những lý do đó mà PLC chỉ được dùng trong những nhà máy và các thiết bị đặc biệt. Ngày nay do giảm giá liên tục, kèm theo tăng khả năng của PLC dẫn đến kết quả là ngày càng được áp dụng rộng rãi cho các thiết bị máy móc. Các bộ PLC đơn khối với 24 kênh đầu vào và 16 kênh đầu ra thích hợp với các máy tiêu chuẩn đơn, các trang thiết bị liên hợp. Còn các bộ PLC với nhiều khả năng ứng dụng và lựa chọn được dùng cho những nhiệm vụ phức tạp hơn.

Có thể kể ra các ưu điểm của PLC như sau:

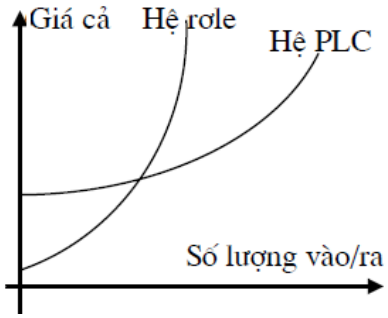
+ Chuẩn bị vào hoạt động nhanh: Thiết kế kiểu module cho phép thích nghi nhanh với mọi chức năng điều khiển. Khi đã được lắp ghép thì PLC sẵn

sàng làm việc ngay. Ngoài ra nó còn được sử dụng lại cho các ứng dụng khác dễ dàng.

+ Độ tin cậy cao: Các linh kiện điện tử có tuổi thọ dài hơn các thiết bị cơ-điện. Độ tin cậy của PLC ngày càng tăng, bảo dưỡng định kỳ thường không cần thiết còn với mạch rơle công tắc tơ thì việc bảo dưỡng định kỳ là cần thiết.

+ Dễ dàng thay đổi chương trình: Những thay đổi chương trình được tiến hành đơn giản. Để sửa đổi hệ thống điều khiển và các quy tắc điều khiển đang được sử dụng, người vận hành chỉ cần nhập tập lệnh khác, gần như không cần mắc nối lại dây (tuy nhiên, có thể vẫn phải nối lại nếu cần thiết). Nhờ đó hệ thống rất linh hoạt và hiệu quả.

+ Đánh giá nhu cầu đơn giản: Khi biết các đầu vào và các đầu ra thì có thể đánh giá được kích cỡ yêu cầu của bộ nhớ hay độ dài chương trình. Do đó, có thể dễ dàng và nhanh chóng lựa chọn PLC phù hợp với các yêu cầu công nghệ đặt ra.



Hình 3.17

+ Khả năng tái tạo: Nếu dùng nhiều PLC với qui cách kỹ thuật giống nhau thì chi phí lao động sẽ giảm thấp hơn nhiều so với bộ điều khiển rơle. Đó là do giảm phần lớn lao động lắp ráp.

+ Tiết kiệm không gian: PLC đòi hỏi ít không gian hơn so với bộ điều khiển rơle tương đương.

+ Có tính chất nhiều chức năng: PLC có ưu điểm chính là có thể sử dụng cùng một thiết bị điều khiển cơ bản cho nhiều hệ thống điều khiển.

Người ta thường dùng PLC cho các quá trình tự động linh hoạt vì dễ dàng thuận tiện trong tính toán, so sánh các giá trị tương quan, thay đổi chương trình và thay đổi các thông số.

+ Về giá trị kinh tế: Khi xét về giá trị kinh tế của PLC ta phải đề cập đến số lượng đầu ra và đầu vào. Quan hệ về giá thành với số lượng đầu vào/ra có

dạng như hình 3.17. Như vậy, nếu số lượng đầu vào/ra quá ít thì hệ rơle tỏ ra kinh tế hơn, những khi số lượng đầu vào/ra tăng lên thì hệ PLC kinh tế hơn hẳn.

Khi tính đến giá cả của PLC thì không thể không kể đến giá của các bộ phân phụ không thể thiếu như thiết bị lập trình, máy in, băng ghi... cả việc đào tạo nhân viên kỹ thuật. Nói chung những phần mềm để thiết kế lập trình cho các mục đích đặc biệt là khá đắt. Ngày nay nhiều hãng chế tạo PLC đã cung cấp chọn bộ đóng gói phần mềm đã được thử nghiệm, nhưng việc thay thế, sửa đổi các phần mềm là nhu cầu không thể tránh khỏi, do đó, vẫn cần thiết phải có kỹ năng phần mềm.

Phân bố giá cả cho việc lắp đặt một PLC thường như sau:

- 50% cho phần cứng của PLC
- 10% cho thiết kế khuân khổ chương trình
- 20% cho soạn thảo và lập trình
- 15% cho chạy thử nghiệm
- 5% cho tài liệu.

Việc lắp đặt một PLC tiếp theo chỉ bằng khoảng 1/2 giá thành của bộ đầu tiên, nghĩa là hầu như chỉ còn chi phí phần cứng.

Có thể so sánh hệ điều khiển rơle và hệ điều khiển PLC như sau:

• Hệ rơle: + Nhiều bộ phận đã được chuẩn hoá
+ ít nhạy cảm với nhiễu
+ Kinh tế với các hệ thống nhỏ
+ Thời gian lắp đặt lâu
+ Thay đổi khó khăn
+ Khó theo dõi và kiểm tra các hệ thống lớn, phức tạp
+ Cần bảo quản thường xuyên
+ Kích thước lớn

• Hệ PLC + Thay đổi dễ dàng qua công nghệ phích cắm
+ Lắp đặt đơn giản
+ Thay đổi nhanh qui trình điều khiển
+ Kích thước nhỏ
+ Có thể nối với mạng máy tính
+ Giá thành cao

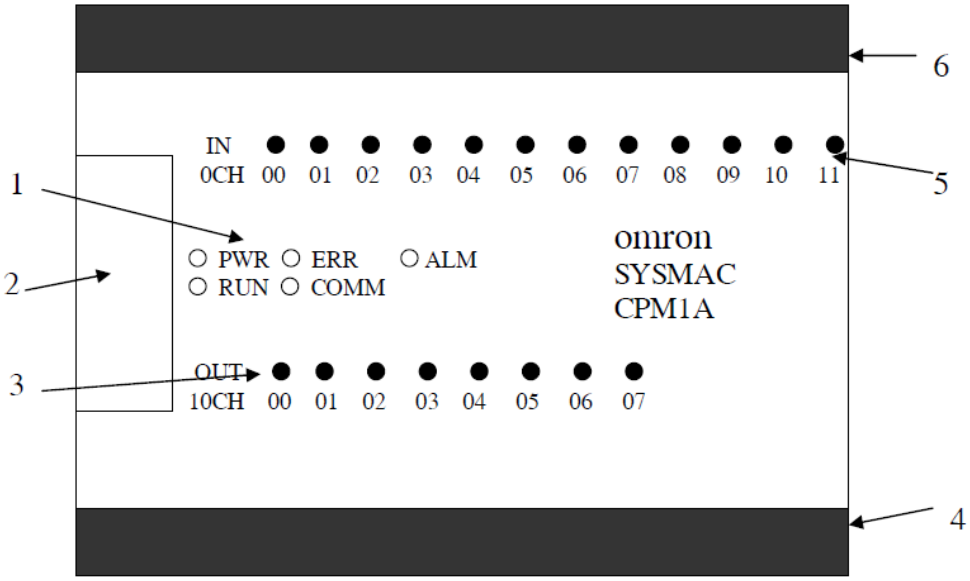
+ Bộ thiết bị lập trình thường đắt, sử dụng ít.

CHƯƠNG 4: BỘ ĐIỀU KHIỂN PLC - CPM1A

4.1. Cấu hình cứng

4.1.1. Cấu tạo của họ PLC - CPM1A.

PLC - CPM1A thuộc họ OMRON do Nhật bản sản xuất. Đây là loại PLC đơn khối có thể lắp ghép thêm các module và lắp ghép nhiều PLC với nhau. Đơn vị cơ bản của PLC CPM1A như hình 4.1



Hình 4.1. Hình khối mặt trước PLC CPM1A

Trong đó:

1. Các đèn báo hệ thống:

+ Đèn PWR (xanh): báo nguồn.

+ Đèn RUN (xanh): PLC đang ở chế độ chạy hoặc kiểm tra, (đèn tắt thì PLC đang ở chế độ lập trình hoặc có lỗi).

+ Đèn ERR/ALM (đỏ): + sáng: Có lỗi, PLC không hoạt động.

+ Nhấp nháy, hoặc tắt: PLC đang hoạt động.

+ COMM (da cam): Dữ liệu đang được truyền tới cổng ngoại vi.

2. Cổng ghép nối với máy tính hoặc thiết bị lập trình (có nắp đậy).

3. Các đèn chỉ thị và địa chỉ ra, (sáng nếu có tín hiệu ra).

4. Chân nối cho đầu ra (có nắp đậy).

5. Các đèn chỉ thị và địa chỉ vào, (sáng nếu có tín hiệu vào).

6. Chân nối cho đầu vào (có nắp đậy).

4.1.2. Các thông số kỹ thuật

a. Các loại CPM1A

Trong họ CPM1A có các PLC sau:

Mã hiệu	Nguồn cung cấp	Số đầu vào	Số đầu ra	Tổng số I/O
CPM1A-10CDR-A	AC	6	4	10
CPM1A-10CDR-D	DC			
CPM1A-20CDR-A	AC	12	8	20
CPM1A-20CDR-D	DC			
CPM1A-30CDR-A	AC	18	12	30
CPM1A-30CDR-D	DC			
CPM1A-40CDR-A	AC	24	16	40
CPM1A-40CDR-D	DC			

b. Thông số chung

Mục		10-đầu I/O	20-đầu I/O	30-đầu I/O	40-đầu I/O
Điện áp cung cấp	Kiểu AC	100 đến 240v AC, 50/60 Hz			
	Kiểu DC	24v DC			
Phạm vi điện áp	Kiểu AC	85 đến 264 v AC			
	Kiểu DC	20,4 đến 26,4v DC			
Tiêu thụ điện	Kiểu AC	max 30 VA		max 60 VA	
	Kiểu DC	max 6 W		max 20 W	
Dòng điện		max 30 A		max 60 A	
Nguồn cấp ra (chỉ có kiểu AC)	Áp	24 VDC			
	Dòng	200 mA		300 mA	
Điện trở cách ly		20 MΩ min. (tại 500v DC) giữa cực AC và cực tiếp địa			
Độ bền xung lực		147m/s2 (20G) ba lần mỗi chiều X, Y và Z			
Nhiệt độ môi trường		Nhiệt độ làm việc: 0 đến 55C Nhiệt độ bảo quản:-20 đến 75C			
Độ ẩm môi trường		10% to 90% (with no condensation)			
Môi trường làm việc		Không làm việc trong môi trường khí đốt			
Thời gian cho gián đoạn nguồn		Kiểu AC: min 10ms; Kiểu DC: min 2ms. (Thời gian gián đoạn tính khi nguồn nhỏ hơn 85% định mức)			
Trọng lượng CPU	Kiểu AC	Max 400 g	Max 500 g	Max 600 g	Max 700 g
	Kiểu DC	Max 300 g	Max 400 g	Max 500 g	Max 600 g

c. Các đặc trưng

Mục		10-đầu I/O	20-đầu I/O	30-đầu I/O	40-đầu I/O
Độ dài lệnh		Từ 1 đến 5 từ cho 1 lệnh			
Kiểu lệnh		Lệnh cơ bản: 14; lệnh đặc biệt: 77 kiểu, tổng 135 lệnh			
Thời gian thực hiện		Lệnh cơ bản: 0.72 đến 16.2 às Lệnh đặc biệt: 12.375 às (lệnh MOV)			
Dung lượng chương trình		2,048 từ (Words)			
Vào ra cực đại	Chỉ CPU	6 input 4 output	12 input 8 output	18 input 12 output	24 input1 16 output
	Có module mở rộng			54 input 36 output	60 input 40 output
Vào dạng bit		00000 đến 00915 (Words 0 đến 9)			
Ra dạng bit		01000 đến 01915 (Words 10 to 19)			
Từ bit (vùng IR)		512 bits: IR20000 to 23115 (words IR 200 to IR 231)			
Bit đặc biệt (vùng SR)		384 bits: SR 23200 to 25515 (words SR 232 to IR 255)			
Bit tạm thời (vùngTR)		8 bits (TR0 to TR7)			
Bit giữ (vùng HR)		320 bits: HR 0000 to HR 1915 (words HR 00 to HR 19)			
Bit bổ trợ (Vùng AR)		256 bits:AR 0000 to AR 1515 (words AR 00 to AR 15)			
Bit liên kết (vùng LR)		256 bits: LR 0000 to LR 1515 (words LR 00 to LR 15)			
Timers/Counters		128 Timers/counters (TIM/CNT 000 to TIM/CNT 127) 100-ms Timers: TIM 000 to TIM 127 10-ms Timers: TIM 00 to TIM 127			
Nhớ dữ liệu		Read/Write:1,024 words (DM 0000 to DM 1023) Read-only: 512 words (DM 6144 to DM 6655)			
Xử lý ngắt		2 điểm (thời gian phản ứng: Max 0.3 ms.)	4 điểm (thời gian phản ứng: Max: 0.3 ms)		
Bảo vệ bộ nhớ		HR, AR, Số liệu trong vùng nhớ nội dung và số đếm được bảo vệ khi nguồn bị gián đoạn.			
Sao lưu bộ nhớ		Tụ điện dự phòng: số liệu nhớ (đọc/viết), bit giữ, bít nhớ bổ trợ, bộ đếm (20 ngày trong điều kiện nhiệt độ 250C)			
Chức năng tự chuẩn đoán		CPU bị hỏng, I/O lỗi đường dẫn, lỗi bộ nhớ			
Chương trình kiểm tra		Không có lệnh kết thúc, lỗi của chương trình (liên tục kiểm tra trong thời gian làm việc)			
Bộ đếm tốc độ cao		1 bộ: 5 kHz 1 pha, hoặc 2.5 kHz 2 pha Kiểu tăng dần: 0 đến 65, 535 (16 bits) Kiểu tăng/giảm: -32,767 đến 32,767 (16 bits)			
Nhập hằng số thờigian		Có thể đặt 1 ms, 2 ms, 4 ms, 8 ms, 16 ms, 32 ms, 64 ms, hoặc 128 ms			
Đặt tín hiệu Analog		2 đường (0 to 200 BCD)			

d. Cấu trúc vùng nhớ

Dữ liệu		Từ	Bit	Chức năng
IR	Vào	IR 000 ÷ IR 009 (10 words)	IR 00000 ÷ IR 00915 (160 bits)	Các bit này có thể làm việc ở vùng vào ra mở rộng
	Ra	IR 010 ÷ IR 019 (10 words)	IR 01000 ÷ IR 01915 (160 bits)	
	làm việc	IR 200 ÷ IR 231 (32 words)	IR 20000 ÷ IR 23115 (512 bits)	Các từ bit này có thể sử dụng tuỳ ý trong chương trình
SR		SR 232 ÷ SR 255 (24 words)	SR 232 ÷ SR 255 (24 words)	SR 23200 đến 25515 (384 bits)
TR				TR 0 đến TR 7 (8 bits)
HR		HR 00 ÷ HR 19 (20 words)	HR 00 ÷ HR 19 (20 words)	HR 0000 đến HR 1915 (320 bits)
AR		AR 00 ÷ HR 15 (16 words)	AR 00 ÷ HR 15 (16 words)	AR 0000 đến HR 1515 (256 bits)
LR		LR 00 ÷ LR 15 (16 words)	LR 00 ÷ LR 15 (16 words)	LR 00000 đến LR 1515 (256 bits)
Timer/ couter		TC 000 ÷ TC 127 (timer/counter)		Số giống nhau sử dụng cho cả time và couter
DM	Đọc/ viết	DM 0000 ÷ DM 0999 DM 1022 ÷ DM 1023 (1,002 words)		DM là dữ liệu chỉ truy cấp dạng từ. Các dữ liệu dạng từ được cất giữ khi mất nguồn
	Ghi lỗi	DM 1000 ÷ DM 1021 (22 words)		Sử dụng để ghi thời gian sự cố và lỗi xuất hiện. Từ đây có thể đọc/ghi khi lỗi xuất hiện
	Chỉ đọc	DM 6144 ÷ DM 6599 (456 words)		Không thể ghi đè lên chương trình
	Cài đặt PC	DM 6600 ÷ DM 6655 (56 words)		Sử dụng đến nhiều vùng tham số để điều khiển làm việc của PC

Chú ý: 1. Bit IR và LR khi chưa sử dụng cho các chức năng chính thì có thể sử dụng như bit làm việc.

2. Nội dung của vùng HR, LR, Counter, và vùng đọc/ghi DM có thể được lưu giữ bằng tụ điện ở nhiệt độ 250C, với thời gian 20 ngày.

3. Khi truy nhập các số PV, TC thì dữ liệu dạng từ; khi truy cấp vào cờ thì dữ liệu dạng bit.

4. Dữ liệu trong DM 6144 đến DM 6655 không thể ghi đè từ chương trình nhưng có thể thay đổi từ thiết bị ngoài "Peripheral Device".

e. Cực vào ra - các bit vùng IR cho vào ra mở rộng

Bảng sau cho biết các bit vùng IR dùng cho module vào ra mở rộng của CPM1A và các loại module mở rộng.

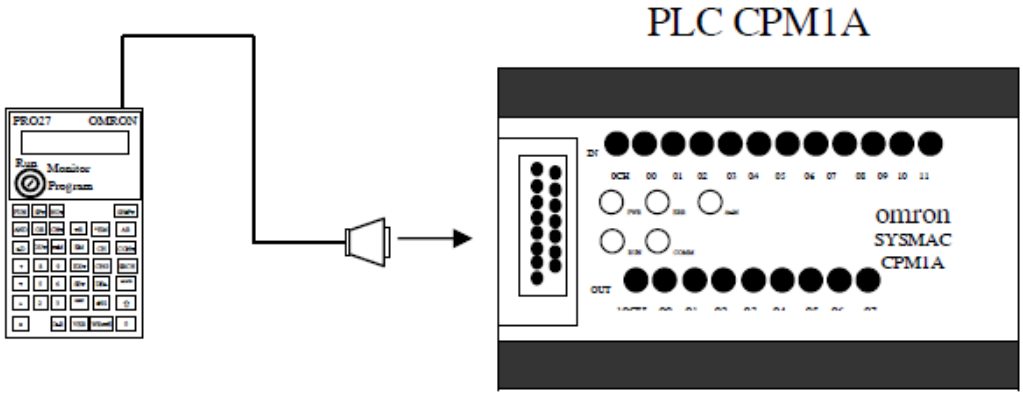
Số vào/ra của CPU	Điểm nối CPU (địa chỉ)		Điểm nối vùng mở rộng (địa chỉ)		Nguồn	Số module
	Vào	Ra	Vào	Ra		
10	6 điểm: 00000 ÷ 00005	4 điểm: 01000 ÷ 01003			AC	CPM1A_10CDR-A
					DC	CPM1A_10CDR-D
20	12 điểm: 00000 ÷ 00011	8 điểm: 01000 ÷ 01007			AC	CPM1A_20CDR-A
					DC	CPM1A_20CDR-D
30	18 điểm: 00000 ÷ 00011 00100 ÷ 00105	12 điểm: 01000 ÷ 01007 01100 ÷ 01103	36 điểm: 00200 ÷ 00211 00300 ÷ 00311 00400 ÷ 00411	24 điểm: 01200 ÷ 01207 01300 ÷ 01307 01400 ÷ 01407	AC	CPM1A_30CDR-A
					DC	CPM1A_30CDR-D
40	20 điểm: 00000 ÷ 00011 00100 ÷ 00111	16 điểm: 01000 ÷ 01007 01100 ÷ 01107			AC	CPM1A_40CDR-A
					DC	CPM1A_40CDR-D

4.2. Ghép nối

PLC CPM1A có thể ghép nối với 32 bộ PLC cùng loại thành hệ thống. Để lập trình cho PLC thì có thể ghép nối nó với thiết bị lập trình cầm tay, bộ lập trình chuyên dụng hoặc máy tính tương thích.

4.2.1. Kết nối với thiết bị lập trình cầm tay

Ta nối trực tiếp cáp của thiết bị cầm tay vào PLC như hình 4.2

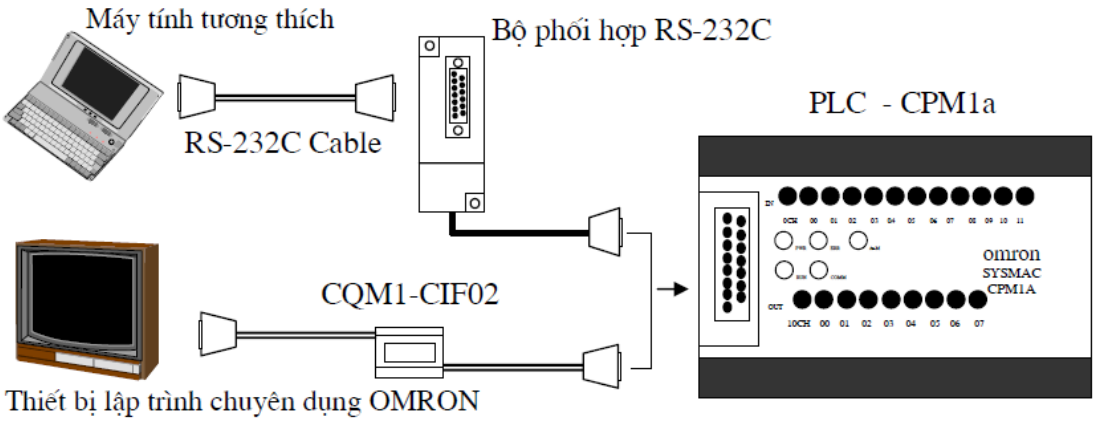


Hình 4.2: Ghép nối PLC với thiết bị lập trình cầm tay

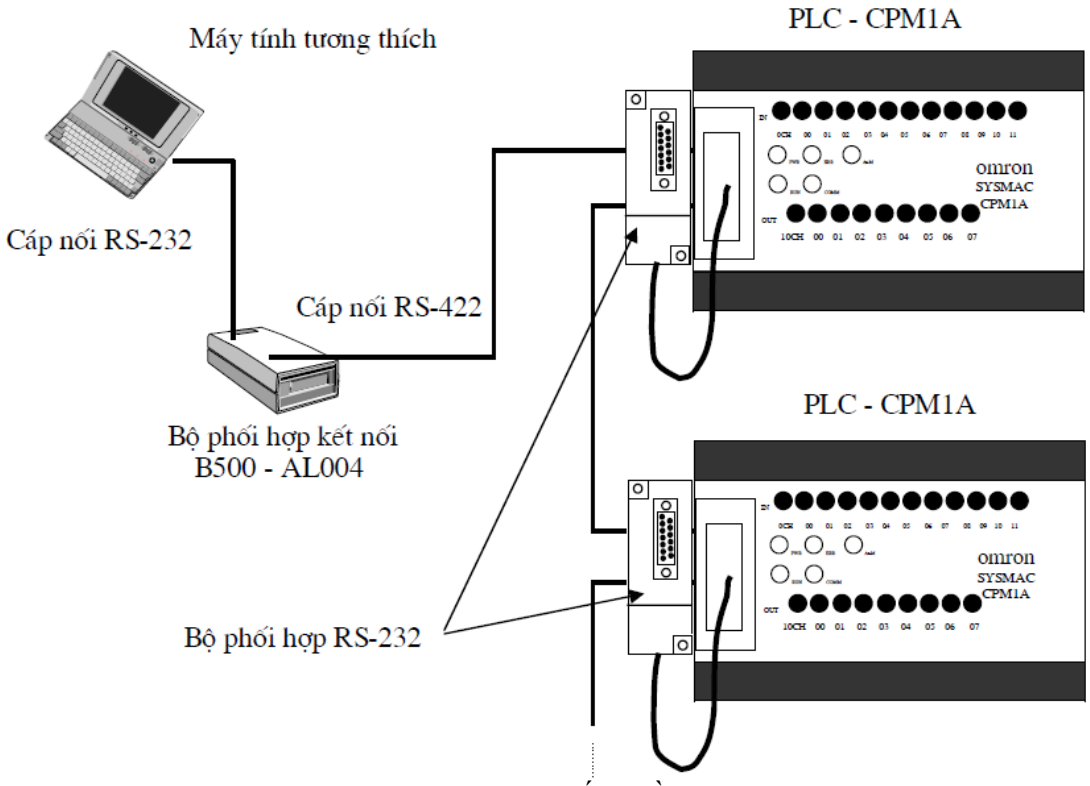
4.2.2. Kết nối với thiết bị lập trình chuyên dụng hoặc máy tính tương thích

Khi ghép nối với máy tính tương thích ta dùng cáp nối chuẩn RS-232C và bộ phối hợp RS-232 (hoặc RS-422) hoặc cáp chuyển đổi loại CQM1-CIF02 khi ghép nối với thiết bị lập trình chuyên dụng như hình 4.3. PLC được ghép nối với cổng nối tiếp (COM) của máy tính.

4.2.3. Kết nối nhiều PLC và máy tính



Hình 4.3: Ghép nối với lập trình chuyên dụng hoặc PC



Hình 4.4: Ghép nối nhiều PLC

Có thể ghép thành hệ thống nhờ nối các PLC - CPM1A với nhau, số PLC - CPM1A có thể ghép tối đa là 32, hệ thống này có thể nối với máy tính tương thích. Sơ đồ như hình 4.4. Chiều dài lớn nhất cho phép của cáp RS-422 là 500m.

4.3. Ngôn ngữ lập trình

4.3.1. Cấu trúc chương trình PLC CPM1A.

Các chương trình điều khiển với PLC CPM1A có thể được viết ở dạng đơn khối hoặc đa khối.

Chương trình đơn khối

Chương trình đơn khối chỉ viết cho các công việc tự động đơn giản, các lệnh được viết tuần tự trong một khối. Khi viết chương trình đơn khối người ta dùng khối OB1. Bộ PLC quét khối theo chương trình, sau khi quét đến lệnh cuối cùng nó quay trở lại lệnh đầu tiên.

Chương trình đa khối (có cấu trúc)

Khi nhiệm vụ tự động hoá phức tạp người ta chia chương trình điều khiển ra thành từng phần riêng gọi là khối. Chương trình có thể xếp lồng khối này vào khối kia. Chương trình đang thực hiện ở khối này có thể dùng lệnh gọi khối để sang làm việc với khối khác, sau khi đã kết thúc công việc ở khối mới nó quay về thực hiện tiếp chương trình đã tạm dừng ở khối cũ.

4.3.2. Bảng lệnh của PLC - PCM1A

Xem phần "Bảng lệnh"

4.3.3. Lập trình các lệnh logic cơ bản của PLC - PCM1A

Với PLC này có:

12 đầu vào với địa chỉ xác định từ 000.00 đến 000.11

8 đầu ra với địa chỉ xác định từ 010.00 đến 010.07

Khi lập trình phần mềm lập trình đã tự hiểu các địa chỉ trên, không cần đưa khái niệm để phân biệt vào/ra.

Nếu đưa thêm khái niệm vào/ra (X/Y) máy sẽ không chấp nhận.

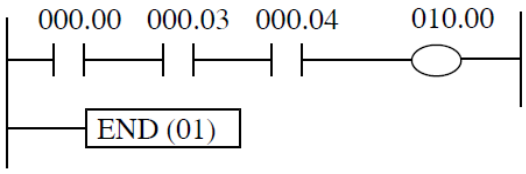
Kết thúc chương trình phải có lệnh kết thúc END chương trình mới chạy.

a. Lệnh AND

Lập trình dạng LAD (có thể lập trình dạng STL và kiểm tra lại dạng LAD).

LD 000.00

```
AND     000.03
AND     000.04
OUT     010.00
```



Hình 4.5 - Lệnh AND

+ Xem lại chương trình từ biểu tượng (phần phụ lục 1)

+ Chọn trạng thái MONITOR hoặc trạng thái PROGRAM (STOP/PRG) nhờ Shift + F10 hoặc biểu tượng "PLC Mode". Đổ chương trình sang PLC từ biểu tượng hoặc từ đường dẫn (như phụ lục 1).

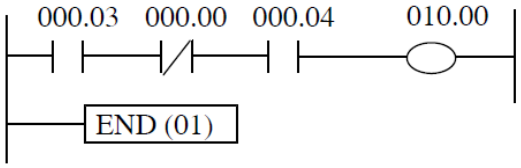
+ Chọn trạng thái MONITOR hoặc trạng thái RUN nhờ Shift + F10 hoặc biểu tượng "PLC Mode" để chạy chương trình.

Quan sát các kết quả.

b. Lệnh AND NOT

Dạng STL

```
LD      000.03
AND NOT 000.00
AND     000.04
OUT     010.01
END
```

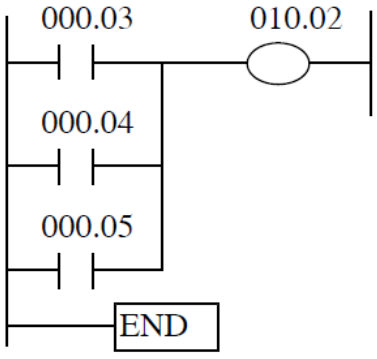


Hình 4.6 - Lệnh AND NOT

c. Lệnh OR

Dạng STL

```
LD      000.03
OR      000.04
OR      000.05
OUT     010.02
END
```



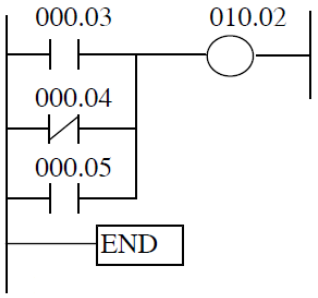
Hình 4.7 - Lệnh OR

d. Lệnh OR NOT

Dạng STL

```
LD 00.03
OR NOT 00.04
```

OR 000.05
OUT 010.02
END

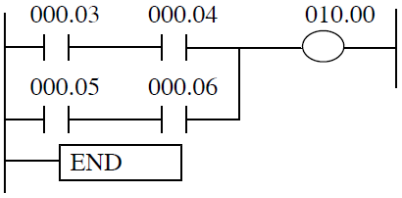


Hình 4.8 - Lệnh OR NOT

e. Lệnh OR giữa hai lệnh AND

Dạng STL.

```
LD      000.03
AND     000.04
LD      000.05
AND     000.06
OR LD
OUT     010.00
END
```

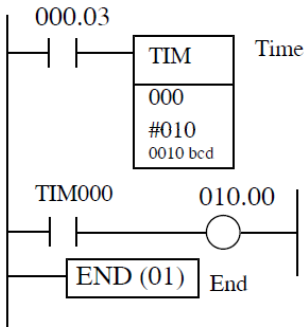


Hình 4.9 - Lệnh OR và AND

g. Lệnh thời gian trễ

Dạng STL

```
LD      000.03
TIM     000 #010
LD TIM  000
OUT     010.00
END
```



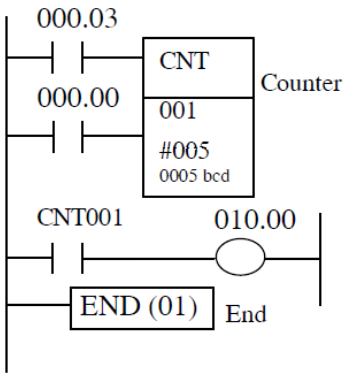
Hình 4.10 - Lệnh thời gian

Chú ý: + Trong lệnh (TIM 000 #010) loạt số đầu chỉ số hiệu của rơle thời gian (rơle thời gian số 0), loạt số thứ hai chỉ thời gian đặt (10s).

+ Khi đầu vào 000.03 có giá trị 1 thì bộ thời gian bắt đầu tính thời gian, khi đủ 10s thì bộ thời gian cho giá trị ra, tức đầu ra 010.00 có giá trị 1.

h. Bộ đếm

```
LD      000.03
LD      000.00
CNT     000 #005
LD      CNT000
OUT     010.00
END
```



Hình 4.11 - Bộ đếm

Chú ý: + Đầu vào thứ nhất (000.03) là đầu vào đếm, mỗi khi đầu vào này nhận giá trị 1 thì bộ đếm đếm một lần.

