CHƯƠNG 3: NGÔN NGỮ LẬP TRÌNH VÀ ỨNG DỤNG.

3.1.Giới thiệu các ngôn ngữ lập trình:

Lập trình cho S7 200 và các PLC khác của hãng Siemens dựa trên 3 phương pháp cơ bản:

Phương pháp hình thang (Ladder logic _ LAD).

Phương pháp khối hàm (Function Block Diagram _ FBD).

Phương pháp liệt kê câu lệnh (Statement List _ STL).

Chương này sẽ giới thiệu các thành phần cơ bản của ba phương pháp và cách sử dụng chúng trong lập trình.

Nếu chương trình được viết theo ngôn ngữ LAD (hoặc FBD) thì có thể chuyển sang ngôn ngữ STL hay FBD (hoặc LAD) tương ứng. Nhưng không phải bất cứ chương trình viết theo STL nào cũng chuyển sang ngôn ngữ LAD hay FBD được. Bộ tập lênh STL được trình bày trong giáo án này đều có một chức năng như các tiếp điểm, cuộn dây, các hộp (trong LAD) hay IC số trong FBD.

Những lệnh này phải phối hợp được trạng thái các tiếp điểm để quyết định về giá trị trạng thái đầu ra hoặc giá trị logic cho phép hoặc không cho phép thực chức năng của một (hay nhiều) cuộn dây hoặc hộp. Trong lập trình lôgic thường hay sử dụng hai ngôn ngữ LAD và STL vì nó gần gũi hơn đối với chuyên ngành điện. Sau đây là những định nghĩa cần phải nắm khi bắt tay vào thiết kế một chương trình:

1. Định nghĩa về LAD: LAD là ngôn ngữ lập trình bằng đồ họa. Nhữnh thành phần cơ bản dùng trong LAD tương ứng với những thành phần cơ bản dùng trong bảng mạch rơle.

+ Tiếp điểm có hai loại: Thường đóng —|/|—

Thường hở —| |—

+ Cuộn dây (coil): —( )

+ Hộp (box): Mô tả các hàm khác nhau, nó làm việc khi có tín hiệu đưa đến hộp. Có các nhóm hộp sau: hộp các bộ định thời, hộp các bộ đếm, hộp di chuyển dữ liệu, hộp các hàm toán học, hộp trong truyền thông mạng...

+ Mạng LAD: Là mạch nối các phần tử thành một mạng hoàn thiện, các phần tử như cuộn dây hoặc các hộp phải được mắc đúng chiều. Nguồn điện có hai đường chính, một đường bên trái thể hiện dây nóng, một đường bên phải là dây trung tính (neutral) nhưng không được thể hiện trên giao diện lập trình. Một mach làm việc được khi các phần tử được mắc đúng chiều và kín mạch.

2. Định nghĩa về STL: Là phương pháp thể hiện chương trình dưới dạng tập hợp các câu lệnh. Để tạo ra một chương trình bằng STL, người lập trình cần phải hiểu rõ phương thức sử dụng 9 bit trong ngăn xếp (stack) logic của S7 200.

Ngăn xếp là một khối 9 bit chồng lên nhau từ S0÷S8, nhưng tất cả các thuật toán liên quan đến ngăn xếp đều làm việc với bit đầu tiên và bit thứ hai (S0 và S1) của ngăn xếp. giá trị logic mới có thể được gởi hoặc nối thêm vào ngăn xếp. Hai bit S0 và S1 phối hợp với nhau thì ngăn xếp được kéo lên một bit.

Ngăn xếp của S7 200 (logic stack):

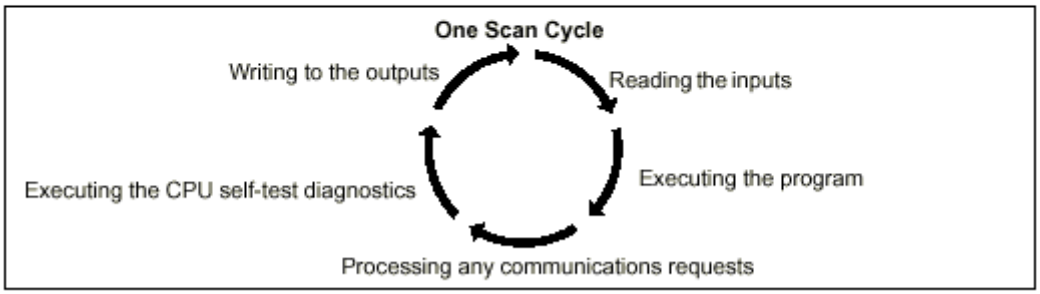
S0	Stack0 bit đầu tiên của ngăn xếp.
S1	Stack1 bit thứ hai của ngăn xếp.
S2	Stack2 bit thứ ba của ngăn xếp.
S3	Stack3 bit thứ tư của ngăn xếp.
S4	Stack4 bit thứ năm của ngăn xếp.
S5	Stack5 bit thứ sáu của ngăn xếp.
S6	Stack6 bit thứ bảy của ngăn xếp.
S7	Stack7 bit thứ tám của ngăn xếp.
S8	Stack8 bit thứ chín của ngăn xếp.

Hình 3.3: Mô tả ngăn xếp của S7 200.

3.2.Vòng quét (thực hiện chương trình) và cấu trúc của một chương trình:

PLC thực hiện chương trình theo vòng lặp. Mỗi vòng lặp được gọi là vòng quét (scan). Các giai đoạn của vòng quét:

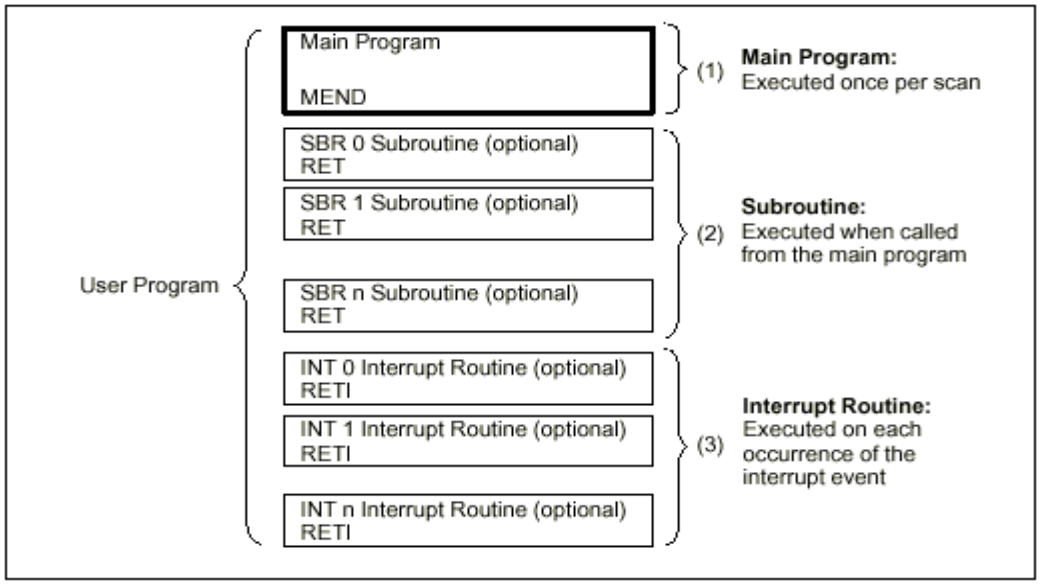
- Reading the inputs
- Executing the program
- Processing any communication requests
- Executing the CPU self-test diagnostics
- Writing to the outputs



Hình 3.1: Chu kỳ vòng quét của CPU S7-200.

Khi gặp lệnh *vào/ra tức thời* ngay lập tức hệ thống dừng tất cả mọi công việc khác, ngay cả chương trình xử lý ngắt để thực hiện chương trình này trực tiếp với cổng vào/ra.

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



Hình 3.2: Cấu trúc của chương trình S7-200

3.3.Tập lệnh S7-200:

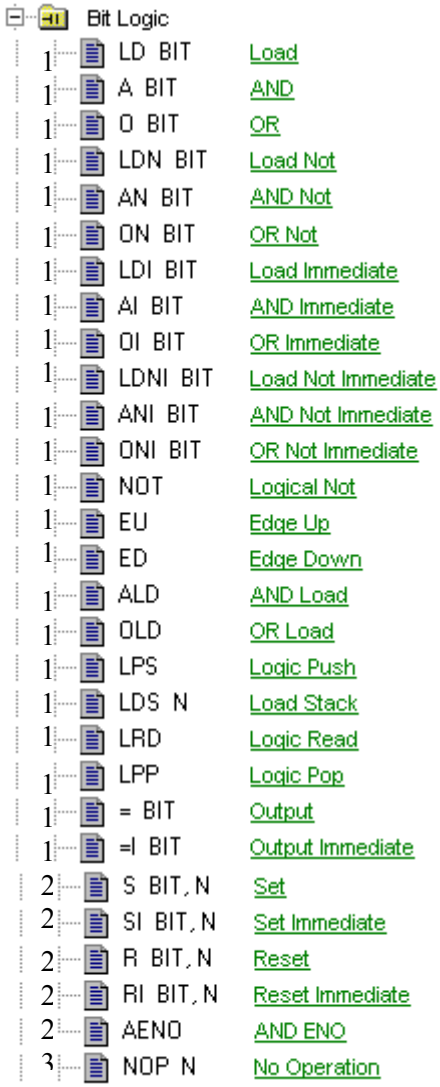
Tập lệnh của S7-200 được chia làm 3 nhóm:

1. Các lệnh mà khi thực hiện thì làm việc độc lập không phụ thuộc vào giá trị logic của bit đầu tiên trong ngăn xếp (gọi là nhóm lệnh không điều kiện).
2. Các lệnh chỉ thực hiện khi bit đầu tiên trong ngăn xếp có giá trị bằng 1 (gọi là nhóm lệnh có điều kiện).
3. Các nhãn lệnh đánh dấu vị trí trong tập lệnh (gọi là nhóm lệnh điều khiển chương trình).

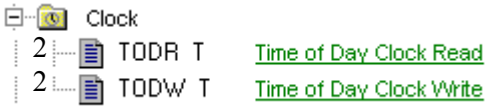
! *Các ngôn ngữ sử dụng chữ I (Immediately) để chỉ ý nghĩa tức thời.*

Instructions	Cây lệnh
Bit Logic	Tập lệnh Bit
Clock	Tập lệnh can thiệp vào thời gian hệ thống
Communications	Tập lệnh truyền thông
Compare	Tập lệnh so sánh
Convert	Tập lệnh biến đổi
Counters	Tập các bộ đếm
Floating-Point Math	Tập lệnh toán học
Integer Math	Tập lệnh toán học
Interrupt	Tập lệnh điều khiển ngắt
Logical Operations	Tập lệnh các phép tính logic biến đổi
Move	Tập lệnh di chuyển dữ liệu
Program Control	Tập lệnh điều khiển chương trình
Shift/Rotate	Tập lệnh thao tác với thanh ghi (dịch/quay vòng thanh ghi)
String	Tập lệnh làm việc với chuỗi
Table	Tập lệnh làm việc với bảng dữ liệu
Timers	Tập các bộ định thời
Subroutine	Tập lệnh gọi chương trình con và chương trình ngắt

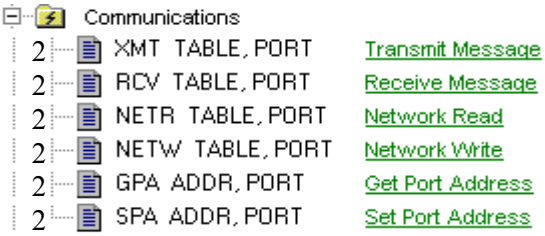
Hình 3.3: Mô tả cây lệnh với SIMATIC S7-200.



Hình 3.4: Mô tả cây lệnh bit.



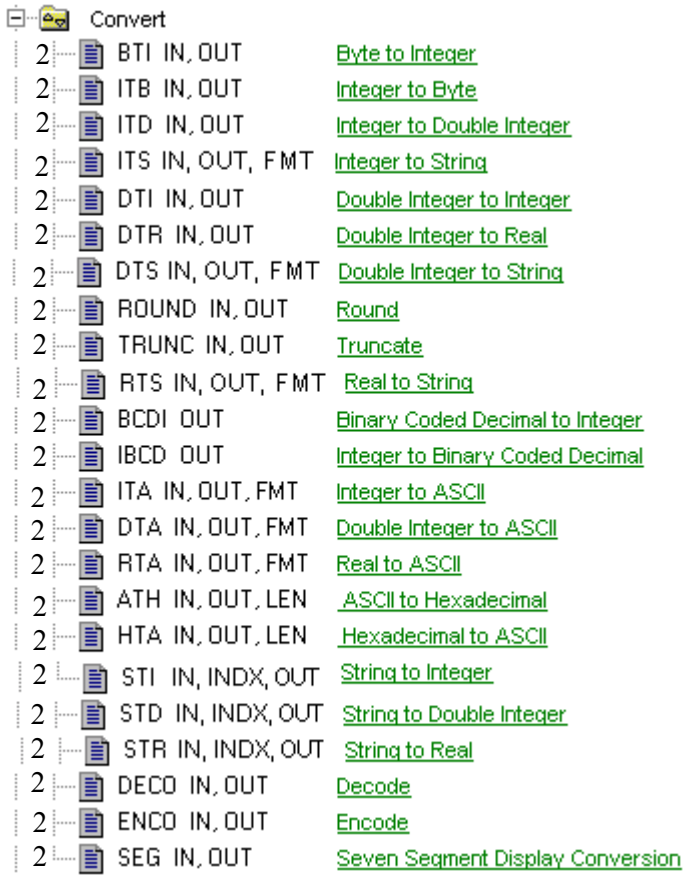
Hình 3.5: Mô tả cây lệnh can thiệp vào thời gian hệ thống.



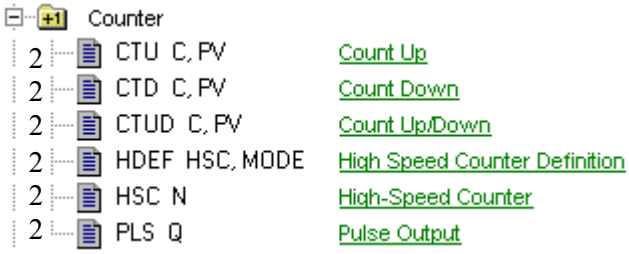
Hình 3.6: Mô tả cây lệnh truyền thông.


Compare
LDB= INI, IN2 Load Byte Equal
AB= IN1, IN2 AND Byte Equal
OB= IN1, IN2 OR Byte Equal
LDB<> IN1, IN2 Load Byte Not Equal
AB<> IN1, IN2 AND Byte Not Equal
OB<> IN1, IN2 OR Byte Not Equal
LDB>= IN1, IN2 Load Byte Greater Than or Equal
AB>= IN1, IN2 AND Byte Greater Than or Equal
OB>= IN1, IN2 OR Byte Greater Than or Equal
LDB<= IN1, IN2 Load Byte Less Than or Equal
AB<= IN1, IN2 AND Byte Less Than or Equal
OB<= IN1, IN2 OR Byte Less Than or Equal
LDB> IN1, IN2 Load Byte Greater Than
AB> IN1, IN2 AND Byte Greater Than
OB> IN1, IN2 OR Byte Greater Thane
LDB< IN1, IN2 Load Byte Less Than
AB< IN1, IN2 AND Byte Less Than
OB< IN1, IN2 OR Byte Less Than
LDW= IN1, IN2 Load Word Equal
AW= IN1, IN2 AND Word Equal
OW= IN1, IN2 OR Word Equal
LDW<> IN1, IN2 Load Word Not Equal
AW<> IN1, IN2 AND Word Not Equal
OW<> IN1, IN2 OR Word Not Equal
LDW>= IN1, IN2 Load Word Greater Than or Equal
AW>= IN1, IN2 AND Word Greater Than or Equal
OW>= IN1, IN2 OR Word Greater Than or Equal
LDW<= IN1, IN2 Load Word Less Than or Equal
AW<= IN1, IN2 AND Word Less Than or Equal
OW<= IN1, IN2 OR Word Less Than or Equal
LDW> IN1, IN2 Load Word Greater Than
AW> IN1, IN2 AND Word Greater Than
OW> IN1, IN2 OR Word Greater Than
LDW< IN1, IN2 Load Word Less Than
AW< IN1, IN2 AND Word Less Than
OW< IN1, IN2 OR Word Less Than
LDD= IN1, IN2 Load Double Word Equal
AD= IN1, IN2 AND Double Word Equal
OD= IN1, IN2 OR Double Word Equal

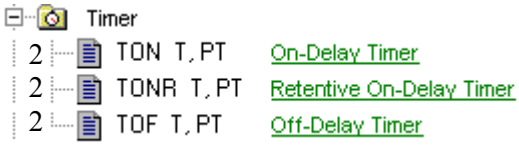
Hình 3.7: Mô tả cây lệnh so sánh



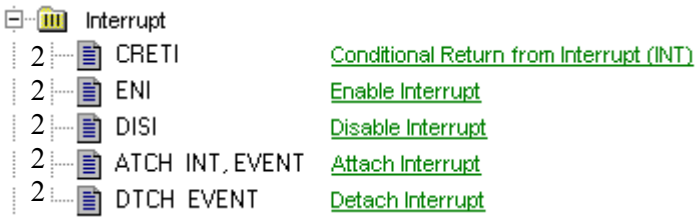
Hình 3.8: Mô tả cây lệnh biến đổi.



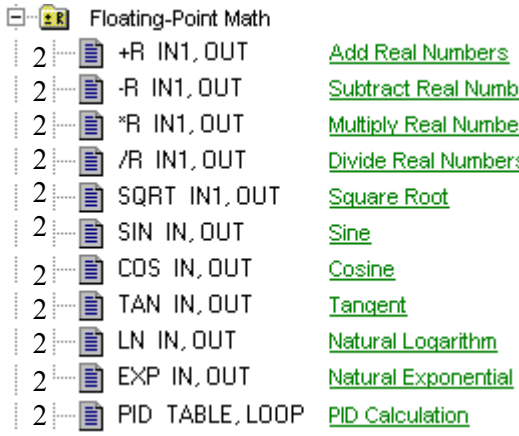
Hình 3.9: Mô tả cây lệnh các bộ đếm.



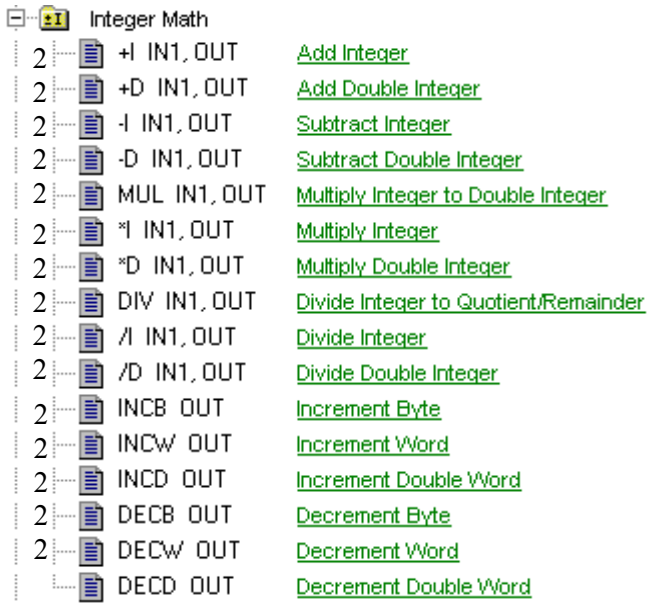
Hình 3.10: Mô tả cây lệnh các bộ định thời.



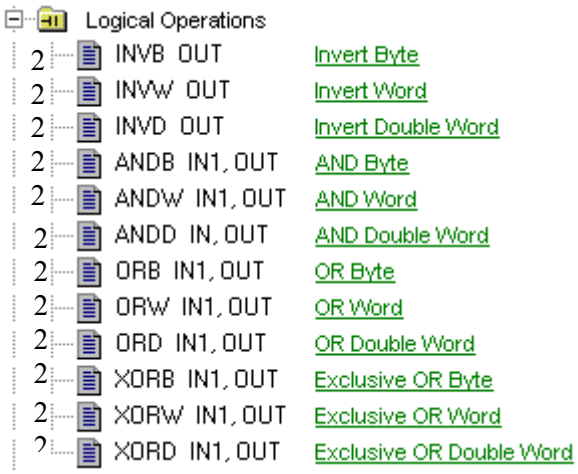
Hình 3.11: Mô tả cây lệnh điều khiển ngắt



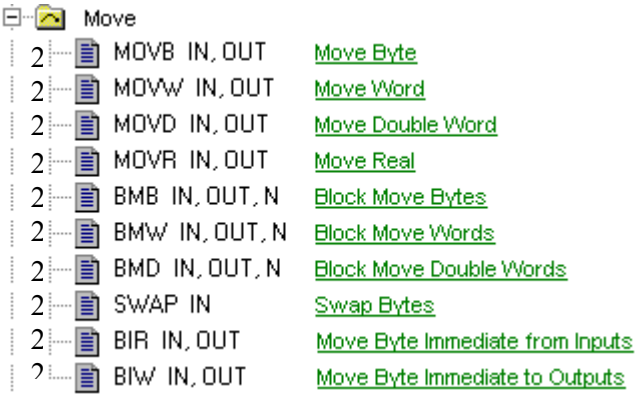
Hình 3.12: Mô tả cây lệnh học kiểu Floating-Point.



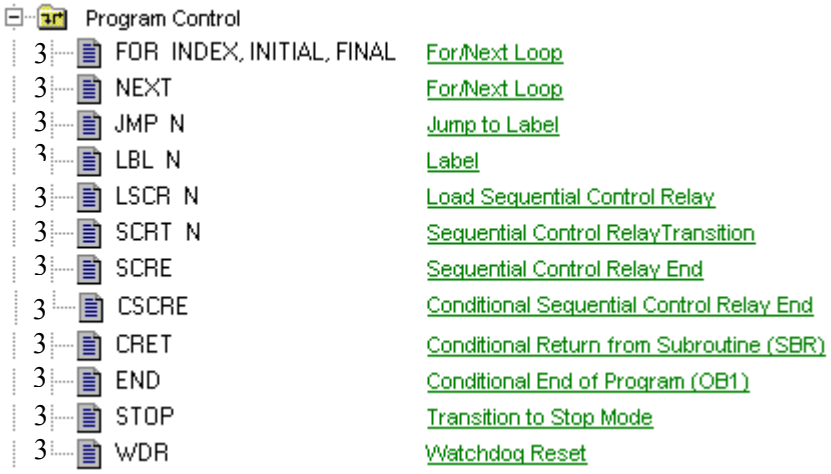
Hình 3.13: Mô tả cây lệnh toán học kiểu Integer.



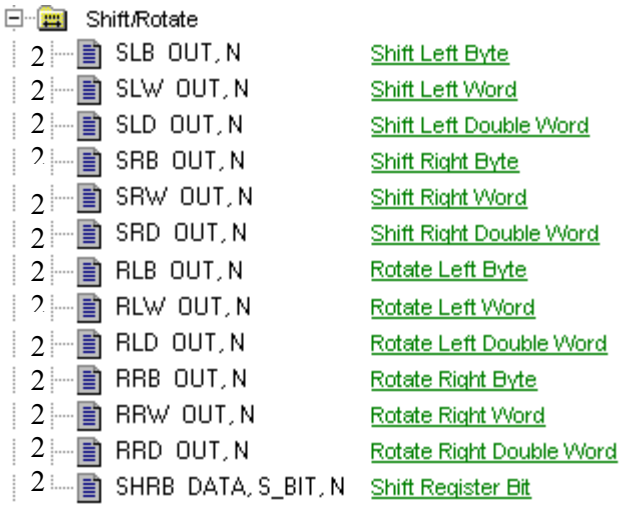
Hình 3.14: Mô tả cây lệnh phép tính logic biến đổi.



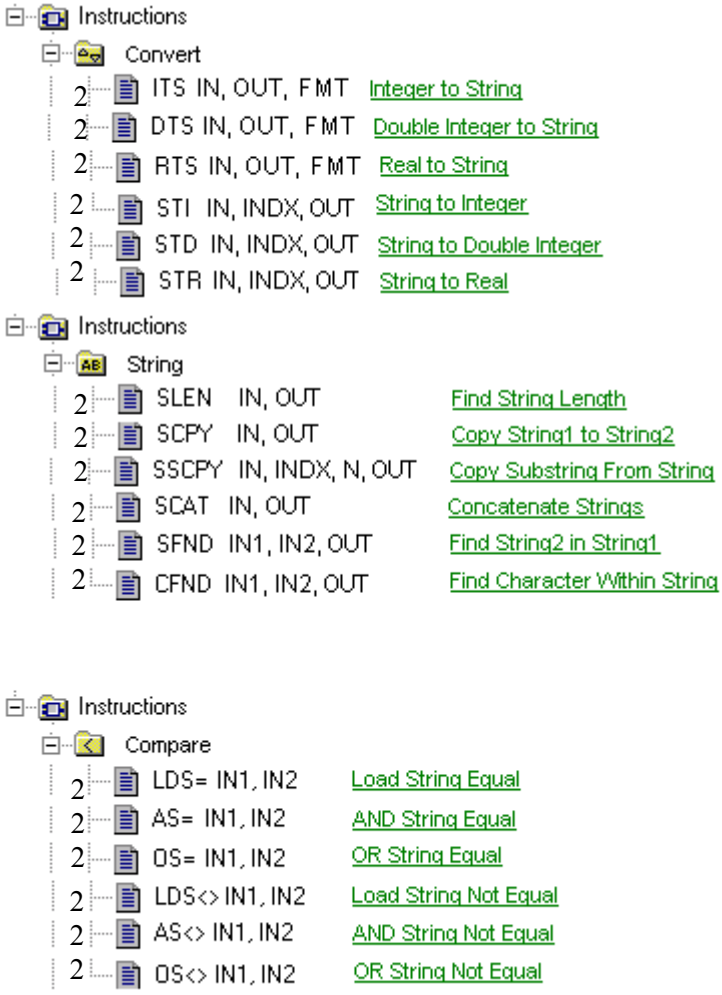
Hình 3.15: Mô tả cây lệnh di chuyển dữ liệu.



Hình 3.16: Mô tả cây lệnh điều khiển chương trình.



Hình 3.17: Mô tả cây lệnh điều khiển chương trình.



Hình 3.18: Mô tả cây lệnh làm việc với chuỗi.

Hình 3.19: Mô tả cây lệnh làm việc với bảng dữ liệu.

1_Các lệnh không điều kiện.
2_Các lệnh có điều kiện.
3_Các lệnh điều khiển chương trình.

3.4. Cú pháp và cách ứng dụng SIMATIC struction S7-200:

3.4.1. Toán hạng và giới hạn cho phép:

Bảng : Giới hạn toán hạng của CPU S7-200 series CPU 22x.

Access Method		CPU 221	CPU 222	CPU 224	CPU 224 XP	CPU 226
Bit access (byte.bit)	I	0.0 to 15.7	0.0 to 15.7	0.0 to 15.7	0.0 to 15.7	0.0 to 15.7
	Q	0.0 to 15.7	0.0 to 15.7	0.0 to 15.7	0.0 to 15.7	0.0 to 15.7
	V	0.0 to 2047.7	0.0 to 2047.7	0.0 to 8191.7	0.0 to 10239.7	0.0 to 10239.7
	M	0.0 to 31.7	0.0 to 31.7	0.0 to 31.7	0.0 to 31.7	0.0 to 31.7
	SM	0.0 to 165.7	0.0 to 299.7	0.0 to 549.7	0.0 to 549.7	0.0 to 549.7
	S	0.0 to 31.7	0.0 to 31.7	0.0 to 31.7	0.0 to 31.7	0.0 to 31.7
	T	0 to 255	0 to 255	0 to 255	0 to 255	0 to 255
	C	0 to 255	0 to 255	0 to 255	0 to 255	0 to 255
	L	0.0 to 63.7	0.0 to 63.7	0.0 to 63.7	0.0 to 63.7	0.0 to 63.7
Byte access	IB	0 to 15	0 to 15	0 to 15	0 to 15	0 to 15
	QB	0 to 15	0 to 15	0 to 15	0 to 15	0 to 15
	VB	0 to 2047	0 to 2047	0 to 8191	0 to 10239	0 to 10239
	MB	0 to 31	0 to 31	0 to 31	0 to 31	0 to 31
	SMB	0 to 165	0 to 299	0 to 549	0 to 549	0 to 549
	SB	0 to 31	0 to 31	0 to 31	0 to 31	0 to 31
	LB	0 to 63	0 to 63	0 to 63	0 to 63	0 to 63
	AC	0 to 3	0 to 3	0 to 3	0 to 255	0 to 255
	KB (Constant)	KB (Constant)	KB (Constant)	KB (Constant)	KB (Constant)	KB (Constant)
Word access	IW	0 to 14	0 to 14	0 to 14	0 to 14	0 to 14
	QW	0 to 14	0 to 14	0 to 14	0 to 14	0 to 14
	VW	0 to 2046	0 to 2046	0 to 8190	0 to 10238	0 to 10238
	MW	0 to 30	0 to 30	0 to 30	0 to 30	0 to 30
	SMW	0 to 164	0 to 298	0 to 548	0 to 548	0 to 548
	SW	0 to 30	0 to 30	0 to 30	0 to 30	0 to 30
	T	0 to 255	0 to 255	0 to 255	0 to 255	0 to 255
	C	0 to 255	0 to 255	0 to 255	0 to 255	0 to 255
	LW	0 to 62	0 to 62	0 to 62	0 to 62	0 to 62

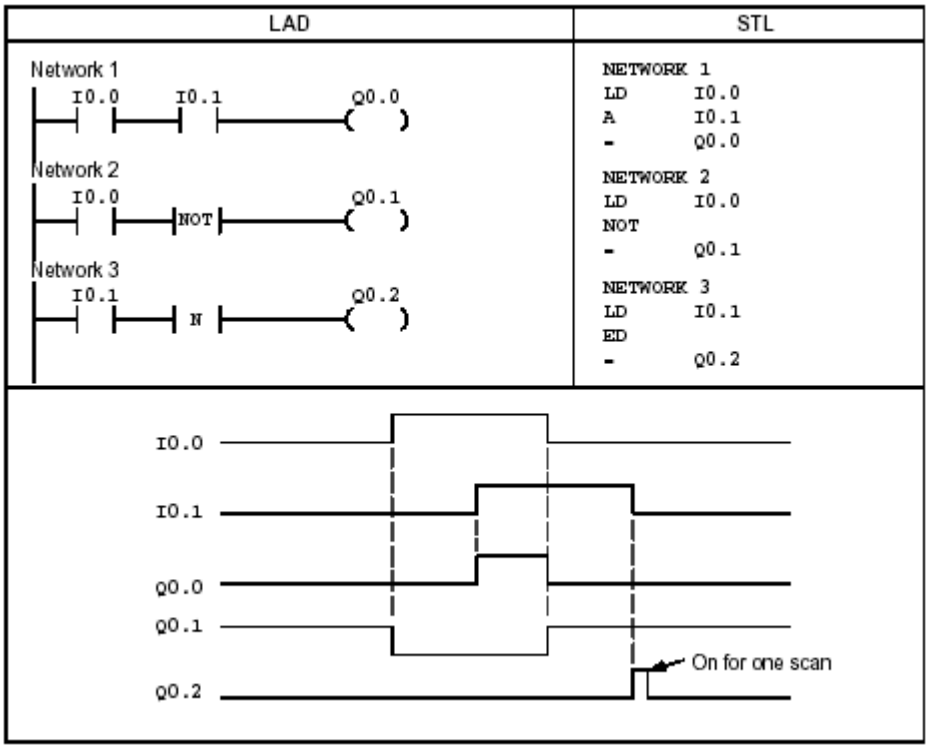
AC	0 to 3	0 to 3	0 to 3	0 to 3	0 to 3
AIW	0 to 30	0 to 30	0 to 62	0 to 62	0 to 62
AQW	0 to 30	0 to 30	0 to 62	0 to 62	0 to 62
KW (Constant)	KW (Constant)	KW (Constant)	KW (Constant)	KW (Constant)	KW (Constant)
Double word access ID	0 to 12	0 to 12	0 to 12	0 to 12	0 to 12
QD	0 to 12	0 to 12	0 to 12	0 to 12	0 to 12
VD	0 to 2044	0 to 2044	0 to 8188	0 to 10236	0 to 10236
MD	0 to 28	0 to 28	0 to 28	0 to 28	0 to 28
SMD	0 to 162	0 to 296	0 to 546	0 to 546	0 to 546
SD	0 to 28	0 to 28	0 to 28	0 to 28	0 to 28
LD	0 to 60	0 to 60	0 to 60	0 to 60	0 to 60
AC	0 to 3	0 to 3	0 to 3	0 to 3	0 to 3
HC	0 to 5	0 to 5	0 to 5	0 to 5	0 to 5
KD (Constant)	KD (Constant)	KD (Constant)	KD (Constant)	KD (Constant)	KD (Constant)

3.4.2. SIMATIC Bit Logic instruction:

Bảng : Standard contacts, Immediate contacts, Not, Positive_Negative transition.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
LD A O	bit ┤ ├	Tiếp điểm thường mở sẽ được đóng khi bit = 1	**bit:** I, Q, M, V, SM, T, C, S, L	Bool
LDN AN ON	bit ┤/├	Tiếp điểm thường đóng sẽ được mở khi bit = 1	**bit:** I, Q, M, V, SM, T, C, S, L	Bool
LDI AI OI	bit ┤I├	Tiếp điểm thường mở sẽ đóng tức thời (không phụ thuộc vào chu kỳ vòng quét)	**bit: I**	Bool
LDNI AIN OIN	bit ┤/I├	Tiếp điểm thường đóng sẽ mở tức thời (không phụ thuộc vào chu kỳ vòng quét)	**bit: I**	Bool
NOT	┤ NOT ├	Đảo giá trị logic của bit đầu tiên trong ngăn xếp	Không	Không
EU	┤ P ├	Bit đầu tiên trong ngăn xếp có giá trị bằng 1 (trong khoảng thời gian đúng bằng 1 chu kỳ vòng quét) khi phát hiện sườn lên của tín hiệu đầu vào.	**bit:** I, Q, M, V, SM, T, C, S, L	Bool

ED	⊣ N⊢	Bit đầu tiên trong ngăn xếp có giá trị bằng 1 (trong khoảng thời gian đúng bằng 1 chu kỳ vòng quét) khi phát hiện sườn xuống của tín hiệu đầu vào.	**bit:** I, Q, M, V, SM, T, C, S, L	Bool

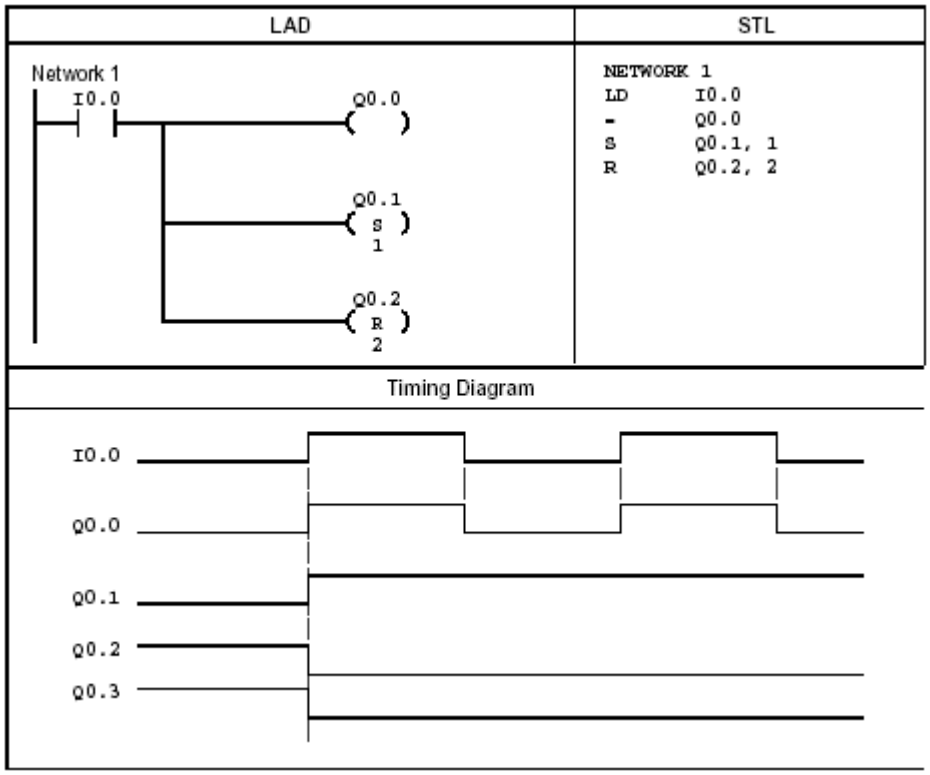


Hình 3.20: Ví dụ minh hoạ lệnh LD, NOT, ED trong chương trình LAD và STL.