+ Đầu vào thứ hai (000.00) là đầu vào reset bộ đếm, khi đầu vào này nhận giá trị 1 thì bộ đếm bị reset về trạng thái ban đầu

+ Trong lệnh (CNT 001 #005) loạt số đầu chỉ số hiệu của bộ đếm (bộ đếm số 1), loạt số thứ hai chỉ số đếm đã đặt (5 số), khi đầu vμo 000.03 đạt năm lần giá trị 1 thì bộ đếm cho giá trị ra, tức đầu ra 010.00 có giá trị 1.

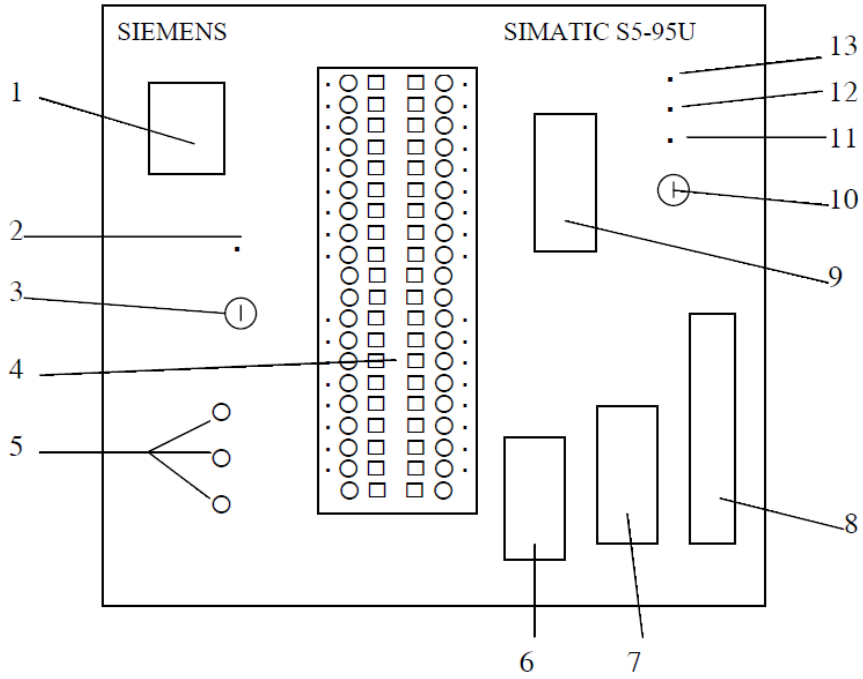
CHƯƠNG 5: BỘ ĐIỀU KHIỂN PLC - S5

5.1. Cấu tạo của họ PLC Step5

PLC Step 5 thuộc họ Simatic do hãng Siemens sản xuất. Đây là loại PLC hỗn hợp vừa đơn khối vừa đa khối. Cấu tạo cơ bản của loại PLC này là một đơn vị cơ bản sau đó có thể ghép thêm các modue mở rộng về phía bên phải, có các module mở rộng tiêu chuẩn S5-100U. Những module ngoài này bao gồm những đơn vị chức năng mà có thể tổ hợp lại cho phù hợp với những nhiệm vụ kỹ thuật cụ thể.

5.1.1. Đơn vị cơ bản

Đơn vị cơ bản của PLC S5- 95U như hình 5.1



Hình 5.1 - Hình khối mặt trước PLC S5-95U siemens Simatic S5-95U

Trong đó:

1. Ngăn để ắc qui

2. Mở điện ắc qui

3. Công tắt mở nguồn.

4. Bảng ổ cắm và đèn báo cho đầu vào và ra logic, có: 16 đầu vào từ I32.0 đến I33.7; 16 đầu ra từ Q32.0 đến Q33.7

5. Đầu nối nguồn 24v cho khối cơ bản.

6. Giao diện cho đầu vào bộ ngắt IW59.0 đến IW59.3 và đầu vào bộ đếm IW36 đến IW38.

7. Giao diện nối tiếp với máy lập trình hoặc máy tính.

8. Giao diện tiếp nhận module nhớ ngoài.

9. Giao diện cho đầu vào ra analog.

10. Công tắc chọn chế độ RUN, STOP,

11. Đèn báo chế độ STOP.

12. Đèn báo chế độ RUN.

13. Đèn báo lỗi.

5.1.2. Các module vào ra mở rộng

Khi quá trình tự động hoá đòi hỏi số lượng đầu và đầu ra nhiều hơn số lượng sẵn có trên đơn vị cơ bản hoặc khi cần những chức năng đặc biệt thì có thể mở rộng đơn vị cơ bản bằng cách gá thêm các modue ngoài. Tối đa có thể gá thêm 8 modue vào ra qua 8 vị trí có sẵn trên panen về phía phải. Thường Step 5 sử dụng các module mở rộng:

+ Modue vào, ra số duy trì.

+ Modue vào, ra số không duy trì lấy từ S5-100U.

+ Modue vào, ra tương tự không duy trì lấy từ S5-100U.

+ Modue thông tin không duy trì CCP.

* Qui ước các chân của module mở rộng như hình 5.2

+ Chân 1: Dương nguồn (L+)

+ Chân 2: Âm nguồn (M)

+ Chân 4: Kênh số 0

+ Chân 3: Kênh số 1

+ Chân 6: Kênh số 2

+ Chân 5: Kênh số 3

+ Chân 8: Kênh số 4

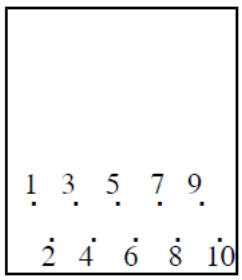
+ Chân 7: Kênh số 5

+ Chân 10: Kênh số 6

+ Chân 9: Kênh số 7

5.2. Địa chỉ và gán địa chỉ

Trong PLC các địa chỉ cần gửi thông tin đến hoặc lấy thông tin đi đều phải có địa chỉ để liên lạc. Địa chỉ là con số hoặc tổ hợp các con số đi theo sau chữ cái. Chữ cái chỉ loại địa chỉ, con số hoặc tổ hợp con số chỉ số hiệu địa chỉ.



Hình 5.2 - Sơ đồ chân module mở rộng

Trong PLC có những bộ phận được gán địa chỉ đơn như bộ thời gian (T), bộ đếm (C) và cờ (F), chỉ cần một trong 3 chữ cái đó kèm theo một số là đủ, *ví dụ*: T1, C32, F6...

Các địa chỉ đầu vào và đầu ra cùng với các module chức năng có địa chỉ phức, cách gán địa chỉ giống nhau. Ta xét cách gán địa chỉ cho các đầu vào, ra.

Có hai loại đầu vào ra:

+ Đầu vào ra trên khối cơ bản (gắn liền với CPU), các đầu vào ra này có địa chỉ không đổi, với S5-95U là I32.0 đến I33.7, Q32.0 đến Q33.3.

+ Đầu vào ra trên các module mở rộng thì địa chỉ phụ thuộc vào vị trí lắp đặt của module trên Panen. Chỗ lắp module trên Panen gọi là khe (Slot), các khe đều có đánh số, khe số 0 đứng liền với đơn vị cơ bản và cứ thế tiếp tục.

5.2.1. Địa chỉ vào/ra trên module số

Khi lắp module số vào ra lên một khe nào lập tức nó được mang số hiệu của khe đó. Trên mỗi module thì mỗi đầu vào, ra là một kênh, các kênh đều được đánh số. Địa chỉ của mỗi đầu vào ra là số ghép của số hiệu khe và kênh, số hiệu khe đứng trước, số hiệu kênh đứng sau, giữa hai số có dấu chấm. Số hiệu khe và kênh như hình 5.3.

Khe số	0	1	2	3	…
Đơn vị cơ bản	1 2 : 7	1 2 : 7	1 2 : 7	1 2 : 7	1 2 : 7

Hình 5.3 - Số hiệu khe và kênh trên module

Ví dụ: địa chỉ của kênh số 2 trên module cắm vào khe số 0 là 0.2.

Mỗi đầu vào ra trên module số chỉ thể hiện được tại một thời điểm một trong hai trạng thái "1" hoặc "0". Như vậy mỗi kênh của module số chỉ được biểu diễn bằng một bit số liệu, vì vậy địa chỉ của kênh trên module số còn được

gọi là địa chỉ bit, mỗi module mang nhiều kênh tức là chứa nhiều bit, thường là 8 bit hay một byte, vì vậy địa chỉ khe còn gọi là địa chỉ byte.

Module số có thể được lắp trên bất kỳ khe nào trên Panen của PLC.

5.2.2. Địa chỉ vào ra trên module tương tự

Để diễn tả một giá trị tương tự ta phải cần nhiều bit. Trong PLC S5 người ta dùng 16 bit (một word). Các lệnh tương tự có thể được gán địa chỉ byte hoặc địa chỉ word khi dùng lệnh nạp hoặc truyền.

Chỉ có thể lắp module tương tự vào khe 0 đến 7. Mỗi khe có 4 kênh, mỗi kênh mang 2 địa chỉ đánh số từ 64+65 (đầu khe 0) đến 126+127 (cuối khe 7) như hình 5.4.

Như vậy mỗi kênh mang địa chỉ riêng không kèm theo địa chỉ khe, đọc địachỉ kênh là đã biết nó nằm ở khe nào.

Khe số	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Đơn vị cơ bản	64+65 66+67 68+69 70+71	72+73 74+75 76+77 78+79	80+81 82+83 84+85 86+87	88+89 90+91 92+93 94+95	96+97 98+99 100+101 102+103	104+105 106+107 108+109 110+111	112+113 114+115 116+117 118+119	120+121 122+123 124+125 126+127

Hình 5.4 - Địa chỉ trên module tương tự

Ví dụ: Một module tương tự lắp vào khe số 2 trên đó kênh số 0 mang địa chỉ byte 80 và 81.

Chú ý: Các khe trống bao giờ cũng có trạng thái tín hiệu "0".

5.3. Vùng đối tượng

TT	**Tên tham số**	**Diễn giải**	**Vùng tham số**
1	ACCUM 1	ắc qui 1	
2	ACCUM 2	ắc qui 2	
3	BN	Hằng số byte	-127 đến 127
4	C	Bộ đếm: - Có nhớ - Không nhớ	0 đến 7 8 đến 127
5	CCO/CC1	Mã điều kiện 1 và mã điều kiện 2	
6	D	Số liệu dạng bit	0.0 đến 255.15
7	DB	Khối số liệu	2 đến 255
8	DL	Từ dữ liệu trái	0 đến 255
9	DR	Từ dữ liệu phải	0 đến 225
10	DW	Từ dữ liệu	0 đến 255
11	F	Cờ - Có nhớ - Không nhớ	0.0 đến 63.7 64.0 đến 255.7

12	FB	Khối hàm	0 đến 255
13	FW	Từ cờ - Có nhớ - Không nhớ	0 đến 62 64 đến 254
14	FY	Từ byte: - Có nhớ - Không nhớ	0 đến 63 64 đến 255
15	I	Đầu vào bit	0.0 đến 127.7
16	IB	Đầu vào byte	0 đến 127
17	IW	Đầu vào từ	0 đến 126
18	KB	Hằng số 1 byte	0 đến 255
19	KC	Hăng số đếm	0 đến 999
20	KF	Hằng số	-32768 đến 32677
21	KH	Hằng số dạng cơ số16	0000 đến FFFF
22	KM	Hằng số bit dạng byte	Mỗi byte 16 bit
23	KS	Hằng số cho ký tự	2 ký tự ASCII
24	KT	Hằng số cho thời gian	0.0 đến 999.3
25	KY	Hằng số	0 đến 255 cho mỗi byte
26	OB	Khối tổ chức (khối đặc biệt: 1, 3,13, 21, 31, 34, 251)	0 đến 255
27	PB	Khối chương trình	0 đến 255
28	PB/PY	Đệm ngoại vi vào ra	0 đến 127
29	PII	Bộ đệm đầu vào	
30	PIQ	Bộ đệm đầu ra	
31	PW	Đệm ngoại vi dạng từ	0 đến 125
32	Q	Đầu ra bit	0.0 đến 127.7
33	QB	Đầu ra dạng byte	0 đến 127
34	QW	Đầu ra dạng từ	0 đến 125
35	RS	Vùng số liệu hệ thống	0 đến 255
36	SB	Khối dãy	0 đến 255
37	T	Bộ thời gian	0 đến 127

5.4. Cấu trúc của chương trình S5

5.4.1. Cấu trúc chương trình

Các chương trình điều khiển với PLC S5 có thể được viết ở dạng đơn khối hoặc đa khối.

Chương trình đơn khối

Chương trình đơn khối chỉ viết cho các công việc tự động đơn giản, các lệnh được viết tuần tự trong một khối. Khi viết chương trình đơn khối người ta dùng khối OB1. Bộ PLC quét khối theo chương trình, sau khi quét đến lệnh cuối cùng nó quay trở lại lệnh đầu tiên.

Chương trình đa khối (có cấu trúc):

Khi nhiệm vụ tự động hoá phức tạp người ta chia chương trình điều khiển ra thành từng phần riêng gọi là khối. Chương trình có thể xếp lồng khối này vào

khối kia. Chương trình đang thực hiện ở khối này có thể dùng lệnh gọi khối để sang làm việc với khối khác, sau khi đã kết thúc công việc ở khối mới nó quay về thực hiện tiếp chương trình đã tạm dừng ở khối cũ.

Người lập trình có thể xếp lồng khối này vào khối kia thành lớp, tối đa là 16 lớp. Nếu số lớp vượt quá giới hạn thì PLC tự động về trạng thái ban đầu.

5.4.2. Khối và đoạn (Block and Segment)

Cấu trúc mỗi khối gồm có:

+ Đầu khối gồm tên khối, số hiệu khối và xác định chiều dài khối.

+ Thân khối: Thể hiện nội dung khối và được chia thành đoạn (Segment) thực hiện từng công đoạn của tự động hoá sản xuất. Mỗi đoạn lại bao gồm một số dòng lệnh phục vụ việc giải bài toán logic. Kết quả của phép toán logic được gửi vào RLO (Result of logic operation). Việc phân chia chương trình thành các đoạn cũng ảnh hưởng đến RLO. Khi bắt đầu một đoạn mới thì tạo ra một giá trị RLO mới, khác với giá trị RLO của đoạn trước.

+ Kết thúc khối: Phần kết thúc khối là lệnh kết thúc khối BE.

Các loại khối:

** Khối tổ chức OB (Organisation Block):*

Khối tổ chức quản lý chương trình điều khiển và tổ chức việc thực hiện chương trình.

** Khối chương trình PB (Program Block):*

Khối chương trình sắp xếp chương trình điều khiển theo chức năng hoặc khía cạnh kỹ thuật.

** Khối dãy SB (Sequence Block):*

Khối dãy là loại khối đặc biệt được điều khiển theo chương trình dãy và được xử lý như khối chương trình.

** Khối chức năng FB (Function Block):*

Khối chức năng là loại khối đặc biệt dùng để lập trình các phần chương trình điều khiển tái diễn thường xuyên hoặc đặc biệt phức tạp. Có thể gán tham số cho các khối đó và chúng có một nhóm lệnh mở rộng.

** Khối dữ liệu DB (Data Block):*

Khối dữ liệu lưu trữ các dữ liệu cần thiết cho việc xử lý chương trình điều khiển.

5.5. Bảng lệnh của S5-95U

Các lệnh của chương trình S5 được chia thành ba nhóm là:

5.5.1. Nhóm lệnh cơ bản

Nhóm lệnh cơ bản gồm những lệnh sử dụng cho các chức năng, thực hiện trong các khối tổ chức OB, khối chương trình PB, khối dãy SB và khối chức năng FB. Ngoại trừ hai lệnh số học +F và -F chỉ được biểu diễn bằng phương pháp dãy lệnh STL, còn lại tất cả các lệnh cơ bản khác đều có thể được biểu diễn bằng cả ba phương pháp đó là bảng lệnh STL, lưu đồ điều khiển CSF và biểu đồ bậc thang LAD.

5.5.2. Nhóm lệnh bổ trợ

Nhóm lệnh bổ trợ bao gồm các lệnh sử dụng cho các chức năng phức tạp, ví dụ như các lệnh thay thế, các chức năng thử nghiệm, các lệnh dịch chuyển hoặc chuyển đổi...

Các lệnh bổ trợ dùng trong khối chức năng và được biểu diễn bằng phương pháp bảng lệnh STL. Chỉ có rất ít lệnh được sử dụng ở phương pháp lưu đồ.

5.5.3. Nhóm lệnh hệ thống

Các lệnh hệ thống được phép thâm nhập trực tiếp vào hệ thống điều hành và chỉ có thể được biểu diễn bằng phương pháp bảng lệnh STL. Chỉ khi thực sự am hiểu về hệ thống mới nên sử dụng các lệnh hệ thống.

Diễn dải của các lệnh xem phần "Bảng lệnh"

5.6. Cú pháp một số lệnh cơ bản của S5

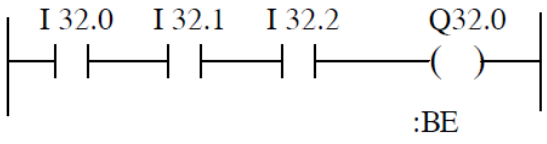
5.6.1. Nhóm lệnh logic cơ bản

Khi thực hiện lệnh đầu tiên của một loạt phép toán logic thì nội dung của đối tượng lệnh được lấy vào sẽ được nạp ngay vào RLO (Kết quả của phép toán logic) mà không cần thực hiện phép toán.

Đối tượng của các lệnh logic là: I, Q, F, T, C

a. Lệnh A

Lập trình dạng STL (có thể lập trình dạng LAD và kiểm tra lại dạng STL).



Hình 5.6 - Lệnh A

```
A    I    32.0
A    I    32.1
A    I    32.2
=    Q    32.0
```

BE

+ Ấn Enter để trở về màn hình Output.

+Ấn Shift-F5 để xem dạng LAD và CSF, dạng LAD như hình 5.6

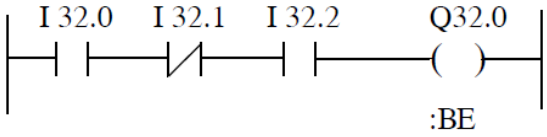
+Ấn Shift-F7 để cất chương trình và đổ chương trình sang PLC, chọn yes để xác nhận việc đổ đè chương trình lên chương trình cũ trong PLC (khi cất thì PLC phải để ở chế độ STOP).

+ Bật công tắc của CPU về chế độ RUN, quan sát kết quả lập trình.

b. Lệnh AN

Lập trình dạng STL.

```
A    I    32.0
AN   I    32.1
A    I    32.2
=    Q    32.0
BE
```

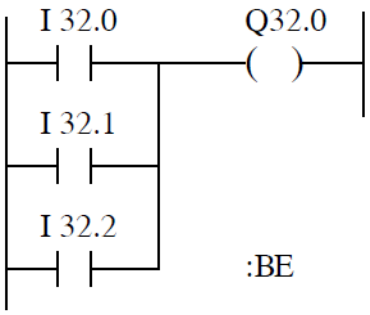


Hình 5.7 - Lệnh AN

c. Lệnh O

Lập trình dạng STL.

```
O    I    32.0
O    I    32.1
O    I    32.2
=    Q    32.0
BE
```

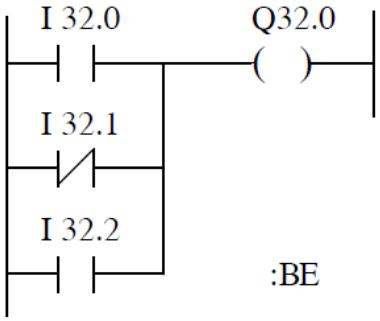


Hình 5.8 - Lệnh O

d. Lệnh ON

Lập trình dạng STL.

```
O I 32.0
ON I 32.1
O I 32.2
= Q 32.0
BE
```



Hình 5.9 - Lệnh ON

e. Lệnh O giữa hai lệnh A

Lập trình dạng STL.

A I 32.0

A I 32.1

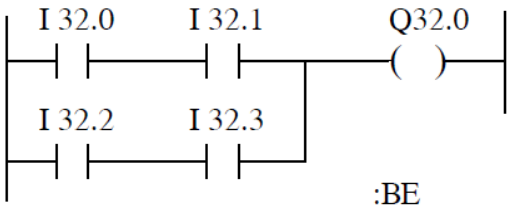
O

A I 32.2

A I 32.3

= Q 32.0

BE



Hình 5.10 - Lệnh O giữa hai lệnh A

g. Lệnh "(" và lệnh ")"

Lập trình dạng STL

O I 32.0

O

A I 32.1

A(

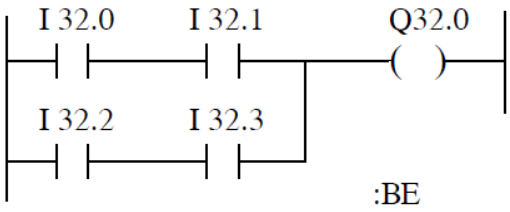
O I 32.2

O I 32.3

)

= Q 32.0

BE



Hình 5.11 - Lệnh "(" và lệnh ")"

5.6.2. Nhóm lệnh set và reset

Các lệnh set và reset lưu giữ kết quả của phép toán logic được hình thành trong bộ xử lý.

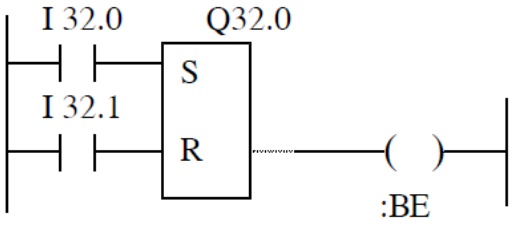
Đối tượng của các lệnh này là I, Q, F.

Ví dụ 1:

A I 32.0

S Q 32.0

A I 32.1



Hình 5.12 - Lệnh set /reset

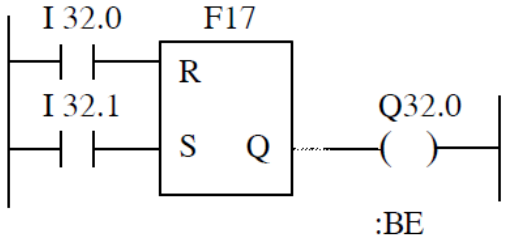
```
R     Q          32.0
NOP   0
```

Khi đầu vào I32.0 có thì đầu ra Q32.0 có và được giữ lại cho dù I32.0 mất, chỉ khi I32.1 có thì lại xoá nhớ làm Q32.0 về không.

Lệnh NOP 0 là lệnh giữ chỗ cho phương pháp LAD. Vì có đầu ra Q chưa dùng, muốn phương pháp LAD vẽ được hình thì phải đưa lệnh NOP 0 vào.

Ví dụ 2:

```
A I 32.0
R F 17
A I 32.1
S F 17
A F 17
= Q 32.0
```



Hình 5.13 - Lệnh set /reset

Đây là ví dụ về lệnh set trội, vì khi I32.0 có trạng thái 1 thì nó sẽ xoá trạng thái tín hiệu trên cờ F17 về "0" cho đến khi I32.1 có trạng thái 1 thì nó sẽ đặt trạng thái 1 cho cờ F17 sau đó không phụ thuộc I32.0 nữa. Khi cờ nhận trạng thái 1 thì sẽ gán cho đầu ra Q32.0 trạng thái 1. Khi cả I32.0 và I32.1 cùng có trạng thái 1 thì cờ sẽ có trạng thái 1 vì lệnh set ở sau, gọi là ưu tiên set.

5.6.3. Nhóm lệnh nạp và truyền

Lệnh nạp và truyền để trao đổi thông tin giữa các vùng đối tượng lệnh khác nhau. Chuẩn bị giá trị thời gian và giá trị đếm cho các lệnh thời gian và lệnh đếm. Nạp hằng số phục vụ việc xử lý chương trình.

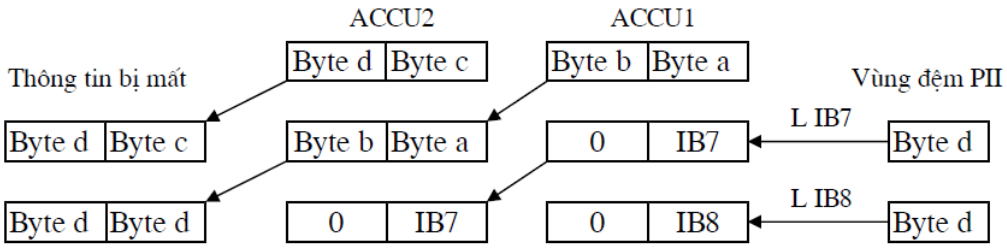
Luông thông tin được nạp và truyền thông qua hai thanh ghi tích luỹ ACCU1 và ACCU2. Thanh ghi tích luỹ là thanh ghi đặc biệt trong PLC dùng để lưu trữtạm thời các thông tin. Mỗi thanh ghi có độ dài 16 bit.

Có thể nạp hoặc truyền các đối tượng theo byte hoặc từ. Để trao đổi theo byte, thông tin lưu trữ trong byte phải tức là byte thấp của thanh ghi, số bit còn thừa (ngoài 8 bit) được đặt không. Có thể dùng các lệnh khác nhau để xử lý các thông tin trong hai thanh ghi.

Các lệnh thuộc nhóm này là:

a. Lệnh nạp L:

Nội dung của đối tượng (đơn vị byte) được chép vào ACCU1 không phụ thuộc vào RLO và RLO cũng không bị ảnh hưởng. Nội dung trước đó của ACCU1 được chuyển dịch sang ACCU2, nội dung cũ của ACCU2 sẽ bị mất.

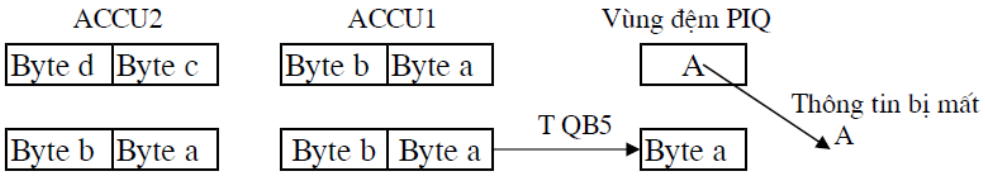


Hình 5.14

Ví dụ: Nạp liên tiếp IB7 và IB8 từ vùng đệm PII vào thanh ghi tích luỹ ta có sơ đồ nạp như hình 5.14.

b. Lệnh truyền T

Nội dung của ACCU1 được gán cho đối tượng lệnh không phụ thuộc RLO và RLO cũng không bị ảnh hưởng. Khi truyền thì thông tin từ ACCU1 được chép vào vùng nhớ đã được địa chỉ hoá (ví dụ vùng đệm đầu ra PIQ). Nội dung của ACCU1 không bị mất. Giá trị trước đó của vùng đệm đầu ra PIQ bị mất. Mô tả lệnh như hình 5.15.



Hình 5.15

c. Lệnh LD:

số đếm và số thời gian được nạp vào ACCU1 dạng mã BCD, không phụ thuộc vào RLO và RLO cũng không bị ảnh hưởng.

Đối tượng của các lệnh này là:

+ Lệnh L: IB, IW, QB, QW, FY, FW, DR, DL, DW, PB/PY, PW, T, C, KM, KH, KF, KY, KB, KS, KT, KC.

+ Lệnh T: IB, IW, QB, QW, FY, FW, DR, DL, DW, PB/PY, PW.

+ Lệnh LD: T, C.

5.6.4. Nhóm lệnh thời gian

Chương trình điều khiển sử dụng các lệnh thời gian để theo dõi, kiểm soát và quản lý các hoạt động có liên quan đến thời gian.

a. Nạp giá trị thời gian

Khi một bộ thời gian được khởi phát thì nội dung trong ACCU1 (dạng từ 16 bit) được dùng làm giá trị tính thời gian. Do đó, muốn dùng các lệnh thời gian phải nạp giá trị thời gian cần đặt vào ACCU1 trước khi bộ thời gian hoạt động.

Có thể nạp các kiểu dữ liệu sau dùng cho các lệnh thời gian:

+ KT: giá trị thời gian hằng số

+ DW: từ (word) dữ liệu

+ IW: từ (word) đầu vào

+ QW: từ (word) đầu ra

+ FW: từ (word) cờ

Trừ loại KT các loại còn lại phải ở dạng mã BCD.

• *Nạp thời gian hằng số*: L KT 40.2

Trong lệnh có: KT chỉ rõ là hằng số

Số 40: hệ số (có thể gán từ 0 đến 999)

Số 2: là mã, có 4 mã: 0 tương ứng 0,01s

1 tương ứng 0,1s

2 tương ứng 1s

3 tương ứng 10s

Với số trên thì thời gian được tính là $\Delta t = 40 \times 1s = 40s$.

Với mã càng nhỏ thì giá trị thời gian càng chính xác, nên dùng mã nhỏ.

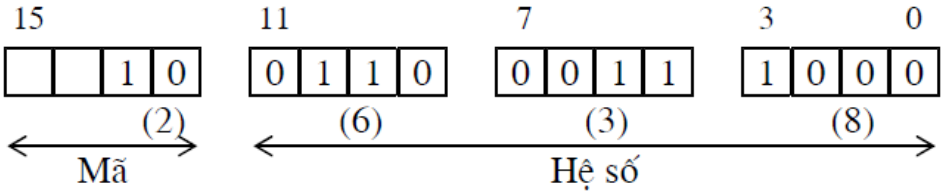
• Nạp thời gian dưới dạng đầu vào, đầu ra, hoặc từ dữ liệu:

Ví dụ muốn nạp một giá trị thời gian từ một từ dữ liệu DW2 vào ACCU1 ta viết lệnh sau:

L DW2

Như vậy, trước khi thực hiện lệnh này thì giá trị thời gian đã được lưu sẵn trong từ dữ liệu DW2 dưới dạng mã BCD.

Ví dụ trong DW2 có các số như hình 5.16.



Hình 5.16

Mã thời gian cũng được sử dụng như trên.

$\Delta t = 638 \times 1s = 638s$

Vậy, trước khi dùng lệnh nạp trên ta phải dùng chương trình điều khiển để viết giá trị thời gian vào từ dữ liệu DW2. Ví dụ để viết giá trị thời gian 27s vào từ dữ liệu DW2 trong khối DB3 rồi sau đó nạp vào ACCU1 như sau:

```
C     DB3
L     KT        270.1
T     DW2
...
L     DW2
```

b. Đọc giá trị thời gian hiện hành

Có thể dùng hai lệnh L và LD để đưa giá trị thời gian hiện hành của bộ thời gian T vào ACCU1 để xử lý.

```
L     T1        % đọc giá trị thời gian dạng nhị phân
LD    T1        % đọc giá trị thời gian dạng BCD
```

Chú ý: Lệnh L và T đi với T và C thì bao giờ cũng đọc giá trị nhị phân còn đi với các đối tượng khác thì cũng có thể đọc giá trị nhị phân hoặc dạng BCD tuỳ theo trường hợp cụ thể.

c. Các lệnh

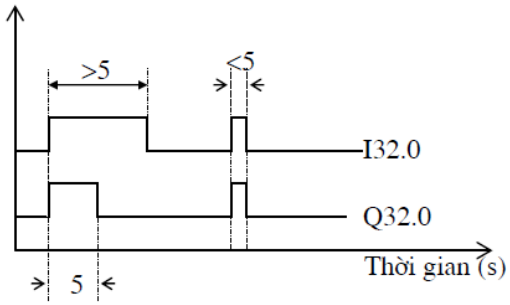
- Bộ thời gian xung SP

Bộ thời gian được khởi phát lên 1 tại sườn lên của RLO khi RLO là 1 thì bộ thời gian vẫn duy trì trạng thái 1 cho đến khi đạt giá trị đặt mới xuống. Nhưng khi RLO về không thì bộ thời gian về không ngay.

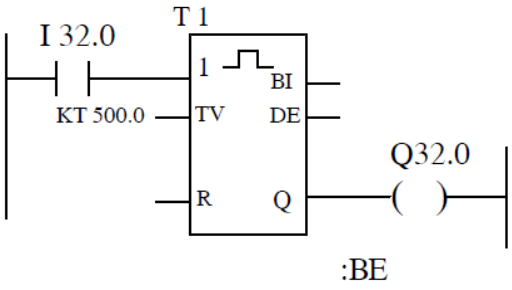
Lập trình dạng STL (có thể lập trình dạng LAD và kiểm tra lại dạng STL).

```
A     I         32.0
L     KT        500.0
SP    T         1
```

```
NOP 0
NOP 0
NOP 0
A     T     1
=     Q     32.0
BE
```



a)



b)

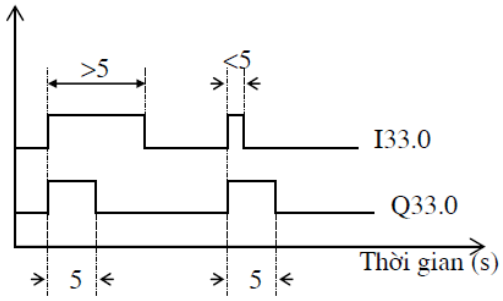
Hình 5.17 - Giản đồ thời gian và dạng LAD lệnh SP

- *Bộ thời gian mở rộng SE*

Bộ thời gian xung mở rộng SE được khởi phát lên 1 tại sườn lên của RLO sau đó không phụ thuộc RLO nữa cho đến khi đủ thời gian đặt mới về không.

Lập trình dạng STL.

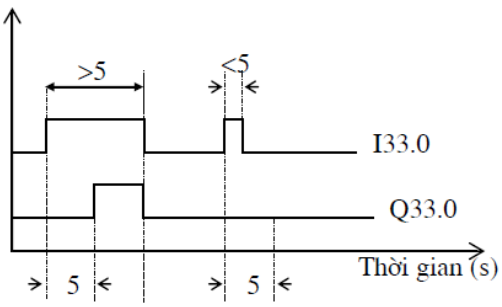
```
C     DB    3
L     KT    500.0
T     IW    16
A     I     33.0
L     IW    16
SE    T     2
NOP 0
NOP 0
NOP 0
A     T     2
=     Q     33.0
BE
```



Hình 5.18 - Giản đồ thời gian lệnh SE

- *Bộ thời gian bắt đầu trễ SD*

Thời gian bắt đầu chậm hơn so với sườn lên của RLO một khoảng



Hình 5.19 - Giản đồ thời gian lệnh SD

bằng thời gian đặt trong lệnh. Khi RLO về không thì bộ thời gian cũng bị đặt ngay về không.

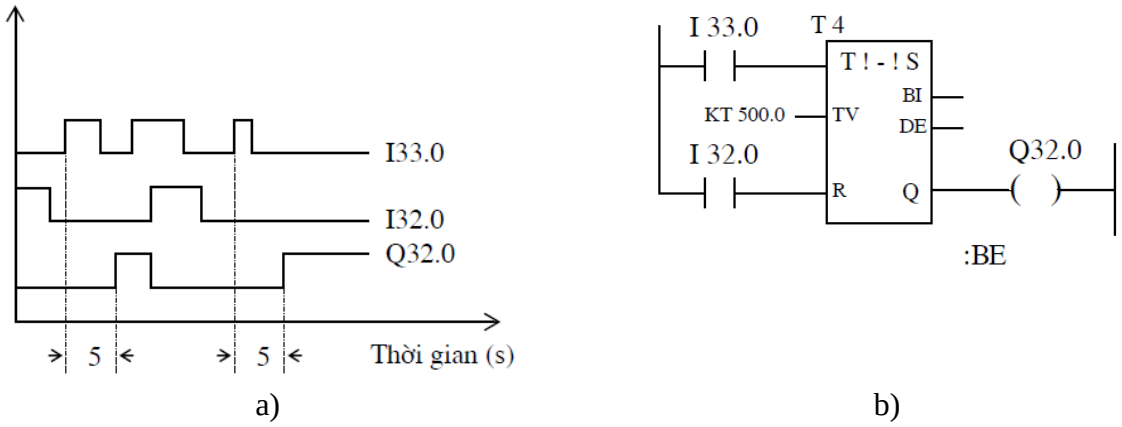
Lập trình dạng STL.

```
C     DB    3
L     KT    50.1
T     FW    16
A     I     33.0
L     FW    16
SD    T     3
NOP 0
NOP 0
NOP 0
A     T     3
=     Q     33.0
BE
```

- *Bộ thời gian bắt đầu trễ lưu trữ SS*

Thời gian bắt đầu chậm hơn so với sườn lên của RLO một khoảng thời gian bằng thời gian đặt trong lệnh và sau đó không phụ thuộc RLO nữa. Nó chỉ về không khi có lệnh xoá R.

```
A     I     33.0
L     KT    500.0
SS    T     4
A     I     32.0
R     T     4
NOP 0
NOP 0
A     T     4
=     Q     32.0
BE
```



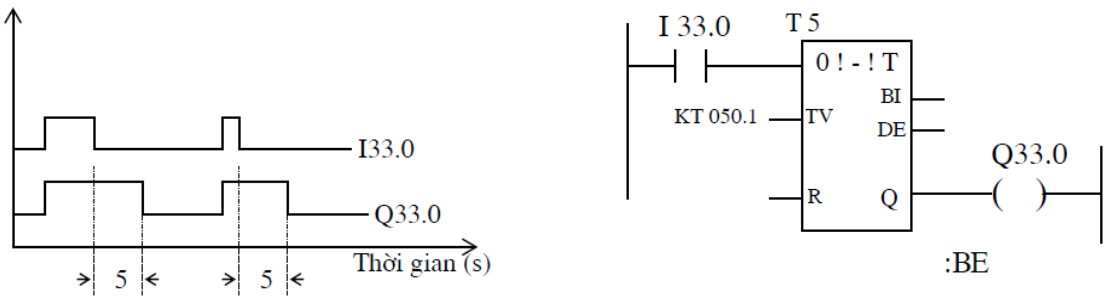
a) b)

Hình 5.20 - Giản đồ thời gian và dạng LAD lệnh SS

- *Bộ thời gian tắt trễ SF*

Bộ thời gian lên 1 tại sườn lên của RLO. Khi RLO về không thì bộ thời gian tiếp tục duy trì trạng thái một khoảng thời gian nữa bằng khoảng đã đặt trong lệnh rồi mới về không. Để xoá thời gian dùng lệnh R, khi có lệnh R từ 0 lên 1 thì bộ thời gian được đặt về không và trạng thái tín hiệu vẫn giữ 0 cho đến khi bộ thời gian được khởi phát lại.

```
A     I     33.0
L     KT    50.1
SF    T     4
NOP 0
NOP 0
NOP 0
A     T     4
=     Q     33.0
BE
```



a) b)

Hình 5.21 - Giản đồ thời gian và dạng LAD lệnh SF

5.6.6. Nhóm lệnh đếm

a. Nạp giá trị đếm

Cũng như bộ thời gian khi một bộ đếm được khởi phát thì nội dung trong ACCU1 (dạng từ 16 bit) được dùng làm giá trị đếm. Do đó, muốn dùng các lệnh đếm phải nạp giá trị đếm vào ACCU1 trước khi bộ đếm hoạt động.

Có các kiểu dữ liệu sau dùng cho các lệnh đếm:

+ KC: giá trị hằng số

+ DW: từ (word) dữ liệu

+ IW: từ (word) đầu vào

+ QW: từ (word) đầu ra

+ FW: từ (word) cờ

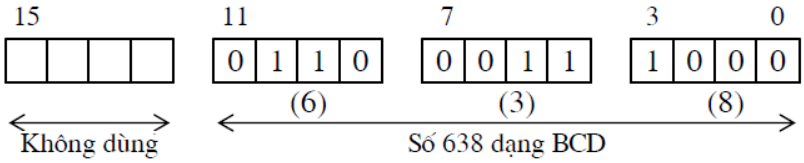
Trừ loại KC các loại còn lại phải ở dạng mã BCD.

• Nạp giá trị đếm hằng số: L KC 38

Số đếm từ 0 đến 999

• *Nạp số đếm dưới dạng đầu vào, đầu ra, hoặc từ dữ liệu*: Ví dụ muốn nạp một giá trị đếm từ một từ dữ liệu DW2 vào ACCU1 ta viết lệnh sau:

L DW 2



Hình 5.22

Như vậy, trước khi thực hiện lệnh này thì giá trị đếm đã được lưu sẵn trong từ dữ liệu DW2 dưới dạng mã BCD. Ví dụ trong DW2 có các số như hình 5.22:

Với lệnh trên thì số 638 được nạp vào DW2.

• *Đối tượng của lệnh*: Cả hai lệnh đếm chỉ có một đối tượng là bộ đếm C với các số hiệu tuỳ thuộc loại PLC.

b. Chuẩn bị thực hiện các lệnh đếm

+ *Đặt bộ đếm*: Sau khi đã nạp giá trị đếm ta dùng lệnh S để cho bộ đếm làm việc.

+ *Xoá bộ đếm*: Khi đã đếm tới một giá trị nào đó ta dùng lệnh R để xoá, tức là ngừng đếm và đưa giá trị đếm về không, nếu không dùng lệnh này khi đếm đủ giá trị đặt bộ đếm giữ nguyên trạng thái không về không.

+ *Quét bộ đếm*: Ta dùng lệnh logic boole để quét bộ đếm (ví dụ lệnh A). Nếu bộ đếm chưa về không thì kết quả quét có trạng thái 1.

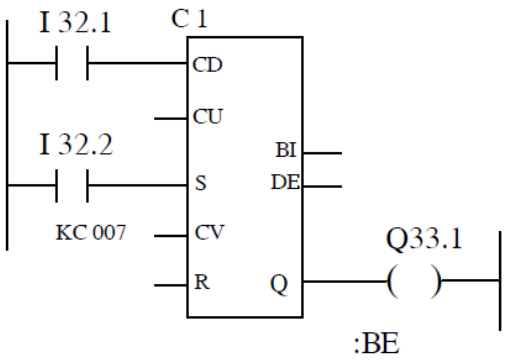
+ *Xuất ra trạng thái bộ đếm hiện hành*: Có thể dùng lệnh L và LD để đưa trạng thái bộ đếm hiện hành vào ACCU1 để xử lý sau này, lệnh L dùng cho số nhị phân, lệnh LD dùng cho số BCD.

c. Các lệnh

- Lệnh đếm xuống CD

Số đếm giảm đi một đơn vị lúc xuất hiện một sườn lên của RLO. Khi RLO về không số đếm không bị ảnh hưởng.

```
A     I     32.1
CD    C     1
NOP 0
A     I     32.2
L     KC    7
S     C     1
NOP 0
NOP 0
:BE
```



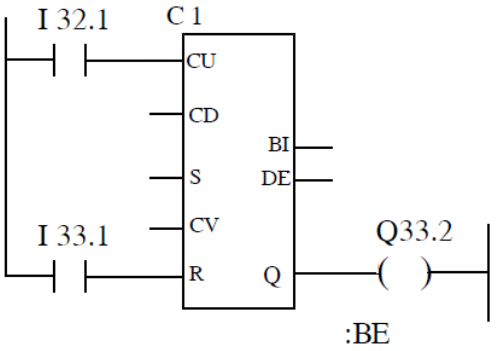
Hình 5.23 - Lệnh đếm xuống CD

2. Lệnh đếm lên CU

Số đếm tăng một đơn vị lúc xuất hiện sườn lên của RLO. Khi RLO về không số đếm không bị ảnh hưởng.

```
A     I     32.1
CU    C     1
```

```
NOP 0
NOP 0
NOP 0
A     I     33.1
R     C     1
NOP 0
NOP 0
A     C     1
=     Q     33.1
BE
```



Hình 5.24 - Lệnh đếm lên CU

CHƯƠNG 6: BỘ ĐIỀU KHIỂN PLC - S7-200

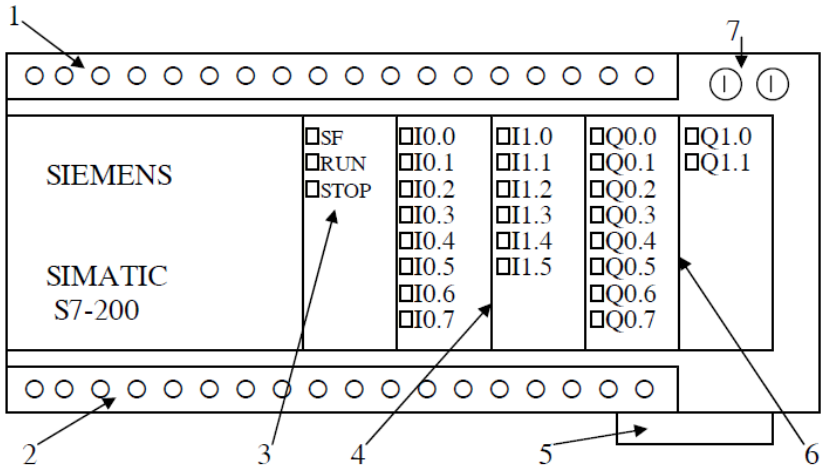
6.1. Cấu hình cứng

PLC Step 7 thuộc họ Simatic do hãng Siemens sản xuất. Đây là loại PLC hỗn hợp vừa đơn khối vừa đa khối. Cấu tạo cơ bản của loại PLC này là một đơn vị cơ bản sau đó có thể ghép thêm các module mở rộng về phía bên phải. Có các module mở rộng tiêu chuẩn. Những module ngoài này bao gồm những đơn vị chức năng mà có thể tổ hợp lại cho phù hợp với những nhiệm vụ kỹ thuật cụ thể.

6.1.1. Đơn vị cơ bản

a. Cấu trúc đơn vị có bản:

Đơn vị cơ bản của PLC S7-200 (CPU 314) như hình 6.1



Hình 6.1 - Hình khối mặt trước PLC S7-200

Trong đó:

1. Chân cắm cổng ra.

2. Chân cắm cổng vào.

3. Các đèn trạng thái: SF (đèn đỏ): Báo hiệu hệ thống bị hỏng.

RUN (đèn xanh): Chỉ định rằng PLC đang ở chế độ làm việc.

STOP (đèn vàng): Chỉ định rằng PLC đang ở chế độ dừng.

4. Đèn xanh ở cổng vào chỉ định trạng thái tức thời của cổng vào.

5. Cổng truyền thông.

6. Đèn xanh ở cổng ra chỉ định trạng thái tức thời của cổng ra.

7. Công tắc.

Chế độ làm việc: Công tắc chọn chế độ làm việc có ba vị trí chuyển về trạng thái STOP khi máy có sự cố, hoặc trong chương trình gặp lệnh STOP, do đó khi chạy nên quan sát trạng thái thực của PLC theo đèn báo.

+ STOP: cưỡng bức PLC dừng công việc đang thực hiện, chuyển về trạng thái nghỉ. ở chế độ này PLC cho phép hiệu chỉnh lại chương trình hoặc nạp một chương trình mới.

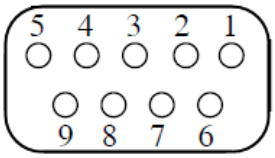
+ TERM: cho phép PLC tự quyết định một chế độ làm việc (hoặc RUN hoặc STOP).

Chỉnh định tương tự: Núm điều chỉnh tương tự đặt dưới nắp đậy cạnh cổng ra, núm điều chỉnh tương tự cho phép điều chỉnh tín hiệu tương tự, góc quay được 2700..

Pin và nguồn nuôi bộ nhớ: Nguồn pin được tự động chuyển sang trạng thái tích cực khi dung lượng nhớ bị cạn kiệt và nó thay thế để dữ liệu không bị mất.

Cổng truyền thông: S7-200 sử dụng cổng truyền thông nối tiếp RS 485 với phích cắm 9 chân để phục vụ cho việc ghép nối với thiết bị lập trình hoặc với các PLC khác. Tốc độ truyền cho máy lập trình kiểu PPI là 9600 boud. Các chân của cổng truyền thông là:

1. Đất
2. 24v DC
3. Truyền và nhận dữ liệu
4. Không dùng
5. Đất
6. 5v DC (điện trở trong 100Ω)
7. 24v DC (120 mA)
8. Truyền và nhận dữ liệu
9. Không dùng



Hình 6.2 - Lệnh đếm lên CU

b. Thông số

• Với CPU 214:

+ 14 cổng vào và 10 cổng ra logic. Có thể mở rộng thêm 7 module bao gồm cả module analog.

+ Tổng số cổng vào và ra cực đại là: 64 vào, 64 ra.

+ 2048 từ đơn (4Kbyte) thuộc miền nhớ đọc/ghi không đổi để lưu chương trình (vùng nhớ giao diện với EFROM).

+ 2048 từ đơn (4Kbyte) thuộc miền nhớ đọc/ghi để ghi dữ liệu, trong đó có 512 từ đầu thuộc miền không đổi.

+ 128 bộ thời gian (Times) chia làm ba loại theo độ phân dải khác nhau: 4 bộ 1ms, 16 bộ 10ms và 108 bộ 100ms.

+ 128 bộ đếm chia làm hai loại: chỉ đếm tiến và vừa đếm tiến vừa đếm lùi.

+ 688 bit nhớ đặc biệt để thông báo trạng thái và đặt chế độ làm việc.

+ Các chế độ ngắt và xử lý ngắt gồm: ngắt truyền thông, ngắt theo sườn lên hoặc xuống, ngắt thời gian, ngắt của bộ đếm tốc độ cao và ngắt truyền xung.

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

+ 2 bộ phát xung nhanh cho dãy xung kiểu PTO hoặc kiểu PWM.

+ 2 bộ điều chỉnh tương tự.

+ Toàn bộ vùng nhớ không bị mất dữ liệu trong khoảng thời gian 190h khi PLC bị mất nguồn cung cấp.

• *Với CPU 212*

+ 8 cổng vào và 6 cổng ra logic. Có thể mở rộng thêm 2 module bao gồm cả module analog.

+ Tổng số cổng vào và ra cực đại là: 64 vào, 64 ra.

+ 512 từ đơn (1Kbyte) thuộc miền nhớ đọc/ghi không đổi để lưu chương trình (vùng nhớ giao diện với EFROM).

+ 512 từ đơn lưu dữ liệu, trong đó có 100 từ nhớ đọc/ghi thuộc miền không đổi.

+ 64 bộ thời gian trễ (Times) trong đó: 2 bộ 1ms, 8 bộ 10ms và 54 bộ 100ms.

+ 64 bộ đếm chia làm hai loại: chỉ đếm tiến và vừa đếm tiến vừa đếm lùi.

+ 368 bit nhớ đặc biệt để thông báo trạng thái và đặt chế độ làm việc.

+ Các chế độ ngắt và xử lý ngắt gồm: ngắt truyền thông, ngắt theo sườn lên hoặc xuống, ngắt thời gian, ngắt của bộ đếm tốc độ cao và ngắt truyền xung.

+ Toàn bộ vùng nhớ không bị mất dữ liệu trong khoảng thời gian 50h khi PLC bị mất nguồn cung cấp.

6.1.2. Các module vào ra mở rộng

Khi quá trình tự động hoá đòi hỏi số lượng đầu và đầu ra nhiều hơn số lượng sẵn có trên đơn vị cơ bản hoặc khi cần những chức năng đặc biệt thì có thể mở rộng đơn vị cơ bản bằng cách gá thêm các module ngoài. Tối đa có thể gá thêm 7 module vào ra qua 7 vị trí có sẵn trên Panen về phía phải. Địa chỉ của các vị trí của module được xác định bằng kiểu vào ra và vị trí của module trong rãnh, bao gồm có các module cùng kiểu. Ví dụ một module cổng ra không thể gán địa chỉ module cổng vào, cũng như module tương tự không thể gán địa chỉ như module số và ngược lại.

Các module số hay rời rạc đều chiếm chỗ trong bộ đệm, tương ứng với số đầu vào ra của module.

Cách gán địa chỉ được thể hiện trên hình 6.3.

CPU 214	**Module 0 (4 vào, 4 ra)**	**Module 1 (8 vào)**	**Module 2 analog (3 vào, 1 ra)**	**Module 3 (8 ra)**	**Module 4 analog (3 vào, 1 ra)**
I0.0 Q0.0 I0.1 Q0.1 I0.2 Q0.2 I0.3 Q0.3 I0.4 Q0.4 I0.5 Q0.5 I0.6 Q0.6 I0.7 Q0.7 I1.0 Q1.0 I1.1 Q1.1 I1.2 I1.3 I1.4 I1.5	I2.0 I2.1 I2.2 I2.3 Q2.0 Q2.1 Q2.2 Q2.3	I3.0 I3.1 I3.2 I3.3 I3.4 I3.5 I3.6 I3.7	AIW0 AIW2 AIW3 AIW4 AQW0	Q3.0 Q3.1 Q3.2 Q3.3 Q3.4 Q3.5 Q3.6 Q3.7	AIW8 AIW10 AIW12 AQW4

Hình 6.3 - Địa chỉ các module mở rộng

6.2. Cấu trúc bộ nhớ

Bộ nhớ được chia thành 4 vùng chính đó là:

6.2.1. Vùng nhớ chương trình

Vùng nhớ chương trình là miền bộ nhớ được sử dụng để lưu giữ các lệnh chương trình. Vùng này thuộc kiểu không đổi (non-volatile) đọc / ghi được.

6.2.2. Vùng tham số

Vùng tham số lưu giữ các tham số như: từ khoá, địa chỉ trạm... vùng này thuộc vùng không đổi đọc / ghi được.

6.2.3. Vùng dữ liệu

Vùng dữ liệu để cất các dữ liệu của chương trình gồm kết quả của các phép tính, các hằng số trong chương trình.... vùng dữ liệu là miền nhớ động, có thể truy nhập theo từng bit, byte, từ (word) hoặc từ kép.

Vùng dữ liệu được chia thành các vùng nhớ nhỏ với các công dụng khác nhau đó là:

TT	Tên tham số	Diễn giải	Tham số	
			CPU 212	CPU214
1	V	Miền đọc ghi	0.0 ÷ 1023.7	0.0 ÷ 4095.7
2	I	Đệm cổng vào	0.0 ÷ 7.7	0.0 ÷ 7.7
3	Q	Đệm cổng ra	0.0 ÷ 7.7	0.0 ÷ 7.7
4	M	Vùng nhớ nội	0.0 ÷ 15.7	0.0 ÷ 31.7
5	SM chỉ đọc	Vùng nhớ đặc biệt	0.0 ÷ 29.7	0.0 ÷ 29.7
6	SM dọc/ghi	Vùng nhớ đặc biệt	30.0 ÷ 45.7	30.0 ÷ 85.7

Địa chỉ truy nhập được qui ước với công thức:

* Truy nhập theo bit: Tên miền + địa chỉ byte.chỉ số bit.

Ví dụ: V150.4 là địa chỉ bít số 4 của byte 150 thuộc miền V.

* Truy nhập theo byte: Tên miền + B và địa chỉ byte.

Ví dụ: VB150 là địa chỉ byte 150 thuộc miền V.

* Truy nhập theo từ (word): Tên miền + W và địa chỉ byte cao của từ.

Ví dụ: VW150 là địa chỉ từ đơn gồm hai byte 150 và 151 thuộc miền V, trong đó byte 150 có vai trò byte cao của từ.

* Truy nhập theo từ kép : Tên miền + D và địa chỉ byte cao của từ.

Ví dụ: VD150 là địa chỉ từ kép gồm bốn byte 150, 151, 152 và 153 thuộc miền V, trong đó byte 150 có vai trò byte cao, 153 có vai trò là byte thấp của từ kép.

Tất cả các byte thuộc vùng dữ liệu đều có thể truy nhập bằng con trỏ. Con trỏ được định nghĩa trong miền V hoặc các thanh ghi AC1, AC2, AC3. Mỗi con trỏ chỉ địa chỉ gồm 4 byte (từ kép). Qui ước sử dụng con trỏ để truy nhập như sau:

& + địa chỉ byte cao

Ví dụ: + AC1 = &VB150 là thanh ghi AC1 chứa địa chỉ byte 150 thuộc miền V.

+ VD100 = &VW150 là từ kép VD100 chứa địa chỉ byte cao của từ đơn VW150 thuộc miền V.

+ AC2 = &VD150 là thanh ghi AC2 chứa địa chỉ byte cao 150 của từ kép VD150 thuộc miền V.

***Toán hạng** ** (con trỏ): là lấy nội dung của byte, từ hoặc từ kép mà con trỏ đang chỉ vào. Với các địa chỉ đã xác định trên ta có các ví dụ:

Ví dụ: + Lấy nội dung của byte VB150 là: *AC1.

+ Lấy nội dung của từ đơn VW150 là: *VD100.

+ Lấy nội dung của từ kép VD150 là: *AC2.

Phép gán địa chỉ và sử dụng con trỏ như trên cũng có tác dụng với những thanh ghi 16 bit của bộ thời gian, bộ đếm thuộc đối tượng.

4. Vùng đối tượng

Vùng đối tượng để lưu giữ dữ liệu cho các đối tượng lập trình như các giá trị tức thời, giá trị đặt trước của bộ đếm, hay bộ thời gian. Dữ liệu kiểu đối tượng bao gồm các thanh ghi của bộ thời gian, bộ đếm, các bộ đếm cao tốc, bộ đệm tương tự và các thanh ghi AC.

Kiểu dữ liệu đối tượng bị hạn chế rất nhiều vì các dữ liệu kiểu đối tượng chỉ được ghi theo mục đích cần sử dụng của đối tượng đó.

TT	Tên tham số	Diễn giải	Tham số	
			CPU 212	CPU 214
1	AC0	ắc qui 0 (Không có khả năng làm con trỏ)		
2	AC	ắc qui	1 ÷ 3	1 ÷ 3
3	C	Bộ đếm	0 ÷ 63	0 ÷ 127
4	HSC	Bộ đếm tốc độ cao		0 ÷ 2
5	AW	Bộ đệm cổng vào tương tự	0 ÷ 30	0 ÷ 30
6	AQW	Bộ đệm cổng ra tương tự	0 ÷ 30	0 ÷ 30
7	T	Bộ thời gian	0 ÷ 63	0 ÷ 127

6.3. Chương trình của S7-200

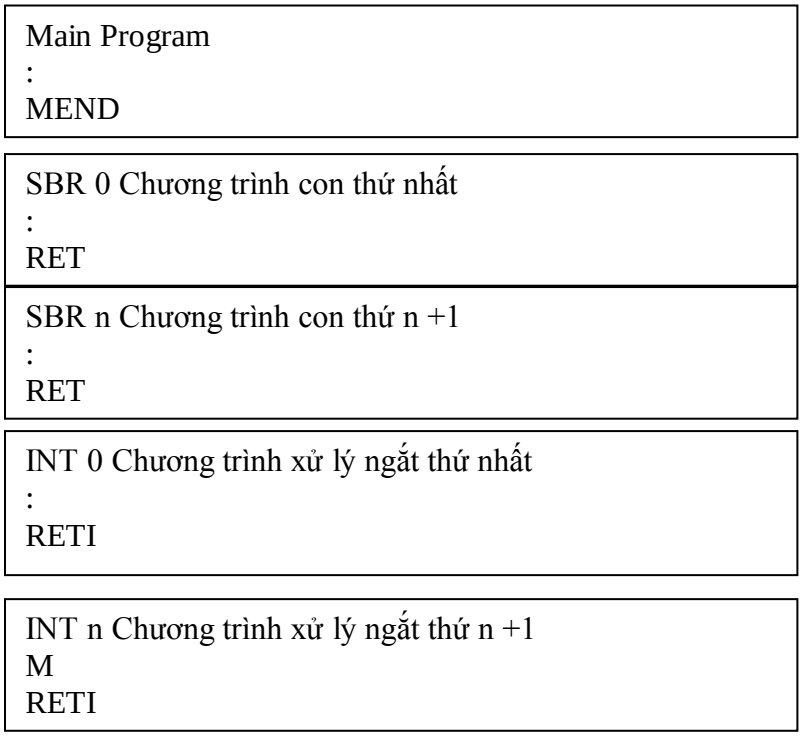
6.3.1. Cấu trúc chương trình S7-200

Các chương trình điều khiển với PLC S7-200 được viết có cấu trúc bao gồm chương trình chính (main program) sau đó đến các chương trình con và các chương trình sử lý ngắt như hình 6.4.

- Chương trình chính được kết thúc bằng lệnh kết thúc chương trình MEND.

- Chương trình con là một bộ phận của chương trình, chương trình con được kết thúc bằng lệnh RET. Các chương trình con phải được viết sau lệnh kết thúc chương trình chính MEND.

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



Hình 6.4 - Cấu trúc chương trình của S7-200

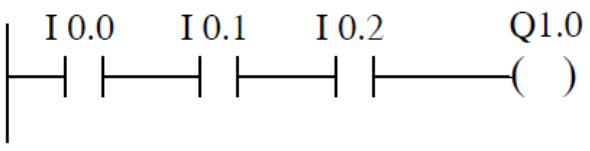
Các chương trình con được nhóm lại thành một nhóm ngay sau chương trình chính. Sau đó đến ngay các chương trình xử lý ngắt. Có thể tự do trộn lẫn các chương trình con và chương trình xử lý ngắt đằng sau chương trình chính.

6.3.2. Bảng lệnh của S7-200

Xem phần phụ lục.

6.4. Lập trình một số lệnh cơ bản của S7-200

6.4.1. Lệnh LD và lệnh A



Hình 6.5 - Lệnh LD và A

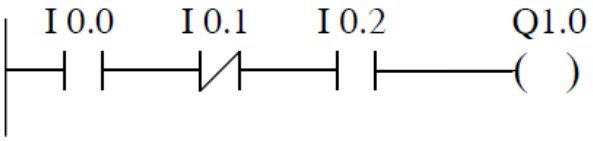
Lập trình dạng STL.

```
LD   I   0.0
A    I   0.1
A    I   0.2
=    Q   1.0
```

6.4.2. Lệnh AN

Lập trình dạng STL.

```
LD   I   0.0
AN   I   0.1
A    I   0.2
=    Q   1.0
```

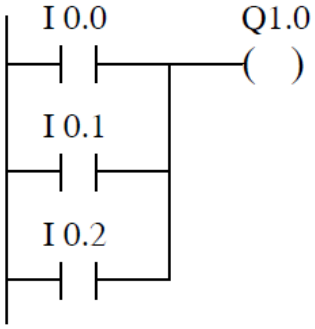


Hình 6.6 - Lệnh AN

6.4.3. Lệnh O

Lập trình dạng STL.

```
LD   I   0.0
O    I   0.1
O    I   0.2
=    Q   1.0
```

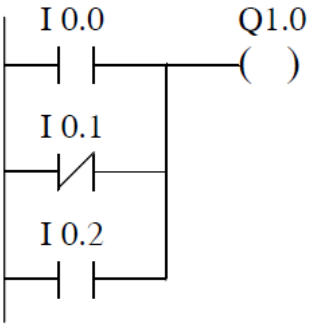


Hình 6.7 - Lệnh O

6.4.4. Lệnh ON:

Lập trình dạng STL.

```
LD   I   0.0
ON   I   0.1
O    I   0.2
=    Q   1.0
```

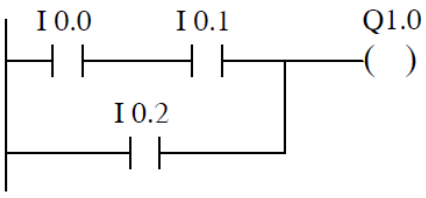


Hình 6.8 - Lệnh ON

6.4.5. Lệnh OLD

Lập trình dạng STL (có thể lập trình dạng LAD và kiểm tra lại dạng STL).

```
LD   I   0.0
A    I   0.1
LD   I   0.2
OLD
=    Q   1.0
```

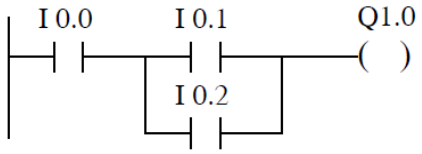


Hình 6.9 - Lệnh OLD

6.4.6. Lệnh ALD

.

```
LD   I   0.0
LD   I   0.1
O    I   0.2
ALD
=    Q   1.0
```

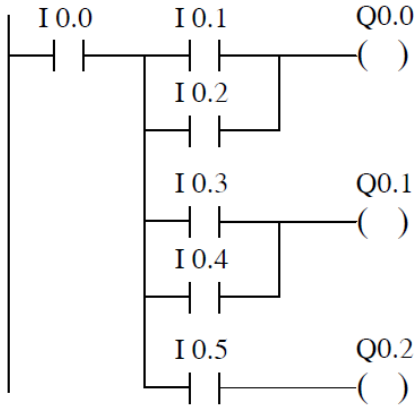


Hình 6.10 - Lệnh ALD

6.4.7. Lệnh LPS, LRD, LPP

Lập trình dạng STL

```
LD   I   0.0
LPS
LD   I   0.1
O    I   0.2
ALD
=    Q   0.0
LRD
LD   I   0.3
0    I   0.4
ALD
=    Q   0.1
LPP
A    I   0.5
=    Q   0.2
```



Hình 6.11 - Lệnh LPS, LRD, LPP

CHƯƠNG 7: BỘ ĐIỀU KHIỂN PLC - S7-300

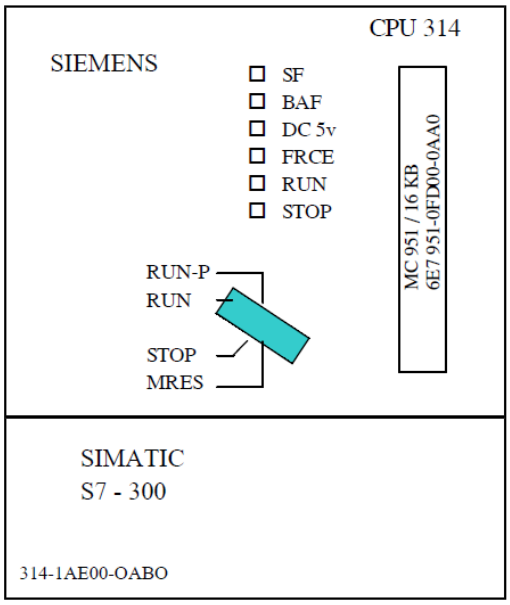
7.1. Cấu hình phần cứng

7.1.1. Cấu tạo của họ PLC- S7-300

PLC Step S7-300 thuộc họ Simatic do hãng Siemens sản xuất. Đây là loại PLC đa khối. Cấu tạo cơ bản của loại PLC này là một đơn vị cơ bản (chỉ để xử lý) sau đó ghép thêm các module mở rộng về phía bên phải, có các module mở rộng tiêu chuẩn. Những module ngoài này bao gồm những đơn vị chức năng mà có thể tổ hợp lại cho phù hợp với những nhiệm vụ kỹ thuật cụ thể.

a. Đơn vị cơ bản

Đơn vị cơ bản của PLC S7-300 như hình 7.1



Trong đó:

1. Các đèn báo:

+ Đèn SF: báo lỗi CPU.