1. SIMATIC Bit Logic Instruction:

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
= bit	—(bit)—	Cuộn dây đầu ra ở trạng thái ON khi có dòng điện điều khiển đi qua.	**bit:** I, Q, M, V, SM, T, C, S, L	Bool
=I bit	—(bit |)—	Cuộn dây đầu ra ở trạng thái ON tức thời (không phụ thuộc vào chu kỳ vòng quét) khi có dòng điện điều khiển đi qua.	**bit:** Q	Bool
S bit, n	—(bit S n)—	Set 1 mảng gồm n tiếp điểm, tính từ tiếp điểm "bit" (n <= 128 tiếp điểm).	**bit:** I, Q, M, V, SM, T, C, S, L n: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC,* LD	Bool

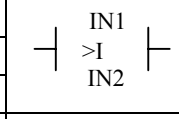
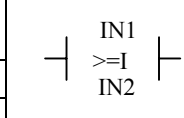
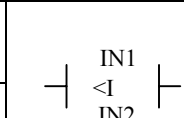
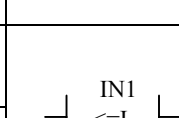
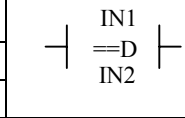
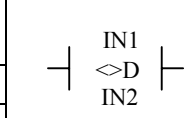
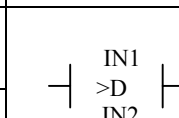
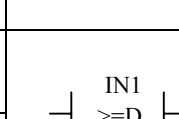
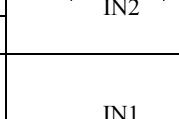
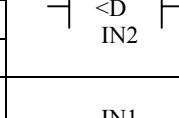
R bit, n	bit R n	Reset 1 mảng gồm n tiếp điểm, tính từ tiếp điểm "bit" (n <= 128 tiếp điểm).	**bit:** I, Q, M, V, SM, T, C, S, L n: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD	Bool
SI bit, n	bit SI n	Set tức thời 1 mảng gồm n tiếp điểm, tính từ tiếp điểm "bit" (n <= 128 tiếp điểm).	**bit:** Q n: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD	Bool
RI bit, n	bit RI n	Reset tức thời 1 mảng gồm n tiếp điểm, tính từ tiếp điểm "bit" (n <= 128 tiếp điểm).	**bit:** Q n: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD	Bool
NOP	n NOP	Lệnh rỗng, không hoạt động n lần.	n: 0 ÷255	Byte



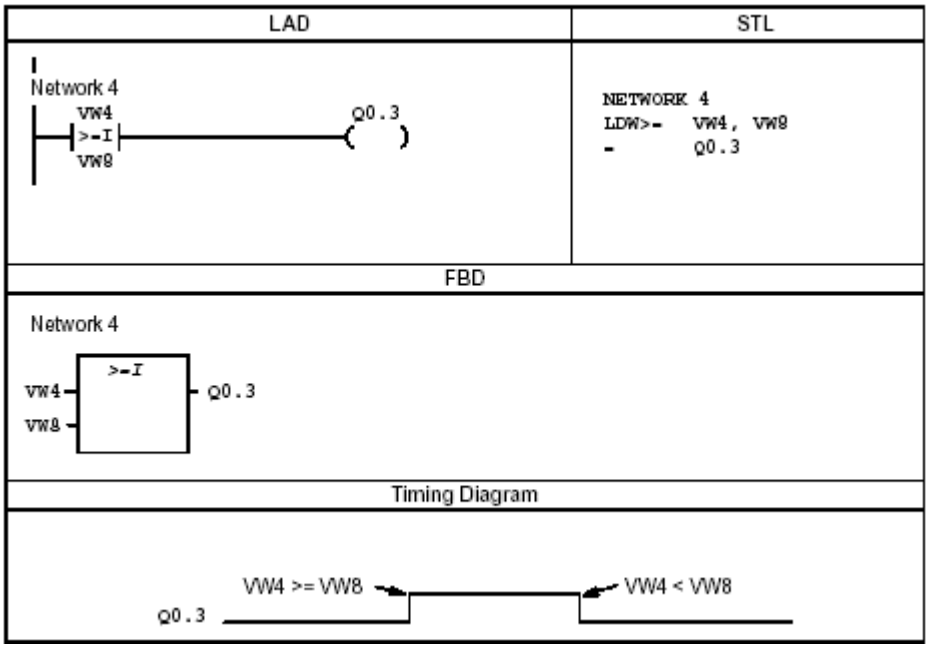
Hình 3.21: Ví dụ minh hoạ lệnh =, S, R trong chương trình LAD và STL.

2. SIMATIC Copare Byte Instructions:

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
COPARE BYTE				
LDB= AB= OB=	IN1 ==B IN2	Lệnh so sánh giá trị của hai byte IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1= IN2 là đúng.	IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD	Byte
LDB<> AB<> OB<>	IN1 <>B IN2	Lệnh so sánh giá trị của hai byte IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1<> IN2 là đúng.	IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD	Byte
LDB< AB< OB<	IN1 <B IN2	Lệnh so sánh giá trị của hai byte IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1< IN2 là đúng.	IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD	Byte
LDB<= AB<= OB<=	IN1 <=B IN2	Lệnh so sánh giá trị của hai byte IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1<= IN2 là đúng.	IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD	Byte
LDB> AB> OB>	IN1 >B IN2	Lệnh so sánh giá trị của hai byte IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1> IN2 là đúng.	IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD	Byte
LDB>= AB>= OB>=	IN1 >=B IN2	Lệnh so sánh giá trị của hai byte IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1>= IN2 là đúng.	IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD	Byte
COPARE WORD (COPARE INTEGER)				
LDW= AW= OW=	IN1 ==I IN2	Lệnh so sánh giá trị của hai Word IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1= IN2 là đúng.	IW, QW, MW, VW, SMW, SW, LW, AC, Constant, ∗VD, ∗AC, ∗LD	Word
LDW<> AW<> OW<>	IN1 <>I IN2	Lệnh so sánh giá trị của hai Word IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1<> IN2 là đúng.	IW, QW, MW, VW, SMW, SW, LW, AC, Constant, ∗VD, ∗AC, ∗LD	Word

LDW> AW> OW>		Lệnh so sánh giá trị của hai Word IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 > IN2 là đúng.	IW, QW, MW, VW, SMW, SW, LW, AC, Constant, ∗VD, ∗AC, ∗LD	Word
LDW>= AW>= OW>=		Lệnh so sánh giá trị của hai Word IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 >= IN2 là đúng.	IW, QW, MW, VW, SMW, SW, LW, AC, Constant, ∗VD, ∗AC, ∗LD	Word
LDW< AW< OW<		Lệnh so sánh giá trị của hai Word IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 < IN2 là đúng.	IW, QW, MW, VW, SMW, SW, LW, AC, Constant, ∗VD, ∗AC, ∗LD	Word
LDW<= AW<= OW<=		Lệnh so sánh giá trị của hai Word IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 <= IN2 là đúng.	IW, QW, MW, VW, SMW, SW, LW, AC, Constant, ∗VD, ∗AC, ∗LD	Word
COPARE DOUBLEWORD				
LDDW= ADW= ODW=		Lệnh so sánh giá trị của hai DoubleWord IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 = IN2 là đúng.	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, ∗VD, ∗AC, ∗LD	Double Word
LDDW<> ADW<> ODW<>		Lệnh so sánh giá trị của hai DoubleWord IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 <> IN2 là đúng.	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, ∗VD, ∗AC, ∗LD	Double Word
LDDW> ADW> ODW>		Lệnh so sánh giá trị của hai DoubleWord IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 > IN2 là đúng.	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, ∗VD, ∗AC, ∗LD	Double Word
LDDW>= ADW>= ODW>=		Lệnh so sánh giá trị của hai DoubleWord IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 >= IN2 là đúng.	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, ∗VD, ∗AC, ∗LD	Double Word
LDDW< ADW< ODW<		Lệnh so sánh giá trị của hai DoubleWord IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 < IN2 là đúng.	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, ∗VD, ∗AC, ∗LD	Double Word
LDDW<= ADW<= ODW<=		Lệnh so sánh giá trị của hai DoubleWord IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 <= IN2 là đúng.	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, ∗VD, ∗AC, ∗LD	Double Word
COPARE REAL				
LDR=		Lệnh so sánh giá trị của hai số	ID, QD, MD, VD,	Real

AR= OR=	IN1 ==R IN2	thực IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 = IN2 là đúng.	SMD, SD, LD, AC, Constant, *VD, *AC, *LD	
LDR<> AR<> OR<>	IN1 <>R IN2	Lệnh so sánh giá trị của hai số thực IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 <> IN2 là đúng	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, *VD, *AC, *LD	Real
LDR> AR> OR>	IN1 >R IN2	Lệnh so sánh giá trị của hai số thực IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 > IN2 là đúng	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, *VD, *AC, *LD	Real
LDR>= AR>= OR>=	IN1 >=R IN2	Lệnh so sánh giá trị của hai số thực IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 >= IN2 là đúng	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, *VD, *AC, *LD	Real
LDR< AR< OR<	IN1 <R IN2	Lệnh so sánh giá trị của hai số thực IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 < IN2 là đúng	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, *VD, *AC, *LD	Real
LDR<= AR<= OR<=	IN1 <=R IN2	Lệnh so sánh giá trị của hai số thực IN1 và IN2. Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1 <= IN2 là đúng	ID, QD, MD, VD, SMD, SD, LD, AC, Constant, *VD, *AC, *LD	Real



Hình 3.22: Ví dụ minh hoạ lệnh so sánh trong chương trình LAD, FBD và STL.

3. SIMATIC Timer Instructions:

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
On_Delay_Timer (TON)				
TON Txxx, PT TON Txxx, PT	Txxx TON EN PT Txxx TONR EN PT	Đây là lệnh đếm thời gian hoạt khi tín hiệu EN là ON. Khi giá trị dếm tức thời trong thanh ghi CT >= giá trị đặt trước trong thanh ghi PT thì bit trạng thái Txxx của bộ Timer là ON. Gía trị đếm tức thời trong thanh ghi CT = 0 và bit trạng thái về off khi tín hiệu ở đầu vào là off. Ngược lại với bộ TON, thanh ghi CV và bit trạng thái vẫn giữ nguyên trừ khi có lệnh Reset bộ TONR. Ngoài ra có thể sử dụng lệnh Reset để xoá thanh ghi tức thời cũng như bit trạng thái của bộ TON. Ta có thể sử dụng toán hạng Word (INT) tương ứng với lệnh INT hay toán hạng tương ứng với bit trạng thái.	Txxx: Constant IN : power flow PT: IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, Constant, ∗VD, ∗AC, ∗LD	word bool INT
TOF Txxx, PT	Txxx TOF EN PT	Khi tín hiệu đầu vào EN = 1 bộ TOF không hoạt động. chỉ hoạt động khi có sườn xuống của tín hiệu đầu vào. Bit trạng thái được bật lên ON khi		

		CV = PT. Reset TOF (cả CV và bit trạng thái) bằng cách cung cấp tín hiệu vào đầu vào EN.		

Bảng : Số Timer và độ phân giải.

Timer Type	Resolution in milliseconds (ms)	Maximum Value in seconds (s)	Timer Number
TONR (retentive)	1 ms	32.767 s (0.546 min.)	T0, T64
	10 ms	327.67 s (0.546 min.)	T1 to T4, T65 to T68
	100 ms	3276.7 s (0.546 min.)	T5 to T31, T69 to T95
TON, TOF (non-retentive)	1 ms	32.767 s (0.546 min.)	T32, T96
	10 ms	327.67 s (0.546 min.)	T33 to T36, T97 to T100
	100 ms	3276.7 s (0.546 min.)	T37 to T63, T101 to T255

Note: Không thể cùng một lúc sử dụng cả 2 bộ TON và TOF cho cùng 1 địa chỉ (ví dụ T37).

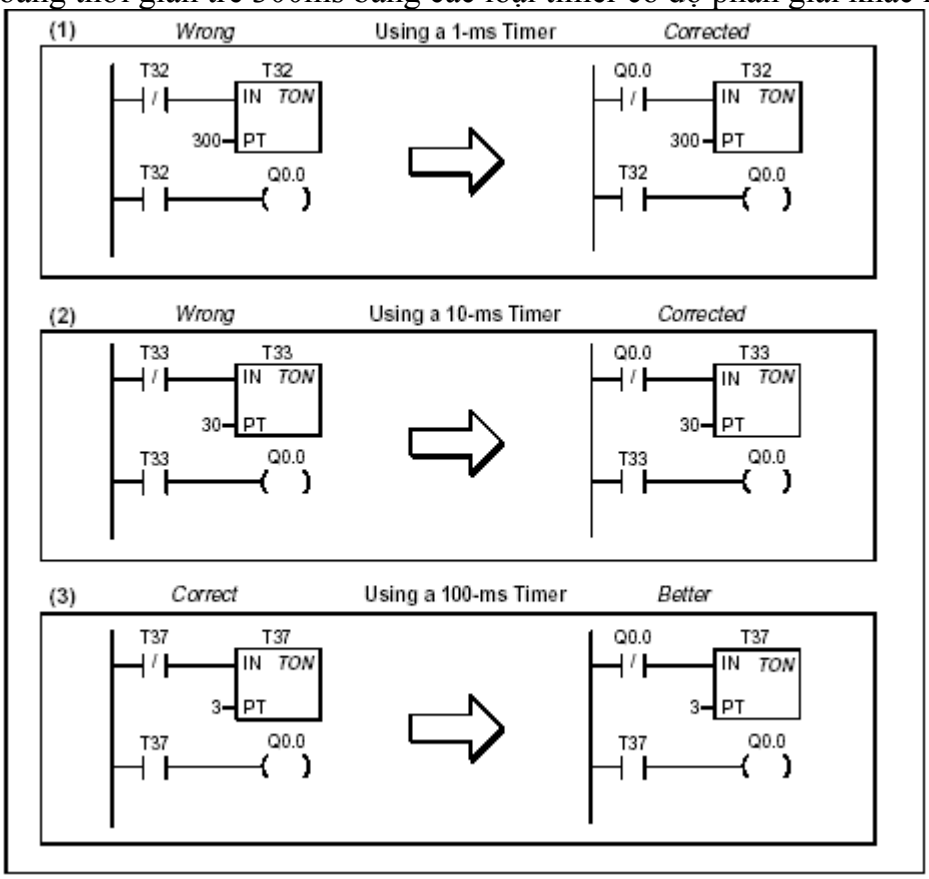
Bảng : Gía trị đặt tối đa cho từng loại và trạng thái làm việc của các loại Timer.

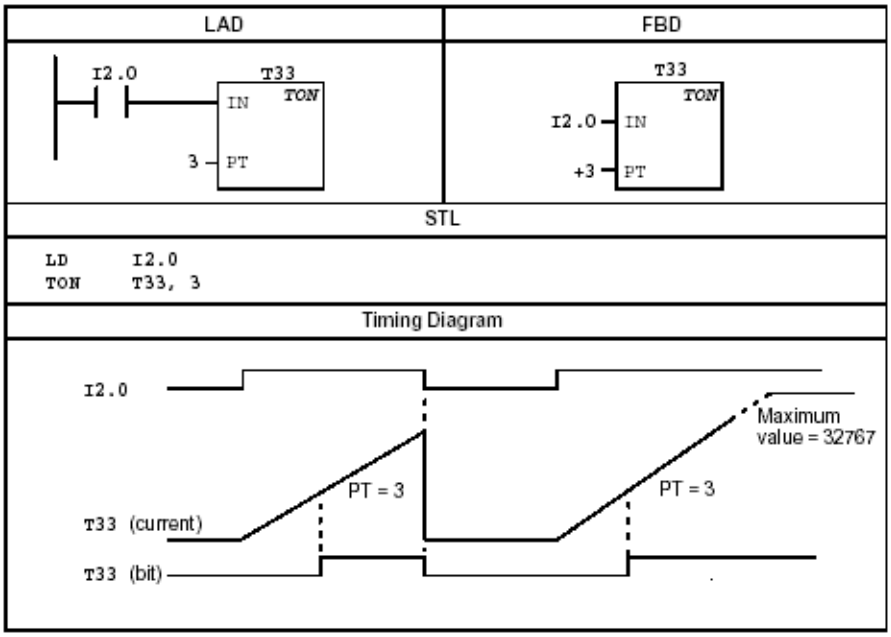
Timer Type	Current >= Preset	Enabling Input ON	Enabling Input OFF	Power Cycle/ First Scan
TON	Timer bit ON, Current continues counting to 32,767	Current value counts time	Timer bit OFF, Current value = 0	Timer bit OFF, Current value = 0
TONR	Timer bit ON, Current continues counting to 32,767	Current value counts time	Timer bit and current value maintain last state	Timer bit OFF, Current value may be maintained[1]
TOF	Timer bit OFF, Current = Preset, stops counting	Timer bit ON, Current value = 0	Timer counts after ON to OFF transition	Timer bit OFF, Current value = 0

[1] The retentive timer current value can be selected for retention through a power cycle. See Section 5.3 for information about memory retention for the S7-200 CPU.

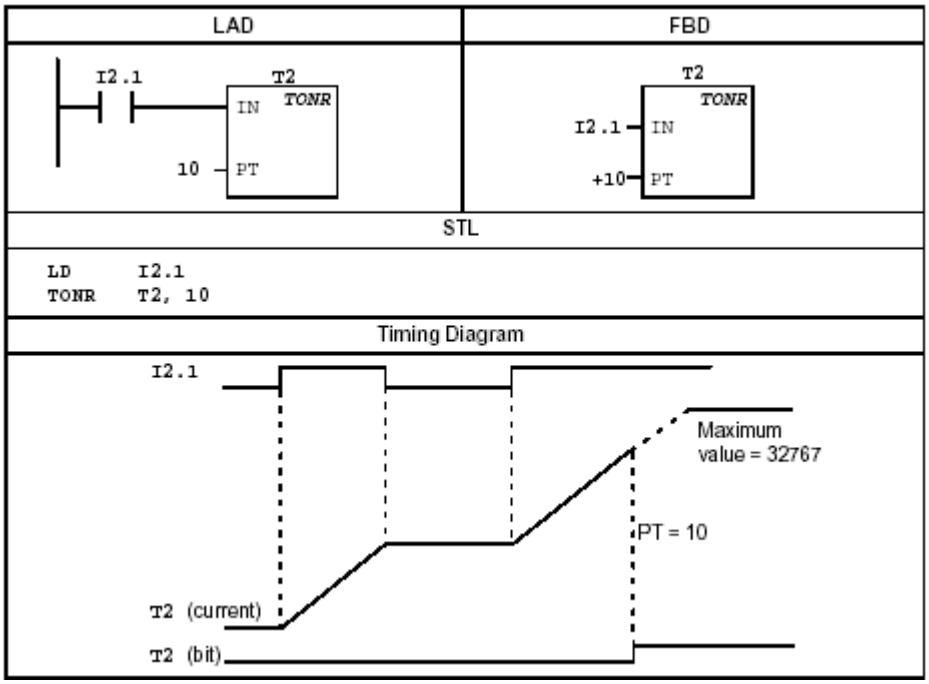
Việc sử dụng tiếp điểm thường đóng Q0.0 bên dưới để đảm làm tín hiệu đầu vào cho Timer đảm bảo cho Q0.0 sẽ có giá trị logic bằng 1 trong một vòng quét ở mỗi thời điểm mà giá trị đếm tức thời của bộ Timer đạt giá trị đặt trước PT.

Tạo khoảng thời gian trễ 300ms bằng các loại timer có độ phân giải khác nhau

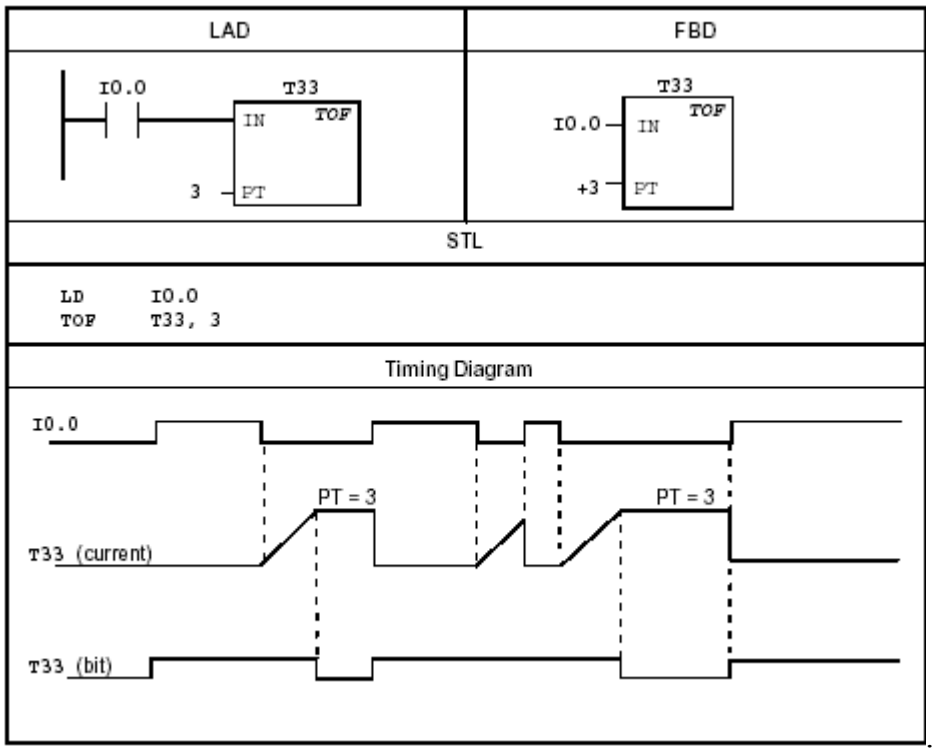




Hình 23: Ví dụ cách sử dụng bộ TON.



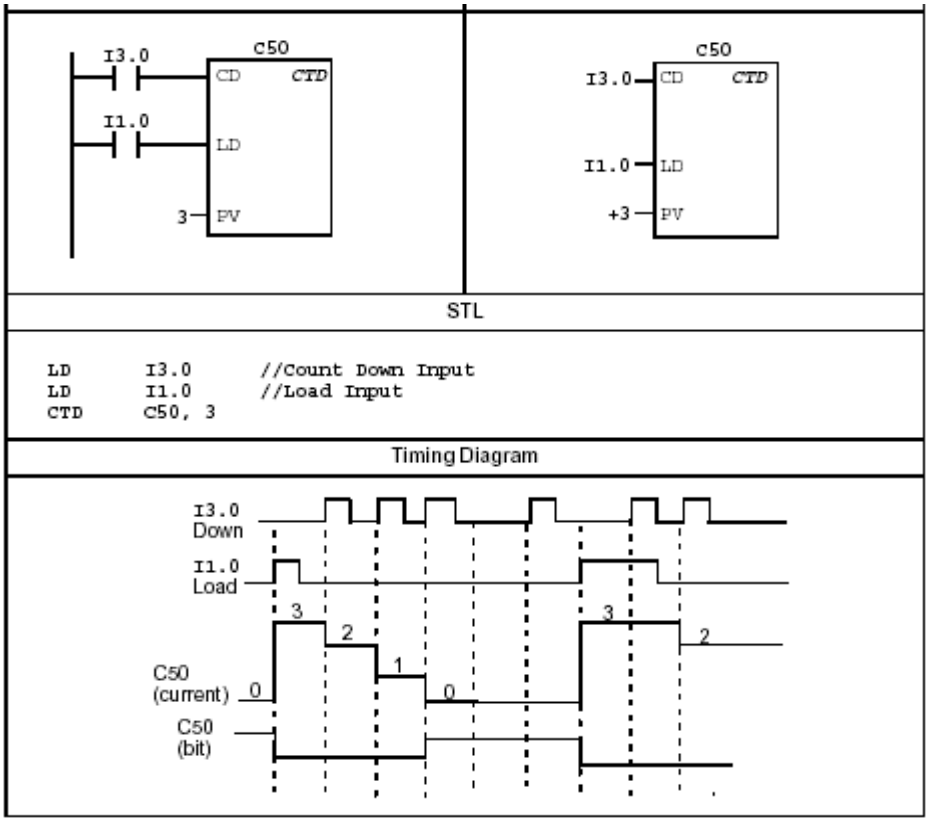
Hình 24: Ví dụ cách sử dụng bộ TONR



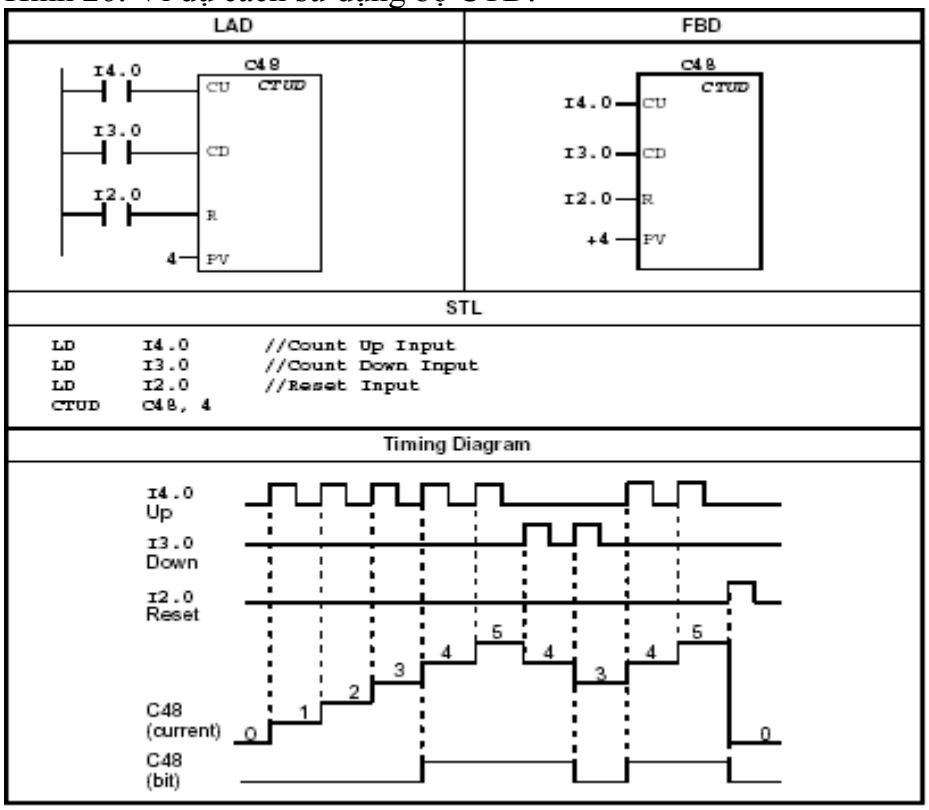
Hình 25: Ví dụ cách sử dụng bộ TOF

4. SIMATIC Counter Instructions (*Count Up, Count Up Down, Count Down*):

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
CTU Cxxx, PV	Cxxx CTU CU R PV	Khai báo bộ đếm tiến theo sườn lên của tín hiệu đầu vào CU. Khi gí trị đếm tức thời C-Word lớn hơn hoặc bằng giá trị đặt trước PV, htì bit trạng thái Cxxx có giá trị bằng 1. Bộ đếm được Reset khi R có giá trị logic bằng 1. Bộ đếm ngừng đếm khi giá trị đếm đạt giá trị cực đại 32767.	Cxxx: Constant	word
			EU, R : power flow.	bool
			PT: IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT
CTUD Cxxx, PV	Cxxx CTUD CU CD R PV	Khai báo bộ đếm tiến/lùi; đếm tiến theo sườn lên của tín hiệu đầu vào CU, đếm lùi theo sườn lên của tín hiệu đầu vào CD. Khi gí trị đếm tức thời C-Word lớn hơn hoặc bằng giá trị đặt trước PV, htì bit trạng thái Cxxx có giá trị bằng 1. Bộ đếm được Reset khi R có giá trị logic bằng 1. Bộ đếm ngừng đếm tiến khi giá trị đếm đạt giá trị cực đại 32767. Bộ đếm ngừng đếm lùi khi giá trị đếm đạt giá trị cực đại -32767. CTUD reset khi đầu vào R có giá trị logic bằng 1.	Cxxx: Constant	word
			EU, ED, R : power flow.	bool
			PT: IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT
CTD Cxxx, PV	Cxxx CTD CD LD PV	Khai báo bộ đếm lùi theo sườn lên của tín hiệu đầu vào C. Khi gí trị đếm tức thời C-Word lớn hơn hoặc bằng giá trị đặt trước PV, htì bit trạng thái Cxxx có giá trị bằng 1. Bộ đếm được Reset khi R có giá trị logic bằng 1. Bộ đếm ngừng đếm khi giá trị đếm đạt giá trị cực đại 32767.	Cxxx: Constant	word
			CD, LD : power flow.	bool
			PT: IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT

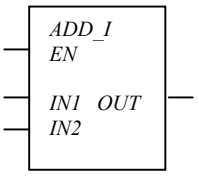
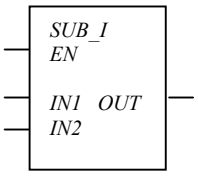
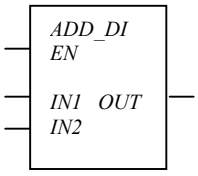
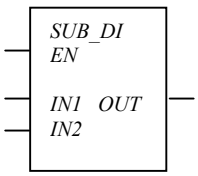
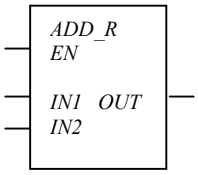


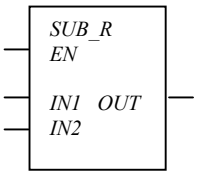
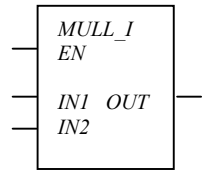
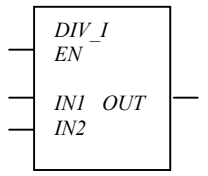
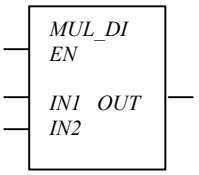
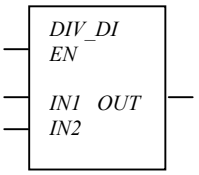
Hình 26: Ví dụ cách sử dụng bộ CTD.