+ Đèn BAF: Báo nguồn ắc qui.

+ Đèn DC 5v: Báo nguồn 5v.

+ Đèn RUN: Báo chế độ PLC đang làm việc.

+ Đèn STOP: Báo PLC đang ở chế độ dừng.

2. Công tắc chuyển đổi chế độ:

+ RUN-P: Chế độ vừa chạy vừa sửa chương trình.

+ RUN: Đưa PLC vào chế độ làm việc.

+ STOP: Để PLC ở chế độ nghỉ.

+ MRES: Vị trí chỉ định chế độ xoá chương trình trong CPU.

Muốn xoá chương trình thì giữ nút bấm về vị trí MRES để đèn STOP nhấp nháy, khi thôi không nhấp nháy thì nhả tay. Làm lại nhanh một lần nữa (không để ý đèn STOP) nếu đèn vàng nháy nhiều lần là xong, nếu không thì phải làm lại.

b. Các kiểu module

Tuỳ theo quá trình tự động hoá đòi hỏi số lượng đầu vào và đầu ra ta phải lắp thêm bao nhiêu module mở rộng cũng như loại module cho phù hợp. Tối đa có thể gá thêm 32 module vào ra trên 4 panen (rãnh), trên mỗi panen ngoài module nguồn, CPU và module ghép nối còn gá được 8 các module về bên phải. Thường Step 7-300 sử dụng các module sau:

+ Module nguồn PS

+ Module ghép nối IM (Intefare Module):

+ Module tín hiệu SM (Signal Module):

- Vào số: 8 kênh, 16 kênh, 32 kênh.

- Ra số: 8 kênh, 16 kênh, 32 kênh.

- Vào, ra số: 8 kênh vào 8 kênh ra, 16 kênh vào 16 kênh ra.

- Vào tương tự: 2 kênh, 4 kênh, 8 kênh.

- Ra tương tự: 2 kênh, 4 kênh, 8 kênh.

- Vào, ra tương tự: 2 kênh vào 2 kênh ra, 4 kênh vào 4 kênh ra.

+ Module hàm (Function Module).

- Đếm tốc độ cao.

- Truyền thông CP 340, CP340-1, CP341.

+ Module điều khiển (Control Module):

- Module điều khiển PID.

- Module điều khiển Fuzzy.

- Module điều khiển rô bot.

- Module điều khiển động cơ bước.

- Module điều khiển động cơ Servo.

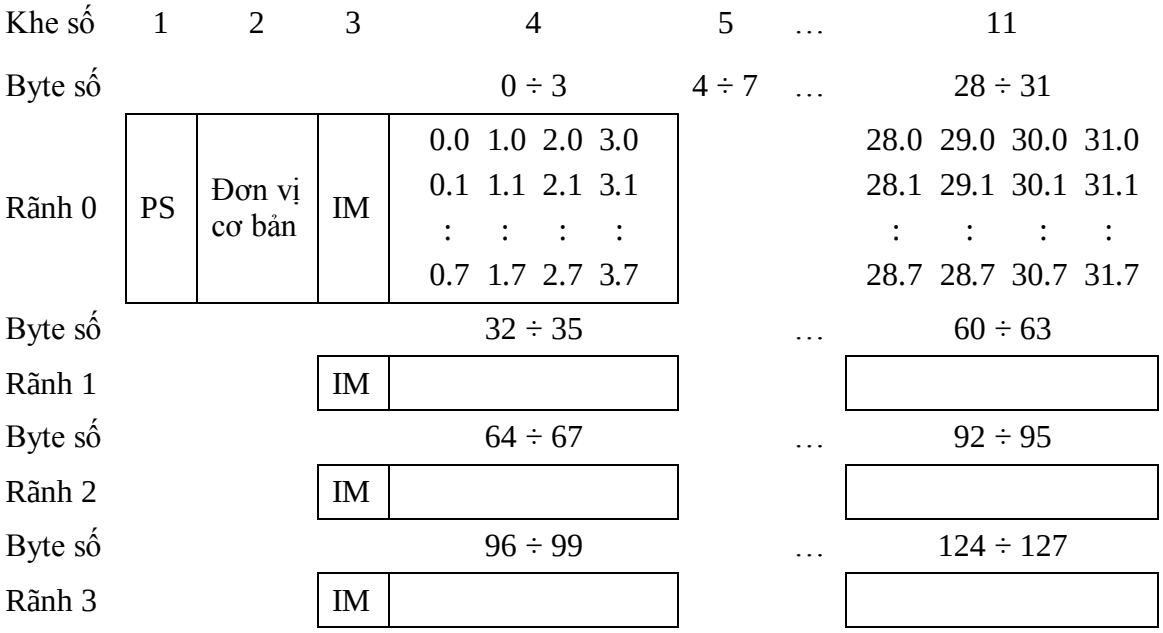
7.1.2. Địa chỉ và gán địa chỉ

Trong PLC các bộ phận cần gửi thông tin đến hoặc lấy thông tin đi đều phải có địa chỉ để liên lạc. Địa chỉ là con số hoặc tổ hợp các con số đi theo sau chữ cái. Chữ cái chỉ loại địa chỉ, con số hoặc tổ hợp con số chỉ số hiệu địa chỉ.

Trong PLC có những bộ phận được gán địa chỉ đơn như bộ thời gian (T), bộ đếm (C).... chỉ cần một trong 3 chữ cái đó kèm theo một số là đủ, *ví dụ*: T1, C32... Các địa chỉ đầu vào và đầu ra cùng với các module chức năng có cách gán địa chỉ giống nhau. Địa chỉ phụ thuộc vào vị trí gá của module trên Panen. Chỗ gá module trên panen gọi là khe (Slot), các khe đều có đánh số, khe số 1 là khe đầu tiên của và cứ thế tiếp tục.

a. Địa chỉ vào ra module số:

Khi gá module số vào, ra lên một khe nào lập tức nó được mạng địa chỉ byte của khe đó, mỗi khe có 4 byte địa chỉ.



Hình 7.2 - Địa chỉ khe và kênh trên module số

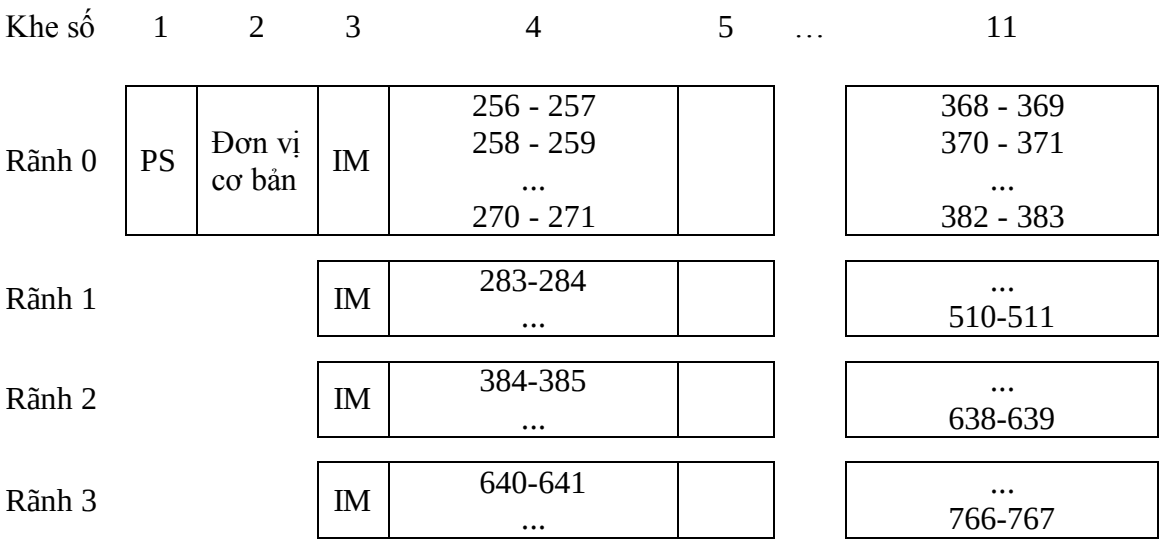
Trên mỗi module thì mỗi đầu vào, ra là một kênh, các kênh đều có địa chỉ bit là 0 đến 7. Địa chỉ của mỗi đầu vào, ra là số ghép của địa chỉ byte và địa chỉ kênh, địa chỉ byte đứng trước, địa chỉ kênh đứng sau, giữa hai số có dấu chấm. Khi các module gá trên khe thì địa chỉ được tính từ byte đầu của khe, các đầu vào và ra của một khe có cùng địa chỉ. Địa chỉ byte và địa chỉ kênh như hình.

Ví dụ: Module 2 đầu vào, 2 đầu ra số gá vào khe số 5 rãnh 0 có địa chỉ là I4.0, I4.1 và Q4.0, Q4.1. Module số có thể được gá trên bất kỳ khe nào trên panen của PLC.

b. Địa chỉ vào ra module tương tự

Để diễn tả một giá trị tương tự ta phải cần nhiều bit. Trong PLC S7-300 người ta dùng 16 bit (một word) cho một kênh.

Một khe có 8 kênh với địa chỉ đầu tiên là PIW256 hoặc PQW256 (byte 256 và 257) cho đến PIW766 hoặc PQW766 như hình 7.3.



Hình 7.3 - Địa chỉ của module tương tự

Module tương tự có thể được gá vào bất kỳ khe nào trên panen của PLC.

Ví dụ: Một module tương tự 2 vào, 1 ra gá vào khe số 6 rãnh 0 có địa chỉ là PIW288, PIW290, PQW288.

Chú ý: Các khe trống bao giờ cũng có trạng thái tín hiệu "0".

7.2. Vùng đối tượng

7.2.1. Các vùng nhớ

Bảng 7.1

TT	Tên tham số	Diễn giải	Vùng tham số
1	I	Đầu vào bit	0.0 đến 65535.7
2	IB	Đầu vào byte	0 đến 65535
3	IW	Đầu vào từ	0 đến 65534
4	ID	Đầu vào từ kép	0 đến 65532
5	Q	Đầu ra bit	0.0 đến 65535.7
6	QB	Đầu ra byte	0 đến 65535
7	QW	Đầu ra từ	0 đến 65534
8	QD	Đầu ra từ kép	0 đến 65532
9	M	Nhớ nội dạng bit	0.0 đến 255.7

10	MB	Nhớ nội dạng byte	0 đến 255
11	MW	Nhớ nội dạng từ	0 đến 254
12	MD	Nhớ nội dạng từ kép	0 đến 252
13	PIB	Vùng đệm đầu vào dạng byte	0 đến 65535
14	PIW	Vùng đệm đầu vào dạng từ	0 đến 65534
15	PID	Vùng đệm đầu vào dạng từ kép	0 đến 65532
16	PQB	Vùng đệm đầu ra dạng byte	0 đến 65535
17	PQW	Vùng đệm đầu ra dạng từ	0 đến 65534
18	PQD	Vùng đệm đầu ra dạng từ kép	0 đến 65532
19	T	Bộ thời gian	0 đến 255
20	C	Bộ đếm	0 đến 255
21	DBX	Khối dữ liệu kiểu BD dạng bit	0.0 đến 65535.7
22	DBB	Khối dữ liệu kiểu BD dạng byte	0 đến 65535
23	DBW	Khối dữ liệu kiểu BD dạng từ	0 đến 65534
24	DBD	Khối dữ liệu kiểu BD dạng từ kép	0 đến 65532
25	DIX	Khối dữ liệu kiểu BI dạng bit	0.0 đến 65535.7
26	DIB	Khối dữ liệu kiểu BI dạng byte	0 đến 65535
27	DIW	Khối dữ liệu kiểu BI dạng từ	0 đến 65534
28	DID	Khối dữ liệu kiểu BI dạng từ kép	0 đến 65532
29	L	Vùng dữ liệu tạm thời dạng bit	0.0 đến 65535.7
30	LB	Vùng dữ liệu tạm thời dạng byte	0 đến 65535
31	LW	Vùng dữ liệu tạm thời dạng từ	0 đến 65534
32	LD	Vùng dữ liệu tạm thời dạng từ kép	0 đến 65532

7.2.2. Nhập các hằng số

Các hằng số được viết gồm phần đầu và tham số đi liền nhau

Ví dụ: B#16#1A là số: (viết dạng byte, cơ số 16, giá trị là 1A tương ứng cơ số thập phân là 26).

Các hằng số về thời gian được viết theo các ký hiệu: D (Date) ngày_ H (Hours) giờ_ M (minuter) phút_ S (seconds) giây_ MS (milliseconds) mili giây

Ví dụ 2D_23H_10M_50S_13MS là: (2 ngày, 23 giờ, 10 phút, 50 giây, 13 mili giây).

Các kiểu viết hằng số được thể hiện trên bảng 7.2:

Bảng 7.2

Loại	**Bit**	**Cơ số**	**Phần đầu**	**Phạm vi tham số**
Byte	8	16	B#16#...	0 đến FF

Từ	16	2 16 BCD 10 không dấu	2#... W#16#... C# B#...	0 đến 1111_1111_1111_1111 0 đến FFFF 0 đến 999 (0,0) đến (255,255)
Từ kép	32	2 16 10 không dấu	2#... DW#16#... B#...	0 đến 1111_1111_1111_1111_ 1111_1111_1111_1111 0000_0000 đến FFFF_FFFF (0,0,0,0) đến (255,255,255,255)
Số thực	16	có dấu	(không có)	-32768 đến 32767
Số thực	32	có dấu	L#	-2147483648 đến +2147483647
Số thực	32	dấu phảy động	(không có)	lớn hơn ± 3,402823 e+38 nhỏ hơn ± 1.175495e-38
Thời gian	16 32	giờ_phút_ giấy_miligiây ngày_giờ_ phút_giây_ miligiây	S5T#..... T#...	0H_0M_0S_10MS đến 2H_46M_30S_0MS -24D_20H_31M_23S_648MS đến 24D_20H_31M_23S_647MS
Ngày		năm- thángngày	D#...	1990-1-1 đến 2168-12-31
Thời gian của ngày	32	giờ:phút: giây.ngày	TOD#...	0:0:0.0 đến 23:59:59.999
Ký tự	8		'....'	viết các ký tự như 'HA'

7.3. Ngôn ngữ lập trình

7.3.1. Cấu trúc chương trình S7-300

Các chương trình điều khiển với PLC S7-300 có thể được viết ở dạng đơn khối hoặc đa khối.

Chương trình đơn khối

Chương trình đơn khối chỉ viết cho các công việc tự động đơn giản, các lệnh được viết tuần tự trong một khối. Khi viết chương trình đơn khối người ta dùng khối OB1. Bộ PLC quét khối theo chương trình, sau khi quét đến lệnh cuối cùng nó quay trở lại lệnh đầu tiên.

Chương trình đa khối (có cấu trúc)

Khi nhiệm vụ tự động hoá phức tạp người ta chia chương trình điều khiển ra thành từng phần riêng gọi là khối. Chương trình có thể xếp lồng khối này vào khối kia. Chương trình đang thực hiện ở khối này có thể dùng lệnh gọi khối để

sang làm việc với khối khác, sau khi đã kết thúc công việc ở khối mới nó quay về thực hiện tiếp chương trình đã tạm dừng ở khối cũ.

Các khối được xếp thành lớp. Mỗi khối có:

+ Đầu khối gồm tên khối, số hiệu khối và xác định chiều dài khối.

+ Thân khối: Thể hiện nội dung khối và được chia thành đoạn (Segment) thực hiện từng công đoạn của tự động hoá sản xuất. Mỗi đoạn lại bao gồm một số dòng lệnh phục vụ việc giải bài toán logic. Kết quả của phép toán logic được gửi vào RLO (Result of logic operation). Việc phân chia chương trình thành các đoạn cũng ảnh hưởng đến RLO. Khi bắt đầu một đoạn mới thì tạo ra một giá trị RLO mới, khác với giá trị RLO của đoạn trước.

+ Kết thúc khối: Phần kết thúc khối là lệnh kết thúc khối BEU.

Các loại khối:

* *Khối tổ chức* OB (Organisation Block)

Khối tổ chức quản lý chương trình điều khiển và tổ chức việc thực hiện chương trình.

** Hàm số FC (Functions)*

Khối hàm số FC là một chương trình do người sử dụng tạo ra hoặc có thể sử dụng các hàm chuẩn sẵn có của SIEMENS.

* *Khối hàm* FB (Function Block)

Khối hàm là loại khối đặc biệt dùng để lập trình các phần chương trình điều khiển tái diễn thường xuyên hoặc đặc biệt phức tạp. Có thể gán tham số cho các khối đó và chúng có một nhóm lệnh mở rộng. Người sử dụng có thể tạo ra các khối hàm mới cho mình, có thể sử dụng các khối hàm sẵn có của SIEMENS.

** Khối dữ liệu: có hai loại là*

+ Khối dữ liệu dùng chung DB (Shared Data Block)

Khối dữ liệu dùng chung lưu trữ các dữ liệu chung cần thiết cho việc xử lý chương trình điều khiển.

+ Khối dữ liệu riêng DI (Instance Data Block)

Khối dữ liệu dùng riêng lưu trữ các dữ liệu riêng cho một chương trình nào đó cho việc xử lý chương trình điều khiển.

Ngoài ra trong PLC S7-300 còn hàm hệ thống SFC (System Function) và khối hàm hệ thống SFB (System Function Block).

7.3.2. Bảg lệnh của S7-300

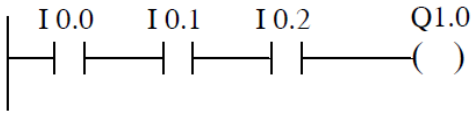
Xem phần phụ lục

7.4. Lập trình một số lệnh cơ bản

7.4.1. Lệnh LD và lệnh A

Lập trình dạng STL.

```
LD   I   0.0
A    I   0.1
A    I   0.2
=    Q   1.0
```

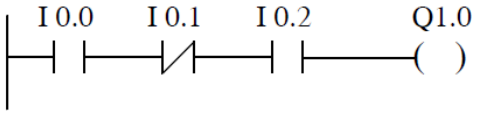


Hình 7.4 - Lệnh LD và A

7.4.2. Lệnh AN

Lập trình dạng STL.

```
LD   I   0.0
AN   I   0.1
A    I   0.2
=    Q   1.0
```

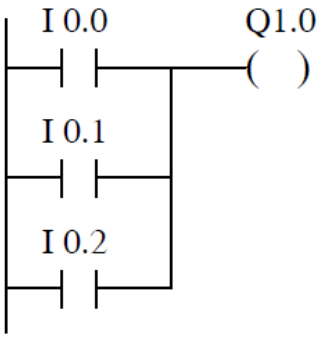


Hình 7.5 - Lệnh AN

7.4.3. Lệnh O

Lập trình dạng STL.

```
LD   I   0.0
O    I   0.1
O    I   0.2
=    Q   1.0
```

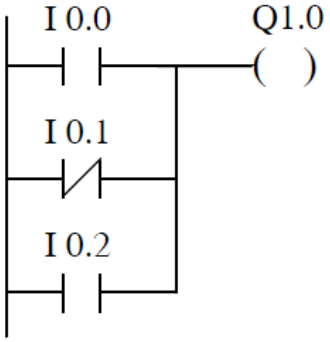


Hình 7.6 - Lệnh O

7.4.4. Lệnh ON

Lập trình dạng STL.

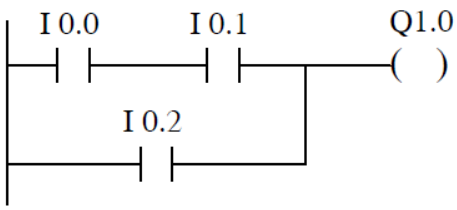
```
LD   I   0.0
ON   I   0.1
O    I   0.2
=    Q   1.0
```



7.4.5. Lệnh OLD

Lập trình dạng STL (có thể lập trình dạng LAD và kiểm tra lại dạng STL).

```
LD   I   0.0
A    I   0.1
LD   I   0.2
OLD
=    Q   1.0
```

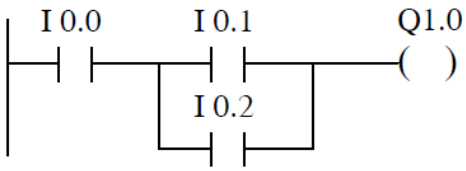


Hình 7.8 - Lệnh OLD

7.4.6. Lệnh ALD

Lập trình dạng STL.

```
LD    I    0.0
LD    I    0.1
O     I    0.2
ALD
=     Q    1.0
```

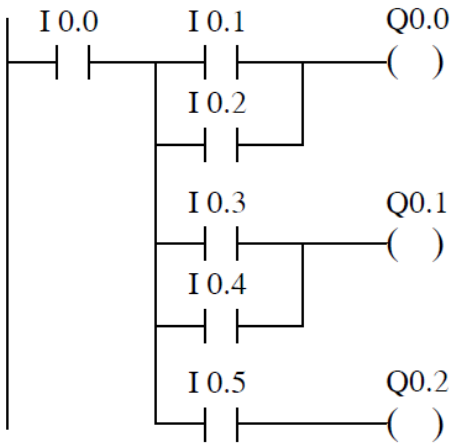


Hình 7.9 - Lệnh ALD

7.4.7. Lệnh LPS, LRD, LPP

Lập trình dạng STL

```
LD    I    0.0
LPS
LD    I    0.1
O     I    0.2
ALD
=     Q    0.0
LRD
LD    I    0.3
0     I    0.4
ALD
```



Hình 7.10 - Lệnh LPS, LRD, LPP

CHƯƠNG 8: CÁC PHẦN MỀM LẬP TRÌNH PLC

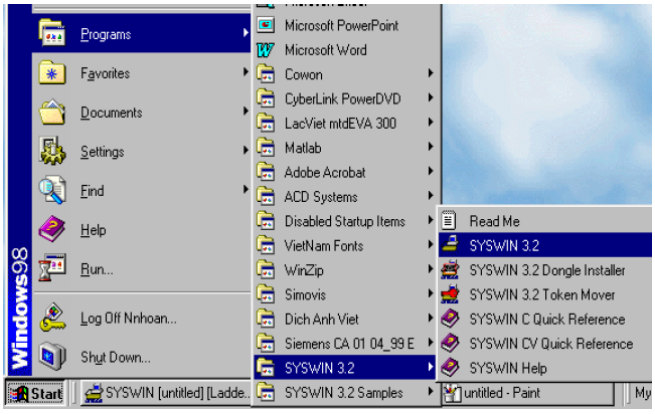
8.1. Lập trình cho OMRON

8.1.1. Phần mềm SYSWIN (cho OMRON)

a. Khởi động

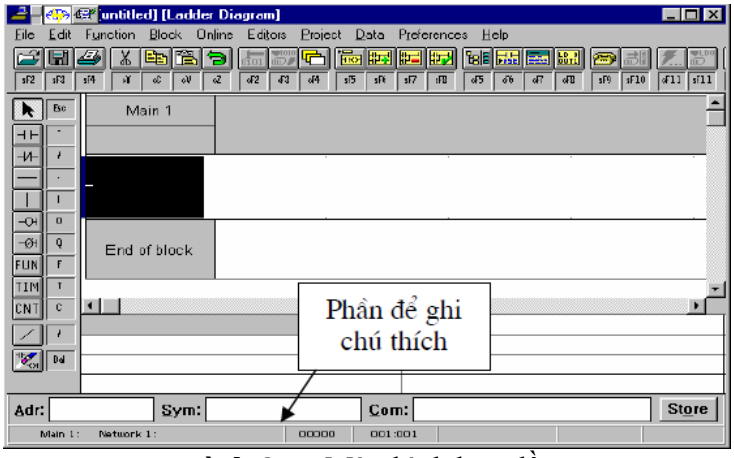
1. Khởi động máy tính ở chế độ Windows, bật công tắc nguồn của khối PLC.

2. Khởi động phần mềm SYSWIN từ biểu tượng hoặc từ file chương trình như hình 8.1.



Hình 8.1 - Khởi động phần mềm SYSWIN

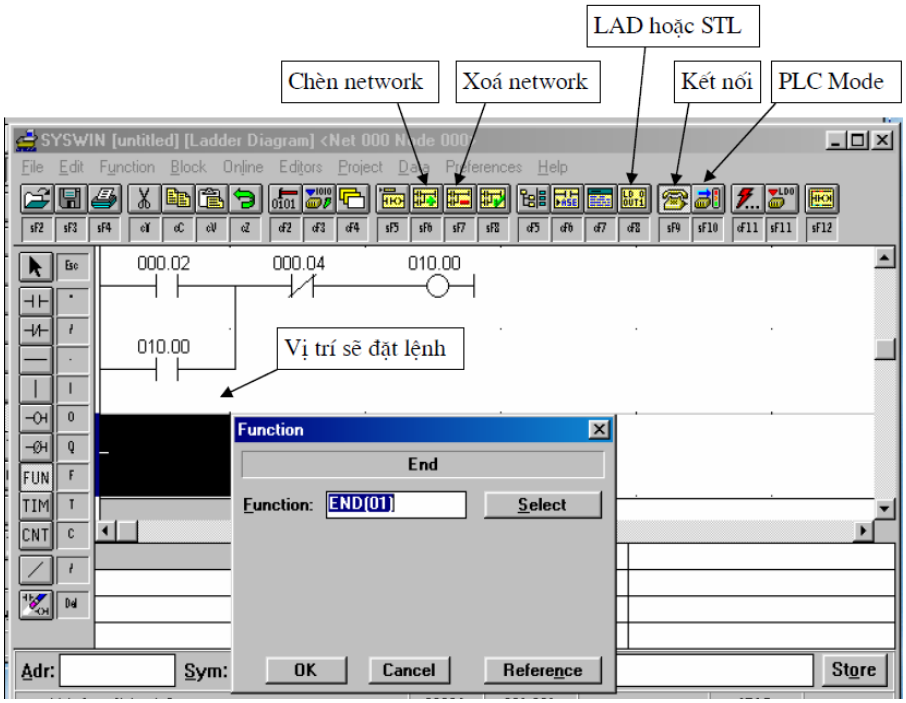
Cửa sổ màn hình ban đầu có dạng như hình 8.2. Trong cửa sổ có 2 thanh công cụ hỗ trợ cho quá trình soạn thảo chính là:



Hình 8.2 - Màn hình ban đầu

• Thanh trên: ngoài một số chức năng như soạn thảo văn bản bình thường còn một số chức năng để soạn thảo lệnh như chỉ ra trên hình 8.3.

• Thanh dọc: Lần lượt từ trên là: Con trỏ (để chọn), tiếp điểm thường hở, thường kín, thanh nối ngang, thanh nối dọc, cuộn dây thường mở, cuộn dây thường đóng, khối hàm (FUN), bộ thời gian (TIM), bộ đếm (CNT)



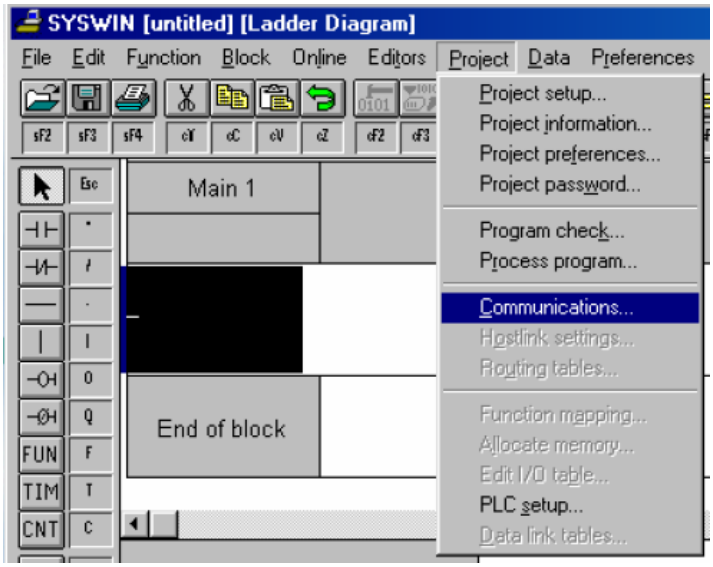
Hình 8.3: Một số chức năng chính

3. Kiểm tra một số điều kiện trước khi lập trình:

+ Kiểm tra xem máy tính đã được kết nối với PLC chưa. Khi máy tính đã được kết nối với PLC thì biểu tượng kết nối sáng, nếu chưa được kết nối thì nháy vào biểu tượng kết nối hệ

thống sẽ tự kết nối với PLC.

+ Nếu sự kết nối không thực hiện được có thể phải khai báo lại cổng như chỉ ra trên hình 8.4. (đường dẫn Project \ Communications).



Hình 8.4: Khai báo cổng kết nối

b. Soạn thảo: Theo LAD

1. Mở một file chương trình mới hoặc một file chương trình đã có (chế độ mặc định đã có một file mới được mở ra).

Chèn network Xoá network Kết nối PLC Mode LAD hoặc STL

Vị trí sẽ đặt lệnh

2. Nháy chuột trái vào khối muốn chọn (tiếp điểm, cuộn dây, khối hàm....)

3. Đưa con chỏ đến vị trí đặt lệnh (vị trí tô đen), nháy chuột trái và vào địa chỉ lệnh (Đầu vào có các địa chỉ: 0, đến 11; đầu ra có các địa chỉ: 1000, đến 1007).

4. Khi cần ghi chú thích dưới mỗi lệnh thì chọn lệnh cần ghi chú thích, vào hộp SYM: (ở phía dưới màn hình như trên hình 8.2) ghi những điều cần chú thích, câu chú thích phải liền nhau (không dùng dấu cách) sau đó chọn Store.