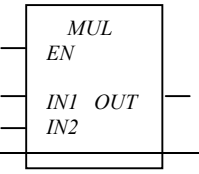


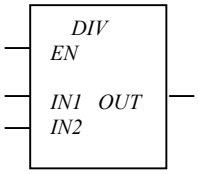
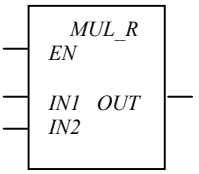
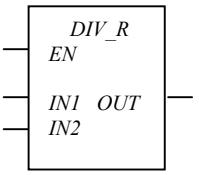
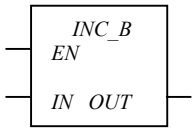
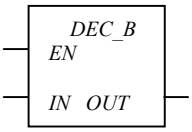
Hình 27: Ví dụ cách sử dụng bộ CTUD.

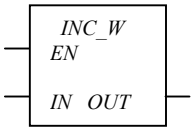
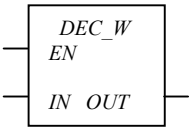
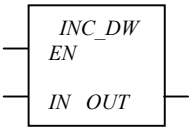
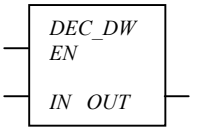
5. SIMATIC Integer Math Instructions:

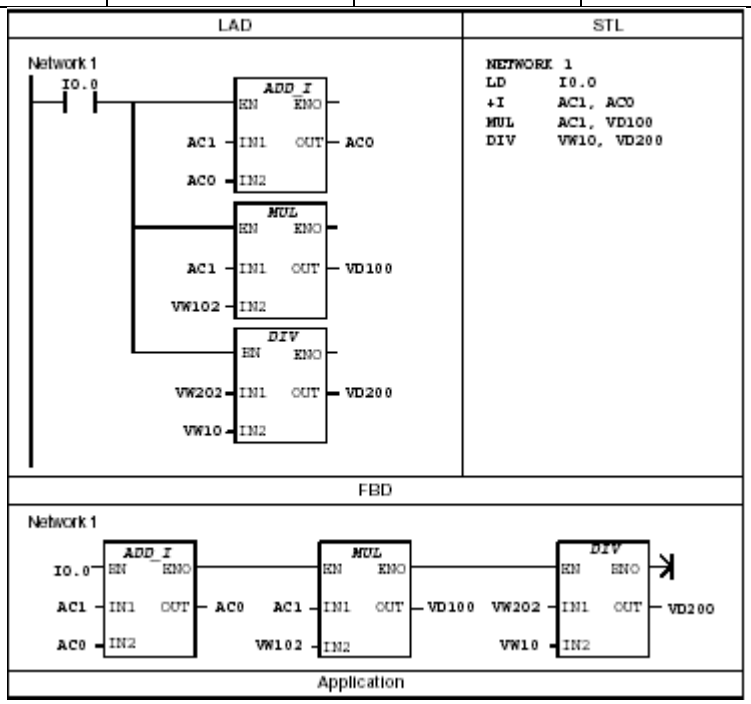
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Add Integer and Subtract Integer				
MOVW IN1, OUT +I IN2, OUT hoặc +I IN1, IN2		Lệnh cộng hai số nguyên 16 bit IN1 + IN2 kết quả chứa trong OUT (16 bit)	**IN1, IN2:** IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, Constant, ∗VD, ∗AC, ∗LD **OUT:** IW, QW, MW, SMW, VW, LW, SW, T, C, AC, ∗VD, ∗AC, ∗LD	INT
MOVW IN1, OUT -I IN2, OUT hoặc -I IN1, IN2		Lệnh trừ hai số nguyên 16 bit IN1- IN2 kết quả chứa trong OUT (16 bit)		
Add Double Integer and Subtract Double Integer				
MOVD IN1, OUT +D IN2, OUT hoặc +D IN1, IN2		Lệnh cộng hai số nguyên 32 bit IN1 + IN2 kết quả chứa trong OUT (32 bit)	**IN1, IN2:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, ∗VD, ∗AC, ∗LD **OUT:** ID, QD, MD, VD, SMD, SD, LD,AC, ∗VD, ∗AC, ∗LD	DINT
MOVD IN1, OUT -D IN2, OUT hoặc -D IN1, IN2		Lệnh trừ hai số nguyên 32 bit IN1 - IN2 kết quả chứa trong OUT (32 bit)		
Add Real and Subtract Real				
MOVR IN1, OUT +R IN2, OUT hoặc +R IN1, IN2		Lệnh cộng hai số thực 32 bit IN1 + IN2 kết quả chứa trong OUT (32 bit)	**IN1, IN2:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, ∗VD, ∗AC, ∗LD **OUT:** ID, QD, MD, VD, SMD, SD, LD,AC, ∗VD, ∗AC, ∗LD	Real
MOVR IN1, OUT		Lệnh trù hai số	**IN1, IN2:** ID, QD, MD,	Real

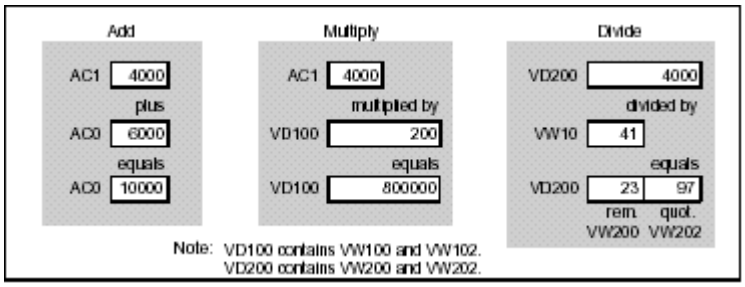
-R IN2, OUT hoặc -R IN1, IN2		thực 32 bit IN1 + IN2 kết quả chứa trong OUT (32 bit)	VD, SMD, SD, LD, HC,AC, Constant, ∗VD, ∗AC, ∗LD **OUT:** ID, QD, MD, VD, SMD, SD, LD,AC, ∗VD, ∗AC, ∗LD	
Miltiply Integer and Divide Integer				
MOVW IN1, OUT *I IN2, OUT hoặc *I IN1, IN2		Lệnh nhân hai số nguyên 16 bit IN1*IN2 kết quả chứa trong OUT (16 bit)	**IN1, IN2:** IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, Constant, ∗VD, ∗AC, ∗LD **OUT:** IW, QW, MW, SMW, VW, LW, SW, T, C, AC, ∗VD, ∗AC, ∗LD	INT
MOVW IN1, UT /I IN2, OUT hoặc /I IN1, IN2		Lệnh chia hai số nguyên 16 bit IN1/IN2 kết quả chứa trong OUT (16 bit)		
Miltiply Double Integer and Divide Double Integer				
MOVD IN1, OUT *D IN2, OUT hoặc *D IN1, IN2		Lệnh nhân hai số nguyên 32 bit IN1*IN2 kết quả chứa trong OUT (32 bit)	**IN1, IN2:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, ∗VD, ∗AC, ∗LD **OUT:** ID, QD, MD, VD, SMD, SD, LD,AC, ∗VD, ∗AC, ∗LD	DINT
MOVD IN1, OUT /D IN2, OUT hoặc /D IN1, IN2		Lệnh chia hai số nguyên 32 bit IN1/IN2 kết quả chứa trong OUT (32 bit)		
Multiply Integer to Double Double Integer and Divide Integer to Double Double Integer				

MOVW IN1, OUT
MUL IN2, OUT
hoặc
MUL IN1, IN2
		Lệnh nhân hai số nguyên 16 bit IN1*IN2 kết quả chứa trong OUT (32	**IN1, IN2:** IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, Constant, ∗VD, ∗AC, ∗LD	INT

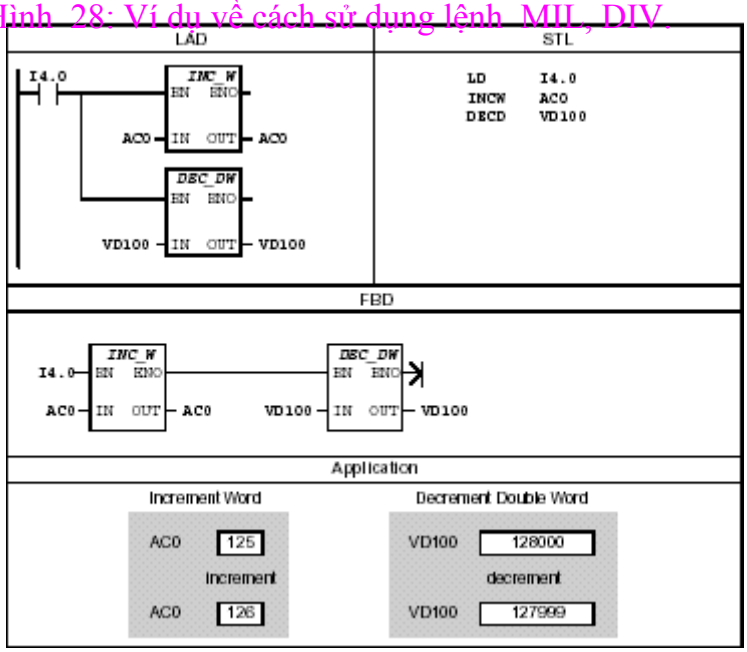
		trong OUT (32 bit)	**OUT:** ID, QD, MD, VD, SMD, SD, LD,AC, *VD, *AC, *LD	DINT
MOVW IN1, OUT DIV IN2, OUT hoặc DIV IN1, IN2		Lệnh chia hai số nguyên 16 bit IN1*IN2 kết quả chứa trong OUT (32 bit)	**IN1, IN2:** IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT
			OUT: ID, QD, MD, VD, SMD, SD, LD,AC, *VD, *AC, *LD	DINT
Miltiply Real and Divide Real				
MOVR IN1, OUT *R IN2, OUT hoặc *R IN1, IN2		Lệnh nhân hai số thực 32 bit IN1*IN2 kết quả chứa trong OUT (32 bit)	**IN1, IN2:** IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, MD, VD, SMD, SD, LD,AC, *VD, *AC, *LD	Real
MOVR IN1, OUT /R IN2, OUT hoặc /R IN1, IN2		Lệnh chia hai số thực 32 bit IN1/IN2 kết quả chứa trong OUT (32 bit)		
Những lệnh này làm đơn giản hoá các vòng điều khiển bên trong chương trình hoặc là các quá trình lặp. Trong LAD hay trong STL các lệnh tăng hoặc giảm đều làm việc với các toán hạng có kiểu Byte, từ đơn, kiểu từ kép theo nguyên tắc cộng hoặc trừ toán hạng với số nguyên 1. Để tiết kiệm ô nhớ ta có thể sủ dụng đầu vào đồng thời làm đầu ra. **Increment Byte and Decrement Byte**				
INCB OUT		Mô tả ở trên.	**IN:** IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC,* LD **OUT:**IB, QB, MB, VB, SMB, SB, LB, AC, *VD, *AC,* LD	Byte
DECB OUT				

Increment Word and Decrement Word				
INCW OUT		Mô tả ở trên	**IN:** IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, Constant, *VD, *AC, *LD **OUT:** IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, *VD, *AC, *LD	INT
DECW OUT				
Increment Double Word and Decrement Double Word				
INCD OUT		Mô tả ở trên	**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD	DINT
DECB OUT				

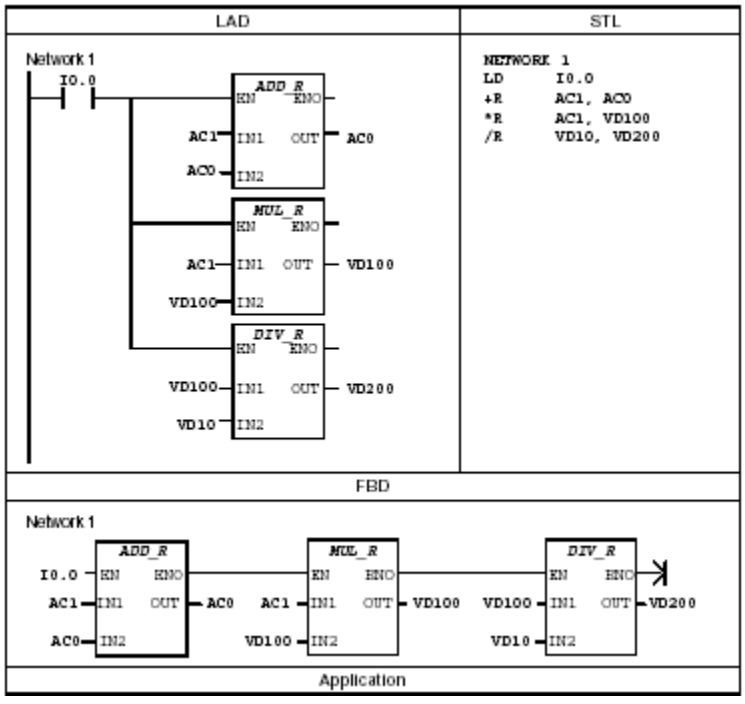


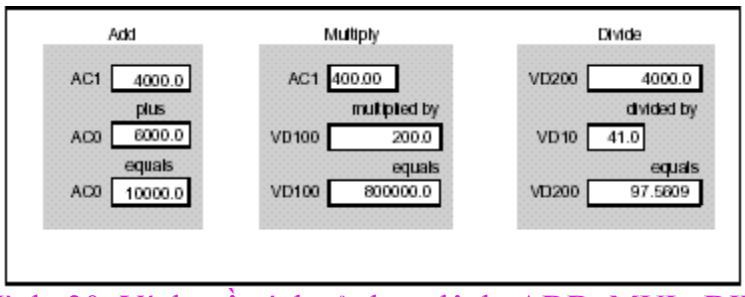


Hình 28: Ví dụ về cách sử dụng lệnh MUL, DIV.



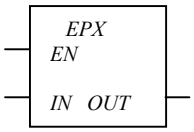
Hình 29: Ví dụ về cách sử dụng lệnh INC, DEC

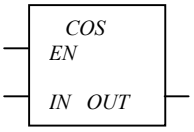
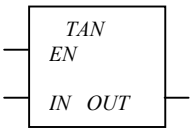
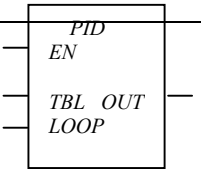




Hình 30: Ví dụ về cách sử dụng lệnh ADD, MUL, DIV với số thực.

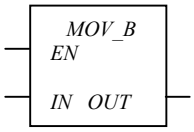
6. SIMATIC Numerical Function Instructions:

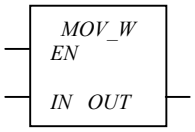
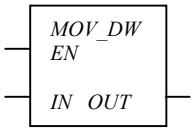
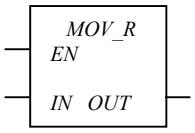
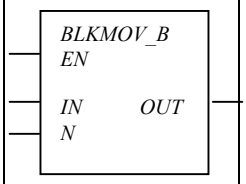
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Square Root				
SQRT IN, OUT		Lệnh thực hiện phép lấy căn bậc hai của số thực 32 bit. Kết quả cũng là số 32 bit được ghi vào từ kép OUT.	**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD	Real
Natural Logarithm (logarit tự nhiên)				
LN IN, OUT		Lệnh Natural Logarithm thực hiện phép logirit tự nhiên của số thực 32 bit, Kết quả được lưu vào từ kép OUT. Lệnh này cũng được sử dụng để thực hiện phép logarit cơ số 10 từ phép lấy logarit tự nhiên.	**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD	Real
Natural Exponential (phép lấy tự nhiên)				
EPX IN, OUT			**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD	Real

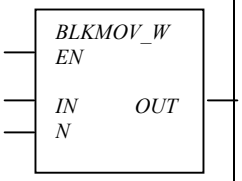
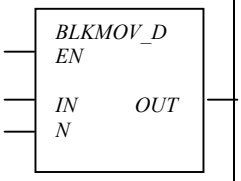
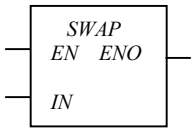
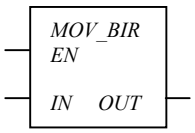
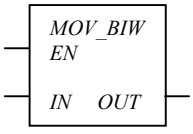
Sine, Cosine and Tangent				
SIN IN, OUT		Lệnh Sine, Cosine và Tangent định giá trị hàm lượng giác của góc IN(số thực 32 bit). Kết quả được lưu vào doubleword OUT. Với điều kiện: IN tính bằng radian, nếu là độ thì phải thực hiện phép chuyển từ độ sang radian bằng cách thực hiện lệnh MUL_R để nhân giá trị IN Với 1.745329E-2 (π/180)	IN: ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, *VD, *AC, *LD OUT: ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD	Real
COS IN, OUT				
TAN IN, OUT				
PID TBL, LOOP		Lệnh thực hiện tính toán vòng lặp, với số thứ tự là LOOP (0<=LOOP<=7) và bảng tham chiếu của quá trình là TBL. ! Trrước khi thực hiện quá trình tính toán vòng lặp PID	**TBL:** VB	BYTE
			LOOP: Constant (0 ÷7)	BYTE

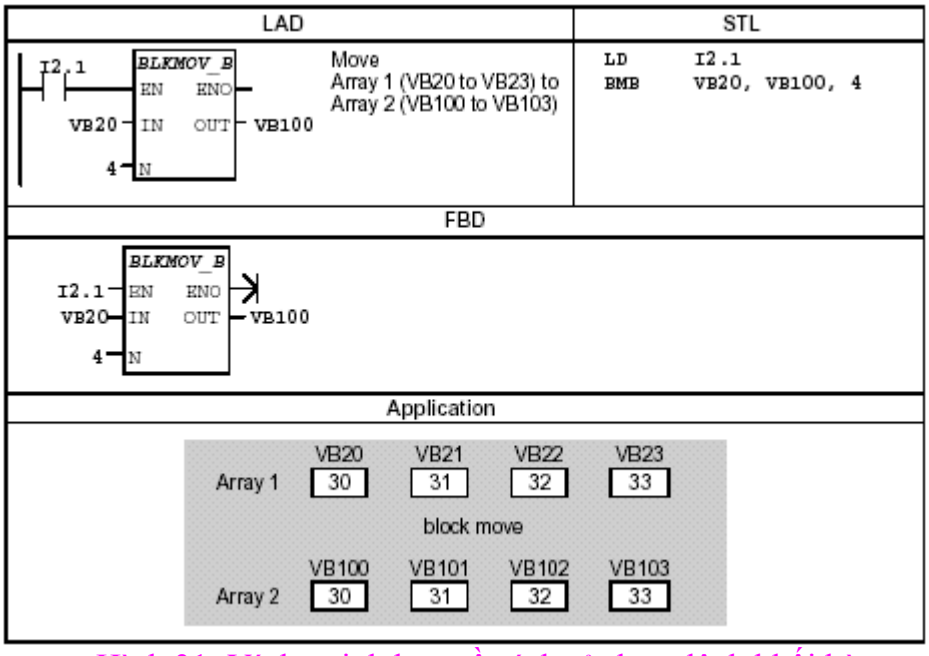
		này cần phải thực hiện một số thủ tục quy định trước khi quá trình tính toán diễn ra như: việc khai báo tham số của hàm, địa chỉ của mảng dữ liệu, lấy mẫu tín hiệu vào analog đầu vào, thực hiện quá trình tính toán, chuẩn hoá, hiệu chỉnh... Phần này sẽ được trình bày cụ thể ở chương sau.		

7. SIMATIC Move Instructions:

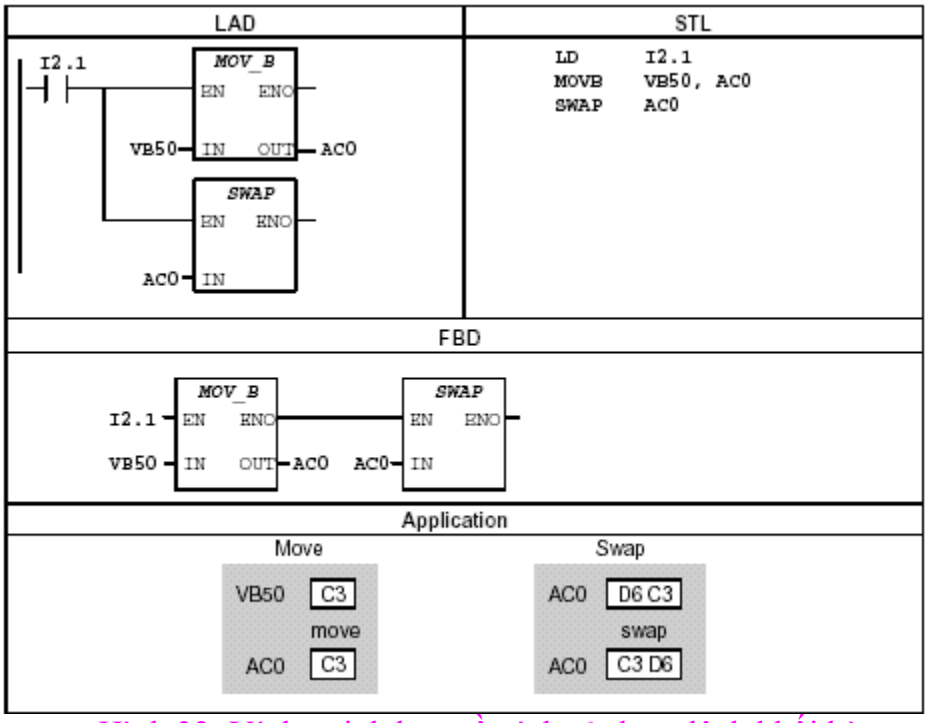
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Move Byte, Move Word, Move Double Word and Move Real				
MOVB IN, OUT		Lệnh thực hiện việc chuyển dữ liệu từ byte IN vào byte OUT khi có sườn lên của tín hiệu vào.	**IN:** IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC,* LD **OUT:**IB, QB, MB, VB, SMB, SB, LB, AC, *VD, *AC,* LD	Byte

MOVW IN,OUT		Lệnh thực hiện việc chuyển dữ liệu từ Word IN vào Word OUT khi có sườn lên của tín hiệu vào.	**IN:** IW, QW, VW, LW, SW, AIW, T, C, AC, Constant, ∗VD, ∗AC, ∗LD **OUT:** IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, ∗VD, ∗AC, ∗LD	Word, INT
MOVD IN, OUT		Lệnh thực hiện việc chuyển dữ liệu từ kép IN vào từ kép OUT khi có sườn lên của tín hiệu vào.	**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, &VB, &IB, &QB, &SB, &MB, &T, &C, Constant, ∗VD, ∗AC, ∗LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, ∗VD, ∗AC, ∗LD	DoubleWord, DINT
MOVR IN, OUT		Lệnh thực hiện việc chuyển dữ liệu là số thực từ từ kép IN vào từ kép OUT khi có sườn lên của tín hiệu vào.	**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, ∗VD, ∗AC, ∗LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, ∗VD, ∗AC, ∗LD	Real
Block Move Byte, Block Move Word, Block Move Double Word and Block Move Real				
BMB IN, OUT, N		Lệnh thực hiện việc chuyển N byte dữ liệu tính từ byte IN vào vùng địa chỉ tính từ byte OUT khi có sườn lên của tín hiệu vào.	**IN, OUT:** IB, QB, MB, VB, SMB, SB, LB, ∗VD, ∗AC, ∗LD.	Byte
			N: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD 1 <= N <= 255	Byte

BMW IN, OUT, N		Lệnh thực hiện việc chuyển N từ đơn dữ liệu tính từ từ đơn IN vào vùng địa chỉ tính từ từ đơn OUT khi có sườn lên của tín hiệu vào.	**IN:** IW, QW, VW, LW, SW, SMW, AIW, T, C, AC, ∗VD, ∗AC, ∗LD **OUT:** IW, QW, VW, LW, SW, SMW, AQW, T, C, AC, ∗VD, ∗AC, ∗LD	Word
			N: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD 1 <= N <= 255	Byte
BMD IN, OUT, N		Lệnh thực hiện việc chuyển N từ kép dữ liệu tính từ từ kép IN vào vùng địa chỉ tính từ từ kép OUT khi có sườn lên của tín hiệu vào.	**IN, OUT:** ID, QD, MD, VD, SMD, SD, LD, ∗VD, ∗AC, ∗LD.	DWord
			N: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD 1 <= N <= 255	Byte
Swap Byte				
SWAP IN		Lệnh đảo dữ liệu của 2 byte trong từ đơn IN.	**IN:** IW, QW, VW, LW, SW, SMW, AIW, T, C, AC.	Word
Move Byte Immedieate Read/ Write				
BIR IN, OUT		Lệnh đọc tức thời giá trị ở byte đầu vào ở cổng vật lý IN và ghi trực tiếp vào byte OUT.	**IN:** IB **OUT:** IB, QB, MB, VB, SMB, SB, LB, AC, ∗VD, ∗AC, ∗LD	Byte
BIW IN, OUT		Lệnh đọc tức thời giá trị ở byte IN và ghi trực tiếp ra đầu ra ở cổng vật lý byte OUT.	**IN:** IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD **OUT:** QB	Byte



Hình 31: Ví dụ minh hoạ về cách sử dụng lệnh khối hàm.



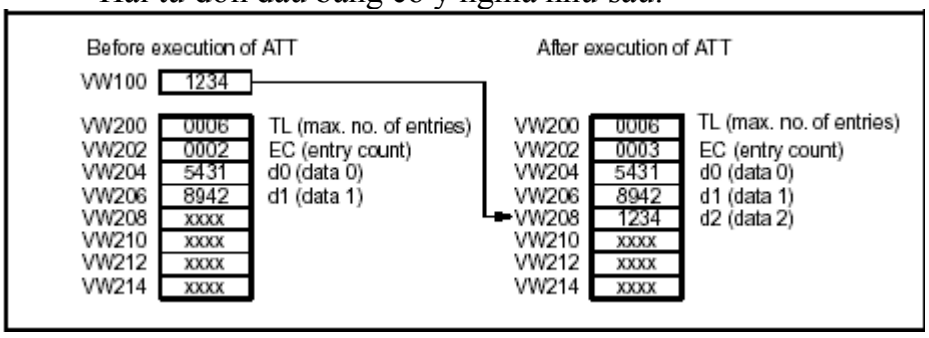
Hình 32: Ví dụ minh hoạ về cách sử dụng lệnh khối hàm

8. SIMATIC Table Instructions:

Các lệnh làm việc với bảng dữ liệu gọi tắc là lệnh bảng, cho phép nhập dữ liệu vào một bảng, sắp xếp số lượng theo thứ tự đã được nhập vào hoặc theo thứ tự ngược lại.

Bảng được định nghĩa là một mảng từ đơn xếp liền nhau từ địa chỉ thấp nhất tính từ đầu bảng đến địa chỉ cao nhất tính đến cuối bảng. Hai từ đơn đầu tiên của bảng dùng để quản lý bảng. Dữ liệu được ghi vào trong bảng bắt đầu từ từ đơn thứ 3 trong bảng, mỗi dữ liệu chiếm một từ đơn, một bảng chỉ chứa tối đa 100 dữ liệu. Có nghĩa là bảng lớn nhất có 204 byte.

Hai từ đơn đầu bảng có ý nghĩa như sau:



Hình 33: Mô tả bảng dữ liệu.

+ Từ đầu ký hiệu bằng TL, chứa kích thước của bảng không kể hai từ đơn quản lý.

+ Từ đơn thứ hai ký hiệu bằng EC, để quản lý số các dữ liệu hiện có trong bảng.

Bit SM1.4 được dùng để báo trạng thái đầy bảng.

Các lệnh làm việc với bảng gồm có các lệnh:

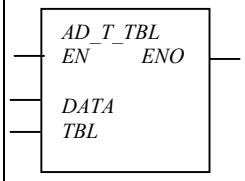
+ Nhập thêm dữ liệu vào bảng : ATT - Add to Table(AT_T_TBL).

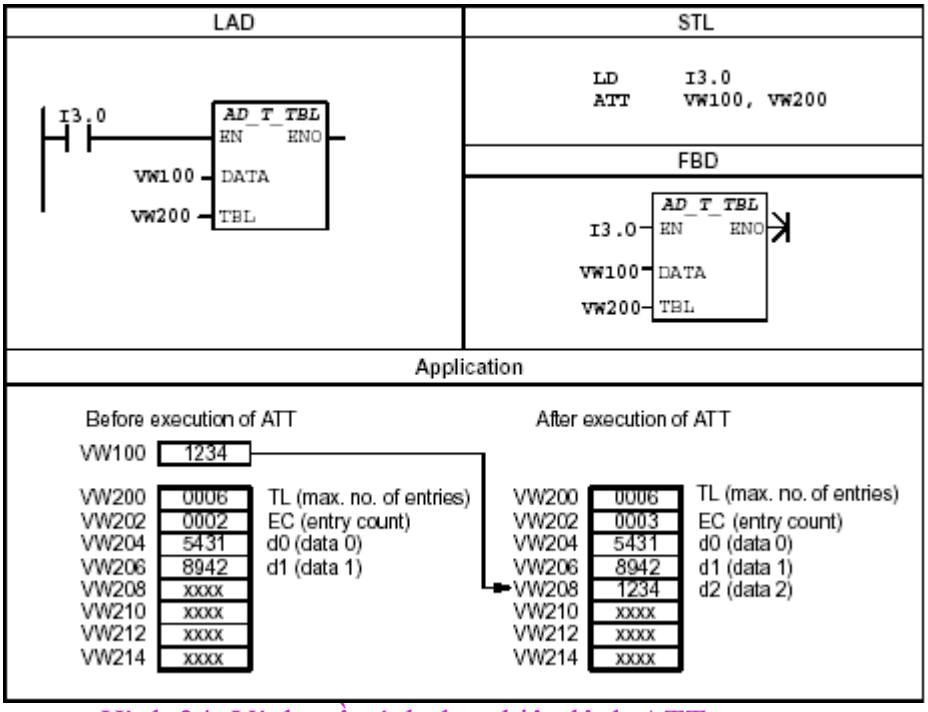
+ Lấy dữ liệu ra khỏi bảng theo thứ tự vào trước ra trước: First - In - First - Out (FIFO).

+ Lấy dữ liệu ra khỏi bảng theo thứ tự vào sau ra trước: Last - In - First - Out (LIFO).

Tip: **Lệnh bảng được thực hiện liên tục (một từ trong một vòng quét) khi đầu vào vẫn còn được kích. Bởi vậy trước khi gọi lệnh làm việc với bảng nên thực hiện lệnh phát hiên sườn lên (EU) cho tín hiệu đầu vào**.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Add to Table				

ATT DATA, TABLE		Lệnh ghi thêm vào bảng một dữ liệu kiểu từ đơn, được xác định bằng nội dung cảu toán hạng DATA trong lệnh. Bảng được chỉ định trong lệnh bằng toán hạng TBL xác định từ đầu tiên của bảng, tức là TL. Nếu bảng đã đầy tức là EC=TL, Bit SM1.4=1. Dữ liệu mới được đưa vào sẽ nằm trong từ chưa dùng đầu tiên, tức là ngay sau dữ liệu được nhập trước đó. Khi lệnh thực hiên xong thì nộ dung của từ EC tăng thêm 1 đơn vị.	**DATA:** IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT
			TBL: IW, QW, VW, LW, SW, MW, SMW, T, C, *VD, *AC, *LD	Word



Hình 34: Ví dụ về cách thực hiện lệnh ATT.

Sử dụng lệnh tìm kiếm để tìm dữ liệu theo mẫu cho trước trong một bảng. Mẫu dữ liệu định trước là nội dung của toán hạng PTN của lệnh. Tham số CMD là luật tìm kiếm, có 4 luật tìm kiếm: =, <>, <, >.

Bảng được chỉ định trong lệnh tìm kiếm được chỉ định bằng nội dung của toán hạng TBL chỉ ô nhớ nằm ngay trước vùng chứa dữ liệu của bảng (ô này chính là ô từ đơn EC).

Bảng quy định cho lệnh tìm kiếm bao gồm bộ đếm EC tức thời có kiểu từ đơn ghi số các dữ liệu có trong bảng và vùng dữ liệu của bảng. Số lượng lớn nhất các dữ liệu của bảng có thể có của bảng là 100.

Mỗi dữ liệu trong bảng có kích thước bằng từ đơn. Dữ liệu trong bảng được đánh số từ 0÷n với n có giá trị cực đại bằng 99. Số các dữ liệu có trong bảng là nội dung của từ đơn EC, ***không bắt buộc lệnh tìm kiếm phải bắt đầu từ đầu bảng***. Lệnh có thể bắt đầu công việc tìm kiếm tại một điểm bất kỳ trong vùng dữ liệu. Toán hạng INDX xác định điểm xuất phát của công việc tìm kiếm bằng việc chỉ ra chỉ số (0÷99) của dữ liệu đầu tiên trong vùng định tìm kiếm. Như vậy muốn tìm từ đầu bảng INDX phải có giá trị bằng 0. Nội dung của INDX là số nguyên trong khoảng từ 0 đến EC.

Nếu sử dụng lệnh tìm kiếm với bảng được tạo bởi các lệnh ATT, FIFO, LIFO thì ô nhớ EC là ô nhớ đầu bảng phải được chỉ định trong lệnh tại toán hạng TBL. Khi sử dụng lệnh ATT, FIFO, LIFO đòi hỏi phải thông báo từ số các đầu vào cực đại cho lệnh (ô nhớ TL) còn khi sử dụng lệnh tìm kiếm TBL_FIND thì không cần. Toán hạng SRC của lệnh tìm kiếm là tên của ô nhớ EC (2 byte).

Cú pháp của lệnh tìm kiếm trong LAD và STL khác nhau. Trong khi cả 4 luật tìm kiếm CMD trong LAD, thì trong STL tương ứng với mỗi luật tìm kiếm có 1 lệnh tìm kiếm riêng. Như vậy trong LAD chỉ có 1 hộp cho 4 lệnh tìm kiếm thì trong STL là: FND=, FND<>, FND<, FND>.

Nội dung của toán hạng trong LAD được quy định như sau:

a) CMD = 1, tìm theo luật = (bằng nhau.).

b) CMD = 2, tìm theo luật <> (khác nhau).

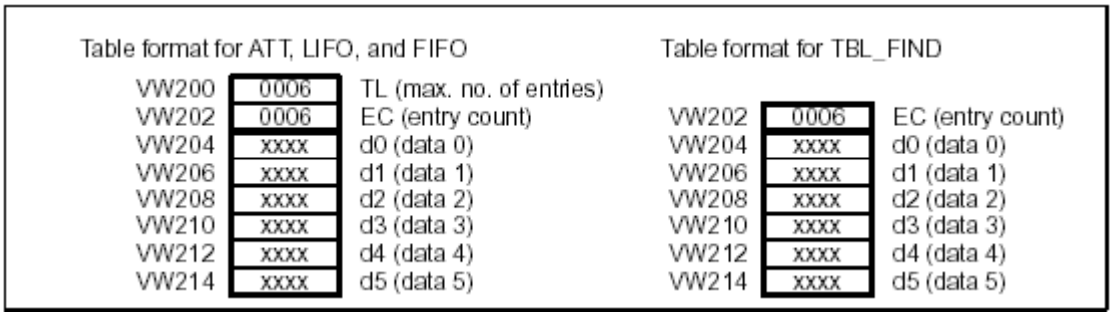
c) CMD = 3, tìm theo luật < (nhỏ hơn).

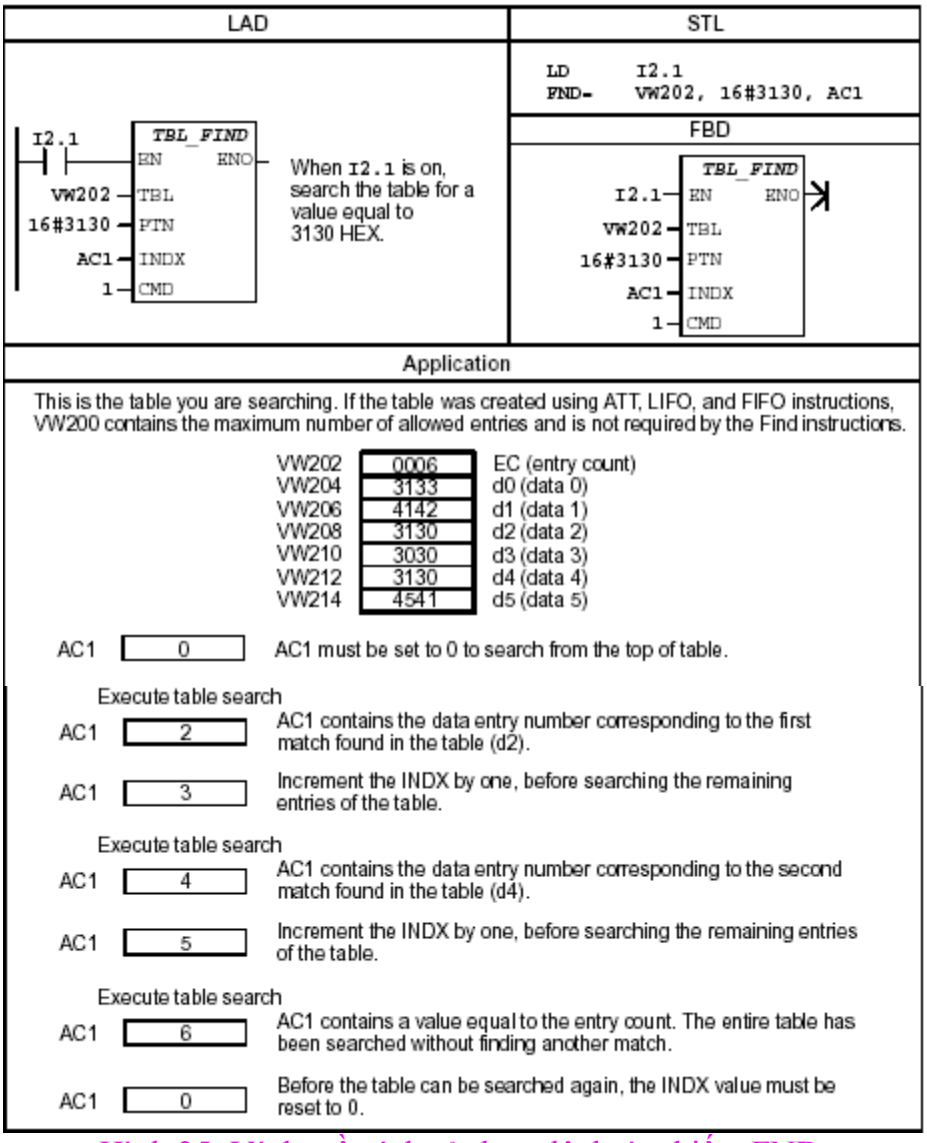
d) CMD = 4, tìm theo luật > (lớn hơn).

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Table Fine				
FND= TBL, PARNT, INDX		Thực hiện việc tìm kiếm trong bảng xác định bởi TBL , bắt đầu từ vị trí dữ liệu INDX ô nhớ chứ dữ liệu PARNT. Luật tìm kiếm được quy định bởi CMD có giá trị từ 1 đến 4 tương ứng =, <>, <, >.	**TBL:** IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, ∗VD, ∗AC, ∗LD	Word
FND<> TBL, PARNT, INDX	AD_T_TBL EN ENO TBL PTN INDX CMD		**PTN:** IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AC, Constant, ∗VD, ∗AC, ∗LD	INT

FND< TBL, PARNT, INDX FND> TBL, PARNT, INDX		<>, <, >. Khi tìm thấy , INDX sẽ chỉ vào ô dữ liệu đầu tiên tìm được trong bảng và lệnh được kết thúc. Do đó để tìm kiếm dữ liệu tiếp theo, INDX phải được tăng giá trị l và gọi lại lệnh này. Nếu như không tìm thấy INDX có giá trị đúng bằng giá trị của bộ đếm EC.	**INDX:** LW, SW, MW, SMW, AIW, T, C, *VD, *AC, *LD	Word
			CMD: Constant	Byte

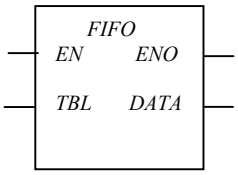
Bảng : Sự khác nhau giữa bảng dữ liệu định nghĩa bằng lệnh ATT, FIFO, LIFO và lệnh FIN.

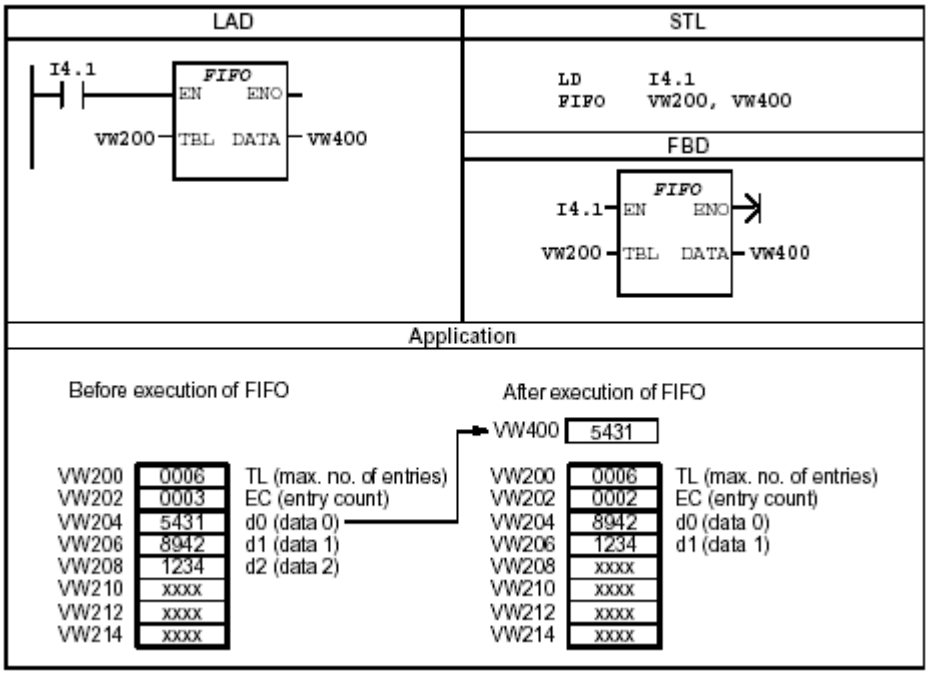




Hình 35: Ví dụ về cách sử dụng lệnh tìm kiếm FND.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Fisrt - In - Fisrf - Out				

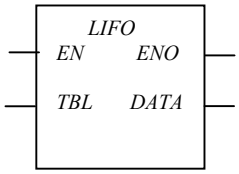
FIFO TABLE, DATA		Lệnh lấy dữ liệu đầu tiên của bảng ra khỏi bảng. Nếu bảng đã trống có nghĩa là dữ liệu trong đó được lấy ra hết, hay EC=0, bit SM1.4=1. Dữ liệu lấy ra đượ ghi vào DATA (kiểu từ). Các dữ liệu còn lại được dồn lên vị trí trên để lấp chỗ trống vừa mới bị lấy đi. Khi lệnh thực hiện xong nội dung của EC giảm đi một đơn vị.	**TBL:** IW, QW, VW, LW, SW, MW, T, C, *VD, *AC, *LD	INT
			DATA: IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AQW, *VD, *AC, *LD	Word

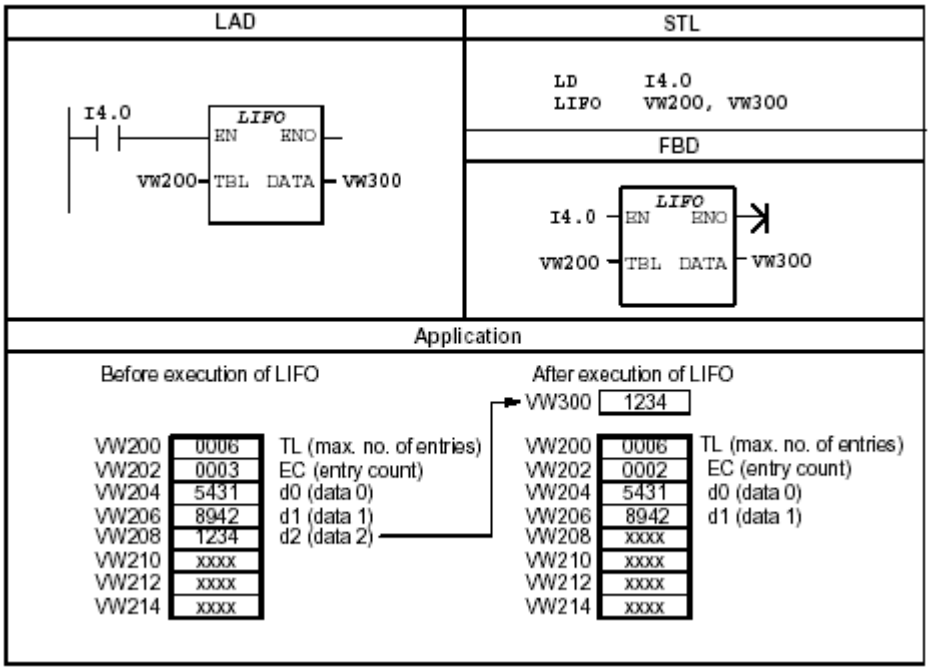


Hình 36: Ví dụ về cách sử dụng lệnh FIFO.

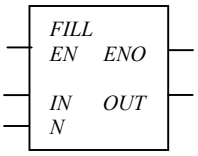
.

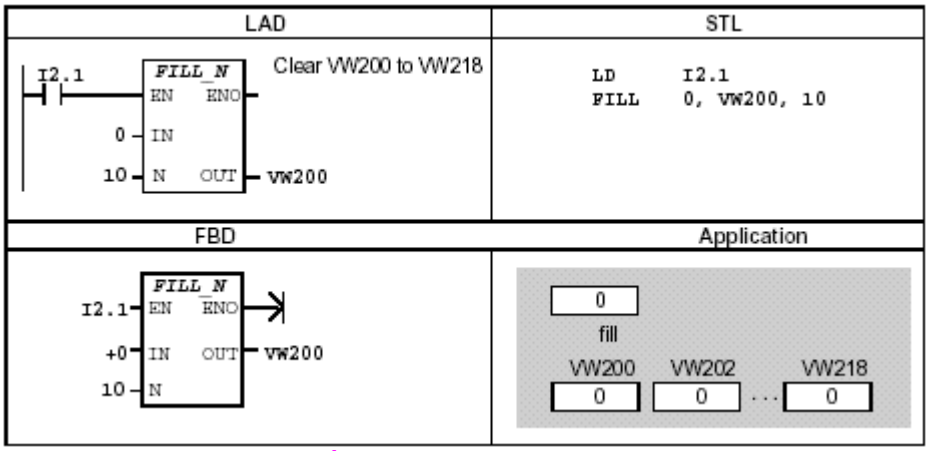
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu

				Data Types
Last - In - Fisrf - Out				
LIFO TABLE, DATA		Lệnh lấy dữ liệu cuối cùng của bảng ra khỏi bảng tức là dữ liệu được nhập sau cùng. Nếu bảng đã trống có nghĩa là dữ liệu trong đó được lấy ra hết, hay EC=0, bit SM1.4=1. Dữ liệu lấy ra đượ ghi vào DATA (kiểu từ). Các dữ liệu còn lại được dồn lên vị trí trên để lấp chỗ trống vừa mới bị lấy đi. Khi lệnh thực hiện xong nội dung của EC giảm đi một đơn vị.	**TBL:** IW, QW, VW, LW, SW, MW, T, C, *VD, *AC, *LD	INT
			DATA: IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AQW, *VD, *AC, *LD	Word



Hình 37: Ví dụ về cách sử dụng lệnh LIFO.

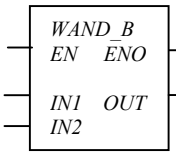
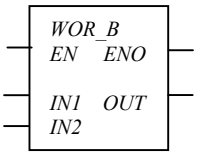
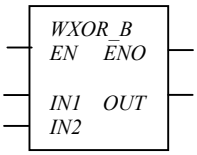
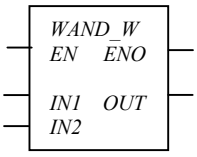
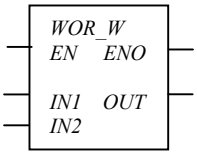
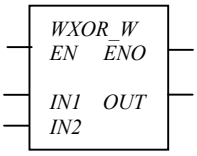
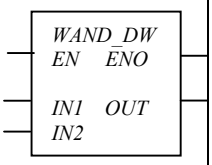
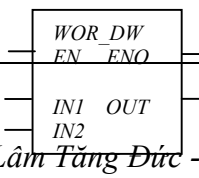
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Memory Fill				
FILL IN, OUT, N		Lệnh điền giá trị chứa trong Word IN vào mảng bắt đầu từ địa chỉ Word OUT. N là số từ đơn của mảng, 1<=N<=255	**IN:** IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AC, Constant, *VD, *AC, *LD	Word
			N: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD	Byte
			OUT: IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, *VD, *AC, *LD	Word

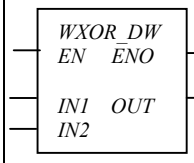


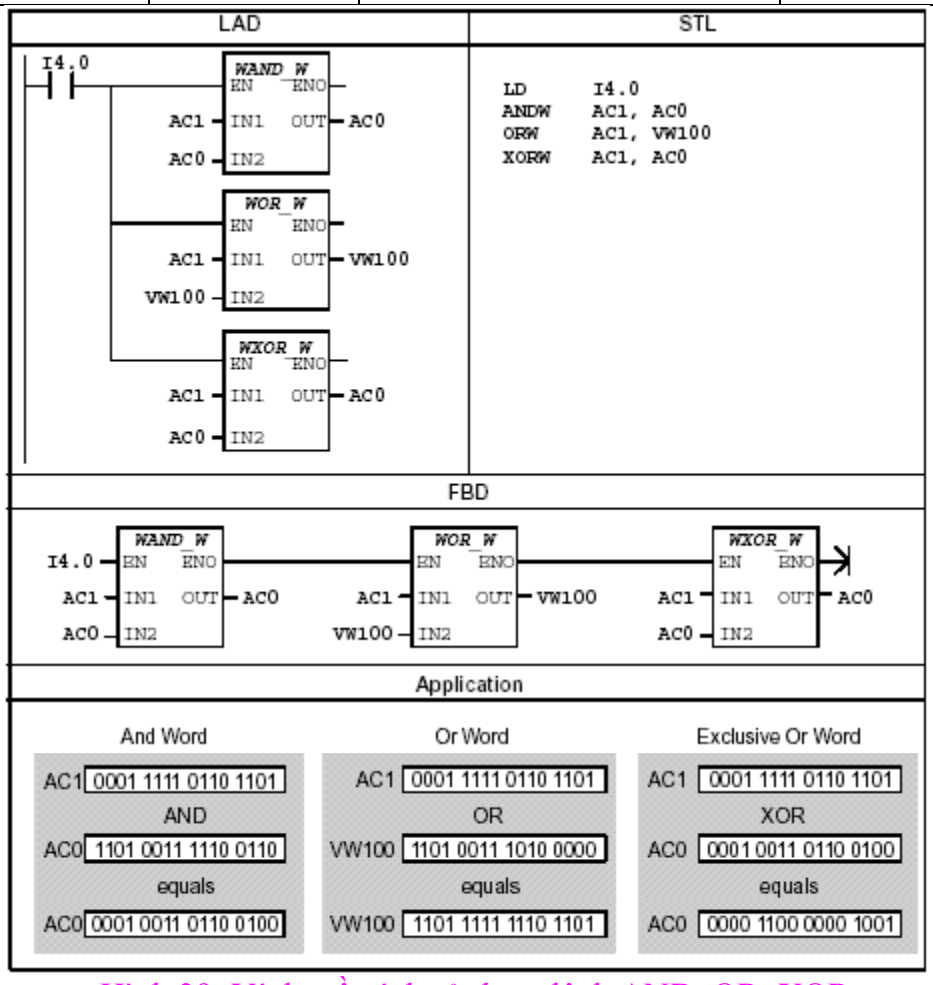
Hình 38: Ví dụ về cách sử dụng lệnh FILL.

9. SIMATIC Logical Operation Instruction:

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
And Byte, Or Byte, Exclusive Or Byte				

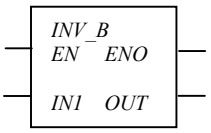
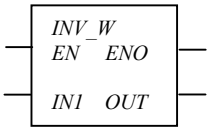
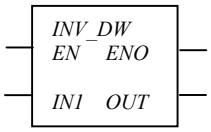
ANDB IN1, OUT		Lệnh thực hiện AND giữa các bit tương ứng của hai Byte IN1 và IN2, kết quả ghi vào Byte OUT.	**IN1, IN2:** IB, QB, MB, VB, SMB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD	Byte
ORB IN1, OUT		Lệnh thực hiện OR giữa các bit tương ứng của hai Byte IN1 và IN2, kết quả ghi vào Byte OUT.	**OUT:** IB, QB, MB, VB, SMB, SB, LB, AC, ∗VD, ∗AC, ∗LD	Byte
XORB IN1, OUT		Lệnh thực hiện XOR giữa các bit tương ứng của hai Byte IN1 và IN2, kết quả ghi vào Byte OUT.		
And Word, Or Word, Exclusive Or Word				
ANDW IN1, OUT		Lệnh thực hiện AND giữa các bit tương ứng của hai Word IN1 và IN2, kết quả ghi vào Word OUT.	**IN1, IN2:** IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AC, Constant, ∗VD, ∗AC, ∗LD	Word
ORW IN1, OUT		Lệnh thực hiện OR giữa các bit tương ứng của hai Word IN1 và IN2, kết quả ghi vào Word OUT.		
XORW IN1, OUT		Lệnh thực hiện XOR giữa các bit tương ứng của hai Word IN1 và IN2, kết quả ghi vào Word OUT.	**OUT:** IW, QW, VW, LW, SW, MW, SMW, T, C, AC, ∗VD, ∗AC, ∗LD	
And DWord, Or DWord, Exclusive Or DWord				
ANDD IN1, OUT		Lệnh thực hiện AND giữa các bit tương ứng của hai từ kép IN1 và IN2, kết quả ghi vào từ kép OUT.	**IN1, IN2:** ID, QD, VD, LD, SD, MD, SMD, HD, AC, Constant, ∗VD, ∗AC, ∗LD	Double Word
ORD 1, OUT		Lệnh thực hiện OR giữa các bit tương ứng của hai từ kép		

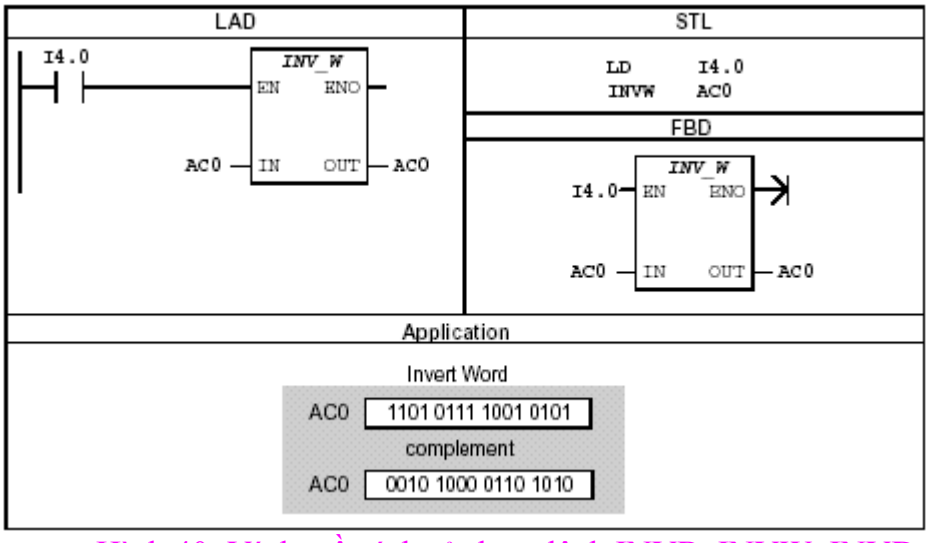
		IN1 và IN2, kết quả ghi vào từ kép OUT.		
XORD IN1, OUT		Lệnh thực hiện XOR giữa các bit tương ứng của hai từ kép IN1 và IN2, kết quả ghi vào từ kép OUT.	**OUT:** ID, QD, VD, LD, MD, SMD, AC, *VD, *AC, *LD	



Hình 39: Ví dụ về cách sử dụng lệnh AND, OR, XOR.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Invert Byte, Invert Word, Invert DWord				

INVB OUT		Lệnh đảo từng bit của byte đầu vào IN, kết qủa đưa ra đầu ra OUT. Thường thì đầu vào và ra cùng địa chỉ.	**IN:**IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD **OUT:**IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD	Byte
INVW OUT		Lệnh đảo từng bit của từ đơn đầu vào IN, kết qủa đưa ra đầu ra OUT. Thường thì đầu vào và ra cùng địa chỉ.	**IN:** IW, QW, VW, LW, SW, MW, SMW, AC, AIW , T, C, Constant, *VD, *AC, *LD **OUT:** IW, QW, VW, LW, SW, MW, SMW, AC, T, C, *VD, *AC, *LD	Word
INVD OUT		Lệnh đảo từng bit của từ kép đầu vào IN, kết qủa đưa ra đầu ra OUT. Thường thì đầu vào và ra cùng địa chỉ.	**IN:** ID, QD, VD, LD, SD, MD, SMD, HD, AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, VD, LD, SD, MD, SMD, AC, *VD, *AC, *LD	DWord



Hình 40: Ví dụ về cách sử dụng lệnh INVB, INVW, INVD.

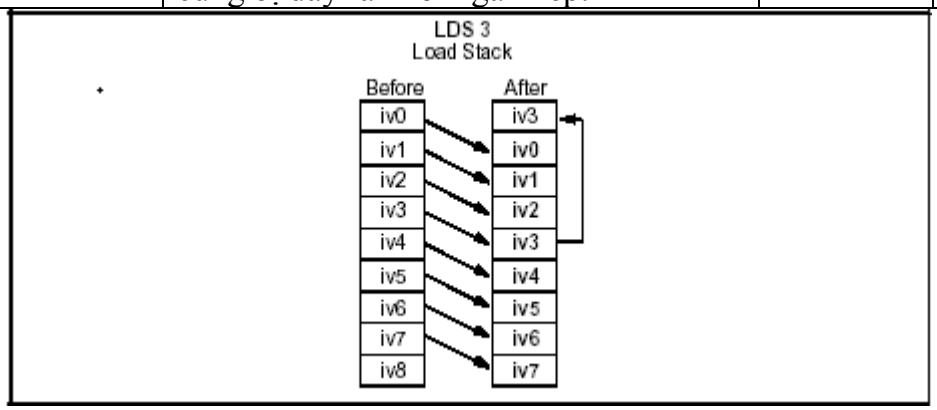
10. SIMATIC Stack Logic Instructions:

Các lệnh tiếp điểm trong đại số Boolean cho phép tạo lập được các mạch logic (không có nhớ). Trong LAD các mạch này biểu diễn thông qua cấu trúc mạch, mắc nối tiếp hay song song các mạch tiếp điểm thường đóng và các tiếp điểm thường mở. STL có thể sử dụng các lệnh A (And) và O (Or) cho các tiếp điểm mắc nối tiếp và song song là thường hở hoặc các lệnh AN (And Not) và ON (Or Not) cho các tiếp điểm mắc nối tiếp và song song là thường đóng. Gía trị của các bit trong ngăn xếp thay đổi tuỳ thuộc vào từng lệnh. Trong phần này chúng ta sẽ đi sâu hơn về sự làm việc của các bit trong ngăn xếp, việc hiểu và nắm bắt về ngăn xếp là điều rất cần thiết trong vấn đề lập trình dùng ngôn ngữ STL.

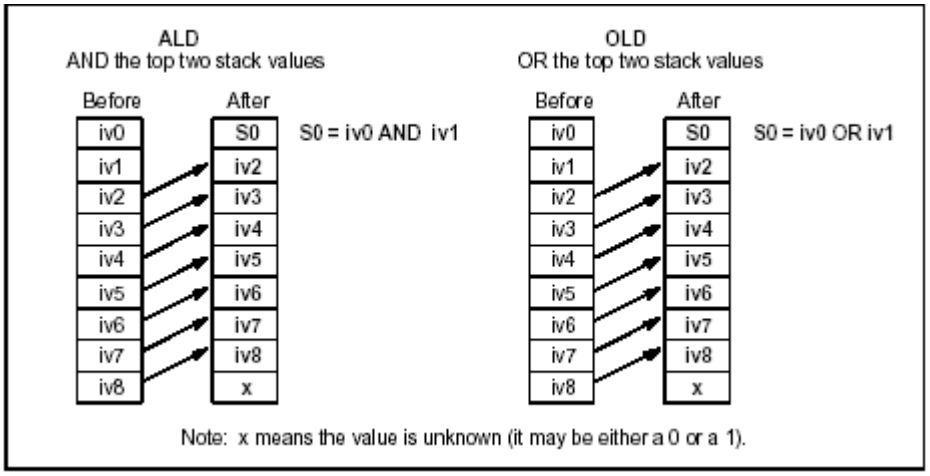
Ngoài những lệnh làm việc trực tiếp với tiếp điểm, S7-200 còn có 5 lệnh đặc biệt biểu diễn cácc phép tính của đại số Boolean cho các bit trong ngăn xếp, được gọi là các lệnh ***stack logic.*** Trong LAD không dùng những lệnh này. STL sử dụng các lệnh này để thực hiện những phép toán của phương trình có nhiều biểu thức con. Sau đây là bảng tóm tắt cú pháp và hướng dẫn cách sử dụng lệnh.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
And Load				
ALD	none	Lệnh tổ hợp giá trị đầu tiên và giá trị của bit thứ hai trong ngăn xếp bằng phép tính ∧. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp. Gía trị còn lại được kéo lên 1 bit.	none	none
Or LoaD				
OLD	none	Lệnh tổ hợp giá trị đầu tiên và giá trị của bit thứ hai trong ngăn xếp bằng phép tính ∨. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp. Gía trị còn lại được kéo lên 1 bit.	none	none
Logic PuSh				
LPS	none	Sao chép giá trị của bit đầu tiên vào bit thứ hai trong ngăn xếp. Gía trị còn lại bị đẩy xuống 1 bit. Bit cuối cùng bị đẩy ra ngoài.	none	none
Logic ReaD				
LRD	none	Lệnh sao chép giá trị của bit thứ hai vào bit đầu tiên của ngăn xếp, các giá trị còn lại của ngăn xếp vẫn giữ nguyên.	none	none
Logic PoP				
LPP	none	Lệnh kéo ngăn xếp lên 1 bit theo nguyên tắc bit sao đè lên bit trước.	none	none

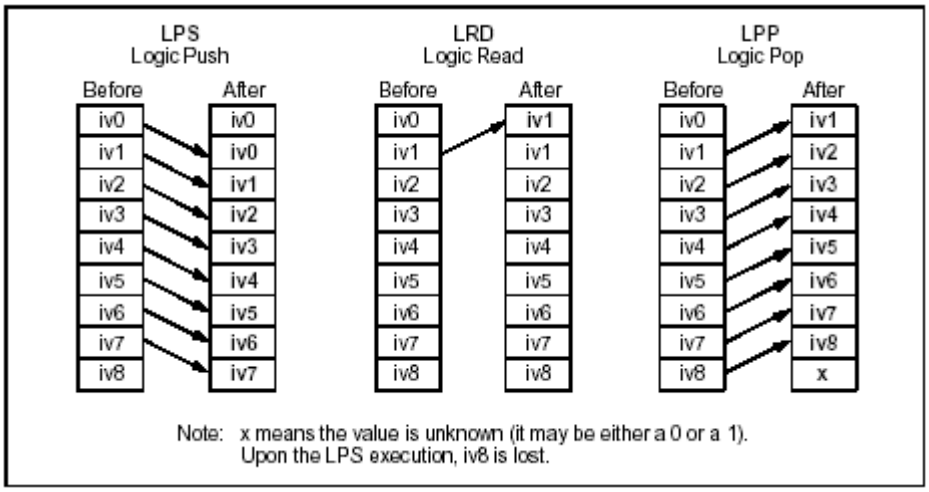
LoaD Stack				
LDS n	none	Lệnh sao chép giá trị của bit thứ n (ngăn xếp có 9 bit thì bit thứ nhì được tính là 1...đến bit cuối cùng là 8) của ngăn xếp lên bit đầu tiên. Các giá trị còn lại của ngăn xếp bi đẩy lùi xuống 1 bit, bit cuối cùng bị đẩy ra khỏi ngăn xếp.	n: 1÷8	Byte



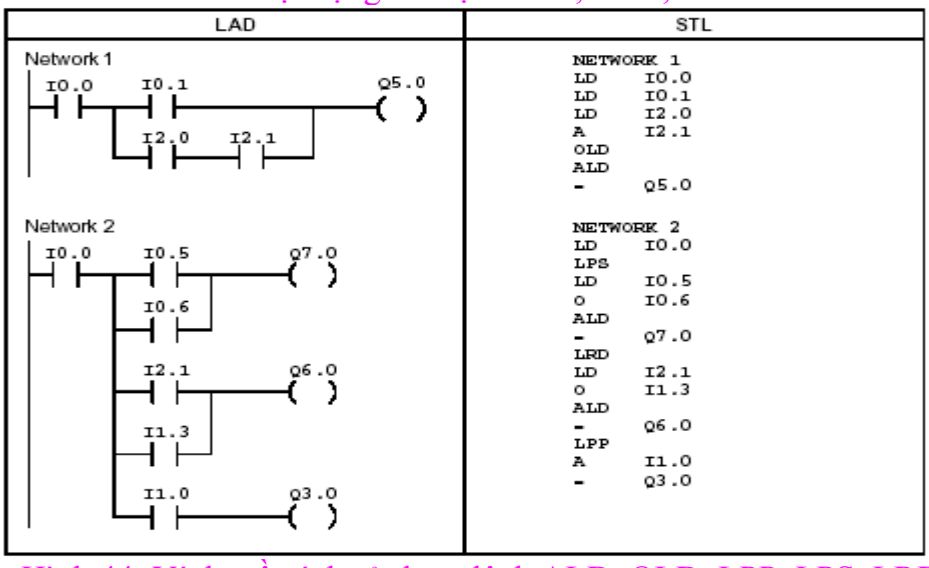
Hình 41: Mô tả hoạt động của lệnh LDS.