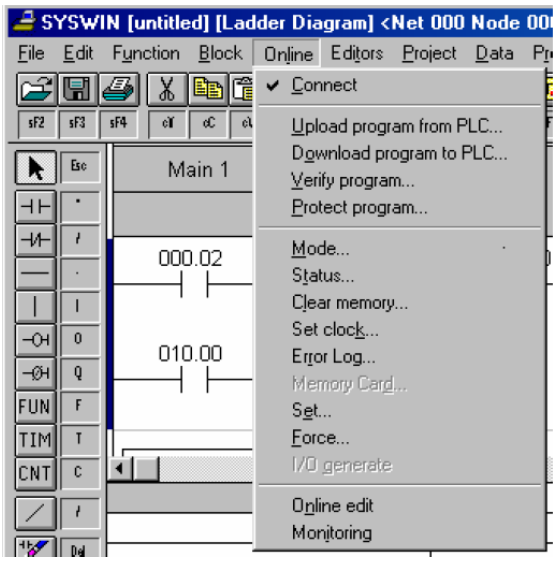
5. Kết thúc một Network chèn thêm Network mớt từ biểu tượng như chỉ ra trên hình 8.3.

6. Nếu soạn sai Network nào thì đánh dấu và xoá Network đó từ biểu tượng hình 8.3.

7. Tiến hành soạn thảo hết các Network.

8. Kết thúc chương trình phải có lệnh kết thúc. Muốn vào lệnh kết thúc thì chọn Netwoks và vị trí lệnh kết thúc, chọn FUN, nháy vào vị trí đặt lệnh, sau đó vào tên lệnh END(01) như chỉ ra trên hình 8.3, hoặc chọn các khối ở mục Select sau đó chọn OK.

9. Đổ chương trình sang PLC chọn Online \ Download program to PLC như trên hình 8.5.



Hình 8.5: Đổ chương trình sang PLC

Chú ý: Khi đổ chương trình sang PLC thì PLC phải đang ở trạng thái MONITOR hoặc trạng thái PROGRAM (STOP/PRG). Muốn chuyển đổi các trạng thái trên thì chọn Shift + F10 hoặc biểu tượng "PLC Mode" như hình 8.3.

10.Để chạy chương trình chọn trạng thái **MONITOR** hoặc **RUN** từ biểu tượng "PLC Mode".

8.1.2. Sử dụng thiết bị lập trình cầm tay (cho OMRON)

a. Cấu tạo thiết bị lập trình cầm tay

Thiết bị lập trình cầm tay có các khối chính như hình P.6.

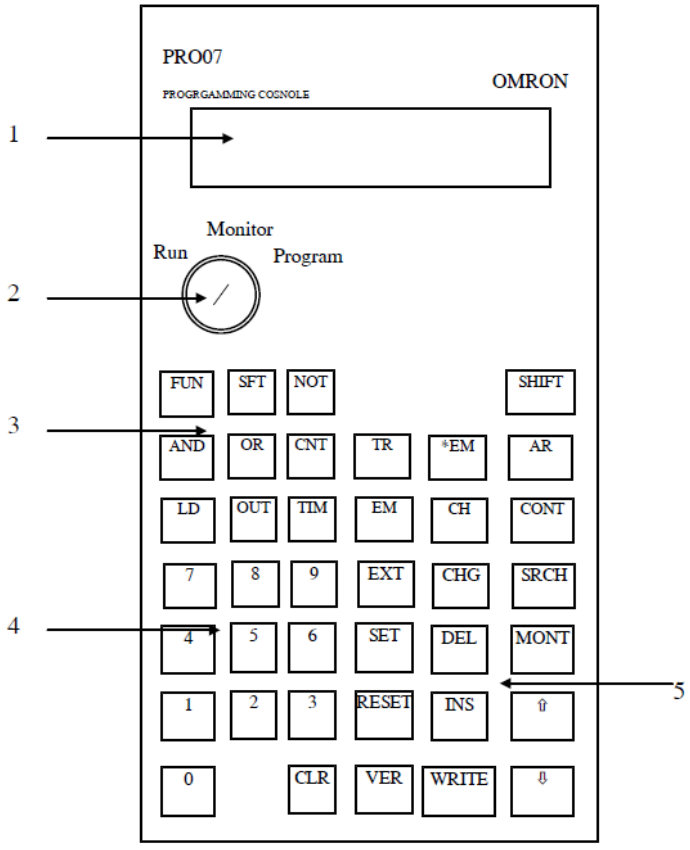
1. Màn hình

2. Công tắc chọn chế độ: có 3 chế độ

* **PROGRAM**: chế độ này để lập trình hoặc thực hiện các thay đổi chương trình.

* **MONITOR**: Chế độ này để thay đổi các giá trị của bộ đếm và thời gian trong khi PLC vẫn đang vận hành.

* **RUN**: Chế độ này để chạy chương trình đã nạp trong PLC (khi PLC đang ở chế độ này thì không đổ chương trình mới sang PLC được).



Hình 8.6: Thiết bị lập trình cầm tay

3. Các phím lệnh
4. Các phím số.
5. Các phím hàm.

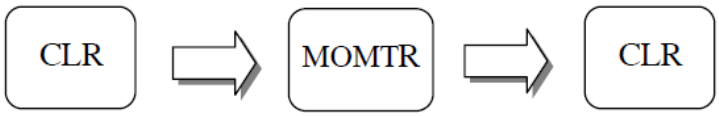
b. Các phím lệnh

FUN	Các lệnh ứng dụng đặc biệt
TIM	Lệnh điều khiển thời gian
LD ⊣⊢	Lệnh nhập các tiếp điểm vào chương trình. (lệnh bắt đầu một Network)
CNT	Lệnh điều khiển bộ đếm
OR ⊣⊢	Lệnh OR (nối song song)
NOT	Dùng kèm với các lệnh LD, AND, OR, OUT để thực hiện phép nghịch đảo
AND ⊣⊢	Lệnh AND (nối nối tiếp)
TR	Thiết lập các rơ le tạm thời
OUT —○⊣	Lệnh ra
AR / HR	Thiết lập các rơ le duy trì
SET	Chỉ thị vận hành của bộ ghi dịch
SHIFT	Dùng để thay đổi các chức năng của các phím nhiều chức năng
A 0	Các phím số 0 đến 9 để nhập số thập phân, hexa

CLR Lệnh xoá trước khi lập trình

c. Thủ tục vào lệnh:

1. Khởi động bộ lập trình cầm tay, công tắc chọn chế độ để ở chế độ **PROGRAM** hoặc chế độ **MONITOR,** vào PASSWORD (từ khoá) theo thứ tự sau:



2. Bắt đầu chương trình mới cần sử dụng lệnh CLR để xoá chương trình cũ.

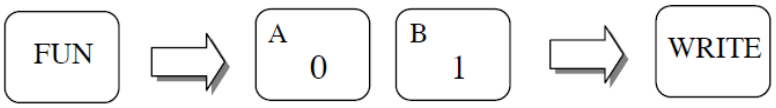
3. Các lệnh được vào theo thứ tự:

+ Tên lệnh (các lệnh bắt đầu một NETWORK là lệnh LD).

+ Tham số của lệnh: Không cần vào các số không đứng trước.

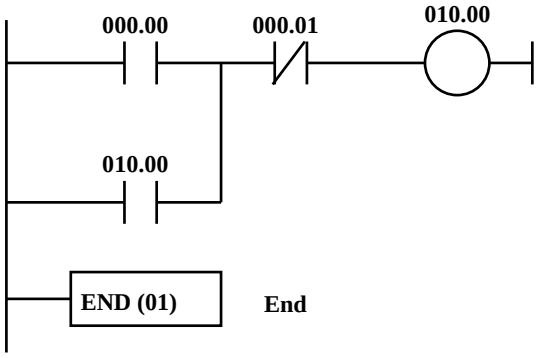
+ Kết thúc một lệnh là WRITE (viết vào PLC)

4. Kết thúc một chương trình phải có lệnh kết thúc. Lệnh kết thúc vào theo thứ tự:



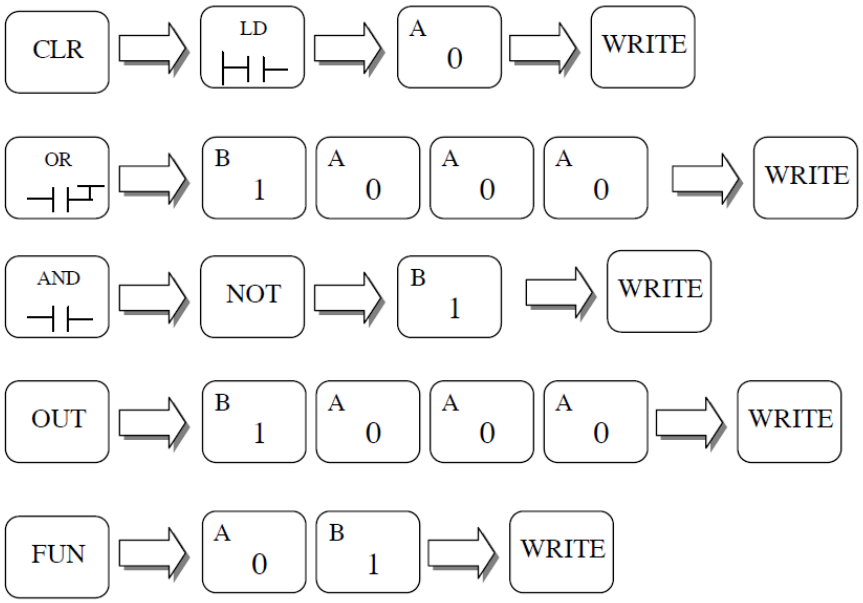
Ví dụ: Chương trình của một mạch tự duy trì dạng LAD và STL như hình 8.7:

```
LD        000.00
OR        010.00
AND NOT   000.01
OUT       010.00
END.
```



Hình 8.7: Mạch tự duy trì

Cách vào chương trình hình 8.7 như sau:



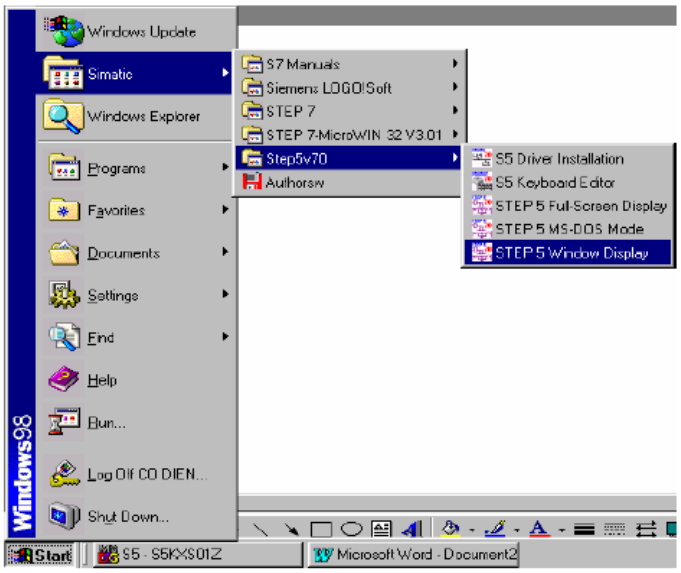
6. Để chạy chương trình chuyển công tắc chọn chế độ sang **RUN**.

8.2. Lập trình cho PLC - S5 (Sử dụng phần mềm Step 5 for Win)

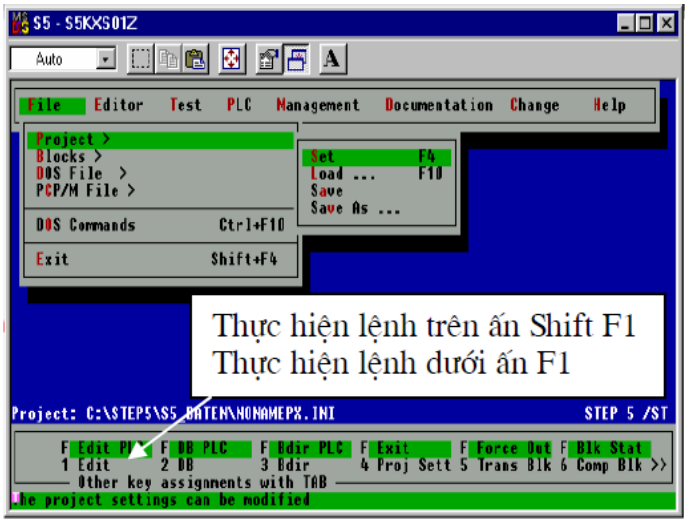
8.2.1. Trình tự thao tác

1. Khởi động máy tính ở chế độ Windows, (bật công tắc nguồn khối thí nghiệm, PLC đặt trong khối thí nghiệm), bật công tắc khối nguồn PS của PLC, công tắc của khối CPU để ở vị trí STOP.

2. Chạy trình Step5 từ fite chương trình như hình 8.8. Màn hình chế độ bắt đầu có dạng như hình 8.9.



Hình 8.8: Khởi động STEP 5



Hình 8.9: Màn hình ban đầu

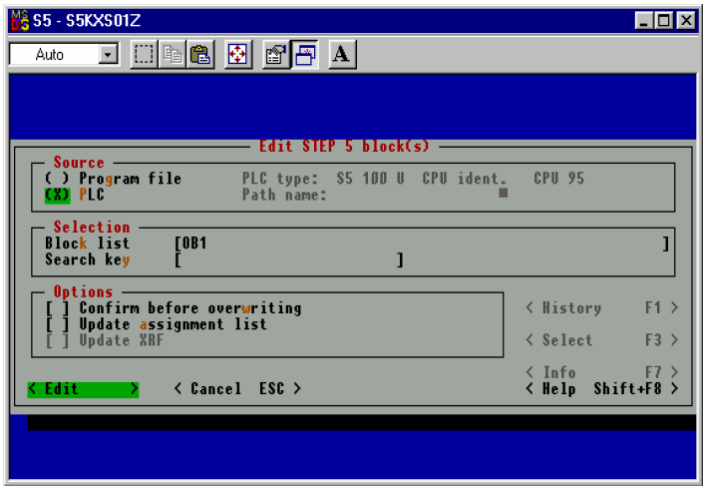
3. Vào File \ Project \ Set. Cần đặt 3 tham số cơ bản.

+ Chọn PLC \ Mode để đặt chế độ Online (chế độ kết nối với PLC).

+ Chọn Blocks \ Representation để đặt chế độ soạn thảo STL.

+ Chọn Blocks \ Program File để tạo file mới, (nếu cần mở một file đã có thì vào đường dẫn và tên file, nếu sử dụng file ngay buổi làm việc trước và chương trình trước đây đã kết nối với PLC thì bỏ qua bước này) sau đó ấn Enter.

4. Vào chế độ soạn thảo từ Editor \ Step 5 Block...., hoặc ấn F1 (Edit). Màn hình trước soạn thảo có dạng như hình 8.10.



Hình 8.10: Màn hình trước soạn thảo

Trong đó:

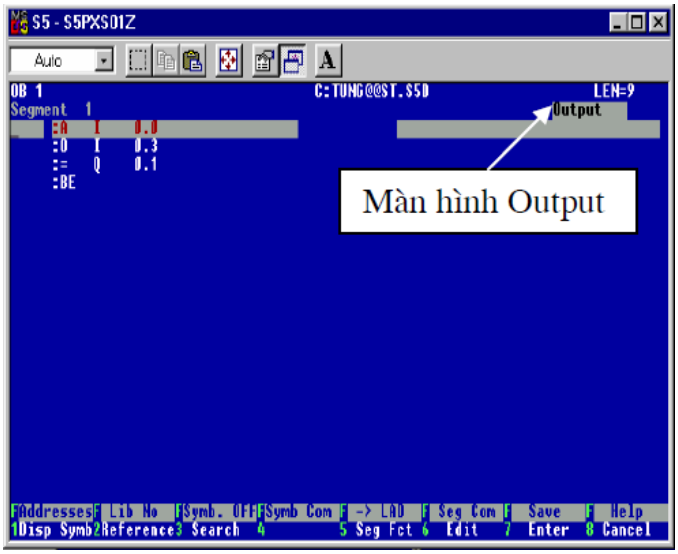
Block list: Vào tên của khối hoặc nhiều khối để soạn thảo.

Confirm before overwriting: Nếu được chọn thì khi ghi đè máy sẽ hỏi lại để khẳng định, không chọn thì khối sửa đổi được ghi đè lên ngay sau khi bấm Enter.

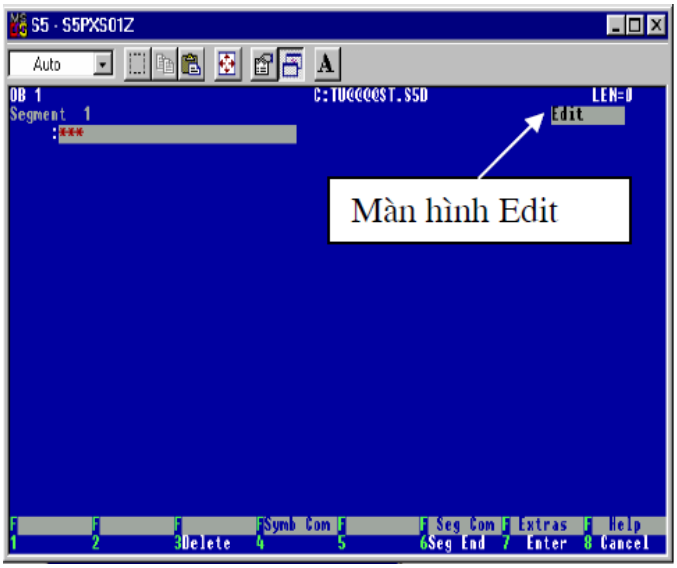
Update assignment: Nếu được chọn thì fite biểu tượng *ZO.INI thay đổi thì fite nguồn *ZO.SEQ cũng được điều chỉnh, nếu không chọn thì fite nguồn *ZO.SEQ không được điều chỉnh.

Update XRF: Nếu được chọn thì fite *XR.INI chứa tham chiếu chéo được điều chỉnh hoặc được tạo nếu chưa tồn tại trước đó, nếu không chọn thì fite *XR.INI chứa tham chiếu chéo không được điều chỉnh.

5. Trong mục Source chọn PLC để kết nối trực tiếp với PLC. Trong mục Selection \ Block list vào khối OB1 để soạn thảo (có thể vào các khối khác nếu cần), trong mục Options không chọn như hình 8.10 sau đó chọn Edit (ấn Enter), nếu làm việc với file mới thì máy tự động vào luôn màn hình Edit như hình 8.11b, nếu làm việc với file cũ thì máy vào màn hình Output như hình 8.11a.



a) Màn hình Output



b) Màn hình Edit

Hình P.11: Màn hình soạn thảo

F1 (Disp Symb): Cho phép thay đổi hoặc đặt tên ký hiệu (symb), chú thích các toán hạng dùng trong khối chương trình đang soạn thảo.

F2 (Reference): Hiển thị tham chiếu chéo.

F3 (Serach): Tìm kiếm các toán hạng đơn lẻ trong khối đang soạn thảo.

F5 (Seg Fct): Hiện các chức năng soạn thảo cho phép làm việc với các đoạn của khối như chép, xoá, chèn,...

F6 (Edit): Chuyển sang chế độ soạn thảo.

F7 (Enter): Lưu trữ khối nếu có sự thay đổi hoặc trở về menu chính.

F8 (Cancel): Trở về menu chính.

Shift-F1 (Addresses): Hiện địa chỉ tương đối của các lệnh trong khối (với STL).

Shift-F2 (Lib no): Cho phép vào số thư viện.

Shift-F3 (Symb.OFF): Cho phép hiển thị toán hạng dưới dạng tuyết đối.

Shift-F4 (Symb Com): Cho phép hiện thị dòng chú thích ký hiệu các toán hạng.

Shift-F5 (→ LAD): Cho phép chuyển đổi các dạng STL, CSF, LAD.

Shift-F6 (Seg com): Cho phép vào soạn thảo tiêu đề và các chú thích của mỗi đoạn chương trình trong khối nếu có chọn *Wich Comments* ở trang *2 (Blocks)* phần phụ lục.

Shift-F7 (Save): Lưu trữ khối soạn thảo vào file.

Shift-F1 (Help): Vào phần trợ giúp.

6. Nếu đang ở màn hình Output cần sửa chữa hoặc soạn thảo mới thì chọn F6 (Edit) để vào màn hình soạn thảo Edit, với chương trình có nhiều đoạn (Segment) thì ấn F5 (Seg Fct) sau đó ấn F1 (-1) hoặc F2 (+1) để chọn các đoạn trước hoặc sau đoạn hiện thời.

7. Khi đang ở màn hình soạn thảo Edit có thể tiến hành soạn thảo:

+ Để vào một câu lệnh ta không cần quan tâm đến cấu trúc mà có thể gõ liên tục liền nhau, hết một dòng ấn Enter máy sẽ tự động chèn vào các ký tự trống ngăn cách.

+ Soạn thảo hết một đoạn (segment) ấn F6 (Seg End) để sang đoạn mới.

+ ***Kết thúc chương trình phải có lệnh BE***, ấn Enter và chọn yes để xác nhận máy sẽ trở về màn hình Output.

8. Ấn Shift-F5 để xem dạng LAD và CSF. Nếu chương trình có nhiều đoạn (Segment) thì ấn F5 (Seg Fct) sau đó ấn F1 (-1) hoặc F2 (+1) để xem lần lượt hết các đoạn trước hoặc sau đoạn hiện thời.

9. Ấn Shift-F7 để cất chương trình và đổ chương trình sang PLC, chọn yes để xác nhận việc đổ đè chương trình lên chương trình cũ trong PLC (khi cất thì PLC phải để ở chế độ STOP).

8.2.2. Đặt tham số cho việc soạn thảo chương trình.

Vào File \ Project \ Set ta sẽ đặt các tham số cần thiết liên quan đến việc soạn thảo chương trình. Các tham số này được hiển thị trong 6 trang màn hình, các trang màn hình có thể chuyển đổi bằng con trỏ. Mỗi trang có các phím chức năng có thể sử dụng như:

+ *Edit F2*: Vào chế độ soạn thảo.

+ *Select F3*: Thay đổi tham số tại vị trí con trỏ.

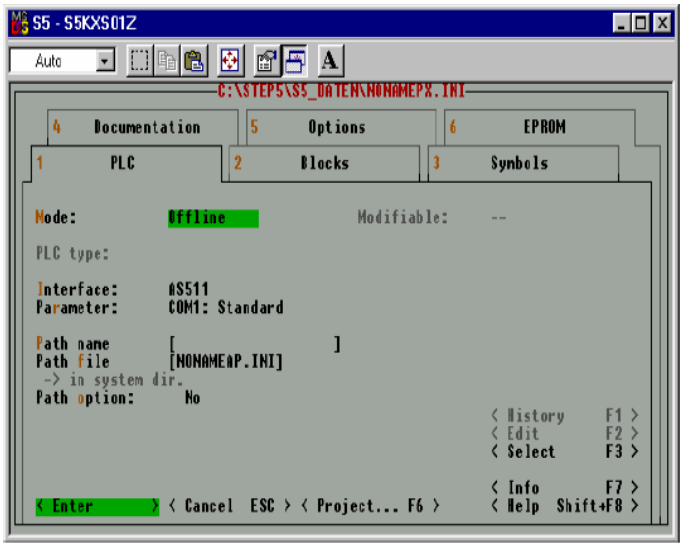
+ *Project... F6*: Cất tham số đã thay đổi.

+ *Info F7*: Hiện thông tin về vùng hiện tại mà tại đó có con trỏ.

+ *Help Shift F8*: Vào phần trợ giúp.

+ *Enter*: Chấp nhận sự thay đổi.

+ *Cancel ESC*: Giữ nguyên trạng thái cũ, trở về màn hình trước đó.



Hình 8.12: Trang 1 - PLC

*Trang 1 (PLC): như hình 8.12

+ *Mode*: Chọn chế độ nối với PLC (Online), và không có PLC (Offline).

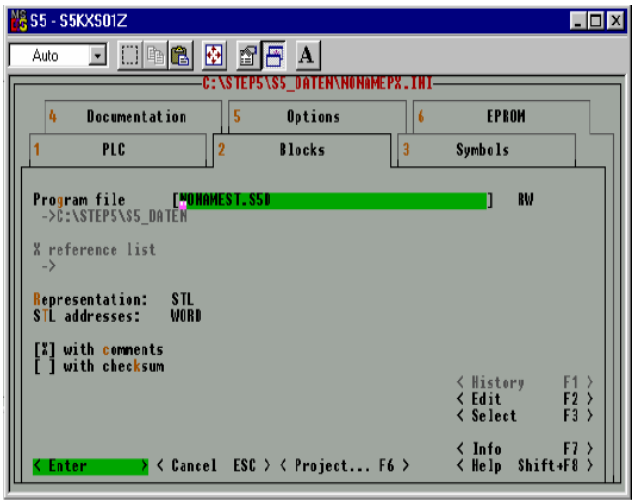
+ *PLC type*: Loại PLC

+ *Interface*: Chọn giao diện.

+ *Parameter*: Địa chỉ cổng giao diện.

+ *Path name*: Đặt tên đường dẫn nối kết. Nếu cả Path name và Path file đều đặt thì hệ thống tìm cách thiết lập hay dừng việc nối kết thông qua đường dẫn đã chọn này mỗi khi có sự thay đổi chế độ làm việc.

+ *Path file*: Tên file chứa đường dẫn Path name.



Hình 8.13: Trang 2 - Blocks

*Trang 2 (Blocks): như hình 8.13

+ *Program File*: Vào đường dẫn, mở file mới hoặc mở file đã có.

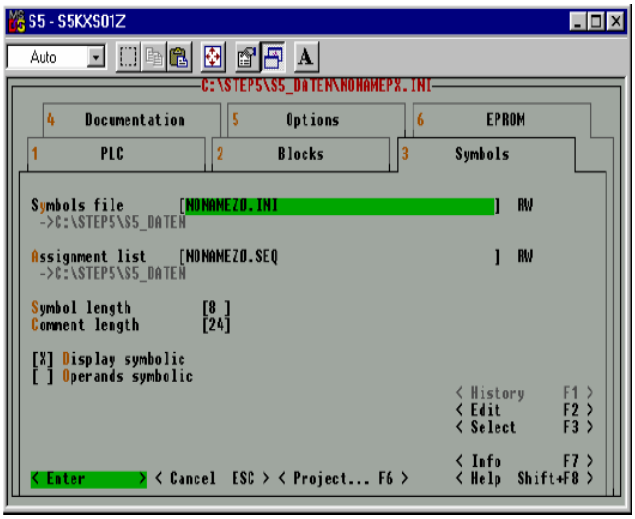
+ *Representation*: Đặt chế độ

soạn thảo STL, LAD, CSF.

+ *STL addresses*: Địa chỉ của STL.

+ *With comments*: Cho phép ẩn, hiện dòng chú thích.

+ *With Checksum*: Kiểm tra việc truyền số liệu ra PLC.



Hình P.14: Trang 3 - Symbols

*Trang 3 (Symbols): như hình 8.14

+ *Symbols file*: Đặt tên file biểu tượng (*ZO.INI).

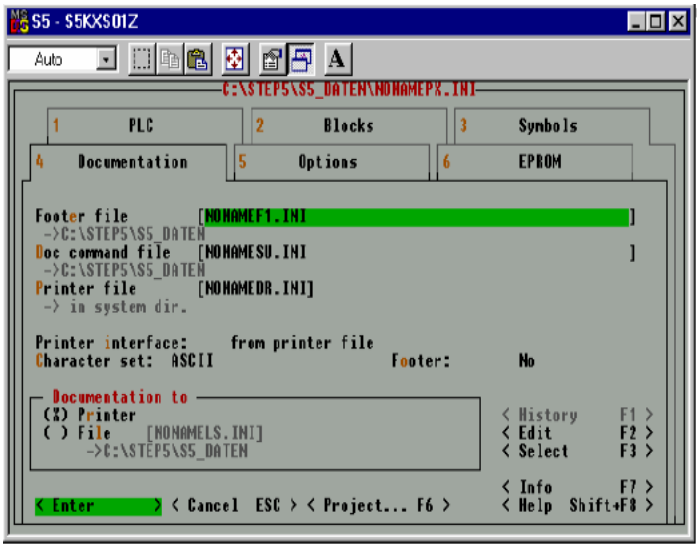
+ *Assigment list*: Đặt tên của file danh sách (ZO.SEQ).

+ *Symbol length*: Đặt độ dài ký hiệu biểu tượng, cho phép 8 đến 24 ký tự.

+ *Comment length*: Đặt độ dài dòng chú thích, cho phép nhiều nhất là 40 ký tự.

+ *Display symbolic*: Cho phép toán hạng thể hiện dưới dạng biểu tượng (symbolic) hay dạng tuyệt đối (absolute).

+ *Operands symbolic*: Cho phép lập trình được với symbolic operands.



Hình 8.15: Trang 4 - Documentation

*Trang 4 (Documetation): như hình 8.15.

+ *Footer file*: Vào tên file chứa các thông tin cần thiết ở cuối mỗi trang khi in và được tạo ra

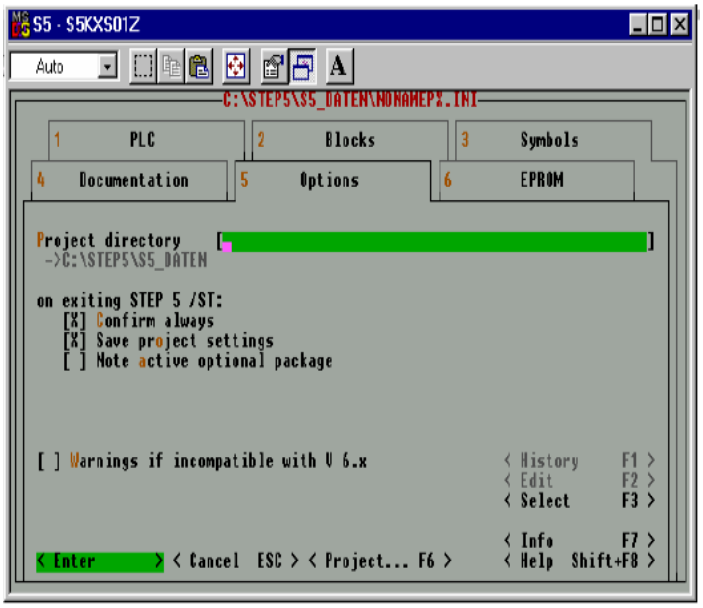
trong Documentation.

+ *Doc comm file*: Đặt tên file (*SU.INI) chứa các lệnh tạo tài liệu.

+ *Printer file*: Đặt tên file chưa thông tin về tham số in được chọn trong menu Documentation như kích cỡ giấy, số dòng trong mỗi trang in, cổng giao tiếp với máy in...

+ *Printer interface*: Chọn giao diện với máy in.

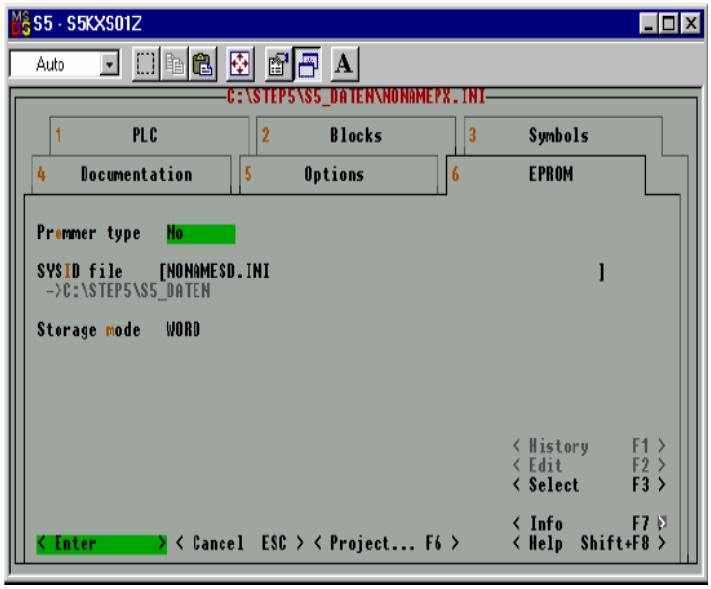
+ *Documetation to*: Đặt chế độ làm việc cho phép in tài liệu.



Hình 816: Trang 5 - Option

*Trang 5 (Options): hình 8.16

+ *Project directory*: Định thư mục làm việc.



Hình 8.17: Trang 6 - EFROM

*Trang 6 (EFROM): như hình 8.17

+ *SYSID file*: Đặt tên file (*SD.INI) chứa các thông tin nhận dạng hệ thống các khối dùng trong việc nạp EFROM.