Hình 42: Mô tả hoạt động của lệnh ALD và OLD.



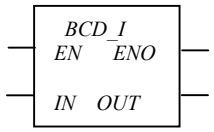
Hình 43: Mô tả hoạt động của lệnh LPS, LRD, LPP.

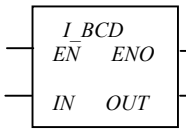
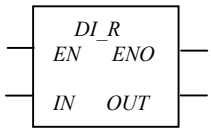
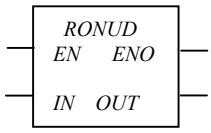
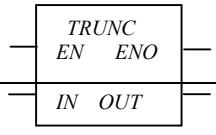


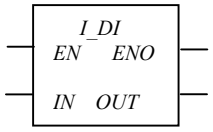
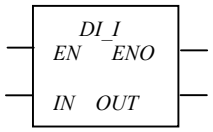
Hình 44: Ví dụ về cách sử dụng lệnh ALD, OLD, LPP, LPS, LRD.

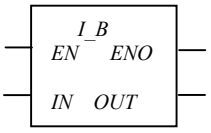
11. SIMATIC Conversion Instructions:

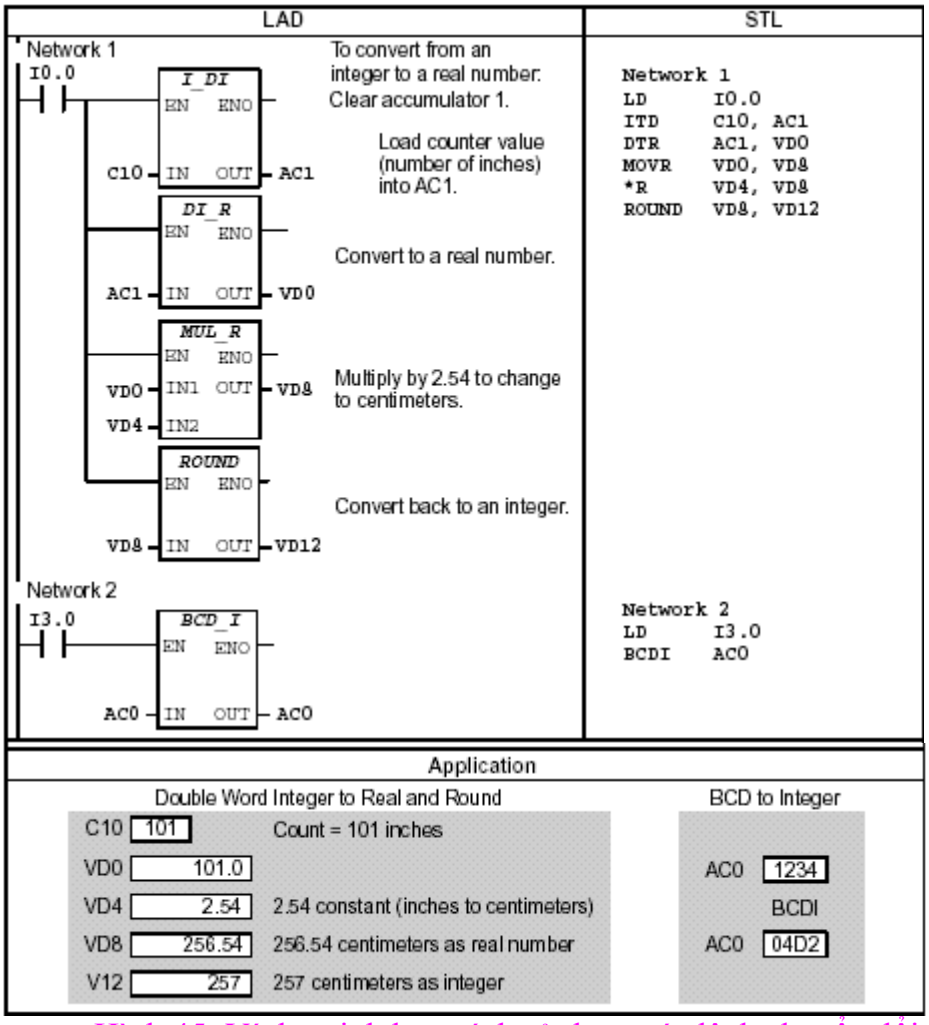
Các hàm đổi kiểu dữ liệu cho phép thực hiện việc đổi kiểu dữ liệu từ kiểu này sang kiểu khác. Sau đây là các lệnh biến đổi kiểu dữ liệu trong STL và LAD:

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
BCD to Integer and Integer to BCD				
BCDI OUT		Lệnh chuyển đổi một số nhị_thập phân IN sang số nguyên và lưu kết quả vào OUT. Giới hạn của IN: 0÷9999.	**IN:** IW, QW, VW, LW, MW, SMW, AIW ,AC, T, C, Constant, *VD, *AC, *LD, SW. **OUT:** IW, QW,	Word

			VW, LW, MW, SMW, AC, T, C, ∗VD, ∗AC, ∗LD, SW.	
IBCD OUT		Lệnh chuyển đổi một số nguyên IN sang số nhị_thập phân và lưu kết quả vào OUT. Giới hạn của IN: 0÷9999.	**IN:** IW, QW, VW, LW, MW, SMW, AIW ,AC, T, C, Constant, ∗VD, ∗AC, ∗LD. **OUT:** IW, QW, VW, LW, MW, SMW, AC, T, C, ∗VD, ∗AC, ∗LD.	Word
Double Integer to Real				
DTR IN, OUT		Lệnh chuyển đổi số nguyên 32 bit IN sang số thực (32 bit) và lưu kết quả vào OUT.	**IN:** ID, QD, VD, LD, MD, SMD, AC, HD, Constant, ∗VD, ∗AC, ∗LD, SD. **OUT:**ID, QD, VD, LD, MD, SMD, AC, HD, ∗VD, ∗AC, ∗LD, SD.	DWord
Round				
ROUND IN, OUT		Lệnh chuyển đổi số thực IN thành số nguyên double Integer (làm tròn số) và kết qủa lưu vào OUT. Nếu phần lẽ >= 0.5 thì được làm tròn về phía lớn hơn 1 đơn vị.	**IN:** ID, QD, VD, LD, MD, SMD, AC, Constant, ∗VD, ∗AC, ∗LD, SD.	Real
			OUT:ID, QD, VD, LD, MD, SMD, AC, HD, ∗VD, ∗AC, ∗LD, SD.	DINT
Truncate				
TRUNC IN,		Hàm chuyển đổi số thực 32 bit có dấu sang số nguyên 32 bit có dấu.	**IN:** ID, QD, VD, LD, MD, SMD, AC, Constant, ∗VD, ∗AC, ∗LD, SD.	Real

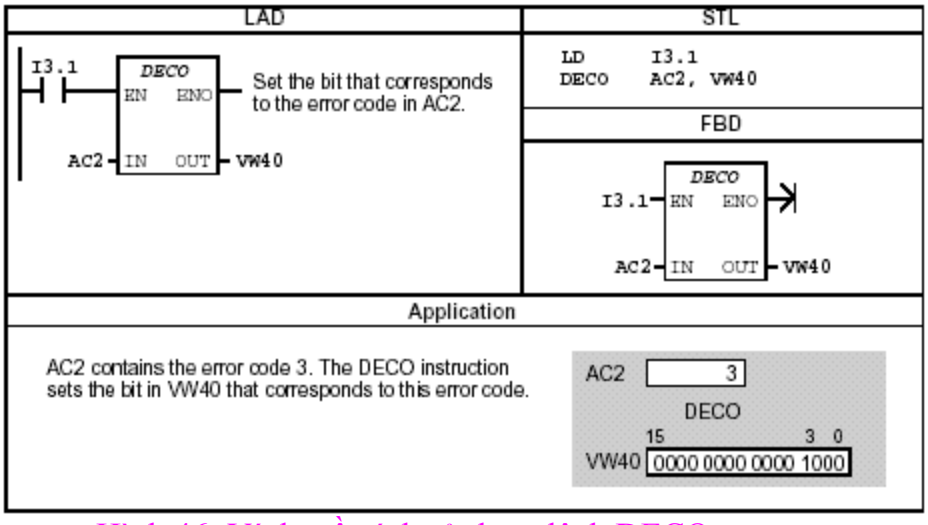
OUT			**OUT:**ID, QD, VD, LD, MD, SMD, AC, HD, ∗VD, ∗AC, ∗LD, SD.	DINT
Double Integer to Integer and Integer to Double Integer				
ITD IN, OUT		Lệnh chuyển đổi số nguyên 16 bit sang số nguyên 32 bit.	**IN:** IW, QW, VW, LW, MW,SW, SMW, AIW ,AC, T, C, Constant, ∗VD, ∗AC, ∗LD.	INT
			OUT: ID, QD,VD, LD, MD,SD, SMD, AC, ∗VD, ∗AC, ∗LD.	DINT
DTI IN, OUT		Lệnh chuyển đổi số nguyên 32 bit sang số nguyên 16 bit.	**IN:** ID, QD,VD, LD, MD,SD, SMD, AC,Constant, ∗VD, ∗AC, ∗LD.	DINT
			OUT: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, ∗VD, ∗AC, ∗LD.	INT
Integer to Real, Byte to Integer and Integer to Byte				
(Integer to Real)	none	Không có lệnh chuyển đổi trực tiếp này. Ta có thể thực hiện được bằng cách dùng lệnh ITD (chuyển số nguyên 16 bit thành số nguyên 32 bit) sau đó dùng tiếp lệnh DTR (chuyển số nguyên 32 bit sang số thực).	none	none
BTI IN, OUT		Lệnh chuyển đổi giá trị của Byte IN thành giá trị Integer 16 bit và lưu vào OUT.	**IN:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD.	Byte

			OUT: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, *VD, *AC, *LD.	INT
IBT IN, OUT		Lệnh chuyển đổi giá trị trong Word IN thành giá Byte và lưu giá trị này vào OUT.	**IN:** IW, QW, VW, LW, MW,SW, SMW, AC, T, C, AIW, Constant, *VD, *AC, *LD.	INT
			OUT: IB, QB, MB, SMB, VB, SB, LB, AC, *VD, *AC, *LD.	Byte

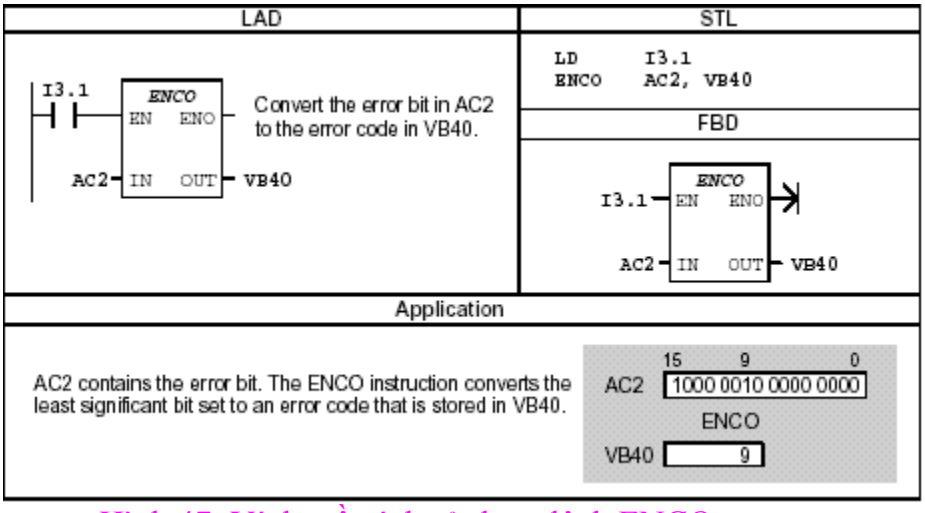


Hình 45: Ví dụ minh hoạ cách sử dụng các lệnh chuyển đổi.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Decode				
DECO IN, OUT		Lệnh đặt giá trị logic 1 vào bit của từ đơn OUT có chỉ số (trọng số của bit thuộc Word) bằng số nguyên nằm trong nibble (4 bit) thấp của byte đầu vào IN. Các bi còn lại của từ đơn có giá trị logic bằng 0.	**IN:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD	Byte
			OUT: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, AIW, ∗VD, ∗AC, ∗LD.	Word
ENCO IN, OUT		Lệnh xác định chỉ số của bit thấp nhất trong từ đơn IN có giá trị logic 1và ghi kết quả này vào nibble thấp nhất của byte đầu ra OUT.	**IN:** IW, QW, VW, LW, MW,SW, SMW, AC, T, C, AIW, ∗VD, ∗AC, ∗LD.	Word
			OUT: IB, QB, MB, SMB, VB, SB, LB, AC, ∗VD, ∗AC, ∗LD.	Byte

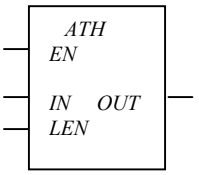
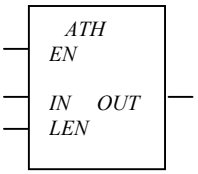


Hình 46: Ví dụ về cách sử dụng lệnh DECO.

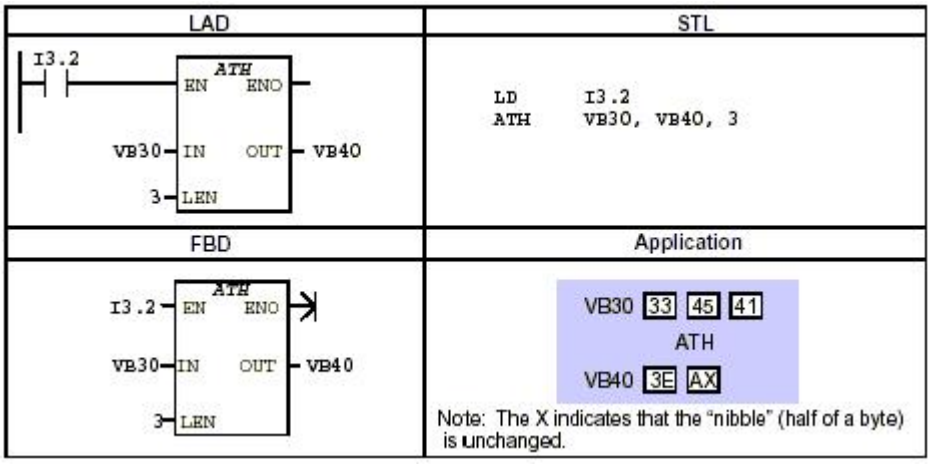


Hình 47: Ví dụ về cách sử dụng lệnh ENCO.

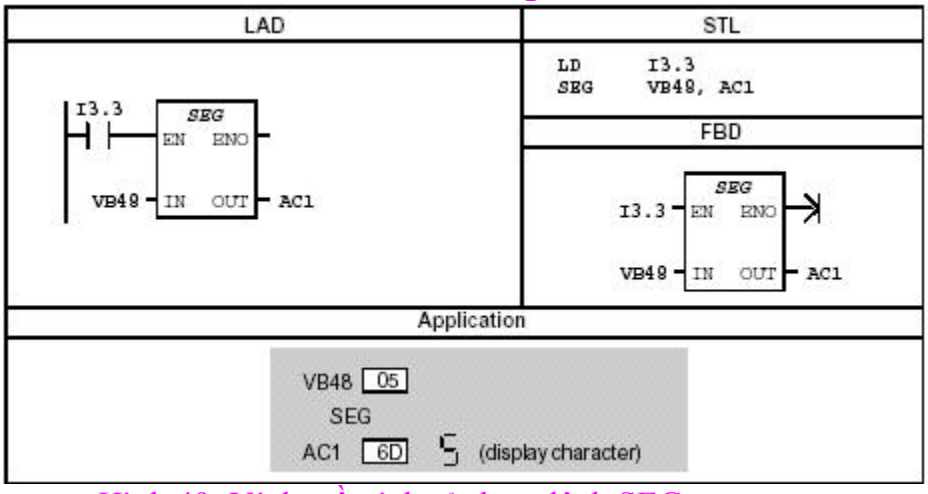
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Segment				

SEG IN, OUT		Lệnh xuất các bit cho thanh ghi 7 đoạn tương ứng với nội dung của 4 bit thấp nhất của byte đầu vào IN. Kết quả được chi vào byte đầu ra.	**IN**: IB, QB, MB, SMB, LB, VB, AC, Constant, ∗VD, ∗AC, SB, ∗LD. **OUT:** IB, QB, MB, SMB, LB, VB, AC, ∗VD, ∗AC, SB, ∗LD.	Byte
ASCII to Hexa and Hexa to ASCII				
ATH IN, OUT, LEN	 	Thực hiện phép biến đổi một chuỗi kí tự có độ dài được chỉ thị trong toán hạng LEN, bắt đầu bằng kí tự chỉ định trong toán hạng IN, sang số nguyên hệ cơ số 16 và ghi vào vùng nhớ kể từ byte được chỉ định bởi OUT. Độ dài cực đại của chuỗi kí tự là 255. Những kí tự hợp lệ là những kí tự có mã ASCII từ 30÷39 và 41÷46 (cơ số 16, ứng với các kí tự từ 0÷9, A÷F). Nếu mã hoá một kí tự bị sai thì quá trình mã hoá bị dừng lại và bit SM1.7 có giá trị logic bằng 1.	**IN, OUT:** IB, QB, MB, SMB, LB, VB, ∗VD, ∗AC, SB, ∗LD. **LEN:** IB, QB, MB, SMB, LB, VB, AC, Constant, ∗VD, ∗AC, SB, ∗LD.	Byte Byte
HTA IN, OUT, LEN		Thực hiện đổi một dãy chữ viết trong hệ cơ số 16 thành chuỗi kí tự mã ASCII. Dãy số đầu vào được lưu trong mảng bắt đầu bằng IN	**IN, OUT:** IB, QB, MB, SMB, LB, VB, ∗VD, ∗AC, SB, ∗LD.	Byte

		và có độ dài là LEN. Độ dài cực đại của dãy số là 255. Chuỗi kí tự đầu ra được ghi vào mảng có byte đầu là OUT.	**LEN:** IB, QB, MB, SMB, LB, VB, AC, Constant, *VD, *AC, SB, *LD.	

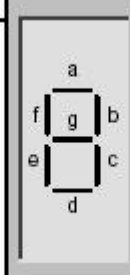


Hình 48: Ví dụ về cách sử dụng lệnh ATH, HTA.



Hình 49: Ví dụ về cách sử dụng lệnh SEG.

(IN) LSD	Segment Display	(OUT) -gfe dcba		(IN) LSD	Segment Display	(OUT) -gfe dcba
0		0011 1111		8		0111 1111
1		0000 0110		9		0110 0111
2		0101 1011		A		0111 0111
3		0100 1111		B		0111 1100
4		0110 0110		C		0011 1001
5		0110 1101		D		0101 1110
6		0111 1101		E		0111 1001
7		0000 0111		F		0111 0001



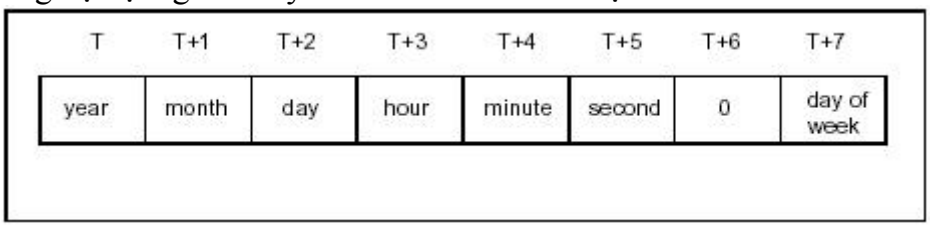
Hình 50: Mã hiển thị thanh ghi 7 đoạn.

12. SIMATIC Clock Instrutions:

Tuyệt đối không sử dụng lệnh đọc /ghi (TODR/TODW) thời gian thực cùng một lúc trong chương trình chính và chương trình xử lý ngắt. Khi một lệnh TODR hoặc TODW đã thực hiện thì khi gọi chương trình xử lý ngắt, các lệnh làm việc với đồng hồ thời gian thực trong chương trình xử lý ngắt sẽ không được thực hiện nữa. Bit SM4.5 sẽ có mức logic 1 trong những trường hợp như vậy.

Đồng hồ thời gian thực chỉ có đối với CPU214 trở lên. Để có thể làm việc với đồng hồ thời gian thực thì CPU sẽ cung cấp 2 lệnh đọc/ghi giá trị cho đồng hồ. Những giá trị đọc được hoặc ghi được với đồng hồ thời gian thực là các giá trị về ngày, tháng, năm và các giá trị về giờ, phút, giây.

Các dữ liệu đọc/ghi với đồng hồ thời gian thực trong LAD, STL có độ dài 1 byte và phải được mã hoá theo kiểu số nhị thập phân BCD (Ex: 16#95 CHO NĂM 95). Chúng năm trong bộ đệm gồm 8 byte liền nhau theo thứ tự như sau:



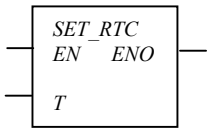
Hình 51: Bộ đệm 8 byte cảu lệnh đồng hồ thời gian thực.

Các giá trị của các thông số phải nằm trong giới hạn:

Year/Month	yymm	yy - 0 to 99	mm - 1 to 12
Day/Hour	ddhh	dd - 1 to 31	hh - 0 to 23
Minute/Second	mmss	mm - 0 to 59	ss - 0 to 59
Day of week	d	d - 0 to 7	1 = Sunday
			0 = disables day of week (remains 0)

CPU S7-200 không thực hiện kiểm tra lại ngày tháng, ngày của tuần để điều chỉnh lại ngày tháng. Giá trị về ngày tháng như là February 30 có thể được chấp nhận. Do đố bạn sẽ phải chắc chắn rằng ngày tháng của bạ đưa vào đó là đúng.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Read Real-Time Clock and Set Real-Time Clock				
TODR T	READ_RTC EN ENO T	Lệnh đọc nội dung của đồng hồ thời gian thực vào bộ đệm 8 byte được chỉ định trong lệnh bằng toán hạng T.	**T:** VB,IB, QB, MB, SMB, SB, LB, *VD, *AC, *LD.	Byte
TODW T		Lệnh ghi nội dung		

		của bộ đệm 8 byte được chỉ định trong lệnh bằng toán hạng T vào đồng hồ thời gian thực.		

13. SIMATIC Program Control Instrutions:

Các lệnh của chương trình, nếu không có những lệnh điều khiển riêng, sẽ được thực hiện tần tự từ trên xuống dưới trong một vòng quét. Lệnh điều khiển chương trình cho phép thay đổi thứ tự thực hiện lệnh. Chúng cho phép chuyển thứ tự như: Đáng lẽ ra là lệnh tiếo theo, tới một lệnh bất cứ nào khác của chương trình; trong đó nơi điều khiển chuyển đến phải được đánh dấu trước bằng ***nhãn chỉ đích***. Nhóm lệnh điều khiển chương trình gồm: lệnh nhảy, lệnh gọi chương trình con, nhãn chỉ đích (hay gọi đơn giản là nhãn), phải được đánh dấu trước khi thực hiện lệnh nhảy hay lệnh gọi chương trình con.

Việc đặt nhãn cho lệnh nhảy phải nằm trong chương trình. Nhãn của chương trình con hay nhãn của chươngtrình xử lý ngắt phải được khai báo ở đầu chương trình. Không thể dùng lệnh JMP để chuyển điều khiển từ chương trình chính vào nhãn bất kỳ trong chương trình con hoặc chương trình xử lý ngắt. Ngược lại cũng không được phép từ một chương trình con hay chương trình xử lý ngắt nhảy ra ngoài chương trình chính đó.

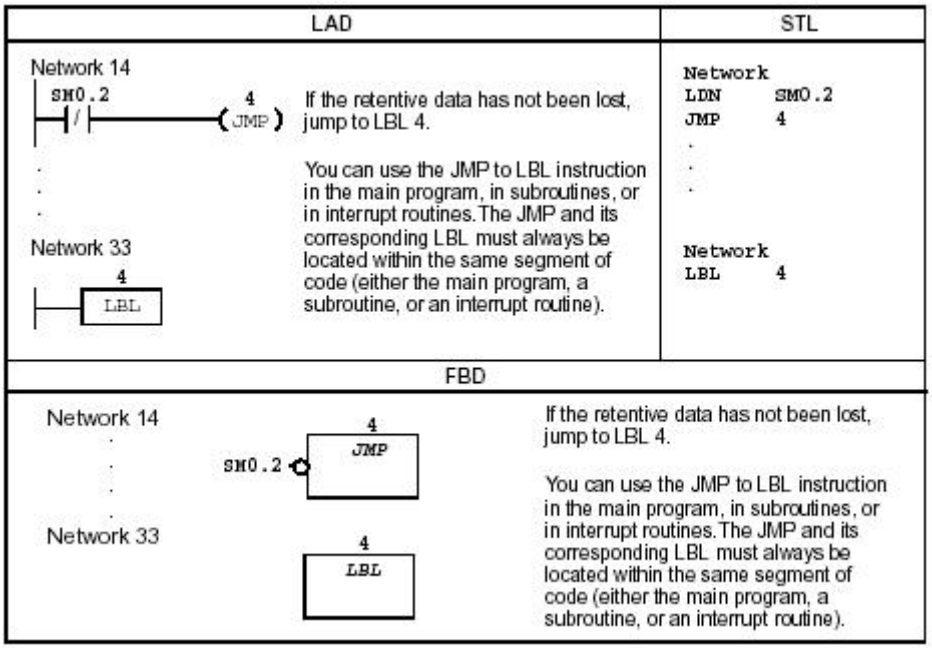
Lệnh gọi chương trình con là lệnh chuyển quyền điều khiển đến chương trình con. Sau khi chương trình con thực hiện xong thì quyền điều khiển lại được chuyển về lệnh tiếp theo trong chương trình chính nagy sau lệnh gọi chương trình con. Từ một chương trình con có thể gọi một chương trình con khác trong nó, có thể gọi như vậy nhiều nhất là 8 lần. Phép đệ quy cũng có thể thực hiện được trong S7-200, mặc dù không bị cấm song phải chú ý đến giới hạn trên.

Trạng thái của ngăn xếp: Nếu lệnh nhảy hay lệnh gọi chương trình con được thực hiện thì đỉnh ngăn xếpluôn có gí trị logic bằng 1. Như vậy trong chương trình con các lệnh có điều kiện được thực hiện như lệnh không có điều kiện. Sau các lệnh **LBL** (lệnh đặt nhãn) và **SBR**, lệnh **LD** trong STL sẽ bị vô hiệu hoá**.**

Khi một chương trình con đươc gọi, toàn bộ nội dung trong ngăn xếp sẽ được cất đi, đỉnh của ngăn xếp nhận giá trị logic mới là 1, các bit khác còn lại của ngăn xếp nhận giá trị logic là 0 và điều khiển được chuyển đến chương trình con đã được gọi. Khi thực hiện xong chương trình con và trước khi quyền điều khiển được chuyển đến chương trình đã gọi nó thì nội dung của ngăn xếp đã được cất giữ trước đó sẽ được chuyển trở lại cho ngăn xếp.

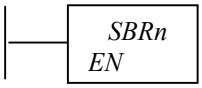
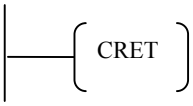
Nội dung của thanh ghi AC không được cất giữ khi gọi chương trình con, nhưng khi một chương trình xử lý ngắt được gọi, nội dung thanh ghi AC sẽ được cất giữ trước khi thực hiện chương trình xử lý ngắt và trả lại sau khi chương trình xử lý ngắt vừa thực hiện xong. Bởi vậy chương trình xử lý ngắt có thể tự do sử dụng 4 thanh ghi AC của S7-200.

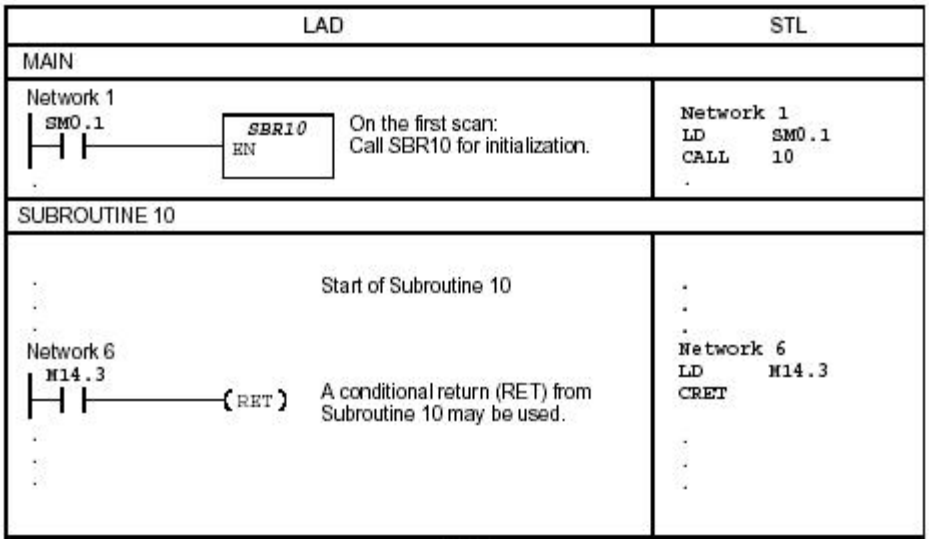
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Jump to Label and Label				
JMP n	n JMP	Lệnh nhảy thực hiện chuyển quyền điều khiển đến nhãn n trong một chương trình.	**n:** CPU 212:0 đến 63 CPU 21x khác từ 0 đến 255.	none
LBL n	n JMP	Lệnh khai báo nhãn n trong một chương trình.		



Hình 52: Ví dụ cách sử dụng lệnh JMP, LBL.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data

				Types
Subroutine and Return Subroutine				
SBR n		Lệnh gọi chương trình con, thực hiện phép chuyển quyền điều khiển đến chương trình con có nhãn n.	**n:** CPU 212:0 đến 15 CPU 21x khác từ 0 đến 255.	none
RET		Lệnh trở về chương trình đã gọi chương trình con không điều kiện.	none	none
CRET		Lệnh trở về chương trình đã gọi chương trình con có điều kiện.		



Hình 53: Ví dụ cách sử dụng lệnh gọi và thoát khỏi chương trình con.

Các lệnh sau sẽ can thiệp vào thời gian vòng quét, nó được dùng để kết thúc chương trình đang thực hiện hoặc kéo dài thêm thời gian của vòng quét.

Trong chương trình chính, kết thúc chương trình bằng lệnh MEND, nhưng trong soạn thảo chương trình chúng ta không cần lệnh kết thúc này mà Step 7 MicroWin đã mặc định rồi. Lệnh END cũng là lệnh kết thúc chương trình nhưng là lệnh kết thúc có điều kiện.

Khi chương trình chính hoặc chương trình con gặp lệnh STOP thì chương trình sẽ kết thúc ngay tại cuối vòng quét hiện thời và CPU chuyển sang chế độ STOP.

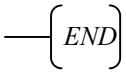
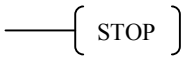
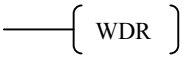
Nếu trong chương trinh xử lý ngắt gặp lệnh STOP thì ngắt cũng được dừng lại ngay lập tức, các tín hiệu xử lý ngắt đang còn nằm trong hàng đợi sẽ bị huỷ bỏ, phần còn lại của chương trình sẽ không thực hiện.Việc thực sự chuyển sang chế độ STOP xảy ra ở cuối chu kỳ vòng quét hiện thời sau giai đoạn xuất tín hiệu cho đầu ra.

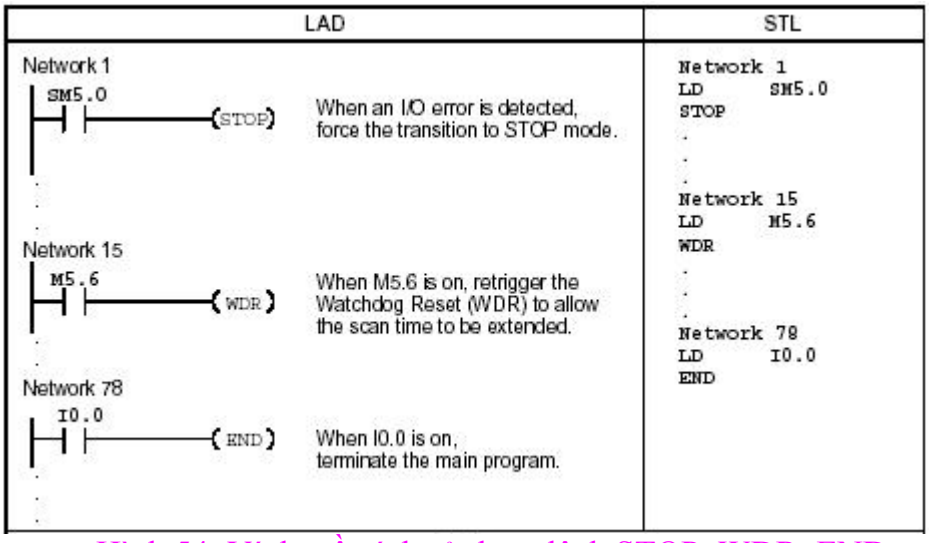
Lệnh WDR sẽ khởi động lại đồng hồ quan sat *(Watchdog Timer),* chương trình tiếp tục thực hiện trong vòng quét ở chế độ quan sat. Nên cẩn thận khi sử dụng lệnh này. Khi trong chương trình sử dụng lệnh lặp, hoặc thời gian trễ quá lớn thì những quá trình sau bị hạn chế:

- Truyền thông (loại trừ kiểu Freeport).
- Cập nhật vào ra (trừ nhẫng lệnh vào ra tức thì).
- Cập nhật cưỡng bức.
- Cập nhật các bit kiểu SM.
- Chuẩn đoán thuời gian chạy.
- Với các vòng quét lớn hơn 25 giây thì các bộ Timer có độ phân giải10ms và 100ms sẽ không được chính xác.

Nếu thời gian của vòng quét lớn hơn 300ms, hoặc khi găpkj một ngắt có chương trình xử lý ngắt với thời gian chạy chương trình lâu hơn 300ms thì cần phải sử cụng lệnh WDR để khởi động lại ***đồng hồ quan sát***.

Việc chuyển công tắc phần cứng sang chế độ STOP hoặc thực hiện lệnh STOP trong chương trình sẽ là nguyên nhân đặt chế độ điều khiển vào chế độ dừng trong khoảng thời gian 1,4s.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
End and Stop and Watchdog Timer				
END		Lệnh kết thúc chương trình hiện hành có đều kiện.	none	none
STOP		Lệnh kết thúc chương trình hiện hành và chuyển sang chế độ STOP.		
WDR		Lệnh khởi động lại đồng hồ quan sát.		



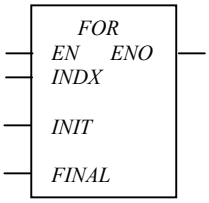
Hình 54: Ví dụ về cách sử dụng lệnh STOP, WDR, END

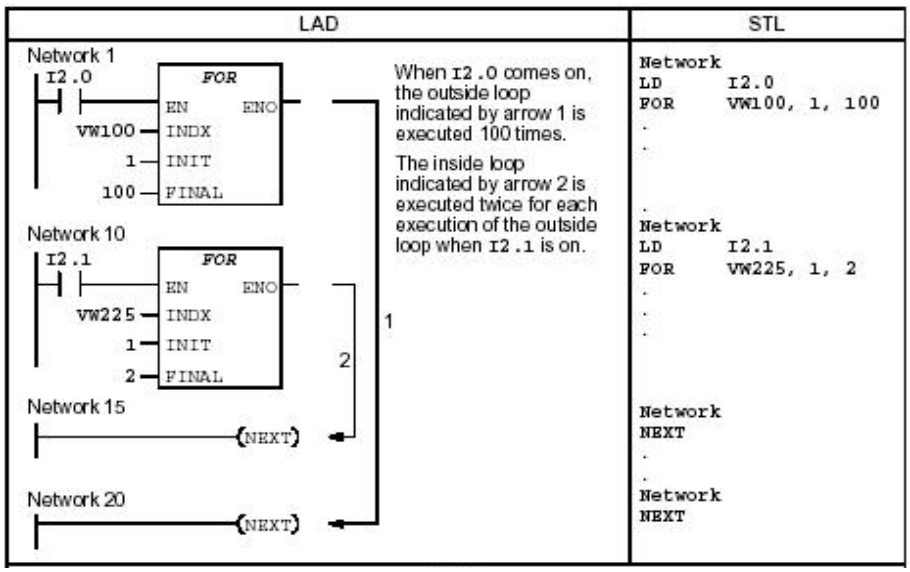
Để xây dựng cấu trúc vòng lặp nhằm thực hiện lặp một khối lệnh riêng biệt trong chương trình. Sử dụng lệnh FOR...NEXT để thiết kế một vòng lặp với số lần có thể định trước bằng hai toán hạng INIT kiểu từ đơn chỉ ***điểm khởi phát*** và FINAL cũng kiểu từ đơn chỉ ***điểm kết thúc***. Ngoài ra lệnh còn sử dụng một từ đơn INDX để lưu ***số vòng lặp tức thời.***

Mỗi một câu lệnh FOR đòi hỏi phải có một câu lệnh NEXT đứng cuối khối lệnh được lặp. Các vòng FOR...NEXT có thể được lồng vào nhau nhưng số lệnh lồng vào nhau không được vượt quá 8 lần.

Tại thời điểm bắt đầu thực hiện lệnh vòng lặp FOR, từ đơn INDX nhận giá trị của INIT. Sau đó, mỗi khi kết thúc một vòng lặp, tức là khi gặp lệnh NEXT, nội dung của INDX được tăng lên 1 đơn vị và được so sánh với nội dung của FINAL. Nếu nội dung của INDX chưa lớn hơn nội dung của FINAL thì chương trình sẽ tiếp tục thực hiện lại vòng lặp, ngược lại khi nội dung của INDX đã lớn hơn nội dung của FINAL thì chương trình sẽ kết thúclệnh FOR...NEXT và tiếp tục thực hiện lệnh kế tiếp nằm ngay sau lệnh NEXT.

Khi lệnh NEXT thực hiện thì bit đầu tiên trong ngăn xếp có giá trị logic bằng 1.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
FOR...NEXT				
FOR INDX, INIT, FINAL		Ví dụ đưa vào INIT giá trị 1, FINAL giá trị là 10. Lệnh sẽ thực hiện lặp đúng 10 lần, số lần lặp được quản lý trong từ đơn INDX. Vợt qúa 10 lần lệnh sẽ kết thúc và chương trình tiếp tục thực hiện các lệnh kế tiếp.	**INDX:** IW, QW, VW, LW, MW,SW, SMW, AC, T, C, *VD, *AC, *LD.	INT
			INIT: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, AIW, Constant, *VD, *AC, *LD.	INT
			FINAL: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, AIW, Constant, *VD, *AC, *LD.	INT
NEXT		Lệnh kết thúc vòng lặp.	none	none



Hình 55: Ví dụ về cách sử dụng lệnh FOR...NEXT.

14. SIMATIC Shift and Rotate Register Instrutions:

Làm việc với thanh ghi có nhóm lệnh sau:

Lệnh dịch chuyển thanh ghi, trong này cũng có hai nhóm:

+ Lệnh dịch chuyển thanh ghi 8 bit, 16 bit, 32 bit.
+ Lệnh dịch chuyển thanh ghi có độ dài tuỳ ý, được định nghĩa trong lệnh.

Lệnh quay vòng thanh ghi, trong này cũng có hai nhóm :

+ Lệnh quay vòng thanh ghi 8 bit, 16 bit, 32 bit.
+ Lệnh quay vòng thanh ghi có độ dài tuỳ ý, được định nghĩa trong lệnh.

Khi sử dụng lệnh dịch chuyển các bit của thanh ghi (Byte, Word, DWord) cần chú ý các điểm sau đây:

1. Không thực hiện việc dich chuyển nếu số lần đẩy bằng 0.
2. Nếu số lần đẩy có giá trị lớn hơn 0, bit nhớ tràn SM1.1 sẽ có giá trị của bit cuối cùng được đẩy ra.
3. Nếu số lần đẩy lớn hơn hoặc bằng 8 đối với byte, 16 đối với Word, 32 đối với từ kép thì lệnh sẽ thực hiện lệnh đẩy lớn nhất chỉ bằng 8, 16, 32.
4. Lệnh SLB (đẩy các bit của byte sang trái), SLW (đẩy các bit của Word sang trái) và SLD (đẩy các bit của từ kép sang trái) sẽ chuyển giá trị 0 vào bit thấp nhất của Byte, Word hoặc DWord sau mỗi lần đẩy. Sau lệnh thực hiện, bit SM1.1 sẽ có giá trị logic của bit thứ 8-N, 16-N hoặc 32-N, trong đó N là số lần đẩy.
5. Lệnh SRB (đẩy các bit của byte sang phải), SRW (đẩy các bit của Word sang phải) và SRD (đẩy các bit của từ kép sang phải) sẽ chuyển giá trị 0 vào bit thấp nhất của Byte, Word hoặc DWord sau mỗi lần đẩy. Sau lệnh thực hiện, bit SM1.1 sẽ có giá trị logic của bit thứ N-1, trong đó N là số lần đẩy.
6. Bit báo kết quả 0 (bit SM1.0) sẽ có giá trị logic bằng 1 nếu như sau khi thực hiện lệnh đẩy nội dung của Byte, Word, DWord bằng 0.

Khi sử dụng lệnh quay vòng các bit của thanh ghi (Byte, Word, DWord) cần chú ý các điểm sau đây:

1. Lệnh quay thực hiệnn phép đẩy vòng tròn sang trái hoặc sang phải các bit của một Byte, Word, DWord. Tại mỗi một lần quay, giá trị của các bit bị đẩy ra ở một đầu của thanh ghi lại được đưa vào đầu kia của thanh ghi đó.
2. Không thực hiện việc quay vòng nếu số lần quay bằng 0. Hay bằng một bội số của 8 (đối với byte), của 16 (đối với word) và của 32 (đối với DWord).
3. Đối với các giá trị của số đếm lần quay lớn hơn 8 (đối với byte), của 16 (đối với word) và của 32 (đối với DWord) lệnh sẽ thực hiện với số đếm lần quay mới bằng phần dư của của phép chia tương ứng.
4. Khi thực hiện lệnh quay sang phải RRB (quay các bit của byte sang phải), RRW (quay các bit của Word sang phải) và RRD (quay các bit của từ kép sang phải), tại mỗi lần quay giá trị của bit thấp nhất được ghi vào bit boá tràn

SM1.1.Sau khi lệnh thực hiện, bit SM1.1 sẽ có giá trị logic của bit thứ 8-N, 16-N hoặc 32-N, trong đó N là số đếm lần quay.

5. Khi thực hiện lệnh quay sang trái RLB (quay các bit của byte sang trái), RLW (quay các bit của Word sang trái) và RLD (quay các bit của từ kép sang trái), tại mỗi lần quay giá trị của bit thấp nhất được ghi vào bit boá tràn SM1.1.Sau khi lệnh thực hiện, bit SM1.1 sẽ có giá trị logic của bit thứ N-1, trong đó N là số đếm lần quay.

6. Bit báo kết quả 0 (bit SM1.0) sẽ có giá trị logic bằng 1 nếu như sau khi thực hiện lệnh quay nội dung của Byte, Word, DWord bằng 0.

Các lệnh dich chuyển hoặc quay vòng ảnh hưởng đến kết quả của các bit đặc biệt như sau:

Lệnh	Kiểu lệnh	SM1.0 (kết quả 0)	SM1.1 (báo tràn)	SM1.2 (kết qủa âm)	SM1.3 (chia cho 0)
SRB	không dấu	có	có	không	không
SLB	không dấu	có	có	không	không
SRW	không dấu	có	có	không	không
SLW	không dấu	có	có	không	không
SRD	không dấu	có	có	không	không
SLD	không dấu	có	có	không	không
RRB	không dấu	có	có	không	không
RLB	không dấu	có	có	không	không
RRW	không dấu	có	có	không	không
RLW	không dấu	có	có	không	không
RRD	không dấu	có	có	không	không
RLD	không dấu	có	có	không	không
SHRB	không dấu	không	có	không	không

Những điều sau đây chỉ đúng với các hàm dịch chuyển bit của byte, từ đơn và từ kép:

+ Nếu bộ đếm chuyển dịch có giá trị lớn hơn 0 thì bit nhớ tràn SM1.1 có giá trị logic của bit cuối cùng được đẩy ra.

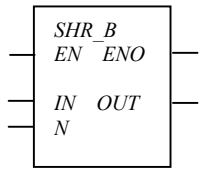
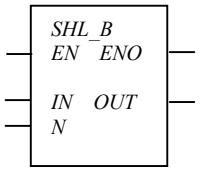
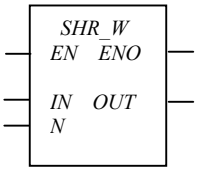
+ Bit báo kết quả 0 SM1.0 có giá trị logic 1 nếu sau khi lệnh được thực hiện, byte, từ hoặc từ kép có nội dung bằng 0.

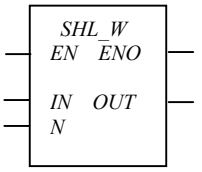
Những điều sau đây chỉ đúng với các hàm dịch chuyển bit của byte, từ đơn và từ kép:

+ Nếu bộ đếm chuyển dịch không phảo là bộ số nguyên của 8, 16, 32 đối với byte, Word, DWordthif giá trị của bit cuối cùng bị đẩy ra ngoài sẽ được gán cho bit nhớ tràn SM1.1.

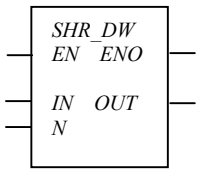
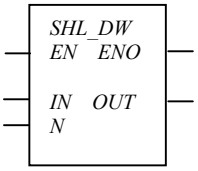
+ Nếu bit báo kết quả 0 có giá trị logic bằng 1 thì giá trị của byte, từ hay từ kép bằng 0.

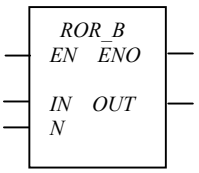
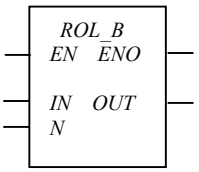
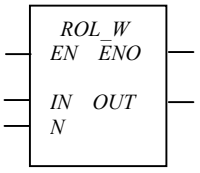
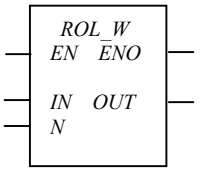
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types

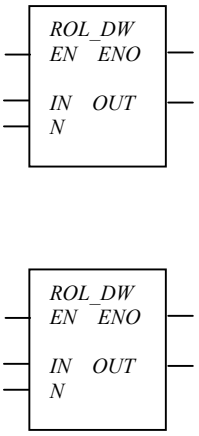
Shift Right Byte and Shift Left Byte				
SRB OUT, N *SLB* OUT, N	 	Lệnh dịch phải hay lệnh dịch trái thực hiện dịch chuyển các bit của Byte đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Lệnh shift điền giá trị zero vào các bit vừa bị dịch chuyển đi, bit cuối cùng bị dịch chuyển ra sẽ được đưa vào bit báo tràn SM1.1. Bit báo kết quả 0 sẽ được set lên 1 nếu giá trị của byte dịch chuyển là 0.	**IN:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD. **OUT:** IB, QB, MB, SMB, VB, SB, LB, AC, ∗VD, ∗AC, ∗LD. **N:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD.	Byte
Shift Right Word and Shift Left Word				
SRW OUT, N		Lệnh dịch phải hay lệnh dịch trái thực hiện dịch chuyển các bit của Word đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Lệnh shift điền	**IN:** IW, QW, VW, LW, MW,SW, SMW, AIW ,AC, T, C, Constant, ∗VD, ∗AC, ∗LD. **OUT:** IW, QW, VW, LW, MW,SW, SMW, AC, T, C, ∗VD, ∗AC,	Word

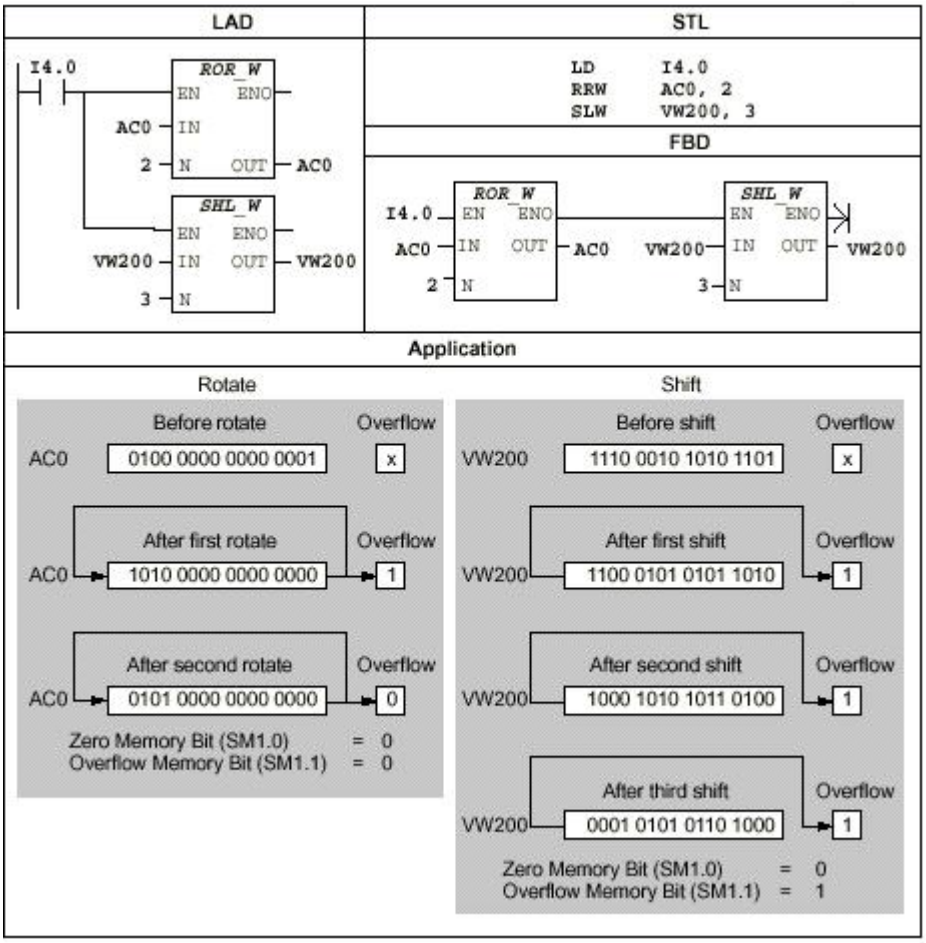
SLW OUT, N		giá trị zero vào các bit vừa bị dịch chuyển đi, bit cuối cùng bị dịch chuyển ra sẽ được đưa vào bit báo tràn SM1.1. Bit báo kết quả 0 sẽ được set lên 1 nếu giá trị của Word dịch chuyển là 0.	AC, T, C, ∗VD, ∗AC, ∗LD. **N:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD.	Byte

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Shift Right Double Word and Shift Left Double Word				

SRD OUT, N *SLD* OUT, N	 	Lệnh dịch phải hay lệnh dịch trái thực hiện dịch chuyển các bit của từ kép đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Lệnh shift điền giá trị zero vào các bit vừa bị dịch chuyển đi, bit cuối cùng bị dịch chuyển ra sẽ được đưa vào bit báo tràn SM1.1. Bit báo kết quả 0 sẽ được set lên 1 nếu giá trị của từ kép dịch chuyển là 0.	**IN:** VD, ID, QD, MD, LD, SD, HC, SMD, AC, Constant, ∗VD, ∗AC, ∗LD. **OUT:** VD, ID, QD, MD, LD, SD, SMD, AC, ∗VD, ∗AC, ∗LD. **N:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD.	DWord Byte
Rotate Right Byte and Rotate Left Byte				

RRB OUT, N *RLB* OUT, N	 	Lệnh quay vòng sang phải hay lệnh quay vòng sang trái thực hiện dịch chuyển các bit của byte đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Tại mỗi lần quay, giá trị của bit cuối cùng (bit 0) được đưa vào bit SM1.1 đồng thời đưa vào bit đầu tiên (bit 7) của byte đó nếu là quay phải, còn ngược lại đối với lệnh quay trái. Bit báo kết quả 0 sẽ có giá trị bằng 1 nếu giá trị trong byte đó bằng 0.	**IN:** IB, QB, MB, SMB, VB, SB, LB, AC, ∗VD, ∗AC, ∗LD. **OUT:** IB, QB, MB, SMB, VB, SB, LB, AC, ∗VD, ∗AC, ∗LD. **N:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD.	Byte
Rotate Right Word and Rotate Left Word				
RRW OUT, N *RLW* OUT, N	 	Lệnh quay vòng sang phải hay lệnh quay vòng sang trái thực hiện dịch chuyển các bit của từ đơn đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Tại mỗi lần quay, giá trị của bit cuối cùng (bit 0) được đưa vào bit SM1.1 đồng thời đưa vào bit đầu tiên (bit 7) của byte đó nếu là quay phải, còn ngược lại đối với lệnh quay trái. Bit báo kết quả 0 sẽ có giá trị bằng 1 nếu giá trị trong từ đơn đó bằng 0.	**IN:** IW, QW, VW, LW, MW,SW, SMW, AIW ,AC, T, C, Constant, ∗VD, ∗AC, ∗LD. **OUT:** IW, QW, VW, LW, MW,SW, SMW, AC, T, C, ∗VD, ∗AC, ∗LD. **N:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD.	Word Byte
Rotate Right Double Word and Rotate Left Double Word				

RRD OUT, N *RLD* OUT, N		Lệnh quay vòng sang phải hay lệnh quay vòng sang trái thực hiện dịch chuyển các bit của từ kép đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Tại mỗi lần quay, giá trị của bit cuối cùng (bit 0) được đưa vào bit SM1.1 đồng thời đưa vào bit đầu tiên (bit 7) của từ kép đó nếu là quay phải, còn ngược lại đối với lệnh quay trái. Bit báo kết quả 0 sẽ có giá trị bằng 1 nếu giá trị trong từ kép đó bằng 0.	**IN:** VD, ID, QD, MD, LD, HC, SMD, AC, Constant, ∗VD, ∗AC, ∗LD. **OUT:** VD, ID, QD, MD, LD, SMD, AC, ∗VD, ∗AC, ∗LD. **N:** IB, QB, MB, SMB, VB, LB, AC, Constant, ∗VD, ∗AC, ∗LD.	DWord Byte



Hình 56: Ví dụ về cách sử dụng lệnh dịch chuyển và quay vòng thanh ghi

<u>Lệnh làm việc với thanh ghi có độ dài tuỳ ý:</u>

Lệnh thuộc nhóm này cung cấp một phương pháp nối tiếp và điều khiển dòng sản phẩm hoặc dữ liệu. Thanh ghi được xác định trong lệnh bởi toán hạng S_BIT chỉ địa chỉ bit thấp của thanh ghi và độ dài là giá trị tuyệt đối của toán hạng N trong lệnh (nghĩa là thanh ghi có độ dài |N| bit). Dữ liệu được chuyển vào trong thanh ghi có tên là DATA (DATA = Bool), một lần trong một vòng quét.

S_BIT là bit thấp nhất của thanh ghi, nếu gọi cao nhất trong thanh ghi là MSB.b thì MSB.b sẽ được tính theo công thức sau:

MSB.b = [(byte của S_BIT) + phần nguyên của(|N| - 1 + bit của S_BIT)/8].[phần còn thừa của phép chia 8]

Lý do trừ đi 1 bởi vì S-BIT đã chiếm mất 1 bit của thanh ghi.

Ví dụ S_BIT là V33.4 và N = 14 thì MSB.b sẽ là:

MSB.b = [(33) + (|14| - 1 + 4)/8].remainder of the division by 8

= (33 + 2).remainder of the division by 8

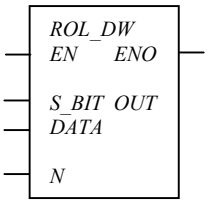
= 35.1

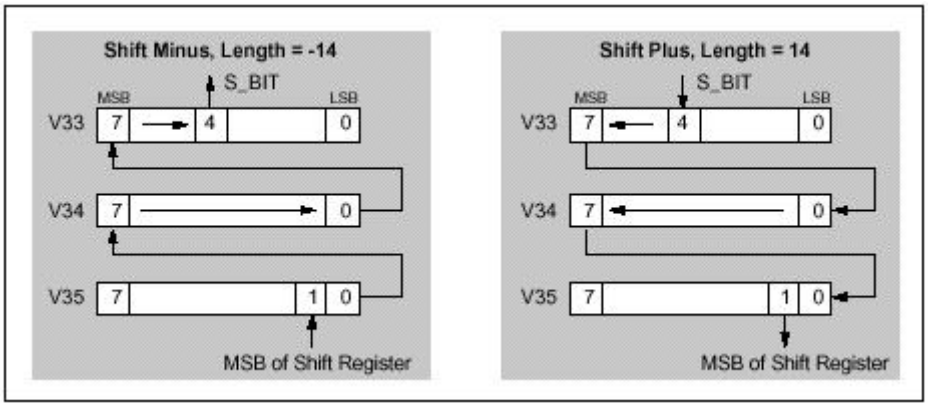
MSB.b là : V35.1

Chiều thực hiện phép dịch chuyển phụ thuộc vào dấu của toán hạng N trong lệnh. Miền giá trị cho phép của toán hạng N là: $-64 \le N \le 64$.

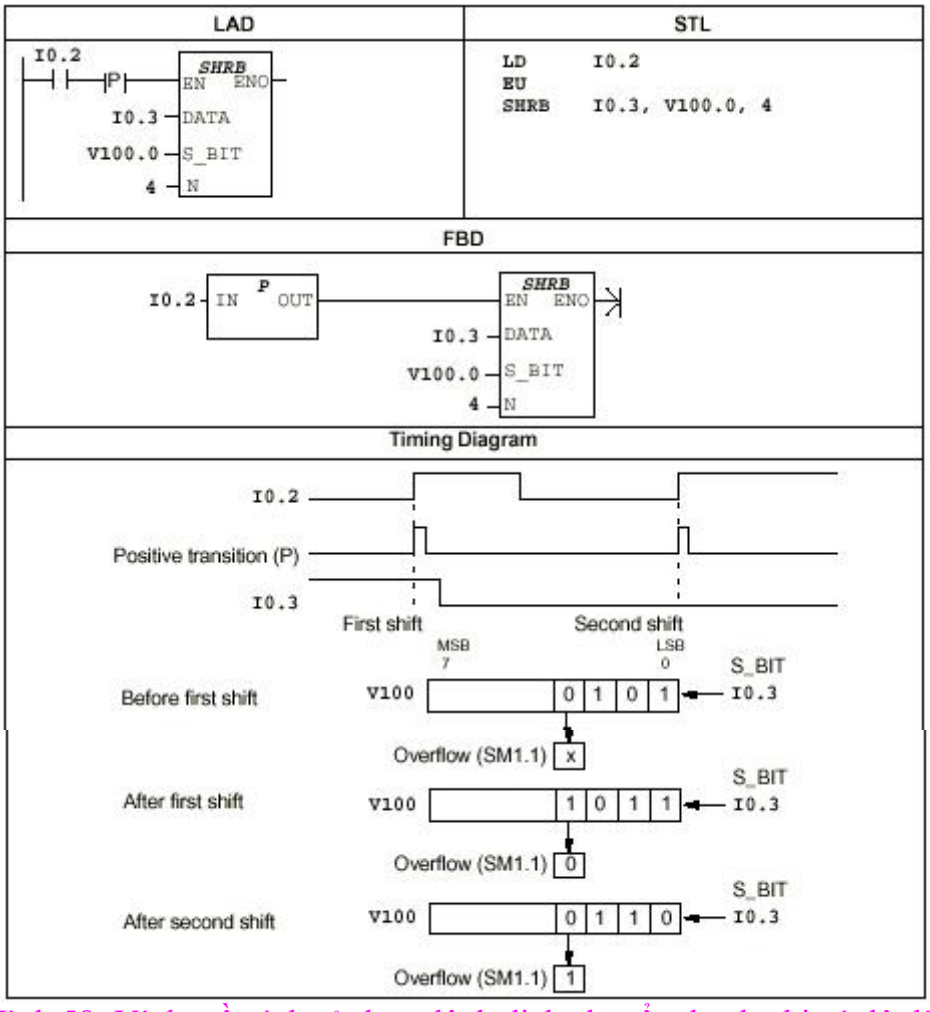
Nếu N dương thì phép dịch chuyển là phép dịch trái, giá trị của DATA được chuyển vào bit thấp nhất, giá trị logic trong bit cao nhất bị đẩy ra ngoài (vào bit báo tràn SM1.1). Ngược lại N là âm thì phép dịch chuyển là phép dịch phải, giá trị của DATA được chuyển vào bit cao nhất, giá trị logic trong bit thấp nhất bị đẩy ra ngoài (vào bit báo tràn SM1.1).

SHRB Lệnh dịch chuyển các bit của thanh ghi một vị trí trong một vòng quét. Thanh ghi được xoá trong lệnh bằng các toán hạng S_BIT chỉ địa chỉ bit thấp trong thanh ghi và |N| chỉ độ dài thanh ghi. Giá trị logic của bit bị đẩy ra khỏi thanh ghi được ghi vào bit báo tràn SM1.1.

STL	LAD	Toán hạng Operands	Kiểu dữ liệu Data Types
Shift Register Bit			
SHRB *DATA, S_BIT, N*		**DATA, S_BIT:** I, Q, V, M, SM, T, C, S, L. **N:** IB, QB, MB, SMB, VB, LB, AC, Constant, *VD, *AC, *LD.	Bool Byte



Hình 57: Mô tả hướng dịch chuyển của thanh ghi với toán hạng âm và dương.



Hình 58: Ví dụ về cách sử dụng lệnh dịch chuyển thanh ghi có độ dài bất kỳ.

15. SIMATIC Interupt and Comunication Instrutions:

Các chế độ ngắt và xử lý ngắt cho phép thực hiện các *quá trình tốc độ cao*, phản ứng kịp thời với các sự kiện ở bên trong và bên ngoài.

Nguyên tắc cơ bản của một chế độ ngắt cũng giống như thực hiện việc gọi một chương trình con, chỉ khác nhau ở đây là chương trình con được *gọi chủ động* bằng lệnh gọi chương trình con CALL, còn chương trình xử lý ngắt được *gọi bị động bằng tín hiệu báo ngắt.*

Khi có một tín hiệu báo ngắt, hệ thống sẽ tổ chức thực hiện gọi và thực hiện chương trình con tương ứng với tín hiệu ngắt đó, hay nói cách khác là hệ thống sẽ tổ chức xử lý tín hiệu báo ngắt đó. Chương trình con này được gọi là *chương trình xử lý ngắt.*

Do việc gọi chương trình xử lý ngắt bằng một tín hiệu báo ngắt mà thời điểm xuất hiện tín hiệu báo ngắt hoàn toàn bị động, bởi vậy hệ thống sẽ phải hỗ trợ thêm cho công việc xử lý ngắt như: cất giữ nội dung ngăn xếp, nội dung thanh ghi AC và các bit nhớ đặc biệt; tổ chức xếp hàng ưu tiên cho các tín hiệu xử lý ngắt trong trường hợp chúng chưa kịp thời xử lý.