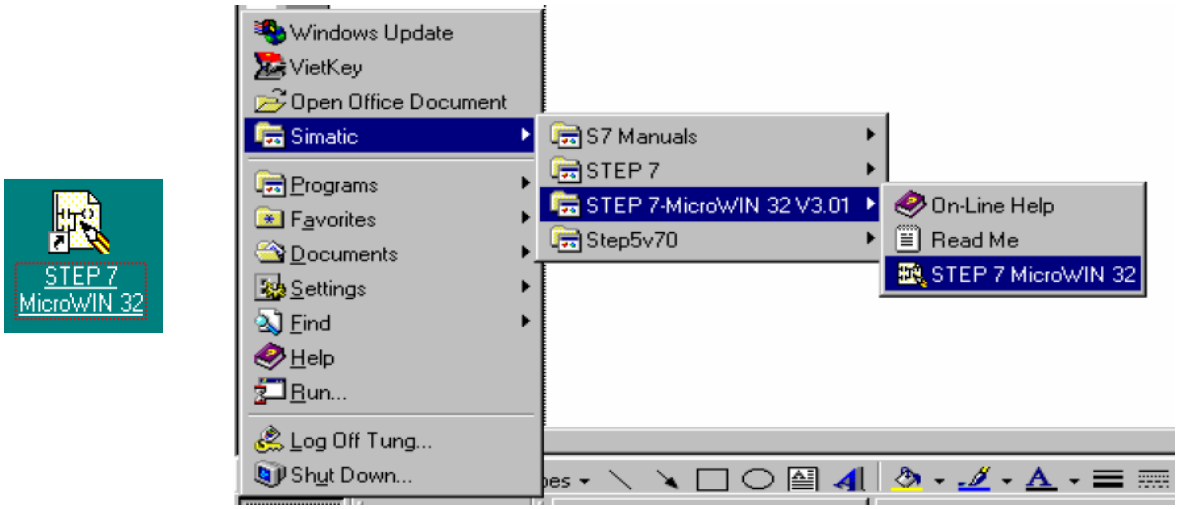
8.3. Lập trình cho PLC - S7-200

8.3.1. Sử dụng phần mềm Step7-200 for Win.

Thao tác chuẩn bị

1. Khởi động máy tính ở chế độ Windows,(bật công tắc nguồn khối thí nghiệm, PLC lắp thành khối thí nghiệm), bật công tắc khối nguồn PS của PLC, công tắc của khối CPU để ở vị trí STOP.

2. Chạy trình Step7 từ biểu tượng hoặc từ fite chương trình như hình 8.18. màn hình chế độ bắt đầu có dạng như hình 8.19.



Hình 8.18: Biểu tượng và đường dẫn file chương trình Step7

3. Nếu ở Project [CPU .....] có loại CPU khác thì nháy nút phải chuột vào Project [CPU .....] để chọn lại CPU.

4. Vào Fite để mở một fite mới hoặc fite đã có.

5. Vào View để chọn chế độ soạn thảo STL (hoặc LAD hoặc FBD).

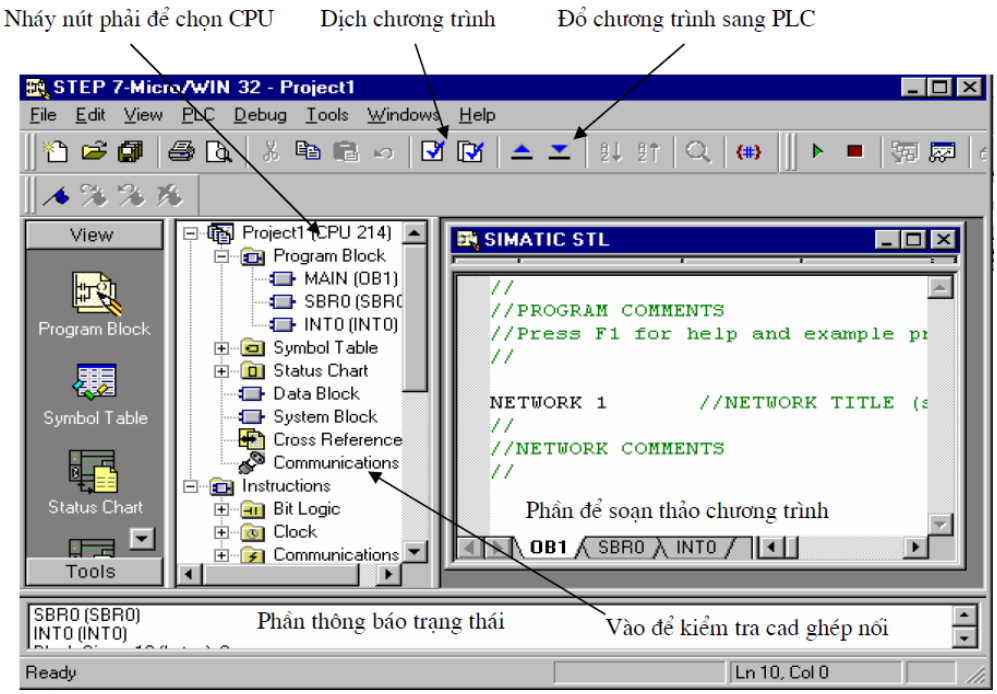
6. Tiến hành soạn thảo chương trình theo STL (nếu soạn thảo chương trình theo LAD thì có thể sử dụng các khâu, khối phía trái màn hình soạn thảo). Khi soạn thảo chỉ cần cách lệnh và đối tượng lệnh một nhịp (dấu cách), không cần chú ý chữ in và chữ thường, máy sẽ tự dịch và chỉnh chữ cho phù hợp. Trong quá trình soạn thảo có thể ghi các chú thích nếu cần.

7. Vào View để xem lại dạng LAD (Ladder) hoặc FBD.

8. Dịch chương trình từ biểu tượng hoặc từ PLC \ compile, nếu muốn dịch cả chương trình thì từ PLC \ Compile All. Khi dịch chương trình các lỗi sẽ được thông báo ở phần thông báo trạng thái.

9. Đổ chương trình sang PLC từ biểu tượng hoặc từ File \ Download, có thể phải kiểm tra lại cad ghép nối cho phù hợp từ Communications.

10. Muốn cất, in chương trình..., có thể thực hiện từ biểu tượng hoặc vào File chọn chế độ cất và chế độ in cần thiết.



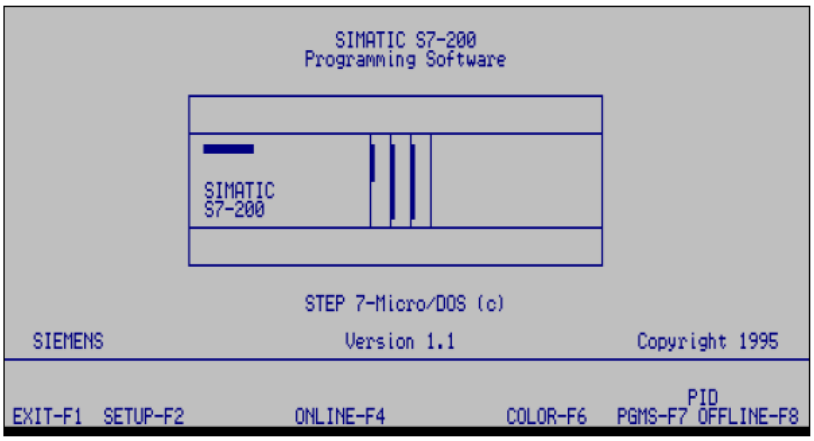
Hình 8.19: Màn hình soạn thảo

8.3.2. Sử dụng phần mềm Step7-200 for Dos.

Thao tác chuẩn bị:

1. Khởi động máy tính ở chế độ Windows.

2. Chạy trình S7-200 từ biểu tượng hoặc từ fite chương trình, màn hình chế độ bắt đầu có dạng như hình 8.20.



Hình 8.20: Màn hình bắt đầu của STEP7-Micro/Dos

Trong đó:

EXIT-F1: Thoát.

SETUP-F2: Chọn ngôn ngữ, đặt cú pháp cho biến nhớ. Chú ý ngôn ngữ giao diện để ở chế độ *International.*

ONLENE-F4: Khi máy tính có nối với PLC.

COLOR-F6: Chọn mầu.

PGMS-F7: Chương trình quản lý fite.

OFLINE-F8: Khi máy tính không nối với PLC.

Chữ PID chỉ tên fite đang sử dụng.

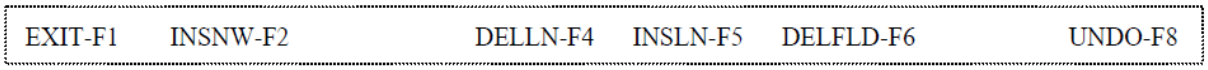
3. Chọn PGMS, ấn phím F7 *(các phần tiếp sau thao tác chọn và ấn phím được viết gọn thành PGMS-F7),* vào chương trình quản lý fite để mở fite mới hoặc fite đã có. Để mở fite mới chọn DIR-F5 vào ổ đĩa, chọn SELECT-F8 để xác nhận, ấn Enter để hiện các thư mục, chọn thư mục sau đó chọn SELECT-F8 để xác nhận, chọn EXIT-F1 thoát về màn hình trước đó, đặt tên fite và chọn SELECT-F8 để xác nhận, chọn ABORT-F1 để về màn hình ban đầu, tên fite và đường dẫn đã được thiết lập.

4. Chọn chế độ ONLINE-F4, rồi xác nhận địa chỉ cổng ghép nối với PLC.

5. ấn F7 để chọn chế độ soạn thảo LAD hoặc STL.

6. Chọn EDIT-F2 để vào chế độ soạn thảo, phía dưới màn hình soạn thảo có dòng thư mục hướng dẫn các cách và các lệnh để soạn thảo.

7a. Soạn thảo với STL dòng hướng dẫn có dạng như hình 8.21:



Hình 8.21: Dòng hướng dẫn soạn thảo STL

Trong đó:

EXIT-F1: thoát về trang trước đó.

INSNW-F2: Chèn một network phía trên con trỏ.

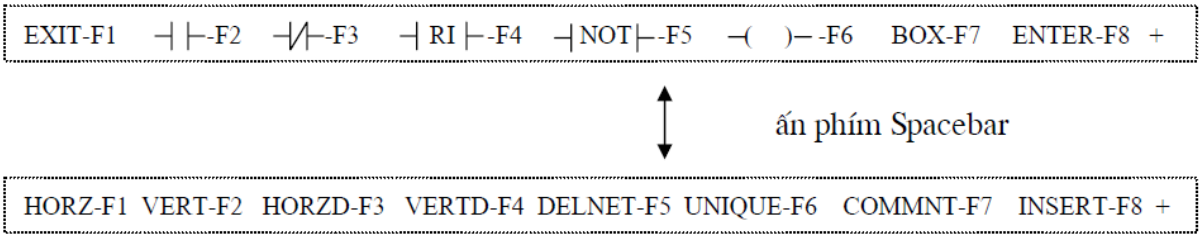
DELLN-F4: Xóa một dòng có con trỏ.

INSLN-F5: Chèn một dòng phía trên con trỏ.

DELFLD-F6: Xóa tham số nơi con trỏ.

Sử dụng các phím và phím ENTER để di chuyển con trỏ đến vị trí soạn thảo.

7b. Soạn thảo với LAD dòng hướng dẫn có dạng như hình 8.22: dấu cộng ở cuối dòng thể hiện thư mục vẫn còn cần ấn phím Spacebar để chuyển đổi.



Hình 8.22: Dòng hướng dẫn soạn thảo LAD

Trong đó:

EXIT-F1: Thoát về trang màn hình trước đó.

Các phím F2 đến F7 (dòng trên) để chọn các tiếp điểm, cuộn dây, hộp.

ENTER-F8: Xác định một network đã được soạn thảo.

HORZ-F1: để kẻ một đoạn ngang từ vị trí con trỏ sang phải.

VERT-F2: để kẻ một đoạn dọc từ vị trí con trỏ xuống dưới.

HORZD-F3: để xóa một đoạn ngang.

VERTD-F4: để xóa một đoạn dọc.

Sử dụng các phím ◄ ▲ ► ▼ để di chuyển con trỏ đến vị trí soạn thảo.

Khi soạn xong một tiếp điểm, hộp... dùng phím ENTER để xác nhận.

Khi soạn xong một network phải dùng F8 để xác nhận, nếu dùng ENTER có nghĩa muốn xuống dòng để mở rộng (nhánh) cho network.

8. Chọn EXIT-F1 để trở về màn hình trước đó.

9. Chọn STL-F7 để xem dạng STL.

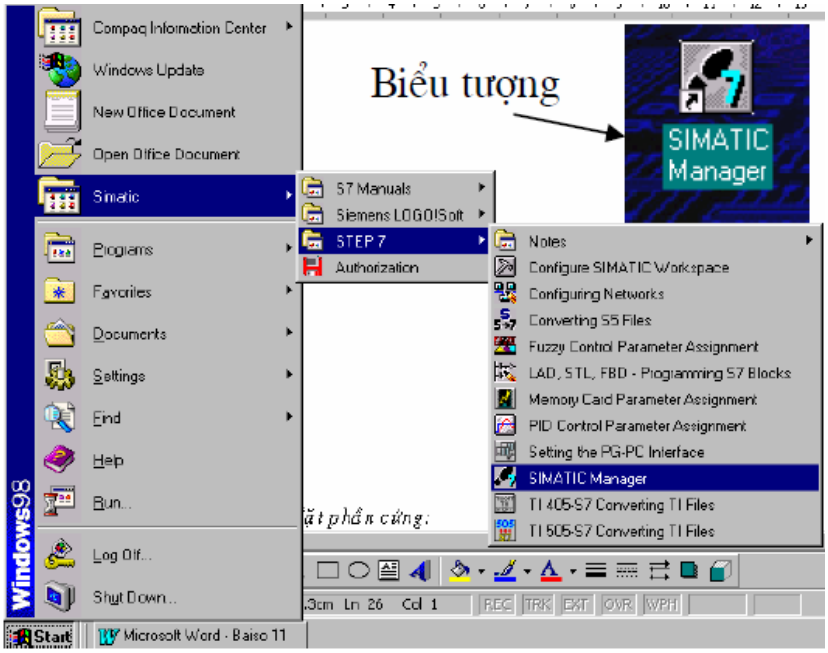
10. Chọn WRITDK-F8 để đổ chương trình sang PLC.

11. Muốn in chương trình, hoặc thực hiện các thao tác lựa chọn khác thì làm theo chỉ dẫn ở dòng thư mục cuối màn hình hoặc vào phần Help.

8.4. Lập trình cho PLC - S7-300 (Sử dụng phần mềm S7-300)

8.4.1. Khởi động:

1. Khởi động máy tính ở chế độ Windows, (bật công tắc nguồn của khối thí nghiệm) bật công tắc nguồn của khối nguồn PS của PLC, công tắc của khối CPU để ở vị trí STOP.



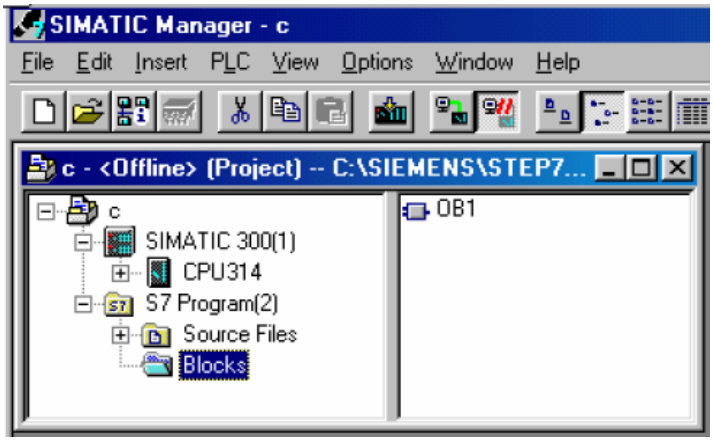
Hình 8.23: Đường dẫn khởi động STEP 7

2. Khởi động phần mềm Step7 từ biểu tượng hoặc từ file chương trình như hình 8.23.

8.4.2. Cài đặt phần cứng:

1. Công tắc của CPU phải để ở chế độ STOP.

2. Vào File để tạo một thư mục chương trình mới (hoặc mở một thư mục chương trình đã có) *(vì một chương trình của S7- 300 là cả một thư mục "Project").* Một chương trình của S7-300 sẽ có dạng như hình 8.24 (khi đã tạo đủ). Nếu mở một thư mục chương trình đã có sẵn chương trình thì có thể bỏ qua một số bước sau.

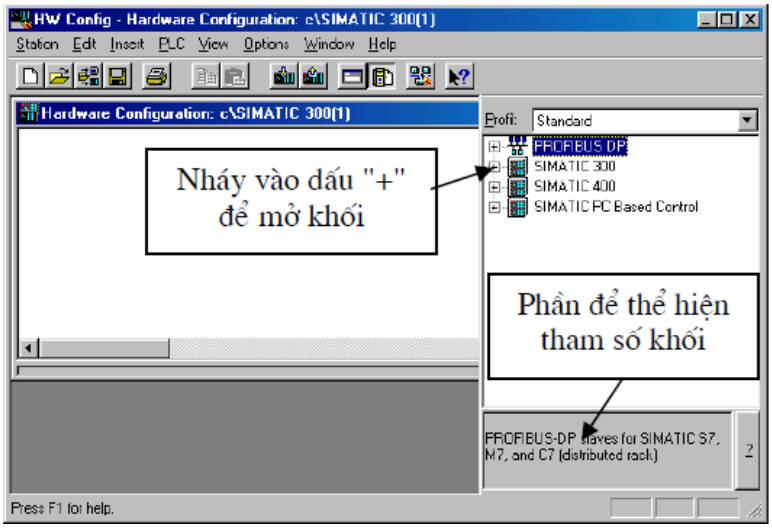


Hình 8.24: Cấu trúc chương trình STEP 7

3. Mở thư mục chương trình *"Project"* để chèn phần cứng từ insert / Station / Simatic 300 Station.

4. Mở thư mục Simatic 300(1) để cài đặt phần cứng.

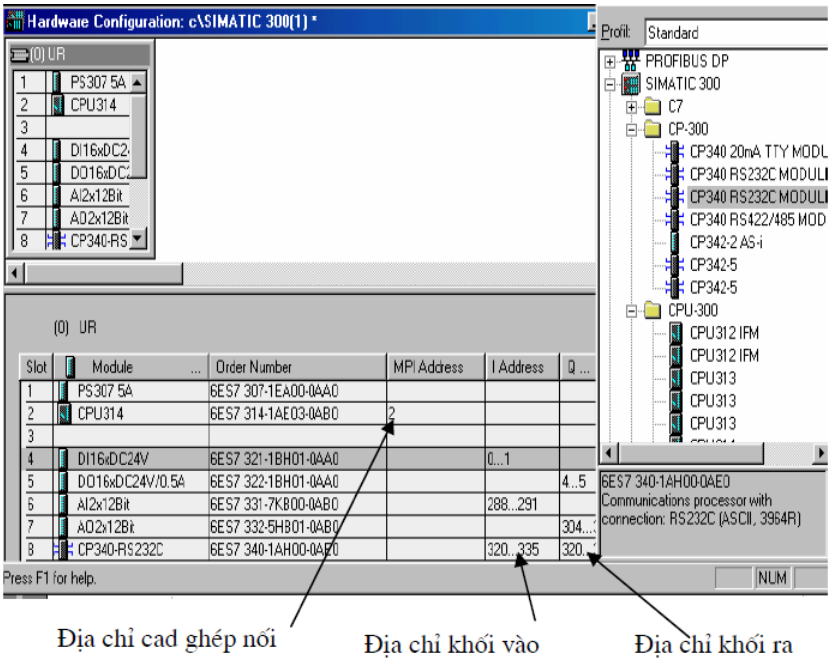
5. Mở thư mục Hardware để bắt đầu cài đặt phần cứng, màn hình ban đầu để cài đặt phần cứng có dạng như hình 8.25.



Hình 8.25: Hướng dẫn cài đặt phần cứng

6. Nháy vào dấu "+" của SIMATIC 300 để chọn lần lượt các khối của cấu hình cứng. (chọn theo các khối hiện có của khối thí nghiệm).

Các khối thực trên PLC như trên hình 8.26.



Hình 8.26: Các khối đã được chọn

Phải nháy vào dấu "+" để mở chương trình.

+ Chọn giá đỡ: Chọn RACK-300 và chọn Rail.

+ Chọn khối nguồn: Chọn PS-300 (và chọn PS307 5A).

+ Chọn khối CPU: Chọn CPU-300 và chọn CPU 314, chọn loại có tham số (được chỉ ra ở phần thể hiện tham số hình 8.26) như tham số của CPU hiện có (được chỉ ra ở dòng trên cùng và dòng dưới cùng của CPU trên khối thí nghiệm). Riêng trong bài thí nghiệm này phần mềm không có loại mã hiệu 6ES7314-1AE04-0AB0 nên chọn loại 6ES7 314-1AE03-0AB0 thay thế.

+ *Bỏ qua khối bị thiếu*: IM (Interfare) nằm trên dòng số 3 của Rail.

+ *Chọn các khối vào ra*: Chọn SM-300 và lần lượt chọn các khối vào ra theo đúng mã hiệu được ghi trên dòng đầu và dòng cuối mỗi khối.

+ *Chọn khối ghép nối*: CP-300 và chọn CP340 RS 232C. Khối ghép nối này để ghép nối với các thiết bị ngoài. Màn hình sau khi chọn khối có dạng như hình 8.26

7. Đổ cấu hình sang PLC từ PLC \ Download hoặc biểu tượng, nhấn OK để xác nhận địa chỉ giá đỡ (Rack), địa chỉ CPU và địa chỉ cổng ghép nối.

8.4.3. Soạn thảo chương trình:

1. Trở về thư mục chương trình chính "Project", xác nhận việc cất cấu hình cứng vài file.

2. Mở thư mục chương trình chính "Project" để chèn chương trình soạn thảo vào từ insert / Program / S7 Program.

3. Mở thư mục S7 Program, trong đó sẽ có các thư mục: Source File, Symbols, Blocks.

4. Mở thư mục Blocks, nếu cần thì chèn thêm các khối (Blocks) cần thiết khác cho chương trình từ insert / S7 Blocks.

5. Mở khối OB1 (bài này chỉ lập trình trên khối OB1), chọn kiểu lập trình STL từ Language (có thể chọn kiểu lập trình khác) rồi chọn OK. Màn hình lập trình có dạng như hình 8.27.

6. Có thể chọn chế độ online để kết nối trực tiếp với PLC hoặc offline không nối trực tiếp với PLC, chọn chế độ offline khi soạn xong chương trình phải đổ sang PLC.

7. Có thể đặt tên cho khối, tên cho đoạn (Networks) và các chú thích.

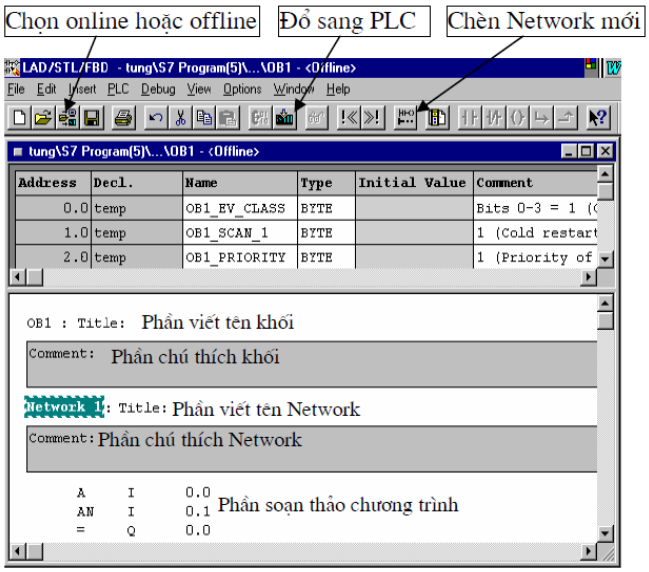
8. Tiến hành soạn thảo, khi soạn thảo chỉ cần cách mã lệnh và đối tượng lệnh một nhịp máy sẽ tự động dịch khoảng cách cho phù hợp.

9. Soạn thảo hết một Networks thì chèn thêm Networks mới từ biểu tượng hoặc insert / Network.

10. Xem lại dạng LAD hoặc FBD từ View / LAD hoặc View / FBD.

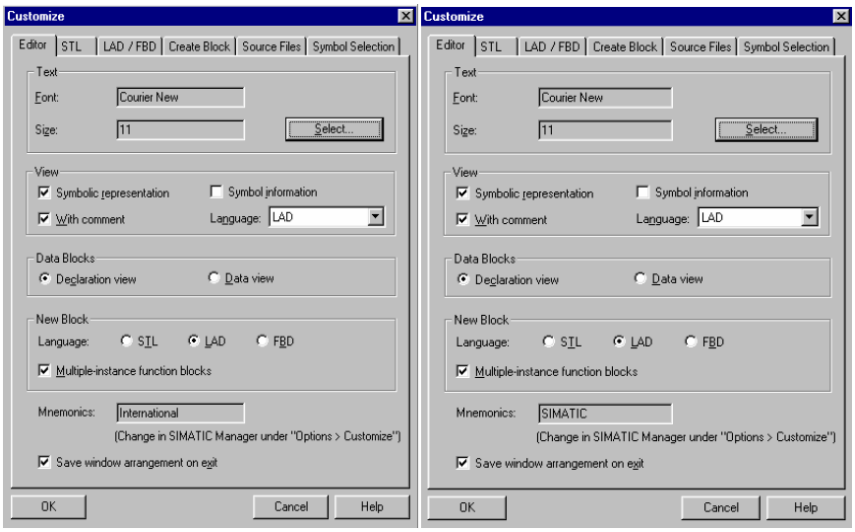
11. Soạn thảo xong đổ chương trình sang PLC từ biểu tượng hoặc từ PLC / Download để kiểm tra, khi đổ chương trình PLC phải để ở trạng thái STOP.

Chú ý: Khi lập trình có thể các ký hiệu không đúng (không lập trình được, chẳng hạn gõ địa chỉ I 0.0 báo lỗi, gõ M 0.0 thì nhận) là do chọn ngôn ngữ không đúng. Để kiểm tra ngôn ngữ làm như sau:



Hình 8.27: Màn hình soạn thảo

+ Từ màn hình soạn thảo như hình 8.27 chọn **Options/Customize**... ta được cửa sổ như hình 8.28.



a) b)

Hình P.28

+ Trong cửa số Editor hình 8.28, hộp kiểm Mnemonics phải là Internectiona như hình 8.28a. nếu trong hộp kiểm Mnemonics là SMATIC như hình 8.28b là sai ngôn ngữ (dùng tiếng Đức). Muốn đổi ngôn ngữ để có thể lập trình được ta phải quay lại màn hình ban đầu như hình 8.24 và tiến hành các bước:

+ Từ màn hình 8.24 chọn Options/Customize... ta được cửa sổ của màn hình Customize như hình 8.29. Trong màn hình Customize ở cửa sổ Language tại hộp kiểm Language phải chọn english, tại hộp kiểm Mnemonics phải chọn English như hình 8.29 sau đó nhấn OK.

PHỤ LỤC

BẢNG LỆNH CỦA CÁC PHẦN MỀM PLC

1. Bảng lệnh của PLC - CPM1A

TT	Tên lệnh	Mô tả
1	AND	Nhận logic trạng thái của bit xác định với điều kiện thực hiện
2	AND LD	Nhân logic các kết quả của các khối xác định
3	AND NOT	Nhân logic giá trị đảo của bit xác định với điều kiện thực hiện
4	CNT	Đếm lùi
5	LD	Khởi động một dãy lệnh với trạng thái của bit xác định hoặc để định nghĩa một khối logic được dùng với ANDLD hoặc ORLD
6	LD NOT	Khởi động một dãy lệnh với nghịch đảo của bit xác định
7	OR	Cộng logic trạng thái của bit xác định với điều kiện thực hiện
8	OR LD	Cộng kết quả của các khối định trước
9	OR NOT	Cộng logic nghịch đảo bit xác định với điều kiện thực hiện
10	OUT	Đưa ra cổng ra giá trị của bit thực hiện
11	OUT NOT	Đưa ra cổng ra giá trị nghịch đảo của bit thực hiện
12	TIM	Quá trình thời gian trễ ON
13	NOP	Không thực hiện gì cả, quá trình chuyển sang lệnh bên cạnh
14	END	Lệnh kết thúc chương trình
15	IL	Nếu điều kiện khoá chéo là OFF tất cả các đầu ra là OFF và toàn bộ thời gian (time) sẽ phục hồi giữa IL này (02) và IL khác (03). Các lệnh khác được điều hành như là lệnh NOP (00), bộ đếm vẫn duy trì
16	ILC	
17	JMP	Nếu điều kiện nhảy bị tắt (OFF) tất cả các lệnh giữa JMP (04) và JME (05) tương ứng bị bỏ qua
18	JME	
19	FAL	Phát một lỗi không tiền định và cho ra số FAL vào bộ lập trình cầm tay
20	FALS	Phát một lỗi tiền định và cho ra số FALS vào bộ lập trình cầm tay
21	STEP	Khi dùng với bit điều khiển sẽ xác định điểm bắt đầu một bước mới và phục hồi (R) bước trước đó. Khi không dùng với bit điều khiển sẽ xác định điểm cuối của việc thực hiện bước.
22	SNXT	Dùng với một bit điều khiển để chỉ ra kết thúc bước, phục hồi bước và bắt đầu bước tiếp theo
23	SET	Tạo ra bộ ghi dịch bit
24	KEEP	Xác định một bit như là một chốt điều khiển bởi các đầu vào đất và phục hồi
25	CNTR	Tăng hoặc giảm số đếm bởi một trong số các tín hiệu vào tăng hoặc giảm chuyển từ OFF sang ON

TT	Tên lệnh	Mô tả
26	DIFU	Bật (On) bit xác định cho một chu kỳ tại sườn trước của xung vào
27	DIFD	Nhân logic trạng thái của bit xác định với điều kiện thực hiện
28	TIMH	Bộ thời gian tốc độ cao có trễ
29	WSFT	Dịch chuyển dữ liệu giữa các từ đầu và cuối trong nhóm từ, viết 0 vào từ đầu
30	CMP	So sánh nội dung của 2 từ và đưa ra kết quả vào các cờ GR, EQ, LE
31	MOV	Chép dữ liệu nguồn (từ hoặc hằng số) vào từ đích
32	MVN	Đảo dữ liệu nguồn (từ hoặc hằng số) sau đó chép nó vào từ đích
33	BIN	Chuyển dữ liệu 4 số dạng BCD trong từ nguồn thành dữ liệu nhị phân 16 bit và đưa dữ liệu đã được chuyển vào từ kết quả
34	BCD	Chuyển dữ liệu nhị phân trong từ nguồn thành BCD sau đó đưa dữ liệu đã chuyển mã ra từ kết quả
35	ASL	Dịch từng bít trong từ đơn của dữ liệu về bên trái có CY
36	ASR	Dịch từng bít trong từ đơn của dữ liệu về bên phải có CY
37	ROL	Quay các bít trong từ đơn của dữ liệu một bít về bên trái có CY
38	ROR	Quay các bít trong từ đơn của dữ liệu một bít về bên phải có CY
39	COM	Đảo trạng thái bít của một từ dữ liệu
40	ADD	Cộng 2 giá trị BCD 4 số với nội dung của CY và đưa kết quả đến từ ghi kết quả đặc biệt
41	SUB	Trừ một giá trị BCD 4 số và CY từ một giá trị BCD 4 bit khác và đưa kết quả
42	MUL	Nhân 2 giá trị BCD 4 số và đưa kết quả tới từ kết quả đặc biệt
43	DIV	Chia số BCD 4 số cho số bị chia BCD 4 số và đưa kết quả tới từ kết quả đặc biệt
44	ANDW	Nhân logic 2 từ vào 16 bit và đặc bit tương ứng vào từ kết quả nếu các bit tương ứng trong các từ vào đều ON
45	ORW	Cộng logic 2 từ vào 16 bit và đặt bit tương ứng vào từ kết quả nếu các bit tương ứng trong dữ liệu vào là ON
46	XORW	Cộng (EXNOR) 2 từ 16 bit và đặt bit vào từ kết quả khi các bit tương ứng trong các từ vào có trạng thái khác nhau
47	XNRW	Cộng đảo (EXNOR) 2 từ 16 bit và đặt bit vào từ kết quả khi các bit tương ứng trong các từ vào có cùng trạng thái
48	INC	Tăng từ BCD 4 số lên 1 đơn vị
49	DEC	Giảm từ BCD 4 số đi 1 đơn vị
50	STC	Đặt cờ mang sang (bật ON, CY)
51	CLC	Xoá cờ mang sang (tắt OF, CY)