Bảng liệt kê các tín hiệu báo ngắt tương ứng với từng loại CPU 21x:

Kiểu ngắt	Mô tả tín hiệu ngắt	CPU 212	CPU 214	CPU 215_2DP	CPU 216
0	Ngắt theo sườn lên của I0.0*	Y	Y	Y	Y
1	Ngắt theo sườn xuống của I0.0*	Y	Y	Y	Y
2	Ngắt theo sườn lên của I0.1		Y	Y	Y
3	Ngắt theo sườn xuống của I0.1		Y	Y	Y
4	Ngắt theo sườn lên của I0.2		Y	Y	Y
5	Ngắt theo sườn xuống của I0.2		Y	Y	Y
6	Ngắt theo sườn lên của I0.3		Y	Y	Y
7	Ngắt theo sườn xuống của I0.3		Y	Y	Y
8	Ngắt để nhận kí tự ở Port 0	Y	Y	Y	Y
9	Ngắt để báo việc truyền dữ liệu đã hoàn tất ở Port 0	Y	Y	Y	Y
10	Ngắt thời gian 0	Y	Y	Y	Y
11	Ngắt thời gian 1		Y	Y	Y
12	Ngắt theo HSC0, khi giá trị tức thời bằng giá trị đặt trước*.	Y	Y	Y	Y
13	Ngắt theo HSC1, khi giá trị tức thời bằng giá trị đặt trước*.	Y	Y	Y	Y
14	Ngắt theo HSC1, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.		Y	Y	Y
15	Ngắt theo HSC1, khi có tín hiệu Reset từ ngoài		Y	Y	Y
16	Ngắt theo HSC2, khi giá trị tức thời bằng giá trị đặt trước*.		Y	Y	Y
17	Ngắt theo HSC2, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.		Y	Y	Y
18	Ngắt theo HSC2, khi có tín hiệu Reset từ ngoài		Y	Y	Y
19	PLS0 Ngắt báo hoàn tất việc đếm xung		Y	Y	Y
20	PLS1 Ngắt báo hoàn tất việc đếm xung		Y	Y	Y
21	Ngắt theo bộ định thời T32, khi giá tức thời CT=PT.			Y	Y
22	Ngắt theo bộ định thời T96, khi giá tức thời CT=PT.			Y	Y
23	Ngắt báo hoàn tất việc nhận 1 gói tin ở Port 0			Y	Y
24	Ngắt báo hoàn tất việc nhận 1 gói tin ở Port 1				Y
25	Ngắt để nhận kí tự ở Port 1				Y
26	Ngắt để báo việc truyền dữ liệu đã hoàn				Y

	tất ở Port 1				
**Nếu khai báo kiểu ngắt 12 (HSC0, PV=CV) thì hai kểu ngắt 0 và 1 bị vô hiệu hoá. Ngược lại, nếu sử dụng kiểu ngắt 0 và 1 thì kiểu ngắt 12 bị vô hiệu hoá.*					

Bảng liệt kê các tín hiệu báo ngắt tương ứng với từng loại CPU 22x:

Kiểu ngắt	Mô tả tín hiệu ngắt	CPU 221	CPU 222	CPU 214, 224XP	CPU 226, 226XM
0	Ngắt theo sườn lên của I0.0	Y	Y	Y	Y
1	Ngắt theo sườn xuống của I0.0	Y	Y	Y	Y
2	Ngắt theo sườn lên của I0.1	Y	Y	Y	Y
3	Ngắt theo sườn xuống của I0.1	Y	Y	Y	Y
4	Ngắt theo sườn lên của I0.2	Y	Y	Y	Y
5	Ngắt theo sườn xuống của I0.2	Y	Y	Y	Y
6	Ngắt theo sườn lên của I0.3	Y	Y	Y	Y
7	Ngắt theo sườn xuống của I0.3	Y	Y	Y	Y
8	Ngắt để nhận kí tự ở Port 0	Y	Y	Y	Y
9	Ngắt để báo việc truyền dữ liệu đã hoàn tất ở Port 0	Y	Y	Y	Y
10	Ngắt thời gian 0, SNB34	Y	Y	Y	Y
11	Ngắt thời gian 1, SMB35	Y	Y	Y	Y
12	Ngắt theo HSC0, khi giá trị tức thời bằng giá trị đặt trước CV=PV.	Y	Y	Y	Y
13	Ngắt theo HSC1, khi giá trị tức thời bằng giá trị đặt trước CV=PV.			Y	Y
14	Ngắt theo HSC1, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.			Y	Y
15	Ngắt theo HSC1, khi có tín hiệu Reset từ ngoài			Y	Y
16	Ngắt theo HSC2, khi giá trị tức thời bằng giá trị đặt trước CV=PV.			Y	Y
17	Ngắt theo HSC2, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.			Y	Y
18	Ngắt theo HSC2, khi có tín hiệu Reset từ ngoài			Y	Y
19	PLS0 Ngắt báo hoàn tất việc đếm xung	Y	Y	Y	Y
20	PLS1 Ngắt báo hoàn tất việc đếm xung	Y	Y	Y	Y
21	Ngắt theo bộ định thời T32, khi giá tức thời CT=PT.	Y	Y	Y	Y
22	Ngắt theo bộ định thời T96, khi giá tức thời CT=PT.	Y	Y	Y	Y
23	Ngắt báo hoàn tất việc nhận 1 gói tin ở Port 0	Y	Y	Y	Y
24	Ngắt báo hoàn tất việc nhận 1 gói tin ở Port 1				Y

25	Ngắt để nhận kí tự ở Port 1				Y
26	Ngắt để báo việc truyền dữ liệu đã hoàn tất ở Port 1				Y
27	Ngắt theo HSC0, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.	Y	Y	Y	Y
28	Ngắt theo HSC0, khi có tín hiệu Reset từ ngoài	Y	Y	Y	Y
29	Ngắt theo HSC4, khi giá trị tức thời bằng giá trị đặt trước CV=PV.	Y	Y	Y	Y
30	Ngắt theo HSC4, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.	Y	Y	Y	Y
31	Ngắt theo HSC4, khi có tín hiệu Reset từ ngoài	Y	Y	Y	Y
32	Ngắt theo HSC3, khi giá trị tức thời bằng giá trị đặt trước CV=PV.	Y	Y	Y	Y
33	Ngắt theo HSC5, khi giá trị tức thời bằng giá trị đặt trước CV=PV.	Y	Y	Y	Y

Thứ tự ưu tiên (priority) và hàng đợi (Queuing) của các kiểu ngắt:

Thứ tự ưu tiên của các kiểu ngắt khác nhau đã được cứng hoá từ trước theo nguyên tắc tín hiệu nào có trước thì xử lý trước. Nếu cùng một lúc có nhiều tín hiệu báo ngắt thì hệ thống sẽ sắp hàng đợi theo thứ tự ưu tiên sau:

Nhóm ngắt truyền thông (nối tiếp).

Nhóm ngắt vào ra(kể cả ngắt cho bộ đếm HSC và ngắt truyền xung).

Nhóm các tín hiệu báo ngắt thời gian.

Tại mỗi thời điểm chỉ có 1 chương trình xử lý ngắt được thực hiện. Cũng nói thêm rằng, nhóm ngắt truyền thông có vị trí ưu tiên cao nhất và ngắt thời gian có vị trí ưu tiên thấp nhất nhưng khi hệ thống đang xử lý ngắt thời gian mà có tín hiệu báo nhắt thời gian thì hệ thống vẫn tiếp tục xử lý đến khi kết thúc mpúi tiếp tục xử lý ngắt truyền thông.

Bảng hàng đợi lớn nhất mà từng CPU có thể có:

Nhóm ưu tiên	212	214	215	216	221	222	224	226
Ngắt truyền thông	4	4	4	8	4	4	4	8
Ngắt vào ra	4	16	16	16	16	16	16	16
Ngắt thời gian	2	4	8	8	8	8	8	8

Riêng đối với tín hiệu báo ngắt truyền thông, mặc dù chưa được xử lý, nhưng kí tự nhận được cùng bit kiểm tra chẵn lẻ vẫn được ghi nhớ lại trong bộ đệm kèm theo đúng thứ tự của tín hiệubáo ngắt.

bit Start	7 hoặc 8 bit của kí tự	Parity	Stop

Khi hàng đợi đã đầy thì bit báo tràn tương ứng cho từng nhóm ngắt sẽ set lên 1:

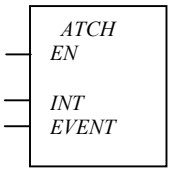
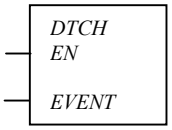
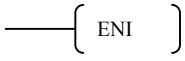
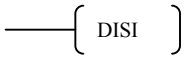
Nhóm ưu tiên	Bit báo tràn
Ngắt truyền thông	SM4.0
Ngắt vào ra	SM4.1
Ngắt thời gian	SM4.2

Cùng với việc chuyển vào chế độ RUN của PLC, tất cả các chế độ ngắt trước đã khai báo trước đó sẽ tự động huỷ (vô hiệu hoá). Nó được kích lại bằng lệnh ENI (kích ngắt toàn cục).

Khai báo một chế độ ngắt phải thực hiện hai việc:

1. Kích tín hiệu báo ngắt cho chế độ ngắt tương ứng (bằng cách khai báo tạ toán hạng EVENT) bằng lệnh ATCH.
2. Sau đó soạn thảo nội dung của chương trình ngắt trong khối INT_x.

Có thể gọp nhiều tín hiệu báo ngắt vào cùng một chương trình (chính hoặc con) nhưng một tín hiệu báo ngắt chỉ có duy nhất một chương trình xử lý ngắt. Khi huỷ tín hiệu ngắt bằng lệnh DISI thì các ngắt vẫn tiếp tục nằm vào hàng đợi cho đến khi chúng được kích klại bằng lệnh ENI.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Attach Interupt				
ATCH INT, EVENT		Lệnh khai báo ngắt mã hiệu INT (khối ngắt), Kiểu ngắt EVENT	**INT:** 0 ÷ 127 **EVENT:** xem bảng liệt kê các tín hiệu báo ngắt tương ứng với từng loại CPU	Byte
Detach Interupt				
DTCH EVENT		Lệnh huỷ ngắt cục bộ tương ứng với kiểu ngắt EVENT.	**EVENT:** xem bảng liệt kê các tín hiệu báo ngắt tương ứng với từng loại CPU	Byte
Enable Interupt				
ENI		Lệnh kích ngắt toàn cục.	none	none
Disable Interupt				
DISI		Lệnh huỷ tất cả các ngắt cùng một lúc.	none	none
Conditional Return from Interupt				
CRETI		Lệnh thoát tức thời khỏi chương trình	none	none

		ngắt khi chương trình ngắt chưa kết thúc.		
Return from Interupt				
RETI	—{ RETI }—	Lệnh kết thúc chương trình xử lý ngắt, ở cuối chương trình.	none	none

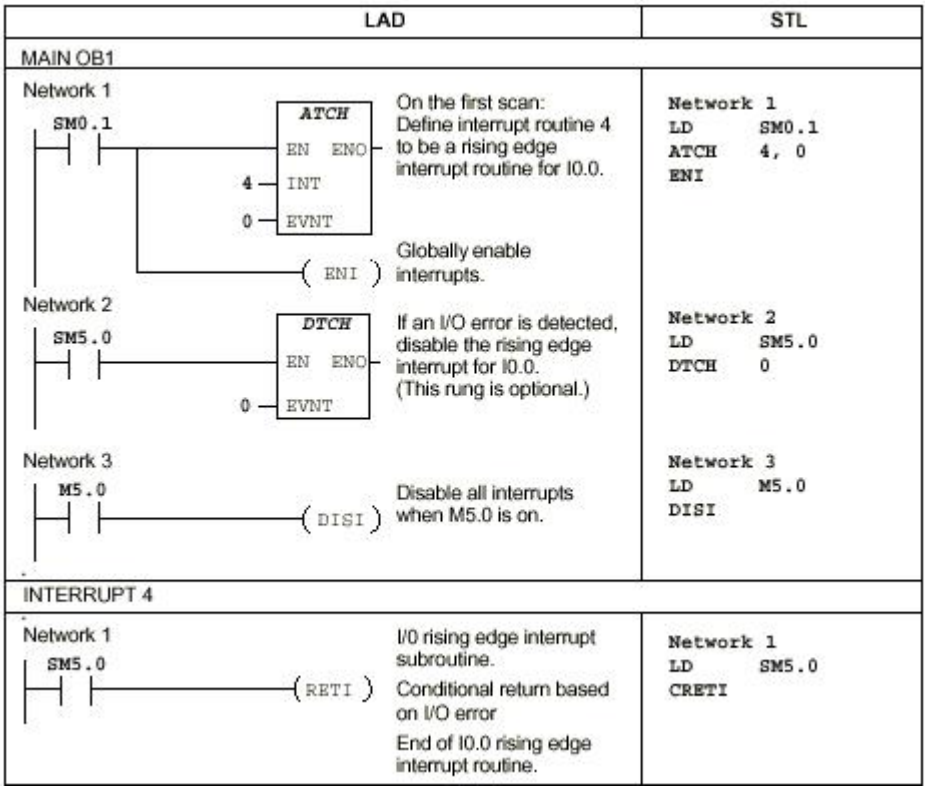
Chương trình xử lý ngắt:

Cũng như chương trình con, mỗi chương trình xử lý ngắt có một nhãn riêng được đánh dấu tại điểm đầu của chương trình. Nhãn náy đwocj khsi báo bằng lệnh INT.

Tất cả các lệnh nằm giữa nhãn cảu chương trình xử lý ngắt và lệnh quay về không điều kiện RETI của chương trình xử lý ngắt đều thuộc về nội dung của chương trình xử lý ngắt. Có thể kết thúc chương trình xử lý ngắt sớm hơn bằng lệnh CRETI, nhưng lệnh RETI vẫn là lệnh kết thúc của chương trình xử lý ngắt. Nhưng lệnh này không không cần khai báo vì chương trình STEP đã tự động khai báo giống như lệnh MEND (kết thúc chương trình chính), lệnh RET (lệnh kết thúc chương trình con).

Chương trình xử lý ngắt cần phải được viết tối ưu, càng nhanh càng tốt, không nên thực hiện chương trình xử lý ngắt quá lâu.

Không được sử dụng các lệnh sau trong CTXLN: DISI, ENI, CALL, HDEF, FOR...NEXT, END.



Hình 59: Ví dụ về cách tổ chức một chương trình xử lý ngắt.

Ngắt tryền thôngnối tiếp:

Cổng truyền thông nối tiếp của PLC có thể điều khiển bằng chương trình viết trong LAD, STL. Chương trình điều khiển này gọi là đèiu khiển cổng tự do (*Freeport Control*). Trước khi thực hiện quá trình truyền thông, các vấn đề sau đây cần phải được thực hiện:

Kiểu biên bản truyền/nhận (giao thức truyền_Protocol).

Tốc độ truyền/nhận tín hiệu.

Số bit được truyền cho 1 kí tự (7 or 8 bit).

Chế độ kiểm tra lỗi (cho kí tự nhận) chẵn lẻ Parity.

Tất cả các vấn đề này đwocj định nghĩa trong byte đặc biệt SMB30 sau:

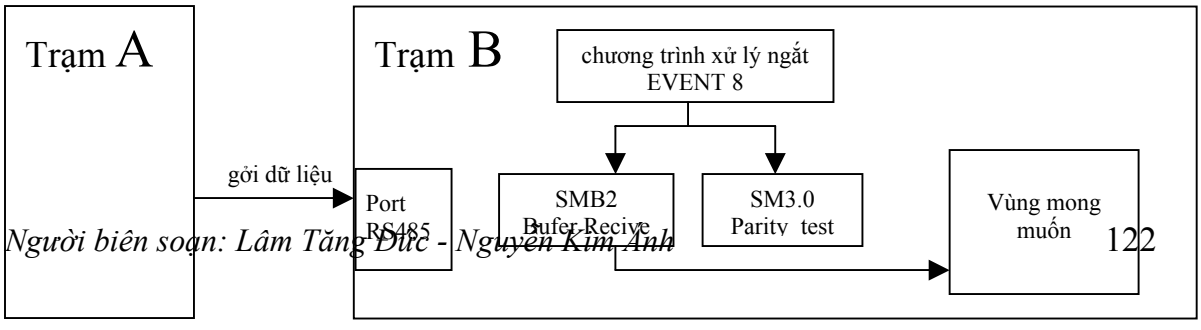
Port 0	Port 1	Description
Format of SMB30	Format of SMB130	MSB 7 … LSB 0: p p d b b b m m — Freeport mode control byte
SM30.6 and SM30.7	SM130.6 and SM130.7	pp: Parity select 00 = no parity 01 = even parity 10 = no parity 11 = odd parity
SM30.5	SM130.5	d: Data bits per character 0 = 8 bits per character 1 = 7 bits per character
SM30.2 to SM30.4	SM130.2 to SM130.4	bbb: Freeport Baud rate 000 = 38,400 baud 001 = 19,200 baud 010 = 9,600 baud 011 = 4,800 baud 100 = 2,400 baud 101 = 1,200 baud 110 = 600 baud 111 = 300 baud
SM30.0 and SM30.1	SM130.0 and SM130.1	mm: Protocol selection 00 = Point-to-Point Interface protocol (PPI/slave mode) 01 = Freeport protocol 10 = PPI/master mode 11 = Reserved (defaults to PPI/slave mode)
Note: One stop bit is generated for all configurations.		

Hình 60: Mô tả byte định nghĩa việc truyền thông nối tiếp.

! Khi truyền thông ở chế độ Freeport thì PLC không làm việc với máy lập trình PG.

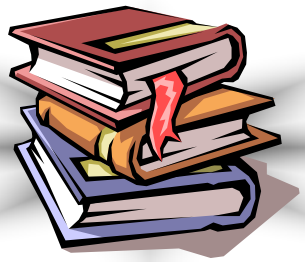
- Byte SMB2 làm bộ đệm ghi nhớ kí tự nhận được
- Bit SM3.0 dùng để kiểm tra lỗi chẵn lẻ kí tự nhận được, nếu có lỗi chẵn lẻ được phát hiện thì SM3.0 set lên 1.
- Sử dụng để thông báo việc truyền thông đã hoàn tất.

Các vấn đề về gởi/nhận message được mô tả như sau:



Hình 61: Mô tả cách nhận message của PLC.

ĐIỀU KHIỂN LOGIC VÀ PLC



TS. NGUYỄN NHƯ HIỀN, TS. NGUYỄN MẠNH TÙNG

ĐIỀU KHIỂN LOGIC VÀ PLC

Sách chuyên khảo dùng cho đào tạo Đại học và Sau đại học ngành Điều khiển & Tự động hoá

NHÀ XUẤT BẢN KHOA HỌC TỰ NHIÊN VÀ CÔNG NGHỆ
HÀ NỘI - 2007

MỤC LỤC

Nội dung *Trang*

PHẦN 1 : LOGIC HAI TRẠNG THÁI VÀ ỨNG DỤNG

CHƯƠNG 1 : LÝ THUYẾT CƠ SỞ

§1.1 Những khái niệm cơ bản

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 của nó 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 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, 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 trạng thái của sự vật hoặc hiện tượng, giá trị 1 đặc trưng cho trạng thái đối lập của sự vật và hiện tượng đó. 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 .

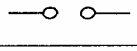
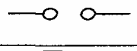
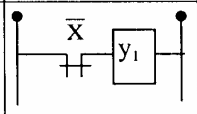
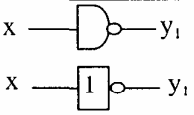
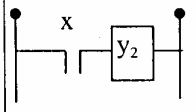
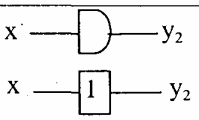
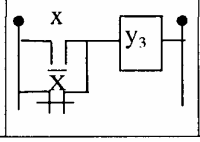
2. Các hàm logic cơ bản

Một hàm $y = f(x_1, x_2, \ldots x_n)$ 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_o, 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

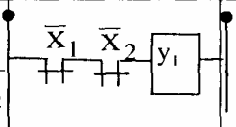
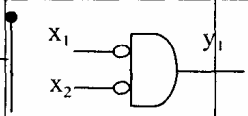
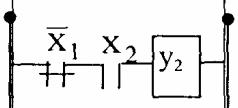
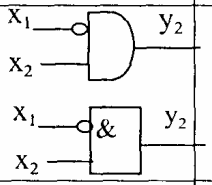
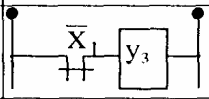
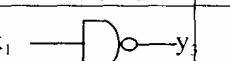
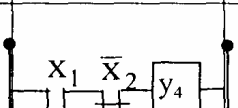
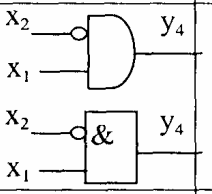
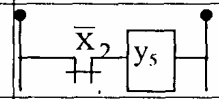
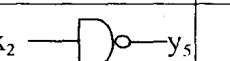
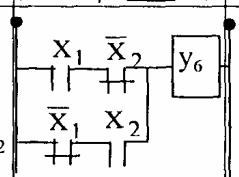
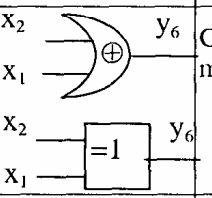
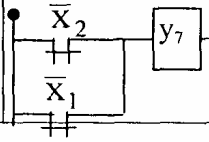
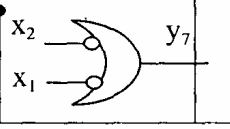
Tên hàm	Bảng chân lý			Thuật toán logic	Ký hiệu sơ đồ		Ghi chú
	x	0	1		Kiểu rơle	Kiểu khối điện tử	
Hàm không	y_0	0	0	$y_0 = 0$ $y_0 = x\bar{x}$			
Hàm đảo	y_1	1	0	$y_1 = \bar{x}$			
Hàm lặp (YES)	y_2	0	1	$y_2 = x$			
Hàm đơn vị	y_3	1	1	$y_3 = 3$ $y_3 = x + \bar{x}$			

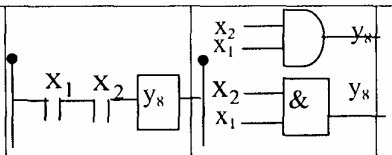
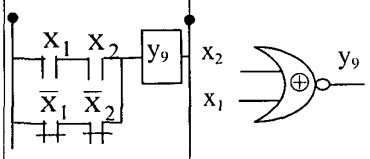
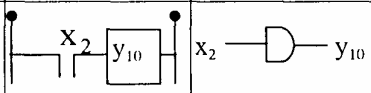
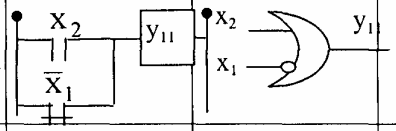
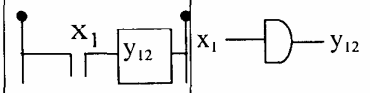
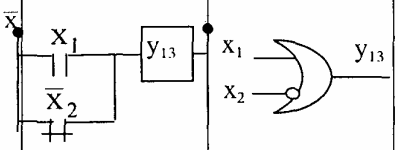
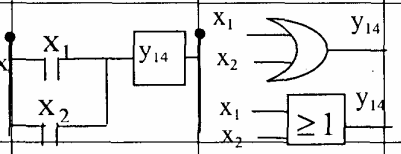
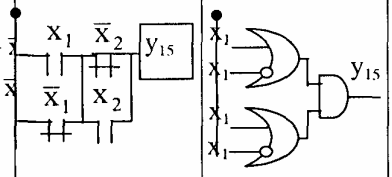
Trong các hàm trên hai hàm y_o và y_3 luôn có giá trị không đổi nên ít được quan tâm, thường chỉ xét hai hàm y_1 và y_2

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	$y_2 = \bar{x}_1 x_2$			
Hàm đảo x_1	y_3	0	0	1	1	$y_3 = \bar{x}_1$			
Hàm cấm x_2 INHIBIT x_2	y_4			x		$y_4 = x_1\bar{x}_2$			
Hàm đảo x_2	y_5	0	1	0	1	$y_5 = \bar{x}_2$			
Hàm hoặc loại trừ XOR	y_6	0	1	1	0	$y_6 = x_1\bar{x}_2 + \bar{x}_1 x_2$			Cộng module
Hàm Chef-fer	y_7	0	1	1	1	$y_7 = \bar{x}_1 + \bar{x}_2 = \overline{x_1 x_2}$			

Hàm và AND	y_8	1	0	0	0	$y_8 = x_1x_2$			
Hàm cùng dấu	y_9					$y_9 = x_1x_2 + \bar{x}_1\bar{x}_2$			
Hàm lặp x_2	y_{10}					$y_{10} = x_2$			Chỉ phụ thuộc x_2
Hàm kéo theo x_2	y_{11}					$y_{11} = \bar{x}_1 + x_2$			
Hàm lặp x_1	y_{12}					$y_{12} = x_1$			Chỉ phụ thuộc x_1
Hàm kéo theo x_1	y_{13}					$y_{13} = x_1 + \bar{x}$			
Hàm hoặc OR	y_{14}	1	1	1	0	$y_{14} = x_1 + x$			
Hàm đơn vị	y_{15}	1	1	1	1	$y_{15} = (x_1 +$ $(x_2 + \bar{x}$			Hàm luôn bằng 1

Các hàm đối xứng nhau qua trục nằm giữa giữa bảng 1.2 là: y_7 và y_8, nghĩa là

$$y_0 = \bar{y}_{15},\ y_1 = \bar{y}_{14} \cdots$$

Hàm logic n biến $y = f(x_1, x_{2,} \ldots x_n)$

Với hàm logic n biến, mỗi biến nhận một trong hai giá trị 0 hoặc 1 nên với hàm logic n biến 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 22. Do đó, 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 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.

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).

4. Tính chất và một số hệ thức cơ bản

4.1. 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:

$$x_1 + x_2 = x_2 + x_1$$

+ 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:

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

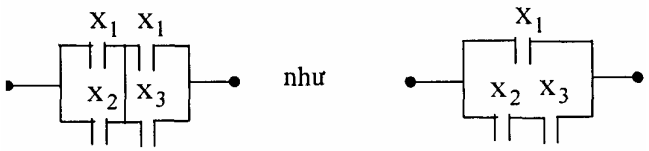
$$x_1 + x_2.x_3 = (x_{1+}x_2) \,.\, (x_{1+}x_3)$$

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. Thể hiện luật phân phối

+ Luật nghịch đảo:

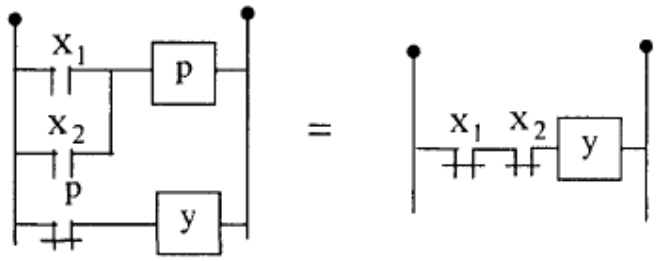
$$\overline{x_1.x_2} = \overline{x}_1 + \overline{x}_2 ; \quad \overline{x_1 + x_2} = \overline{x}_1.\overline{x}_2$$

Cũng minh hoạ tính đúng đắn của luật nghịch đảo bằng cách thành lập bảng 1.4.

Bảng 1.4

x_1	x_2	$\overline{x}_1$	$\overline{x}_2$	$\overline{x_1 + x_2}$	$\overline{x}_1.\overline{x}_2$	$\overline{x}_1 + \overline{x}_2$	$\overline{x_1.x_2}$
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. Thể hiện luật nghịch đảo

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

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

4.2. Các hệ thức cơ bản

Một số hệ thức cơ bản thường dùng trong đại số logic được cho ở bảng 1.5.

Bảng 1.5

1	$x + 0 = x$	10	$x_1.x_2 = x_2.x_1$
2	$x.1 = x$	11	$x_1 + x_1x_2 = x_1$
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. Phương pháp biểu diễn bằng bảng trạng thái

Ở phương pháp này các giá trị của hàm được trình bày trong một bảng. Nếu hàm có n biến thì bảng có n + 1 cột (n cột cho biến và 1 cột cho hàm) và 2^n hàng tương ứng với 2^n 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.

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.

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 $\overline{x_i}$. 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 trên, có hàm ở dạng tổng chuẩn đầy đủ là:

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

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 $\overline{x_i}$. 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 trên, có hàm ở dạng tích chuẩn đầy đủ là:

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

4. Phương pháp biểu diễn bằng bỏng Karnaugh (bìa canô)

Nguyên tắc xây dựng bảng Karnaugh là:

- Để biểu diễn hàm logic n biến cần thành lập một bảng có 2^n 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 l: 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

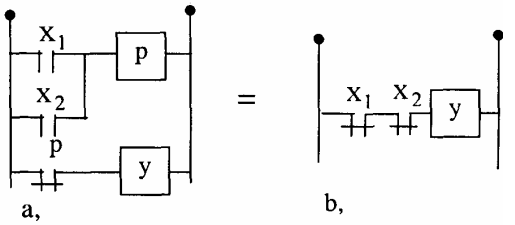
Ví dụ 2: Bảng Karnaugh cho hàm bốn biến như bảng 1.8 sau:

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, 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 sẽ có số biến (thiết bị) cũng như các kết nối (thiết bị) 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 1.3a và hình 1.3b 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, trong khi đó sơ đồ b chỉ cần 2 tiếp điểm, không cần rơle trung gian.



Hình 1.3: Tối giản hàm logic

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

Ở phương pháp này cần dựa vào các tính chất và các hệ thức cơ bản của đại số Boole 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

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

2. Phương pháp tối thiểu hoá hàm logic dùng thuật toán

Phương pháp 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.

Quy tắc của phương pháp là: nếu có 2^n ô có giá trị 1 nằm kề nhau hợp thành một khối vuông hay chữ nhật thì có thể thay 2^n 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 l).

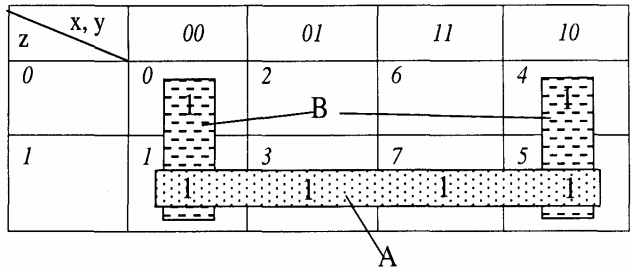
Quy lắc này áp dụng theo thứ tự giảm dần độ lớn các ô, sao cho cuối cùng toàn bộ các ô chứa giá trị 1 đều được bao phủ. Cũng có thể tiến hành tối thiểu theo giá trị 0 của hàm nếu số lượng của nó ít hơn nhiều so với giá trị 1, lúc bấy giờ hàm là hàm phủ định.

Ví dụ: Tối thiểu hàm

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

+ Lập bảng Karnaugh được như bảng 1.9. Bảng Karnaugh có 3 biến với 6 mintec có giá trị 1.

Bảng 1.9



+Tìm nhóm các ô (hình chữ nhật) chứa các ô có giá trị bằng 1, 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 $\bar{y}$ không đổi vậy mintec mới B chỉ còn biến $\bar{y}$: B = $\bar{y}$.

Kết quả tối thiểu hoá là: f = a+b = z+$\bar{y}$.

Phương pháp Quine Mc. Cluskey

Đây là phương pháp có tính tổng quát, cho phép tối thiểu hoá mọi hàm logic với số lượng biến lớn.

a. Một số định nghĩa

+ Đỉnh: là một tích chứa đầy đủ các biến của hàm, nếu hàm có n biến thì đỉnh là tích của n biến.

Đỉnh 1 là đỉnh mà hàm có giá trị bằng 1 .

Đỉnh 0 là đỉnh mà hàm có giá trị bằng 0.

Đỉnh không xác định là đỉnh mà tại đó hàm có thể lấy một trong hai giá trí 0 hoặc 1 .

+ Tích cực tiểu: là tích có số biến là cực tiểu để hàm có giá trị bằng 1 hoặc không xác định.

+ Tích quan trọng: là tích cực tiểu mà giá trị hàm chỉ duy nhất bằng 1 ở tích này.

b. Tối thiểu hoá bằng phương pháp Quine Mc. Cluskey

Để rõ phương pháp hãy xét ví dụ minh hoạ, tối thiểu hoá hàm $f(x_1,x_2,x_3,x_4)$ Với Các đỉnh bằng 1 là L = 2, 3, 7, 12, 14, 15 và các đỉnh có giá trị hàm không xác định là N = 6, 13. Các bước tiến hành như sau:

Bước 1: Tìm các tích cực tiểu

- Lập bảng biểu diễn các giá trị hàm bằng 1 và các giá trị không xác định ứng với mã nhị phân của các biến theo thứ tự số số 1 tăng dần (bảng 1.10a.
- Xếp thành từng nhóm theo số lượng chữ số 1 với thứ tự tăng dần. (bảng 1.10b có 4 nhóm: nhóm 1 có 1 số chứa 1 chữ số 1 ; nhóm 2 gồm 3 số chứa 2 chữ số 1 ; nhóm 3 gồm 3 số chứa 3 chữ số 1, nhóm 4 có 1 số chứa 4 chữ số 1).
- So sánh mỗi tổ hợp thứ i với tổ hợp thứ i + 1, nếu hai tổ hợp chỉ khác nhau ở một cột thì kết hợp 2 tổ hợp đó thành một tổ hợp mới, đồng thời thay cột số khác nhau của 2 tổ hợp cũ bằng một gạch ngang (-) và đánh dấu v vào hai tổ hợp cũ (bảng 1.10c). Về cơ sở toán học, ở đây để thu gọn các tổ hợp đã dùng tính chất:

$$xy + x\bar{y} = x\,.$$

- Cứ tiếp tục c ông việc, từ bảng 1.10c chọn ra các tổ hợp chỉ khác nhau 1 chữ số 1 và có cùng vị trí gạch ngang (-) trong một cột, nghĩa là có cùng biến vừa được giản ước ở bảng 1.10c, như vậy có bảng 1.10d.