52	TRSM	Khởi đầu viết dữ liệu không dùng với CQM1-CPU 11/21-E
TT	**Tên lệnh**	**Mô tả**
53	MSG	Hiển thị thông báo 16 vị trí tên bộ lập trình
54	ADB	Cộng 2 giá trị Hexa 4 số với nội dung của CY và gửi kết quả tới từ kết quả xác định
55	SBB	Trừ giá trị Hexa 4 số cho một giá trị Hexa 4 số, CY và gửi kết quả tới từ kết quả
56	MLB	Nhân 2 số trị Hexa 4 số và gửi kết quả tới từ kết quả xác định
57	DVB	Chia số trị Hexa 4 số cho số Hexa 4 số và gửi kết quả tới từ kết quả xác định
58	ADDL	Cộng 2 giá trị 8 số (2 trừ một) và nội dung của CY và gửi kết quả tới các từ kết quả xác định
59	SUBL	Trừ giá trị BCD 8 số cho một giá trị BCD 8 số và CY và gửi kết quả vào từ kết quả
60	MULL	Nhân 2 giá trị BCD 8 số và gửi kết quả vào các từ kết quả xác định
61	DIVL	Chia số BCD 8 số cho số BCD 8 số và gửi kết quả đến các từ kết quả xác định
62	BINL	Chuyển giá trị BCD thành các từ nhị phân nguồn liên kết và đưa dữ liệu chuyển đổi đến 2 từ kết quả liên tiếp
63	BCDL	Chuyển giá trị nhị phân thành hai từ BCD nguồn liên tiếp và đưa dữ liệu đã chuyển đổi đến 2 từ kết quả liên tiếp
64	XFER	Chuyển 1 số nội dung từ nguồn liên tiếp thành từ đích liên tiếp
65	BSET	Sao chép nội dung 1 từ hoặc 1 hằng số thành một số từ liên tiếp
66	ROOT	Bình phương (khai căn) của giá trị BCD 8 số và đưa ra kết quả số nguyên 4 chữ số đã cắt ngắt và gửi kết quả ra 1 từ định trước
67	XCHG	Trao đổi nội dung của hai từ khác nhau
68	@COLM	Chép 16 bit của một từ xác định vào một cột bit của các từ 16 bit liên tiếp
69	CPS	So sánh hai giá trị nhị phân 16 bit (4 số) đã đánh dấu và đưa kết quả đến các cờ GR, EQ, LE
70	CPSL	So sánh hai giá trị nhị phân 32 bit (8 số) đã đánh dấu và đưa kết quả đến các cờ GR, EQ, LE
71	@DBS	Chia 1 giá trị nhị phân 16 bit đã đánh dấu cho một giá trị khác và đưa kết quả nhị phân 32 bit đã đánh dấu vào từ R đến R+1
72	@DBSL	Chia 1 giá trị nhị phân 32 bit đã đánh dấu cho một giá trị khác và đưa kết quả nhị phân 64 bit đã đánh dấu vào từ R+3 đến R
73	@FCS	Kiểm tra lỗi trong dữ liệu truyền bởi lệnh Host link
74	@FPD	Tìm lỗi trong cụm các lệnh
75	@HEX	Chuyển đổi dữ liệu ASCII thành dữ liệu hexa

76	@HKY	Vào dữ liệu hexa đến 8 số từ bàn 16 phím
77	@HMS	Chuyển đổi dữ liệu giây (s) thành dữ liệu giờ (h) và phút (mm)
TT	**Tên lệnh**	**Mô tả**
78	@LINE	Chép một bit của cụm 16 từ liên tiếp vào từ xác định
79	@MAX	Tìm giá trị cực đại trong không gian dữ liệu xác định và đưa giá trị này tới từ khác
80	@MBS	Nhân nội dung nhị phân đánh dấu của hai từ và đưa kết quả nhị phân 8 bit đã đánh dấu vào R+1 và R
81	@MBSL	Nhân hai giá trị nhị phân 32 bit (8 số) đã đánh dấu và đưa kết quả nhị phân 16 bit đã đánh dấu vào R+3 đến R
82	@MIN	Tìm giá trị cực tiểu trong không gian dữ liệu xác định và đưa giá trị này vào từ khác
83	@NEG	Chuyển đổi nội dung hexa 4 chữ số của từ nguồn thành phần bù modul 2 của nó và đưa kết quả vào R
84	@NEGL	Chuyển đổi nội dung hexa 8 chữ số của từ nguồn thành phần bù modul 2 của nó và đưa kết quả vào R và R+1
85	PID	(Chỉ có CQM1-CPV43E) thể hiện điều khiển PID dựa trên các thông số xác định
86	@PLS2	(Chỉ có CQM1-CPV43E) Tăng tốc độ xung ra từ 0 tới tần số đích
87	@PWM	(Chỉ có CQM1-CPV43E) Đưa ra cổng một và hai các xung có tỷ số luân phiên xác định (0%-99%)
88	@RXD	Nhập dữ liệu thông qua cổng liên lạc
89	@SCL2	(Chỉ có CQM1-CPV43E) Chuyển đổi tuyến tính một giá trị hexa 4 số đã đánh dấu thành giá trị số BCD 4 chữ số
90	@SCL3	(Chỉ có CQM1-CPV43E) Chuyển đổi tuyến tính một giá trị BCD 4 chữ số thành giá trị hexa 4 chữ số đã đánh dấu
91	@SEC	Chuyển đổi dữ liệu giờ (h) và phút (mm) thành dữ liệu giây (s)
92	@SBBL	Trừ đi một giá trị nhị phân 8 chữ số (bình thường hoặc đánh dấu) trừ giá trị khác và đưa kết quả ra R và R+1
93	@SRCH	Kiểm tra phạm vi xác định của bộ nhớ dùng cho dữ liệu xác định. Đưa các địa chỉ từ các từ trong phạm vi chứa dữ liệu
94	@SUM	Tính tổng nội dung các từ trong phạm vi xác định của bộ nhớ
95	@XFRB	Chép trạng thái của nhiều nhất là 255 bit nguồn xác định vào các bít đích xác định
96	@ZCP	So sánh một từ với một dải xác định bởi giới hạn thấp và cao và đưa kết quả đến các cờ GR, EQ, LE
97	ZCPL	So sánh một giá trị 8 chữ số với một dải xác định bởi các giới hạn thấp và cao sau đó đưa kết quả đến các cờ GR, EQ, LE
98	SLD	Dịch trái dữ liệu giữa các từ đầu và cuối một chữ số (4 bit) về bên trái

99	SRD	Dịch phải dữ liệu giữa các từ đầu và cuối một chữ số (4 bit) về bên phải
100	MLPX	Chuyển đổi 4 chữ số hexa trong từ nguồn thành giá trị thập phân từ 0 đến 15 và ghi vào các từ hoặc các bit kết quả có vị trí tương ứng
TT	**Tên lệnh**	**Mô tả**
101	DMPX	Xác định vị trí ON cao nhất trong từ nguồn và chuyển các bit tương ứng vào từ kết quả
102	SDEC	Chuyển giá trị hexa từ nguồn đến dữ liệu cho hiện thị 7 thanh
103	DIST	Chuyển một từ của dữ liệu nguồn đến từ cuối mà địa chỉ của nó được cho bởi từ cuối cộng với OFF SET
104	COLI	Lỗi dữ liệu từ nguồn và viết nó vào từ cuối
105	MOVB	Truyền bit xác định của từ hoặc bằng số nguồn đến bit xác định của từ cuối
106	MOVD	Chuyển nội dung hexa của các chữ số nguồn 4 bit xác định đến các chữ số cuối xác định, tối đa là 4 chữ số
107	SFTR	Dịch dữ liệu trong từng nguồn hoặc chữ cuối các từ nguồn xác định về bên trái hoặc bên phải
108	TCMP	So sánh giá trị hexa 4 chữ số với giá trị trong bảng gồm 16 từ
109	ASC	Chuyển đổi các giá trị hexa từ nguồn thành mã ASII 8 bit bắt đầu tại nửa tận cùng bên trái hoặc phải của từ đầu xác định
110	SBS	Gọi và thực hiện chương trình con
111	SBN	Đánh dấu bắt đầu của chương trình con
112	RET	Kết thúc của chương trình con và trở về chương trình chính
113	IORF	Làm tươi tất cả đầu vào và ra giữa từ đầu và từ cuối
114	MACRO	Gọi và thực hiện chương trình con để thay thế các từ vào ra
115	@ASFT	Tạo một bộ ghi dịch để trao đổi nội dung của các từ liên kết khi một trong các từ là 0
116	@MCMP	So sánh một cụm 16 từ liên tiếp với một cụm 16 từ liên tiếp khác
117	@RXD	Đảo dữ liệu thông qua một cổng liên lạc (cổng COM)
118	@TXD	Gửi dữ liệu thông qua một cổng liên lạc
119	CMPL	So sánh 2 đại lượng hexa 8 chữ số
120	@INI	Khởi động và dừng quá trình đếm, so sánh và chuyển PV của bộ đếm, dừng đầu ra xung
121	@PRV	Đọc PV của bộ đếm và dữ liệu trạng thái cho bộ đếm có tốc độ cao nhất
122	@CTBL	So sánh PV của bộ đếm và phát một bản trực tiếp hoặc là khởi động quá trình chạy
123	@SPED	Đưa ra các xung với tần số xác định (10Hz - 50kHz trong các bộ 10Hz) tần số ra có thể thay đổi trong khi các xung đang được đưa ra

124	@PULS	Đưa ra một số xác định các xung có tần số xác định, đầu ra xung không dừng cho đến khi số lượng xung đã được đưa ra hết
125	@SCL	Thể hiện sự đổi thang đo cho giá trị tính toán
TT	**Tên lệnh**	**Mô tả**
126	@BCNT	Đếm tổng số các bit đang chạy (ON) trong cụm từ xác định
127	@BCMP	Quyết định xem giá trị của một từ có nằm trong phạm vi xác định bởi giới hạn dưới và trên
128	@STIM	Điều khiển Time khoảng dùng cho các ngắt thủ tục
129	DSW	Đưa vào dữ liệu BCD 4 hoặc 8 chữ số từ một chuyển mạch số
130	7SEG	Chuyển dữ liệu BCD 4 hoặc 8 chữ số thành dạng hiển thị 7 thanh
131	@INT	Thể hiện điều khiển và ngắt như là mặt nạ hoặc không mặt nạ các bit ngắt cho các ngắt vào ra
132	@ACC	Cho (CQM1-CPV43-E) cùng với PVLS (-) ACC (-) điều khiển tăng tốc và giảm tốc các xung ra từ cổng 1 và 2
133	@ABDL	Cộng hai giá trị nhị phân 8 chữ số (dữ kiện thường hoặc đóng dấu) và đưa kết quả ra R và R+1
134	@APR	Thể hiện các phép tính sin, cosin hoặc các tiệm cận tuyến tính
135	AVG	Cộng một số xác định các từ hexa và tính giá trị chính, quay dấu thập phân đi một khoảng 4 chữ số

2. Bảng lệnh của PLC - S5 (Siemens)

TT	**Tên lệnh**	**Mô tả**
2.1. Các lệnh cơ bản: (Sử dụng với khối OB, PB, FB, SB)		
2.1.1. Nhóm lệnh đại số logic Bool		
	)	Dùng để đóng ngoặc biểu thức đã mở ngoặc trước đó, lệnh này không có đối tượng
	A n	Thực hiện lệnh AND giữa nội dung của RLO với giá trị của điểm n (đơn vị bit) chỉ dẫn trong lệnh, kết quả ghi vào RLO
	A(	Thực hiện lệnh AND giữa nội dung trong RLO với phép toán trong ngoặc (có đóng ngoặc), kết quả phép toán nạp vào RLO
	AN n	Thực hiện lệnh AND giữa nội dung của RLO với giá trị nghịch đảo của điểm n (đơn vị bit) chỉ dẫn trong lệnh, kết quả ghi vào RLO
	O n	Thực hiện lệnh OR giữa nội dung của RLO với giá trị của điểm n (đơn vị bit) chỉ dẫn trong lệnh, kết quả ghi vào RLO
	O(	Thực hiện lệnh OR giữa nội dung trong RLO với phép toán trong ngoặc (có đóng ngoặc), kết quả phép toán nạp vào RLO
	ON n	Thực hiện lệnh OR giữa nội dung của RLO với giá trị nghịch đảo của điểm n (đơn vị bit) chỉ dẫn trong lệnh, kết quả ghi vào RLO

2.1.2. Lệnh set, reset		
	= n	Nội dung của RLO hiện hành được gán cho đối tượng n
	R n	Nếu nội dung của RLO là 1 thì trạng thái tín hiệu 0 sẽ được gán cho đối tượng n và trạng thái này không thay đổi khi RLO thay đổi
TT	**Tên lệnh**	**Mô tả**
	S n	Nếu nội dung RLO là 1 thì trạng thái tín hiệu 1 sẽ được gán cho đối tượng n và trạng thái này không thay đổi khi RLO thay đổi
2.1.3. Lệnh nạp và truyền		
	L n	Nội dung của đối tượng lệnh (đơn vị byte) được sao chép vào ACCU1 không phụ thuộc vào RLO, nội dung trước đó của ACCU1 chuyển sang ACCU2
	LD n	Nạp nội dung đối tượng n (dạng mã BCD) vào ACCU1 không phụ thuộc RLO
	T n	Nội dung của ACCU1 truyền cho đối tượng n (đơn vị byte) không phụ thuộc RLO, ví dụ truyền cho vùng đệm đầu ra
2.1.4 Lệnh về thời gian		
	R T	Xoá bộ thời gian nếu RLO = 1
	SD	Bộ thời gian chậm sau sườn lên của RLO một khoảng bằng thời gian đặt, khi RLO về 0 thì bộ thời gian về không ngay
	SE	Bộ thời gian lên 1 khi RLO chuyển từ 0 lên 1 (sườn lên) và duy trì đủ thời gian đặt, không phụ thuộc RLO nữa
	SF	Bộ thời gian lên 1 tại sườn lên của RLO, khi RLO về không thì bộ thời gian còn duy trì một khoảng thời gian bằng thời gian đặt
	SP	Bộ thời gian lên 1 khi RLO chuyển từ 0 lên 1 (sườn lên) và duy trì cho đến khi đạt thời gian đã đặt (RLO=1), khi RLO =0 thì bộ thời gian về 0 ngay
	SS	Bộ thời gian chậm sau sườn lên của RLO một khoảng bằng thời gian đặt và không phụ thuộc RLO nữa, nó chỉ về không khi có lệnh xoá R
2.1.5. Lệnh của bộ đếm		
	CD	Số đếm giảm 1 đơn vị tại sườn lên của RLO sau đó không phụ thuộc RLO nữa
	CU	Số đếm tăng 1 đơn vị tại sườn lên của RLO sau đó không phụ thuộc RLO nữa
	R C	Xoá bộ đếm nếu RLO = 1
	S C	Đặt bộ đếm nếu RLO = 1
2.1.6. Các lệnh toán học		
	!=F	So sánh bằng nhau của hai thanh ghi ACCU1 và ACCU2 (dạng bit)
	+F	Cộng nội dung hai thanh ghi ACCU1 và ACCU2, kết quả nạp vào ACCU1 (lệnh này chỉ có ở STL)
	<=F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có nhỏ hơn hay bằng ở ACCU1 không ?

	<F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có nhỏ hơn ở ACCU1 không?
	><F	So sánh đối tượng lệnh trong hai thanh ghi ACCU1 và ACCU2 xem có khác nhau không ?
	>=F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có lớn hơn hay bằng ở ACCU1 không ?
TT	**Tên lệnh**	**Mô tả**
	>F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có lớn hơn ở ACCU1 không ?
	-F	Trừ nội dung ở thanh ghi ACCU1 với nội dung ở thanh ghi ACCU2, kết quả nạp vào ACCU1 (lệnh này chỉ có ở STL)
2.1.7. Các lệnh gọi khối.		
	C n	Gọi khối dữ liệu DB, không phụ thuộc vào RLO, quét chương trình không bị gián đoạn, RLO không bị ảnh hưởng
	G	Tạo lập hoặc xoá khối dữ liệu độc lập với RLO
	JC n	Nhảy sang làm việc ở khối n nếu RLO =1
	JU n	Nhảy sang làm việc ở khối n, không phụ thuộc RLO và RLO không bị ảnh hưởng
2.1.8. Các lệnh kết thúc.		
	BE	Lệnh kết thúc khối
	BEC	Lệnh kết thúc có điều kiện giữa khối (RLO=1)
	BEU	Lệnh kết thúc không điều kiện giữa khối, không phụ thuộc RLO
2.1.9. Các lệnh khống.		
	NOP 0	Mã lệnh 16 bit trong RAM đều bằng 0 (để giữ chỗ)
	NOP 1	Mã lệnh 16 bit trong RAM đều bằng 1 (để giữ chỗ)
2.1.10. Lệnh dừng		
	STP	Lệnh dừng cuối chương trình, bộ PLC đi vào trạng thái nghỉ
2.2. Các lệnh thay thế (chỉ dùng với khối FB)		
2.2.1. Các lệnh đại số logic Bool thay thế.		
	A=	Lệnh AND thay thế
	AN=	Lệnh AND đảo thay thế
	AW	Tổ hợp từng bit theo luật logic AND
	DO=	Lệnh DO thay thế
	O=	Lệnh OR thay thế
	ON=	Lệnh OR đảo thay thế
	OW	Tổ hợp từng bit theo luật logic OR
	XOR	Tổ hợp từng bit theo luật logic OR đặc biệt

2.2.2. Các lệnh về bit		
	RU	Lệnh xoá bít không điều kiện
	SU	Đặt một bit vô điều kiện
	TB	Trắc nghiệm bit cho trạng thái tín hiệu 1
	TBN	Trắc nghiệm bit cho trạng thái tín hiệu 0
TT	**Tên lệnh**	**Mô tả**
2.2.3. Lệnh set, reset thay thế		
	==	Lệnh gán thay thế
	RB=	Lệnh xoá đối tượng lệnh hình thức
	RD=	Lệnh xoá đối tượng lệnh hình thức dạng số
	S=	Lệnh đặt đối tượng lệnh hình thức
2.2.4. Các lệnh về thời gian và đếm		
	FR=	Lệnh khả thi thay thế
	SD=	Lệnh khởi động bộ thời gian bắt đầu trễ hình thức
	SEC=	Khởi động bộ thời gian mở rộng hoặc bộ đếm
	SFD=	Lệnh khởi động bộ thời gian tắt trễ hoặc bộ đếm xuống
	SP=	Lệnh khởi động bộ thời gian xung hình thức
	SSU=	Lệnh khởi động bộ thời gian bắt đầu trễ
2.2.5. Các lệnh nạp và truyền		
	L=	Lệnh nạp thay thế
	LD=	Lệnh nạp đối tượng hình thức dạng cơ số BCD
	LW=	Lệnh nạp mẫu bit của đối tượng lệnh hình thức
	T=	Lệnh truyền đối tượng lệnh hình thức
2.2.6. Các lệnh chuyển đổi		
	CFW	Nội dung ACCU1 được chuyển đổi từng bit một
	CSW	Bổ sung cho 2
2.2.7. Các lệnh dịch chuyển		
	SLW	Dãy bit trong ACCU1 dịch sang trái
	SRW	Dãy bit trong ACCU1 dịch sang phải
2.2.8. Các lệnh nhảy		
	JC=	Nhảy có điều kiện (RLO=1)
	JM=	Nhảy nếu kết quả là âm (CC1=0, CC0=1)
	JN=	Nhảy nếu kết quả là (0,0) (CC1=1, CC0=0)
	JO=	Nhảy khi cờ tràn

	JP=	Nhảy nếu kết quả là dương (CC1=1, CC0=0)
	JU=	Nhảy không điều kiện
	JZ=	Nhảy nếu kết quả là 0 (CC1=0, CC0=0)
2.2.9. Các lệnh khác		
	D	Giảm nội dung trong ACCU1
	DO	Xử lý từ cờ hoặc từ dữ liệu
TT	**Tên lệnh**	**Mô tả**
	FR TC	Tác động vào TIME hoặc COUTER cả khi không có biến đổi sườn để khởi động bộ thời gian, đặt một bộ đếm đếm lên hoặc đếm xuống
	I	Tăng nội dung trong ACCU1
	IA	Lệnh cấm ngắt
	LRS	Nạp miền dữ liệu hệ thống (nạp miền RS... vào ACCU1)
	RA	Cho phép ngắt
2.2.10. Nhóm lệnh hệ thống		
	ADD	Lệnh cộng một hằng số
	JC n	Nhảy sang làm việc ở khối n nếu RLO =1
	JU n	Nhảy sang làm việc ở khối n, không phụ thuộc RLO và RLO không bị ảnh hưởng
	LIR	Lệnh nạp gián tiếp thanh ghi
	RU	Lệnh xoá bít không điều kiện
	STS	Lệnh dừng tức khắc
	SU	Đặt một bit vô điều kiện
	TAK	Lệnh trao đổi nội dung thanh ghi
	TIR	Lệnh truyền gián tiếp thanh ghi
	TNB	Lệnh truyền một trường dữ liệu

3. Bảng lệnh của PLC - S7-200 (Siemens)

TT	Tên lệnh		Mô tả
1	=	n	Giá trị bit đầu tiên trong ngăn xếp được sao chép sang điểm n chỉ dẫn trong lệnh
2	=I	n	Giá trị bit đầu tiên trong ngăn xếp được sao chép trực tiếp sang điểm n chỉ dẫn ngay khi lệnh được thực hiện
3	A	n	Giá trị bít đầu tiên của ngăn xếp được thực hiện bằng phép tính AND với điểm n chỉ dẫn trong lệnh. Kết quả được ghi lại vào bít đầu tiên của ngăn xếp

4	AB<= n_1,n_2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu giá trị byte n_1 không lớn hơn giá trị của byte n_2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
5	AB= n_1,n_2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu giá trị của hai byte n_1 và n_2 thoả mãn $n_1 = n_2$. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
6	AB>= n_1,n_2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu giá trị của hai byte n_1 và n_2 thoả mãn $n_1 \geq n_2$. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
TT	**Tên lệnh**	**Mô tả**
7	AD<= n_1,n_2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ kép (4byte) n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
8	AD>= n_1,n_2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ kép (4byte) n_1 và n_2 thoả mãn $n_1 \geq n_2$. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
9	AD= n_1,n_2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ kép (4byte) n_1 và n_2 thoả mãn $n_1 = n_2$. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
10	AI n	Lệnh AND được thực hiện tức thời giữa giá trị của bít đầu tiên trong ngăn xếp với điểm n được chỉ dẫn. Kết quả được ghi lại vào bit đầu của ngăn xếp
11	ALD	Thực hiện lệnh AND giữa giá trị của bít đầu tiên và của bit thứ hai trong ngăn xếp. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp, các giá trị còn lại trong ngăn xếp được kéo lên một bit
12	AN n	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị nghịch đảo của điểm n trong chỉ dẫn. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
13	ANI n	Thực hiện tức thời lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị nghịch đảo của điểm n trong chỉ dẫn. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
14	AR<= $n_1,n_2^{(5)}$	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai số thực n_1 và n_2 thoả mãn $n_1 \leq n_2$. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
15	AR= $n_1,n_2^{(5)}$	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai số thực n_1 và n_2 thoả mãn $n_1 = n_2$. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
16	AR>= $n_1,n_2^{(5)}$	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai số thực n_1 và n_2 thoả mãn $n_1 \geq n_2$. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
17	AW<= n_1,n_2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ n_1 và n_2 thoả mãn $n_1 \leq n_2$. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
18	AW= n_1,n_2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ n_1 và n_2 thoả mãn $n_1 = n_2$. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp
19	AW>= n_1,n_2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ n_1 và n_2 thoả mãn $n_1 \geq n_2$. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp

20	CTU Cxx,PV	Khởi động bộ đếm tiến theo sườn lên tín hiệu đầu vào. Bộ đếm được đặt lại trạng thái ban đầu *(Reset)* nếu đầu vào R của bộ đếm được kích
21	CTUD Cxx,PV	Khởi động bộ đếm tiến theo sườn lên tín hiệu đầu vào thứ nhất và đếm lùi theo sườn lên tín hiệu thứ hai. Bộ đếm được đặt lại trạng thái ban đầu *(Reset)* nếu đầu vào R của bộ đếm được kích
22	ED	Đặt giá trị logic 1 vào bit đầu tiên của ngăn xếp khi xuất hiện sườn xuống của tín hiệu
23	EU	Đặt giá trị logic 1 vào bit đầu tiên của ngăn xếp khi xuất hiện sườn lên của tín hiệu
TT	**Tên lệnh**	**Mô tả**
24	LD n	Nạp giá trị logic của điểm n chỉ dẫn trong lệnh vào bit đầu tiên của ngăn xếp
25	LDB<= n_1,n_2	Bit đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai byte n_1 và n_2 thoả mãn $n_1 \leq n_2$
26	LDB= n_1,n_2	Bit đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai byte n_1 và n_2 thoả mãn $n_1 = n_2$
27	LDB>= n_1,n_2	Bit đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai byte n_1 và n_2 thoả mãn $n_1 \geq n_2$
28	LDD= n_1,n_2	Bit đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ kép n_1 và n_2 thoả mãn $n_1 = n_2$
29	LDD>= n_1,n_2	Bit đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ kép n_1 và n_2 thoả mãn $n_1 \geq n_2$
30	LDI n	Lệnh nạp tức thời giá trị logic của tiếp điểm n chỉ dẫn trong lệnh vào bít đầu tiên trong ngăn xếp
31	LDN n	Lệnh nạp giá trị logic nghịch đảo của tiếp điểm n chỉ dẫn trong lệnh vào bít đầu tiên trong ngăn xếp
32	LDNI n	Lệnh nạp tức thời giá trị logic nghịch đảo của tiếp điểm n chỉ dẫn trong lệnh vào bít đầu tiên trong ngăn xếp
33	LDR<= n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai số thực n_1 và n_2 thoả mãn $n_1 \leq n_2$
34	LDR= n_1,n_2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai số thực n_1 và n_2 thoả mãn $n_1 = n_2$
35	LDR>= n_1,n_2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai số thực n_1 và n_2 thoả mãn $n_1 \geq n_2$
36	LDW<= n_1,n_2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ n_1 và n_2 thoả mãn $n_1 \leq n_2$
37	LDW= n_1,n_2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ n_1 và n_2 thoả mãn $n_1 = n_2$
38	LDW>= n_1,n_2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ n_1 và n_2 thoả mãn $n_1 \geq n_2$
39	LPP	Kéo nội dung của ngăn xếp lên một bit. Giá trị mới của bit trên là giá trị cũ của bit dưới, độ sâu của ngăn xếp giảm đi một bit
40	LPS	Sao chép giá trị bit đầu tiên trong ngăn xếp vào bit thứ hai. Nội dung còn lại của ngăn xếp bị đẩy xuống một bit
41	LRD	Sao chép giá trị của bit thứ hai vào bit đầu tiên trong ngăn xếp. Các giá trị còn lại của ngăn xếp giữ nguyên
42	MEND (1) (2)	Kết thúc phần chương trình trong một vòng quét

43	NOT	Đảo giá trị của bit đầu tiên trong ngăn xếp
44	O n	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với điểm n chỉ dẫn trong lệnh. Kết quả được ghi vào bit đầu tiên trong ngăn xếp
45	OB<= n_1,n_2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai byte n_1 và n_2 thoả mãn $n_1 \leq n_2$. Kết quả được ghi vào bit đầu tiên trong ngăn xếp
46	OB= n_1,n_2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai byte n_1 và n_2 thoả mãn $n_1 = n_2$. Kết quả được ghi vào bit đầu tiên trong ngăn xếp
47	OB>= n_1,n_2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp
TT	**Tên lệnh**	**Mô tả**
48	OD<= n_1,n_2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai từ kép n_1 và n_2 thoả mãn $n_1 \leq n_2$. Kết quả được ghi vào bit đầu tiên trong ngăn xếp
49	OD= n_1,n_2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai từ kép n_1 và n_2 thoả mãn $n_1 = n_2$. Kết quả được ghi vào bit đầu tiên trong ngăn xếp
50	OD>= n_1,n_2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai từ kép n_1 và n_2 thoả mãn $n_1 \geq n_2$. Kết quả được ghi vào bit đầu tiên trong ngăn xếp
51	OI n	Thực hiện tức thời toán tử OR giữa bit đầu tiên của ngăn xếp với điểm n chỉ dẫn trong lệnh. Kết quả được ghi vào bit đầu tiên trong ngăn xếp
52	OLD	Thực hiện toán tử OR giữa bit đầu và bit thứ hai trong ngăn xếp. Kết quả được ghi vào bit đầu tiên trong ngăn xếp, các giá trị còn lại của ngăn xếp được chuyển lên một bit
53	ON n	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic nghịch đảo của điểm n chỉ dẫn trong lệnh. Kết quả được ghi vào bit đầu tiên trong ngăn xếp
54	ONI n	Thực hiện tức thời toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic nghịch đảo của điểm n chỉ dẫn trong lệnh. Kết quả được ghi vào bit đầu tiên trong ngăn xếp
55	OR<= $n_1,n_2^{(5)}$	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai số thực n_1 và n_2 thoả mãn $n_1 \leq n_2$. Kết quả được ghi lại vào bit đầu trong ngăn xếp
56	OR= $n_1,n_2^{(5)}$	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai số thực n_1 và n_2 thoả mãn $n_1 = n_2$. Kết quả được ghi lại vào bit đầu trong ngăn xếp
57	OR>= $n_1,n_2^{(5)}$	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai số thực n_1 và n_2 thoả mãn $n_1 \geq n_2$. Kết quả ghi lại vào bit đầu trong ngăn xếp
58	OW<= $n_1,n_2^{(5)}$	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai từ n_1 và n_2 thoả mãn $n_1 \leq n_2$. Kết quả được ghi lại vào bit đầu trong ngăn xếp
59	OW= $n_1,n_2^{(5)}$	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai từ n_1 và n_2 thoả mãn $n_1 = n_2$. Kết quả được ghi lại vào bit đầu trong ngăn xếp

60	OW>=	n_1,n_2[5]	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai từ n_1 và n_2 thoả mãn $n_1 \geq n_2$. Kết quả được ghi lại vào bit đầu trong ngăn xếp
61	RET	(1) (3) (4)	Lệnh thoát khỏi chương trình con và trả điều khiển chương trình đã gọi nó
62	RETI	(2) (3) (4)	Lệnh thoát khỏi chương trình xử lý ngắt *(interrupt)* và trả điều khiển chương trình chính
3.2. Các lệnh có điều kiện			
63	*R	IN1, IN2[5]	Thực hiện phép nhân hai số thực (32bit) IN1 và IN2. Kết quả được ghi lại vào IN2
64	/R	IN1, IN2[5]	Thực hiện phép chia hai số thực (32bit) IN1 và IN2. Kết quả được ghi lại vào IN2
TT	**Tên lệnh**		**Mô tả**
65	+D	IN1, IN2	Thực hiện phép cộng hai số nguyên kiểu từ kép IN1 và IN2. Kết quả được ghi lại vào IN2
66	+I	IN1, IN2	Thực hiện phép cộng hai số nguyên kiểu từ IN1 và IN2. Kết quả được ghi lại vào IN2
67	+R	IN1, IN2[5]	Thực hiện phép cộng hai số thực (32bit) IN1 và IN2. Kết quả được ghi lại vào IN2
68	ANDD	IN1, IN2	Thực hiện toán tử AND giữa các giá trị kiểu từ kép IN1 và IN2. Kết quả được ghi lại vào IN2
69	ANDW	IN1, IN2	Thực hiện toán tử AND giữa các giá trị kiểu từ IN1 và IN2. Kết quả được ghi lại vào IN2
70	ATCH	INT, EVENT	Khai báo chương trình xử lý ngắt INT theo kiểu EVENT
71	ATH	IN, OUT, LEN	Biến đổi một sâu ký tự từ mã ASCII từ vị trí IN (kiểu byte) với độ dài LEN (kiểu byte) sang mã hexa (cơ số 16) và ghi vào mảng kể từ byte OUT
72	ATT	DATA TABLE	Nối một giá trị kiểu từ DATA (2 byte) vào bảng TABLE
73	BCDI	IN	Biến đổi một giá trị từ mã BCD có độ dài 2 byte sang kiểu nguyên. Kết quả được ghi lại vào IN
74	BMB	IN, OUT, N	Sao chép một mảng gồm N byte kể từ vị trí đầu IN (byte) vào mảng có vị trí là OUT (kiểu byte)
75	BMW	IN, OUT, N	Sao chép một mảng từ (2 byte) với độ dài N (1 byte) và vị trí đầu IN (2 byte) vào mảng có vị trí đầu OUT (2 byte).
76	CALL	n (1) (6)	Gọi chương trình con được đánh nhãn n
77	CRET	(1)(3)(4)	Kết thúc một chương trình con và trả điều khiển về chương trình đã gọi nó
78	CRETI	(2)(3)(4)	Kết thúc một chương trình xử lý ngắt và trả điều khiển về chương trình chính
79	-D	IN1, IN2	Thực hiện phép trừ hai số nguyên kiểu từ kép IN1 và IN2. Kết quả được ghi lại vào IN2
80	DECD	IN	Giảm giá trị của từ kép IN đi một đơn vị
81	DECO	IN, OUT	Giải mã giá trị của một byte IN sau đó gán giá trị 1 vào bit của từ OUT (2 byte) có chỉ số là IN

82	DECW	IN	Giảm giá trị của từ IN đi một đơn vị
83	DISI	(1)	Vô hiệu hoá tất cả các ngắt (interrupt)
84	DIV	IN1, IN2	Chia số nguyên 16 bit, được xác định là từ thấp của IN2 (kiểu từ kép), cho IN1 kiểu từ. Kết quả được ghi lại vào từ IN2
85	DTCH	EVENT	Vô hiệu hoá một ngắt kiểu EVENT
86	DTR	IN, OUT(5)	Chuyển đổi một số nguyên 32 bit IN có dấu sang thành một số thực 32 bit OUT
87	ENCO	IN, OUT	Chuyển đổi chỉ số của bit thấp nhất có giá trị logic 1 trong từ IN sang thành một số nguyên và ghi vào bit cuối của byte OUT
88	ENI	(1)	Đặt tất cả các ngắt vào chế độ tích cực
89	FIFO	TABLE, DATA(5)	Lấy giá trị đã được cho vào đầu tiên ra khỏi bảng và chuyển nó đến vùng dữ liệu DATA được chỉ dẫn trong lệnh
TT	**Tên lệnh**		**Mô tả**
90	FILL	IN, OUT, N	Đổ giá trị từ IN vào một mảng nhớ gồm N từ (N có kiểu byte) bắt đầu từ vị trí OUT (kiểu từ)
91	FND<	SRC, PATRR INDX(5)	Xác định vị trí ô nhớ trong bảng SRC (kiểu từ), kể từ ô cho bởi INDX (kiểu từ, =0 nếu từ đầu bảng) mà ở đó giá trị nhỏ hơn giá trị của PATRN (kiểu từ).
92	PATRR	INDX(5)	Xác định vị trí ô nhớ trong bảng SRC (kiểu từ), kể từ ô cho bởi INDX (kiểu từ, =0 nếu từ đầu bảng) mà ở đó giá trị khác giá trị của PATRN (kiểu từ)
93	FND=	SRC, PATRR INDX(5)	Xác định vị trí ô nhớ trong bảng SRC (kiểu từ), kể từ ô cho bởi INDX (kiểu từ, =0 nếu từ đầu bảng) mà ở đó giá trị bằng giá trị của PATRN (kiểu từ)
94	FND>	SRC, PATRR INDX(5)	Xác định vị trí ô nhớ trong bảng SRC (kiểu từ), kể từ ô cho bởi INDX (kiểu từ, =0 nếu từ đầu bảng) mà ở đó giá trị lớn hơn giá trị của PATRN (kiểu từ)
95	FOR	INDEX INITIAL FINAL(1)(5)	Thực hiện các lệnh nằm giữa FOR và NEXT theo kiểu xoay vòng với bộ đếm số vòng INDEX (kiểu từ), bắt đầu từ vòng số INITIAL (kiểu từ) và kết thúc tại vòng FINAL (từ)
96	HDEF	HSC, MODE(1)	Xác định kiểu thuật toán MODE cho bộ đếm tốc độ cao HSC (byte).
97	HSC	n	Đưa bộ đếm tốc độ cao số n vào trạng thái tích cực
98	HTA	IN, OUT, LEN	Chuyển đổi một số hệ hexa IN (kiểu byte) thành dãy ký tự mã ASCII và ghi vào mảng byte bắt đầu bằng byte OUT với độ dài LEN (kiểu byte)
99	-I	IN1, IN2	Thực hiện phép trừ hai số nguyên kiểu từ IN1 và IN2. Kết quả được ghi lại vào IN2
100	IBCD	IN	Chuyển đổi giá trị nguyên IN (kiểu từ) thành giá trị BCD và ghi lại vào IN
101	INCD	IN	Tăng giá trị của từ kép IN lên một đơn vị
102	INCW	IN	Tăng giá trị của từ IN lên một đơn vị
103	INT	n(1)(2)(4)	Khai báo nhãn n cho chương trình xử lý ngắt

104	INVD	IN	Lấy phần bù kiểu một (đảo giá trị logic của các bit) của một từ kép IN và ghi lại vào IN
105	JMP	xx	Chuyển điều khiển vào ô nhớ định bằng nhãn xx trong chương trình được khai báo bởi lệnh LBL
106	LBL	xx	Đặt nhãn xx trong chương trình, định hướng cho lệnh nhảy JMP
107	LIFO	TABLE, DATA(5)	Lấy giá trị đã được cho vào bảng sau cùng ra khỏi bảng TABLE và chuyển nó đến vùng dữ liệu DATA (kiểu từ)
108	MOVB	IN, OUT	Sao giá trị của byte IN sang byte OUT
109	MOVD	IN, OUT	Sao giá trị của từ kép IN sang từ kép OUT
110	MOVR	IN, OUT(5)	Sao số thực IN sang OUT
TT	**Tên lệnh**		**Mô tả**
111	MOVW	IN, OUT	Sao giá trị của từ IN sang từ OUT
112	MUL	IN1, IN2	Nhân hai số nguyên 16 bit IN1 với hai byte thấp của số nguyên 32 bit IN2 sau đó ghi lại kết quả vào IN2
113	NETR	TABLE, PORT(5)	Khởi tạo truyền thông để đọc dữ liệu từ ngoại vi qua cổng PORT vào bảng TABLE
114	NETW	TABLE, PORT(5)	Khởi tạo truyền thông để ghi dữ liệu của bảng TABLE ra ngoại vi qua cổng PORT
115	NEXT	(1)(5)(7)	Lệnh kết thúc vòng lặp FOR ... NEXT
116	NOP		Lệnh rỗng
117	ORD	IN1, IN2	Thực hiện toán tử OR cho hai từ kép IN1 và IN2, sau đó ghi kết quả lại vào IN2
118	ORW	IN1, IN2	Thực hiện toán tử OR cho hai từ IN1 và IN2, sau đó ghi kết quả lại vào IN2
119	PLS	xx(5)	Đưa bộ phát xung nhanh đã được định nghĩa trong bộ nhớ đặc biệt vào trạng thái tích cực. Xung được đưa ra cổng Qx.x
120	R	S_BIT,n	Xoá một mảng gồm n bit kể từ địa chỉ S_BIT (kiểu bit)
121	-R	IN1, IN2(5)	Thực hiện phép trừ hai số thực (32bit) IN1 và IN2. Kết quả được ghi lại vào IN2
122	RI	S_BIT,n	Xoá tức thời một mảng gồm n bit kể từ địa chỉ S_BIT
123	RLD	IN, n	Quay tròn từ kép IN sang trái n bit
124	RLW	IN, n	Quay tròn từ IN sang trái n bit
125	RRD	IN, n	Quay tròn từ kép IN sang phải n bit
126	RRW	IN, n	Quay tròn từ IN sang phải n bit
127	S	S_BIT,n	Đặt giá trị logic 1 vào một mảng n bit kể từ địa chỉ S_BIT
128	SBR	n(1)(2)(4)	Khai báo nhãn n cho chương trình con

129	SEG	IN, OUT	Chuyển đổi giá trị của 4 bit thấp trong byte IN sang thành mã tương ứng cho thanh ghi 7 nét và ghi vào OUT
130	SHRB	DATA, S_BIT,n	Dịch thanh ghi gồm *n* bit có bit thấp nhất là S_BIT sang trái nếu n>0, hoặc sang phải nếu n<0. Giá trị của bit DATA được đưa vào bit trống của thanh ghi sau khi dịch (bit S_BIT nếu n>0, hoặc bit S_BIT nếu n<0)
131	SI	S_BIT,n	Đặt tức thời giá trị logic 1 vào mảng n bit kể từ bit S_BIT
132	SLD	IN,n	Dịch từ kép IN sang trái một bit
133	SLW	IN,n	Dịch từ IN sang trái một bit
134	SQRT	IN, OUT(5)	Lấy căn bậc hai của số thực 32 bit IN và ghi kết quả vào OUT (32bit).
135	SRD	IN,n	Dịch từ kép IN sang phải một bit
TT	**Tên lệnh**		**Mô tả**
136	SRW	IN,n	Dịch từ IN sang phải một bit
137	STOP		Dừng "mềm" chương trình
138	SWAP	IN	Đổi chỗ hai bit đầu tiên và cuối cùng của byte IN cho nhau
139	TODR	T(5)	Đọc giờ và ngày tháng sau hiện thời từ đồng hồ và ghi vào bộ đệm 8 byte đầu là T
140	TODW	T(5)	Ghi vào đồng hồ giá trị thời gian, ngày, tháng từ bộ đệm 8 byte với byte đầu là T
141	TON	Txx, PT	Khởi động bộ phát thời gian trễ Txx với thời gian trễ đặt trước là tích của PT (kiểu từ) và độ phân giải của thời gian Txx chọn
142	TONR	Txx, PT	Khởi động bộ phát thời gian trễ có nhớ Txx với thời gian trễ đặt trước là tích của PT (kiểu từ) và độ phân giải của bộ thời gian Txx được chọn
143	TRUNG	IN, OUT(5)	Chuyển đổi một số thực 32 bit IN thành một số nguyên 32 bit có đấu và ghi vào OUT
144	WDR		Đặt chuẩn lại bộ phát xung kiểm tra.
145	XMT	TABLE, PORT	Truyền nội dung của bảng TABLE đến cổng PORT
146	XORD	IN1, IN2	Thực hiện toán tử exclusive OR cho các bit của hai từ kép IN1 và IN2. Kết quả được ghi lại vào IN2
147	XORW	IN1, IN2	Thực hiện toán tử exclusive OR cho các bit của hai từ IN1 và IN2. Kết quả được ghi lại vào IN2
(1)	Những lệnh không thực hiện được trong chương trình xử lý ngắt. Lệnh INT chỉ có thể là lệnh bắt đầu của chương trình xử lý ngắt		
(2)	Những lệnh không thực hiện được trong chương trình con. Lệnh SBR chỉ có thể là lệnh bắt đầu của chương trình con		
(3)	Những lệnh có kèm chức năng ghi lại nội dung của ngăn xếp trước đó		
(4)	Những lệnh không sử dụng được trong chương trình chính		
(5)	Những lệnh chỉ có trong CPU 214		

(6)	Ghi nhớ lại nội dung tức thời của ngăn xếp. Đặt TOS lên 1 và gán giá trị logic 0 vào các bit còn lại của ngăn xếp
(7)	Đặt TOS lên 1

4. Bảng lệnh của PLC - S7-300 (Siemens)

TT	Tên lệnh	Mô tả
1.	+ n	Cộng với hằng số được viết ở điểm n
2.	= n	Nội dung của RLO hiện hành được gán cho đối tượng n
3.	)	Dùng để đóng ngoặc biểu thức đã mở ngoặc trước đó, lệnh này không có đối tượng
4.	+ AR1 n	Cộng nội dung của ACCU1 hoặc nội dung tại con trỏ n với nội dung có địa chỉ ở thanh ghi 1
5.	+AR2 n	Cộng nội dung của ACCU1 hoặc nội dung tại con trỏ n với nội dung có địa chỉ ở thanh ghi 2
TT	**Tên lệnh**	**Mô tả**
6.	+D	Cộng 2 số nguyên 32 bit ở ACCU1 và ACCU2, kết quả để ở ACCU 1
7.	-D	Trừ số nguyên 32 bit ở ACCU2 cho số nguyên 32 bit ở ACCU1, kết quả để ở ACCU1
8.	*D	Nhân 2 số nguyên 32 bit ở ACCU1 và ACCU2, kết quả để ở ACCU1
9.	/D	Chia số nguyên 32 bit ở ACCU2 cho số nguyên 32 bit ở ACCU1, kết quả để ở ACCU1
10.	==D	So sánh hai số nguyên 32 bit ở ACCU1 và ACCU2 có bằng nhau không
11.	<>D	So sánh hai số nguyên 32 bit ở ACCU1 và ACCU2 xem có khác nhau không
12.	>D	So sánh số nguyên 32 bit ở ACCU2 có lớn hơn số nguyên 32 bit ở ACCU1 không
13.	<D	So sánh số nguyên 32 bit ở ACCU2 có nhỏ hơn số nguyên 32 bit ở ACCU1 không
14.	>=D	So sánh số nguyên 32 bit ở ACCU2 có lớn hơn hay bằng số nguyên 32 bit ở ACCU1 không
15.	<=D	So sánh số nguyên 32 bit ở ACCU2 có nhỏ hơn hay bằng số nguyên 32 bit ở ACCU1 không
16.	+I	Cộng 2 số nguyên 16 bit ở ACCU1 và ACCU2, kết quả để ở ACCU1
17.	-I	Trừ số nguyên 16 bit ở ACCU2 cho số nguyên 16 bit ở ACCU1, kết quả để ở ACCU1
18.	*I	Nhân 2 số nguyên 16 bit ở ACCU1 và ACCU2, kết quả để ở ACCU1
19.	/I	Chia số nguyên 16 bit ở ACCU2 cho số nguyên 16 bit ở ACCU1, kết quả để ở ACCU1
20.	==I	So sánh hai số nguyên 16 bit ở ACCU1 và ACCU2 có bằng nhau không
21.	<>I	So sánh hai số nguyên 16 bit ở ACCU1 và ACCU2 xem có khác nhau không

22.	>I	So sánh số nguyên 16 bit ở ACCU2 có lớn hơn số nguyên 16 bit ở ACCU1 không
23.	<I	So sánh số nguyên 16 bit ở ACCU2 có nhỏ hơn số nguyên 16 bit ở ACCU1 không
24.	>=I	So sánh số nguyên 16 bit ở ACCU2 có lớn hơn hay bằng số nguyên 16 bit ở ACCU1 không
25.	<=I	So sánh số nguyên 16 bit ở ACCU2 có nhỏ hơn hay bằng số nguyên 16 bit ở ACCU1 không
26.	+R	Cộng 2 số thực 32 bit ở ACCU1 và ACCU2, kết quả để ở ACCU1
27.	-R	Trừ số thực 32 bit ở ACCU2 cho số thực 32 bit ở ACCU1, kết quả để ở ACCU1
28.	*R	Nhân 2 số thực 32 bit ở ACCU1 và ACCU2, kết quả để ở ACCU1
29.	/R	Chia số thực 32 bit ở ACCU2 cho số thực 32 bit ở ACCU1, kết quả để ở ACCU1
30.	==R	So sánh hai số thực 32 bit ở ACCU1 và ACCU2 có bằng nhau không
TT	**Tên lệnh**	**Mô tả**
31.	<>R	So sánh hai số thực 32 bit ở ACCU1 và ACCU2 xem có khác nhau không
32.	>R	So sánh số thực 32 bit ở ACCU2 có lớn hơn số thực 32 bit ở ACCU1 không
33.	<R	So sánh số thực 32 bit ở ACCU2 có nhỏ hơn số thực 32 bit ở ACCU1 không
34.	>=R	So sánh số thực 32 bit ở ACCU2 có lớn hơn hay bằng số thực 32 bit ở ACCU1 không
35.	<=R	So sánh số thực 32 bit ở ACCU2 có nhỏ hơn hay bằng số thực 32 bit ở ACCU1 không
36.	A n	Thực hiện lệnh AND giữa nội dung của RLO với giá trị của điểm n (đơn vị bit) chỉ dẫn trong lệnh, kết quả ghi vào RLO
37.	A(	Thực hiện lệnh AND giữa nội dung trong RLO với phép toán trong ngoặc (có đóng ngoặc), kết quả phép toán nạp vào RLO
38.	ABS	Lấy giá trị tuyệt đối của số thực 32 bit
39.	AD	Thực hiện lệnh AND giữa nội dung trong ACCU1 và ACCU2, kết quả để ở ACCU1 (32 bit)
40.	AN n	Thực hiện lệnh AND giữa nội dung của RLO với giá trị nghịch đảo của điểm n (đơn vị bit) chỉ dẫn trong lệnh, kết quả ghi vào RLO
41.	AN(	Thực hiện lệnh AND giữa nội dung của RLO với giá trị nghịch đảo của biểu thức trong ngoặc (có đóng ngoặc), kết quả ghi vào RLO
42.	AW	Thực hiện lệnh AND giữa nội dung trong ACCU1 và ACCU2, kết quả để ở ACCU1 (16 bit)
43.	BEC	Lệnh kết thúc có điều kiện giữa khối (RLO=1)
44.	BEU	Lệnh kết thúc khối không điều kiện, không phụ thuộc RLO
45.	BLD	Hiển thị lệnh của chương trình
46.	BTD	Chuyển số dạng mã BCD sang số nguyên 32 bit
47.	BTI	Chuyển số dạng mã BCD sang số nguyên 16 bit

48.	CAD	Đổi thứ tự byte trong ACCU1 (32 bit)
49.	CAR	Chuyển nội dung thanh ghi 1 với nội dung thanh ghi 2
50.	CAW	Đổi thứ tự byte trong ACCU1 (16 bit)
51.	CALL	Lệnh gọi khối
52.	CC	Lệnh gọi khối có điều kiện
53.	CD	Số đếm giảm 1 đơn vị tại sườn lên của RLO sau đó không phụ thuộc RLO nữa
54.	CDB	Chuyển khối dữ liệu chung thành khối dữ liệu riêng
55.	CLR	Xoá RLO (RLO = 0)
56.	CU	Số đếm tăng 1 đơn vị tại sườn lên của RLO sau đó không phụ thuộc RLO nữa
57.	DEC	Giảm nội dung trong ACCU1 đi một đơn vị
58.	DTB	Đổi số nguyên 32 bit thành số dạng mã BCD
59.	DTR	Đổi số nguyên 32 bit thành số thực
TT	**Tên lệnh**	**Mô tả**
60.	FN	Chọn lấy sườn âm của RLO
61.	FP	Chọn lấy sườn dương của RLO
62.	FR T	Khởi tạo bộ thời gian TIME cả khi không có biến đổi sườn để khởi động bộ thời gian
63.	FR C	Khởi tạo bộ đếm COUTER cả khi không có biến đổi sườn để đặt một bộ đếm đếm lên hoặc đếm xuống
64.	INC	Tăng số trong ACCU1 lên một đơn vị
65.	INVD	Lấy phần bù một của số nguyên 32 bit
66.	INVI	Lấy phần bù một của số nguyên 16 bit
67.	ITB	Đổi số nguyên 16 bit thành số dạng mã BCD
68.	ITD	Đổi số nguyên 16 bit thành số nguyên 32 bit
69.	JBI n	Nhảy sang làm việc ở nhãn n nếu BR = 1
70.	JC n	Nhảy sang làm việc ở nhãn n nếu RLO =1
71.	JCB n	Nhảy sang làm việc ở nhãn n nếu RLO =1 và BR = 1
72.	JCN n	Nhảy sang làm việc ở nhãn n nếu RLO =0
73.	JL n	Nhảy đến nhãn ghi ở n
74.	JM	Nhảy nếu kết quả là âm (CC1 = 0, CC0 = 1).
75.	JMZ	Nhảy nếu kết quả là âm hoặc bằng không (CC1=0 hoặc 0, CC0=0 hoặc 1)
76.	JN	Nhảy nếu kết quả là khác không (CC1 = 1 hoặc 0, CC0 = 0 hoặc 1)
77.	JNB n	Nhảy sang làm việc ở nhãn n nếu RLO =0 và BR = 0
78.	JNBI n	Nhảy sang làm việc ở nhãn n nếu BR = 0

79.	JO	n	Nhảy sang làm việc ở nhãn nếu VO = 1
80.	JOS	n	Nhảy sang làm việc ở khối n nếu OS = 0
81.	JP		Nhảy nếu kết quả là dương (CC1 = 1, CC0 = 0)
82.	JPZ		Nhảy nếu kết quả là lớn hơn hoặc bằng không (CC1 = 0 hoặc 1, CC0 = 0 hoặc 0)
83.	JU	n	Nhảy sang làm việc ở nhãn n, không phụ thuộc RLO và RLO không bị ảnh hưởng
84.	JUO		Nhảy nếu (CC1 = 1, CC0 = 1)
85.	JZ		Nhảy nếu kết quả là không (CC1 = 0, CC0 = 0)
86.	L	n	Nội dung của đối tượng lệnh (đơn vị byte) được sao chép vào ACCU1 không phụ thuộc vào RLO, nội dung trước đó của ACCU1 chuyển sang ACCU2
87.	L	C	Nạp giá trị tức thời (số nguyên) của bộ đếm vào ACCU1
88.	L	T	Nạp giá trị tức thời (số nguyên) của bộ thời gian vào ACCU1
89.	L	DBLG	Nạp độ dài của khối dữ liệu DB vào ACCU1
90.	L	DBNO	Nạp số của khối dữ liệu DB vào ACCU1
TT	**Tên lệnh**		**Mô tả**
91.	L	DILG	Nạp độ dài của khối dữ liệu DI vào ACCU1
92.	L	DINO	Nạp số của khối dữ liệu DI vào ACCU1
93.	L	STW	Nạp từ trạng thái vào ACCU1
94.	LAR1		Nạp địa chỉ vào thanh ghi 1 từ ACCU1
95.	LAR1	n	Nạp địa chỉ vào thanh ghi 1 từ vị trí n ghi trong lệnh
96.	LAR1	AR2	Nạp địa chỉ vào thanh ghi 1 từ thanh ghi 2
97.	LAR1	P#	Nạp vào thanh ghi 1 từ địa chỉ tại con trỏ (số thực kép)
98.	LAR2		Nạp địa chỉ vào thanh ghi 2 từ ACCU1
99.	LAR2	n	Nạp địa chỉ vào thanh ghi 2 từ vị trí n ghi trong lệnh
100.	LAR2	P#	Nạp vào thanh ghi 2 từ địa chỉ tại con trỏ (số thực kép)
101.	LC	C	Nạp số đếm hiện thời dạng mã BCD vào ACCU1
102.	LC	T	Nạp giá trị thời gian hiện thời dạng mã BCD vào ACCU1
103.	LOOP	n	Lặp lại từ nhãn n
104.	MCR(		Cất kết quả của phép tính logic vào vùng MCR
105.	)MCR		Kết thúc vùng MCR
106.	MCRA		Kích hoạt vùng MCR
107.	MCRD		Thôi kích hoạt vùng MCR
108.	MOD		Phép chia lấy phần dư của số nguyên 32 bit ở ACCU2 cho số nguyên 32 bit ở ACCU1, kết quả để ở ACCU1
109.	NEGD		Lấy số bù hai của số nguyên 32 bit

110.	NEGI	Lấy số bù hai của số nguyên 16 bit
111.	NEGR	Lấy dấu âm cho số thực 32 bit
112.	NOP 0	Mã lệnh 16 bit trong RAM đều bằng 0 (để giữ chỗ)
113.	NOP 1	Mã lệnh 16 bit trong RAM đều bằng 1 (để giữ chỗ)
114.	NOT	Đặt trạng thái không cho RLO
115.	O n	Thực hiện lệnh OR giữa nội dung của RLO với giá trị của điểm n (đơn vị bit) chỉ dẫn trong lệnh, kết quả ghi vào RLO
116.	O(	Thực hiện lệnh OR giữa nội dung trong RLO với phép toán trong ngoặc (có đóng ngoặc), kết quả phép toán nạp vào RLO
117.	OD	Thực hiện lệnh OR giữa nội dung trong ACCU1 và ACCU2, kết quả để ở ACCU1 (32 bit)
118.	ON n	Thực hiện lệnh OR giữa nội dung của RLO với giá trị nghịch đảo của điểm n (đơn vị bit) chỉ dẫn trong lệnh, kết quả ghi vào RLO
119.	ON(	Thực hiện lệnh OR giữa nội dung của RLO với giá trị nghịch đảo phép toán trong ngoặc (có đóng ngoặc), kết quả ghi vào RLO
120.	OPN	Mở khối dữ liệu
121.	OW	Thực hiện lệnh OR giữa nội dung trong ACCU1 và ACCU2, kết quả để ở ACCU1 (16 bit)
TT	**Tên lệnh**	**Mô tả**
122.	POP	Chuyển nội dung ở ACCU2 sang ACCU1
123.	PUSH	Chuyển nội dung ở ACCU1 sang ACCU2
124.	R n	Nếu nội dung của RLO là 1 thì trạng thái tín hiệu 0 sẽ được gán cho đối tượng n và trạng thái này không thay đổi khi RLO thay đổi
125.	R T	Xoá bộ thời gian nếu RLO = 1
126.	R C	Xoá bộ đếm nếu RLO = 1
127.	RLD n	Quay tròn từ kép ở ACCU1 sang trái n bit
128.	RLDA	Quay tròn từ kép ở ACCU1 sang trái 1 bit qua CC 1
129.	RND	Đổi số thực 32 bit thành số nguyên 32 bit (bỏ phần thập phân)
130.	RND+	Đổi số thực 32 bit thành số nguyên 32 bit, nếu là số dương thì làm tròn tăng, là số âm thì bỏ phần thập phân
131.	RND-	Đổi số thực 32 bit thành số nguyên 32 bit, nếu là số âm thì làm tròn tăng, là số dương thì bỏ phần thập phân
132.	RRD n	Quay tròn từ kép ở ACCU1 sang phải n bit
133.	RRDA	Quay tròn từ kép ở ACCU1 sang phải 1 bit qua CC 1
134.	S n	Nếu nội dung RLO là 1 thì trạng thái tín hiệu 1 sẽ được gán cho đối tượng n và trạng thái này không thay đổi khi RLO thay đổi
135.	S C	Đặt bộ đếm nếu RLO = 1
136.	SAVE	Cất kết quả của phép tính logic vào thanh ghi BR
137.	SD	Bộ thời gian chậm sau sườn lên của RLO một khoảng bằng thời gian đặt, khi RLO về 0 thì bộ thời gian về không ngay
138.	SE	Bộ thời gian lên 1 khi RLO chuyển từ 0 lên 1 (sườn lên) và duy trì đủ thời gian đặt, không phụ thuộc RLO nữa

139.	SET		Đặt RLO =1
140.	SF		Bộ thời gian lên 1 tại sườn lên của RLO, khi RLO về không thì bộ thời gian còn duy trì một khoảng thời gian bằng thời gian đặt
141.	SLD	n	Dịch từ kép trong ACCU1 sang trái n bit hoặc số bit dịch được nạp vào ACCU2
142.	SLW	n	Dịch từ đơn trong ACCU1 sang trái n bit hoặc số bit dịch được nạp vào ACCU2
143.	SP		Bộ thời gian lên 1 khi RLO chuyển từ 0 lên 1 (sườn lên) và duy trì cho đến khi đạt thời gian đã đặt (RLO=1), khi RLO =0 thì bộ thời gian về 0 ngay
144.	SRD	n	Dịch từ kép trong ACCU1 sang phải n bit hoặc số bit dịch được nạp vào ACCU2
145.	SRW	n	Dịch từ đơn trong ACCU1 sang phải n bit hoặc số bit dịch được nạp vào ACCU2
146.	SS		Bộ thời gian chậm sau sườn lên của RLO một khoảng bằng thời gian đặt và không phụ thuộc RLO nữa, nó chỉ về không khi có lệnh xoá R
147.	SSD	n	Dịch số nguyên 32 bit trong ACCU1 sang phải n bit hoặc số bit dịch được nạp vào ACCU2, bit trống được chèn bit dấu của số nguyên
148.	SSI	n	Dịch số nguyên 16 bit trong ACCU1 sang phải n bit hoặc số bit dịch được nạp vào ACCU2, các bit trống được chèn bit dấu của số nguyên
TT	**Tên lệnh**		**Mô tả**
149.	T	n	Nội dung của ACCU1 truyền cho đối tượng n (đơn vị byte) không phụ thuộc RLO, ví dụ truyền cho vùng đệm đầu ra
150.	T	STW	Truyền từ trạng thái tới ACCU1
151.	TAK		Lệnh trao đổi nội dung trong ACCU1 và ACCU2
152.	TAR1		Truyền địa chỉ trong thanh ghi 1 đến ACCU1
153.	TAR1	n	Truyền địa chỉ trong thanh ghi 1 đến vị trí được chỉ trong lệnh
154.	TAR1	AR2	Truyền địa chỉ trong thanh ghi 1 đến thanh ghi 2
155.	TAR2		Truyền địa chỉ trong thanh ghi 2 đến ACCU1
156.	TAR2	n	Truyền địa chỉ trong thanh ghi 2 đến vị trí được chỉ trong lệnh
157.	TRUNC		Chuyển số thực 32 bit trong ACCU1 thành số nguyên 32 bit có dấu
158.	UC		Lệnh gọi khối không điều kiện
159.	X	n	Thực hiện lệnh OR (đặc biệt) giữa nội dung của RLO với giá trị của điểm n (đơn vị bit) chỉ dẫn trong lệnh, kết quả ghi vào RLO
160.	X(		Thực hiện lệnh OR (đặc biệt) giữa nội dung trong RLO với phép toán trong ngoặc (có đóng ngoặc), kết quả phép toán nạp vào RLO
161.	XN	n	Thực hiện lệnh OR (đặc biệt) giữa nội dung của RLO với giá trị nghịch đảo của điểm n, kết quả ghi vào RLO
162.	XN(		Thực hiện lệnh OR (đặc biệt) giữa nội dung của RLO với giá trị nghịch đảo phép toán trong ngoặc (có đóng ngoặc), kết quả ghi vào RLO
163.	XOD		Thực hiện lệnh OR (đặc biệt) giữa các bit của hai từ kép
164.	XOW		Thực hiện lệnh OR (đặc biệt) giữa các bit của hai từ đơn

TÀI LIỆU THAM KHẢO

1. Nguyễn Trọng Thuần, Điều khiển logic và ứng dụng, NXB Khoa học và kỹ thuật, 2000.

2. Nguyễn Doãn Phước, Phan Xuân Minh, Vũ Văn Hà. Tự động hoá tới Simatic S7-300, NXB Khoa học và kỹ thuật, 2000.

3. Tăng Văn Mùi. Nguyễn Tiến Dũng, Điều khiển logic lập trình PLC, NXB thống kê, 2003.

4. Nguyễn Doãn Phước, Phan Xuân Minh, Tự động hoá với Simatic S7-200, NXB Khoa học và kỹ thuật, 2000.

5. A Bigincr's guide to PLC, OMROM ASIA PACIFIC, Singapor 1996.

6. SIMATIC S5. Program examplesfor Programmable Conlrollers.1992.

7. Simatic Step 7 Statemenl Lisl Reference Manual, Siemen AG, Automation Group, Industrial Automation Systems, 1995.

MỤC LỤC

PHẠM KHÁNH TÙNG

TÀI LIỆU DÀO TẠO KỸ THUẬT VIÊN

LẬP TRÌNH PLC

Hà Nội, 3/2011