Bảng 1.10

a		b			c		d	
Số thập phân	Cơ số 2 $x_1x_2x_3x_4$	Số chữ số 1	Số thập phân	Cơ số 2 $x_1x_2x_3x_4$	Liên kết	$x_1x_2x_3x_4$	Liên kết	$x_1x_2x_3x_4$
2	0010	1	2	0010v	2,3	001-v	2,3,6,7 2,6,3,7	0-1-
3	0011	2	3	0011v	2,6	0-10v	6,7,14,15 6,14,7,15	-11-
6 *	0110		6	0110v	3,7	0-11v	12,13,14,15	11- -
12	1100		12	1100v	6,7	011-v		
7	0111	3	7	0111v	6,14	-110v		
13 *	1101		13	1101v	12,13	110-v		
14	1110		14	1110v	12,14	110v		
15	1111	4	15	1111v	7,15	-111v		
					13,15	11-1v		
					14,15	111-v		

Quá trình tiếp tục cho đến khi không còn khả năng kết hợp nữa. Các tổ hợp tìm được ở bảng 1.10d là tổ hợp cuối cùng, các tổ hợp này không còn khả năng kết hợp nữa, đây chính là các tích cực tiểu của hàm đã cho. Theo thứ tự $x_1x_2x_3x_4$, các x_k ở vị trí có dấu (-) được lược bỏ, các x_k ở vị trí giá trị 0 được lấy nghịch đảo, các tích cực tiểu trong ví dụ được viết như sau:

0-1- (phủ các đỉnh 2, 3, 6, 7) ứng với: $\bar{x}_1x_3$.

-11- (phủ các đỉnh 6, 7, 14, 15) ứng với: x_2x_3.

1 1- - (phủ các đỉnh 12, 13, 14, 15) ứng với : x_1x_2.

Bước 2: Tìm các tích quan trọng

Việc tìm các tích quan trọng cũng được tiến hành theo các bước nhỏ.

Gọi L_i là tập các đỉnh 1 đang xét ở bước nhỏ thứ i, lúc này không quan tâm đến các đỉnh có giá trị không xác định nữa.

Z_i là tập các tích cực tiểu đang ở bước nhỏ thứ i.

E_i là tập các tích quan trọng ờ bước nhỏ thứ i.

Với i = 0

$$L_0 = (2,3,7,12,14,15)$$

$$Z_0 = (\bar{x}_1x_3, x_2x_3, x_1x_2)$$

Xác định các tích quan trọng E_o từ tập L_o và Z_o như sau:

+ Lập bảng trong đó mỗi hàng ứng với một tích cực tiểu thuộc Z_o mỗi cột ứng với một đỉnh thuộc L_o. Đánh dấu "x" vào các ô trong bảng ứng với tích cực tiểu bảng 1.11 (tích $\bar{x}_1x_3$ ứng với các đỉnh 2, 3, 7; tích x_2x_3 ứng với các đỉnh 7, 14, 5; tích x_1x_2 ứng với các đỉnh 12, 14, 15 bảng 1.10).

Bảng 1.11

Z_0 \ L_0	*2*	*3*	*7*	*12*	*14*	*15*
$\bar{x}_1x_3$	(x)	(x)	x			
x_2x_3			x		x	x
x_1x_2				(x)	x	x

Xét từng cột, cột nào chỉ có một dấu "x" thì tích cực tiểu (hàng) ứng với nó là tích quan trọng, đổi thành dấu "(x)". Vậy tập các tích quan trọng ở bước này là:

$$E_0 = (\bar{x}_1x_3, x_1x_2)$$

- Với i = 1

Tìm L_1 từ L_o bằng cách loại khỏi L_o các đỉnh 1 của E_o

Tìm Z_1 từ Z_o bằng cách loại khỏi Z_o các tích trong E_o và các tích đã nằm trong

hàng đã được chọn từ E_o. Khi đã tìm được L_1, và Z_1, làm lại như bước i = 0 sẽ tìm được tích quan trọng E_1.

Công việc cứ tiếp tục cho đến khi $L_k = 0$.

Trong ví dụ này vì $E_o = (\overline{x_1}\overline{x_3}, x_1 x_2)$ mà các định 1 của $\overline{x_1}\overline{x_3}$ là 2, 3, 7; các đỉnh 1 của x_1, x_2 là 12, 14, 15 (bỏ qua đỉnh 6, 13 là các đỉnh không xác định); do đó $L_1 = 0$, quá trình kết thúc. Kết quả dạng hàm tối thiểu chính là tổng của các tích cực tiểu. Vậy hàm cực tiểu là:

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

§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 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:

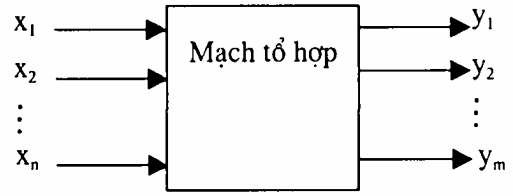
+ Mạch logic tổ hợp.

+ Mạch logic trình tự.

1. 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 bởi trạng thái tín hiệu đầu ra.

Sơ đồ mạch logic tổ hợp như hình 1.4.



Hình 1.4. Mạch tổ hợp

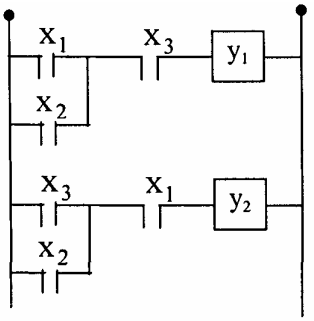
Với mạch logic tổ hợp tồn tại hai loại bài toán là bài toán phân tích và bài toán tổng hợp.

+ 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.

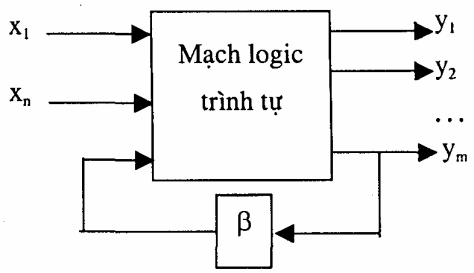
+ Bài toán tổng hợp thực chất là thiết kế mạch tổ hợp. Nhiệm vụ chính là thiết kế được mạch tổ hợp thoả mãn yêu cầu kỹ thuật nhưng mạch phải tối giản. Bài toán tổng

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 các 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: Mạch logic tổ hợp như hình 1.5.



Hình 1.5. Sơ đồ tổ hợp



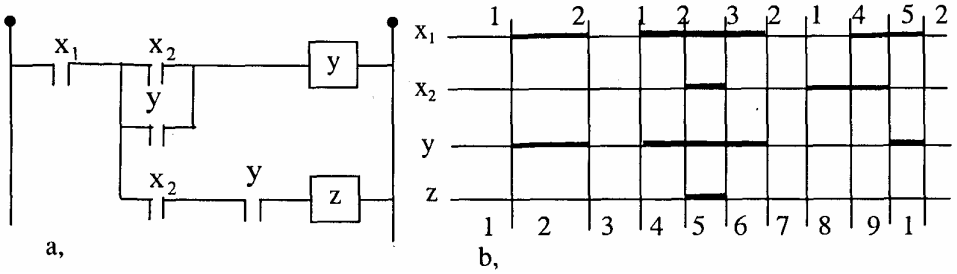
Hình 1.6. Mạch trình tự

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 mạch 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.

Xét mạch logic trình tự như hình 1.7. 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.



Hình 1.7. Sơ đồ mạch trình tự

Từ biểu đồ hình l.7b 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 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 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 cũng có bài toán phân tích và bài toán tổng hợp.

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

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

Trong dây chuyền sản xuất công nghiệp, các thiết bị máy móc thường hoạt động theo một trình tụ 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 khiển 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 chuyền có rất nhiều loại, khi thiết kế 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 chuyền 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 chuyền hoạt động lúc có hư hỏng và sẵn sàng chấp nhận lại diều 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 phận thao tác...).

Khi thiết kế hệ thống phải tính đến các phương 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.

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.

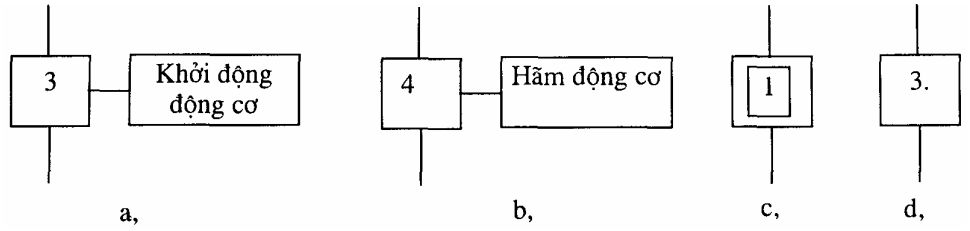
Mạng grafcet cho một quá trình sản xuất luôn luôn là một đồ hình khép kín từ trạng thái đầu đến trạng thái cuối và từ trạng thái cuối về trạng thái đầu.

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 l.8a và b. Một trạng thái có thể tương ứng với một hoặc nhiều hành động của quá trình sản xuất

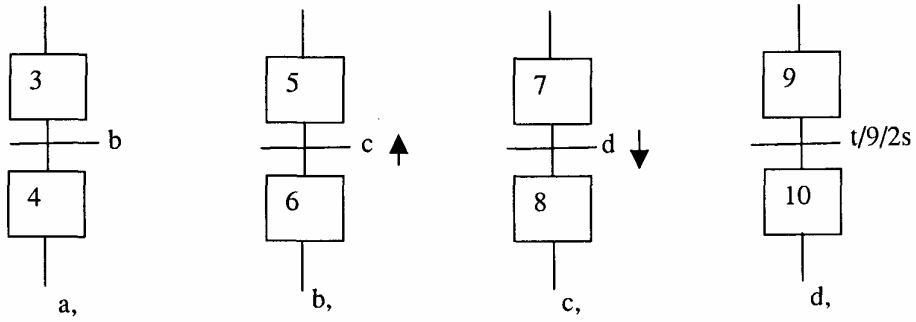
Trạng thái khởi động được thể hiện bằng 2 hình vuông lồng vào nhau, thứ tự thường là 1 hình l.8c.

- Trạng thái hoạt động (tích cực) có thêm dấu ở trong hình vuông trạng thái hình l.8d.



Hình 1.8. Các trạng thái trong grafcet

- 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 l.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. Điều kiện chuyển tiếp

- 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 1.1a và b .

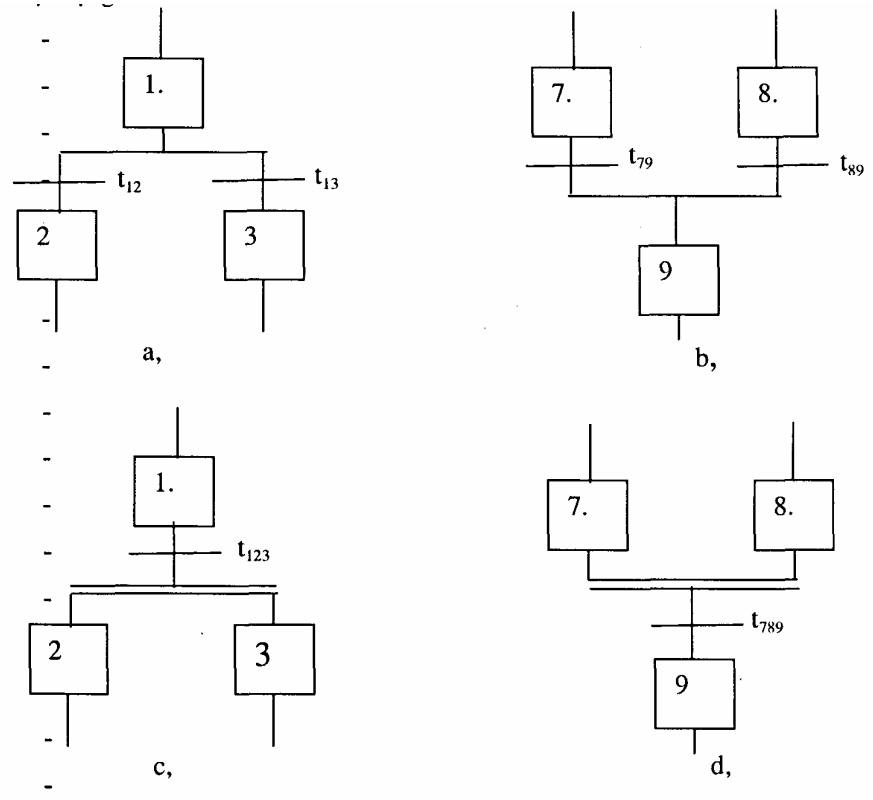
Sơ đồ song song là sơ đồ chỉ có một điều kiện liên hệ giữa 3 trạng thái như hình 1.10c và d .

Ở hình 1.10a, khi trạng thái 1 đang hoạt động, nếu chuyển tiếp t_{12} thoả mãn thì trạng thái 2 hoạt động; nếu chuyển tiếp t_{13} thoả mãn thì trạng thái 3 hoạt động.

Ở hình 1.10b nếu trạng thái 7 đang hoạt động và có t_{79} thì trạng thái 9 hoạt động, nếu trạng thái 8 đang hoạt động và có t_{89} thì trạng thái 9 hoạt động.

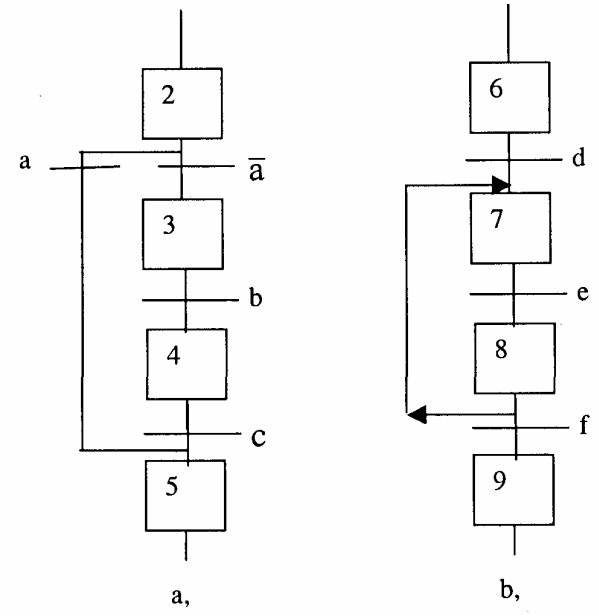
Ở hình 1.10c nếu trạng thái 1 đang hoạt động và có t_{123} thì trạng thái 2 và 3 đồng thời hoạt động.

Ở hình 1.10d nếu trạng thái 7 và 8 đang cùng hoạt động và có t_{789} thì trạng thái 9 hoạt động



Hình 1.10. Ký hiệu phân nhánh

Ký hiệu bước nhảy như hình 1.11 .



Hình 1.11. Ký hiệu bước nhảy

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.

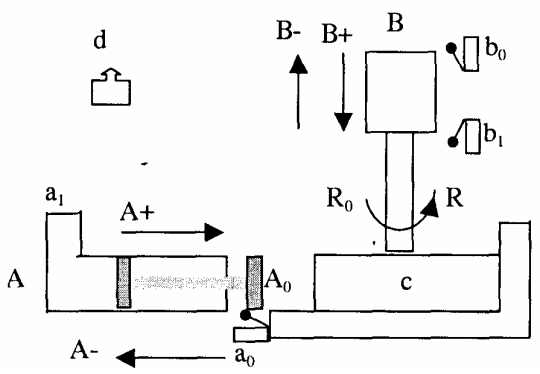
Hình 1.11b khi trạng thái 8 đang hoạt động nếu thoả mãn điều kiện f thì quá trình

chuyển sang trạng thái 9, nếu không thoả mãn điều kiện 8 thì quá trình quay lại trạng thái 7.

4. Cách xây dựng mạng grafcet

Để xây dựng mạng grafcet cho một quá trình nào đó thì trước tiên 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. Sơ đồ quy trình khoan

+ 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_o 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_o khi đó muông ngừng chuyển động, kết thúc một chu kỳ gia công.

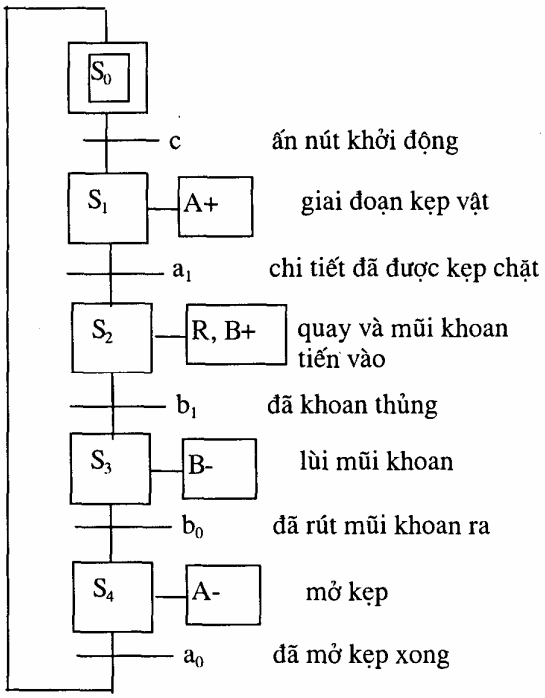
Sơ đồ grafcet như hình 1.13 .

5. Phân tích mạng grafcet

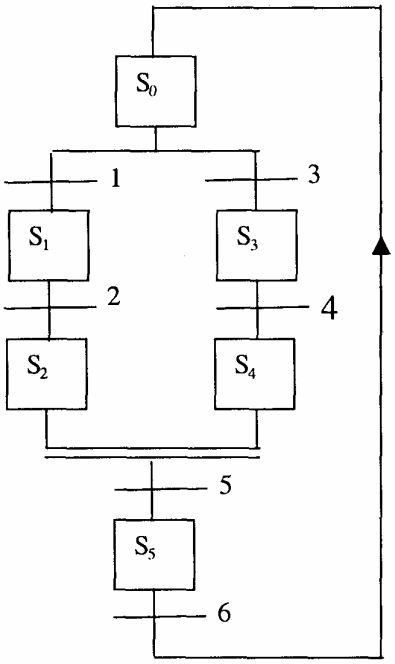
5.1. Quy tắc vượt qua, chuyển tiếp

- 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)



Hình 1.13. Mạng grafcet quá trình khoan



Hình 1.14. Sơ đồ có nhánh chết

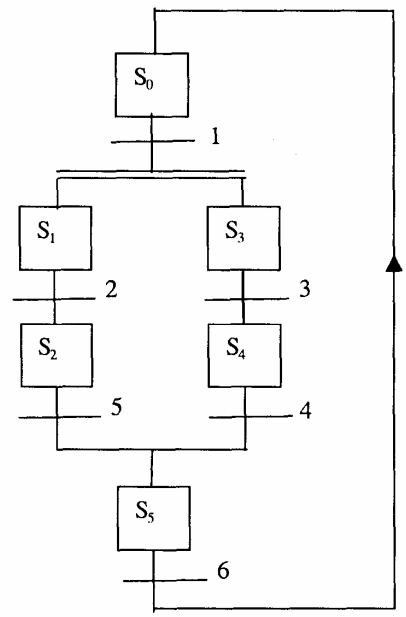
Với các điều kiện hoạt động như trên thì có nhiều khi sơ đồ không hoạt động được hoặc hoạt động không tốt. Người ta gọi:

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

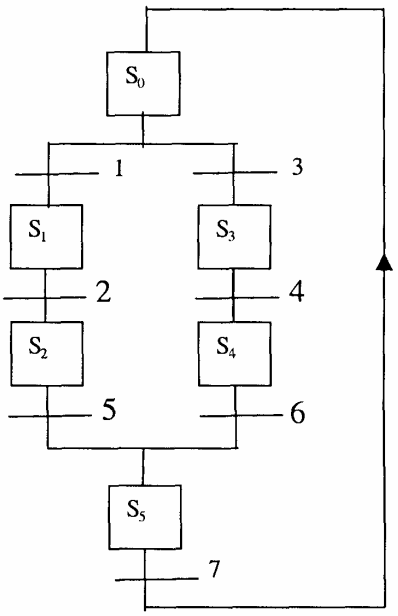
+ Sơ đồ không sạch là sơ đồ mà tại một vị trí nào đó được phát lệnh hai lần.

Ví dụ 1 : Sơ đồ hình 1.14 là sơ đồ có nhánh chết. 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_o nếu có điều kiện 3 thì S_o đã 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_o 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 cùng tích cực kèm điều kiện 5 như vậy hệ sẽ nằm im ở vị trí S_4

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



Hình 1.15. Sơ đồ không sạch



Hình 1.16. Sơ đồ sạch

Ví dụ 2: Sơ đồ hình 1.15 là sơ đồ không sạch. Giả sử 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 (diều kiện 3) thì nếu có điều kiện 1 cũng không có nghĩa vì S_o 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.

5.2. 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, phải xét một cách phân tích mạng grafcet là dùng phương pháp giản đồ điểm.

Để thành lập giản đồ điểm cần 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, lại lặp lại bốn bước nhỏ trên.

+ Quá trình cứ như vậy tiếp tục, 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 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.

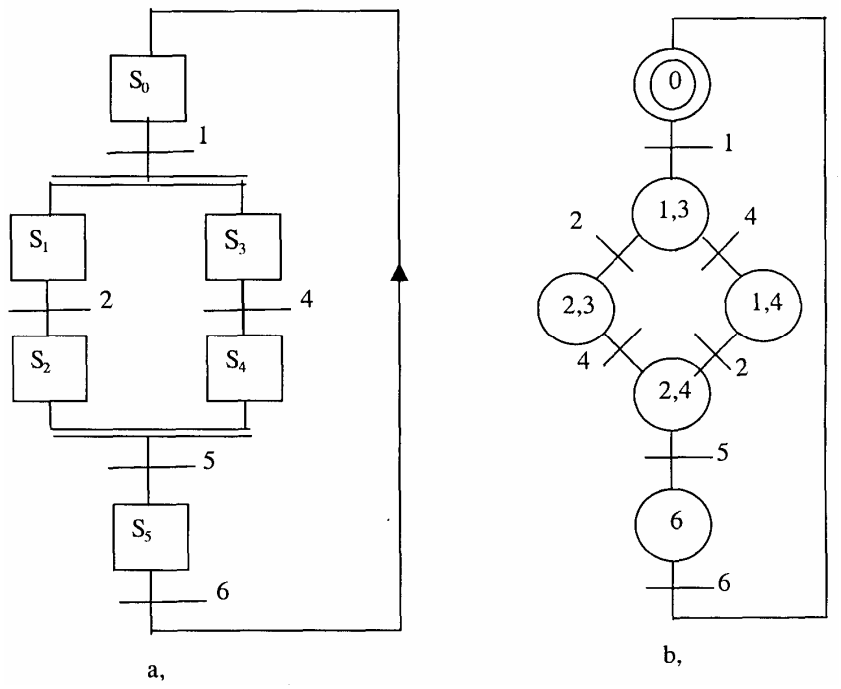
- Nếu vẽ được hết mà ở vị trí nào đó có các điểm làm việc cùng tên thì là sơ đồ không sạch ví dụ 3.

- 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_o (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_o. Vậy, sau điều kiện 1 tạo ô mới và trong ô này cần 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.

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.



Hình 1.17. Giản đồ điểm sơ đồ sạch

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 l,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.

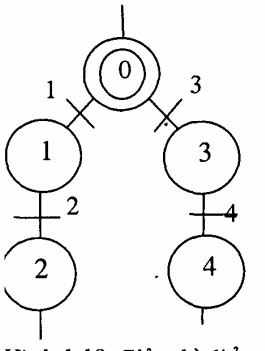
Khi hệ đang ở 1,4 nếu điều kiện 2 được thực hiện thì giai đoạn 2 tích cực (thêm dấu "."), giai đoạn 1 hết tích cực (mất dấu "."). Vậy sau điều kiện 2 tạo ô mới (nối với ô l,4), ô này ghi hai trạng thái tích cực còn lại trên grafcet là 2,4.

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.

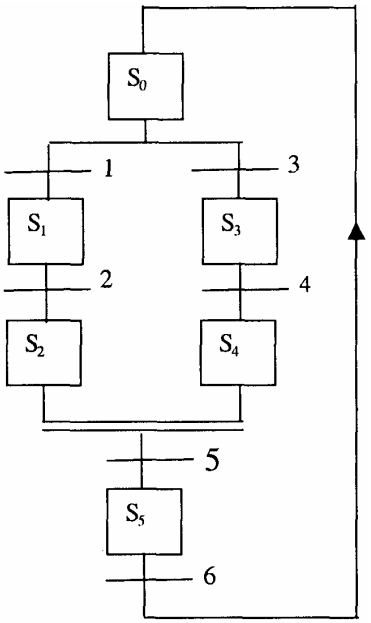
Khi hệ đang ở 2,4 nếu điều kiện 5 được thực hiện thì giai đoạn 5 tích cực (thêm dấu "."), giai đoạn 2 và 4 hết tích cực (mất dấu "."). Vậy sau điều kiện 5 tạo ô mới (nối với ô 2,4), ô này ghi trạng thái tích cực còn lại trên grafcet là 5.

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



Hình 1.18. Giản đồ điểm có nhánh chết



Hình 1.14. Sơ đồ có nhánh chết

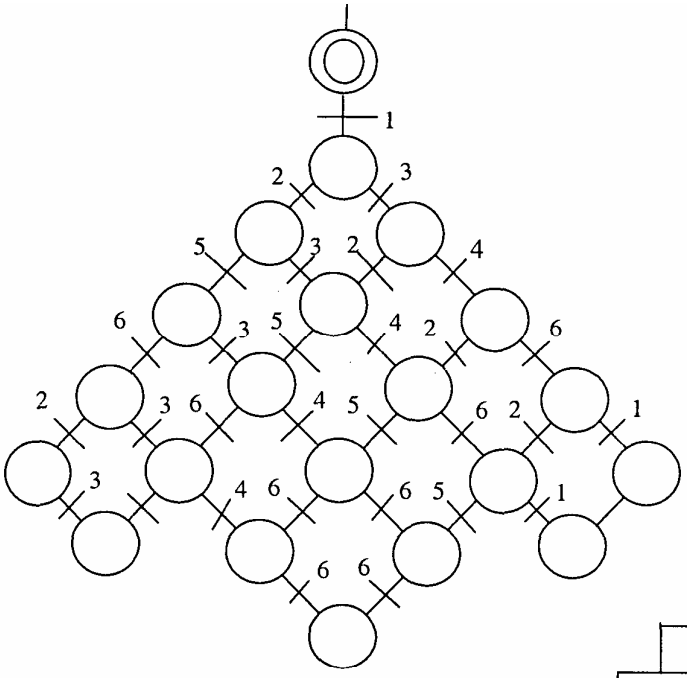
Ví dụ 2 : Vẽ giản đồ điểm cho sơ đồ có nhánh chết hình 1.14

Giản đồ điểm như hình 1.18. Trong trường hợp này không thể vẽ tiếp được nữa vì để S_5 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 1.5.

Cách tiến hành vẽ giản đồ điểm như trên, giản đổ điểm như hình 1.19. Từ giản đồ

điểm nhận thấy có nhiều ô 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.

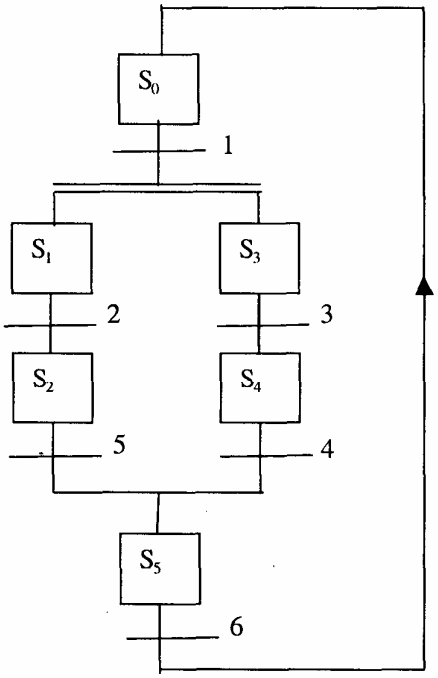


Hình 1.19. Giản đồ điểm không sạch

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.



Hình 1.15

CHƯƠNG 2: MỘT SỐ ỨNG DỤNG MẠCH LOGIC TRONG ĐIỀU KHIỂN

§2.l. Các thiết bị điều khiển

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. 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, 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*: Biết tốc độ động cơ do mô men động cơ xác định, mà mô men 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. Các thiết bị điều khiển

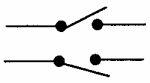
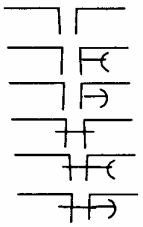
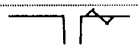
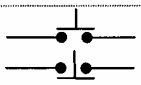
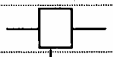
Để điều khiển sự làm việc của các thiết bị cần phải có các thiết bị điều khiển.

Để đóng cắt không thường xuyên thường dùng áptômát. Trong áptômát hệ thống tiếp điểm có bộ phân dập hổ 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 khi bị đất 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.

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	

Mạch từ của rơle có dòng điện một chiều chạy qua làm bằng thép khối, còn mạch từ của rơle dòng điện 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 tết.

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 của các thiết bị điều khiển có cấu tạo khác nhau và thường mạ bạc hay thiếc để đảm bảo tiếp xúc tết. 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.

§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. Xét một số sơ đồ đơn giản.

1. Mạch khống chế đơn giản

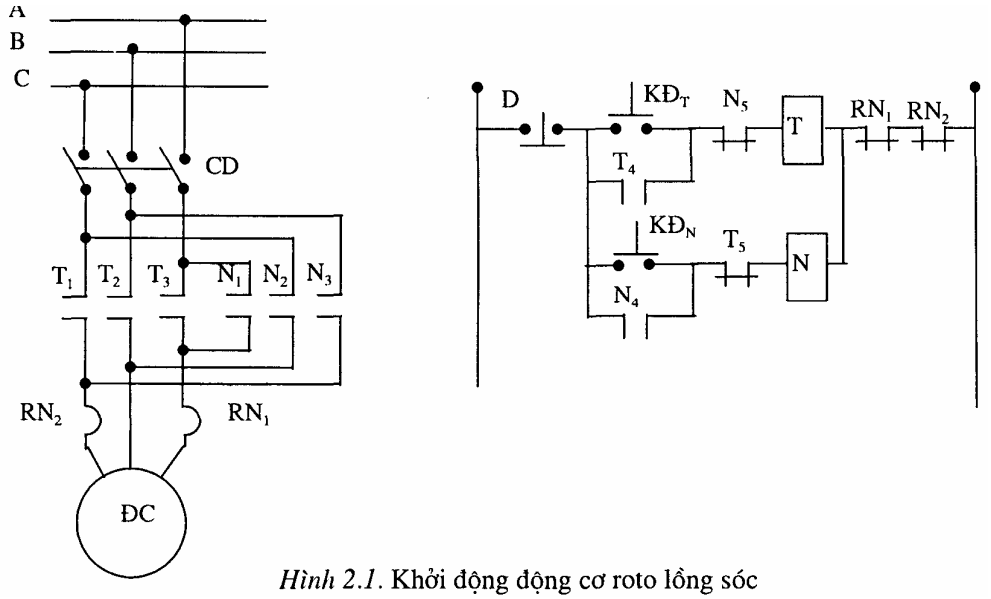
Với động cơ công suất nhỏ có thể đóng trực tiếp vào lưới điện. Nếu động cơ chỉ quay theo một chiều thì mạch đóng cắt có thể dùng cầu dao, áptômát. Với thiết bị đóng cắt này có nhược điểm là khi đang làm việc nếu mất điện, thì khi có điện trở lại động cơ sẽ tự khởi động. Để tránh điều đó dùng khởi động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

Cầu dao 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



Hình 2.1. Khởi động động cơ roto lồng sóc

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) ấn nút $KĐ_T$ (hoặc $KĐ_N$) 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.

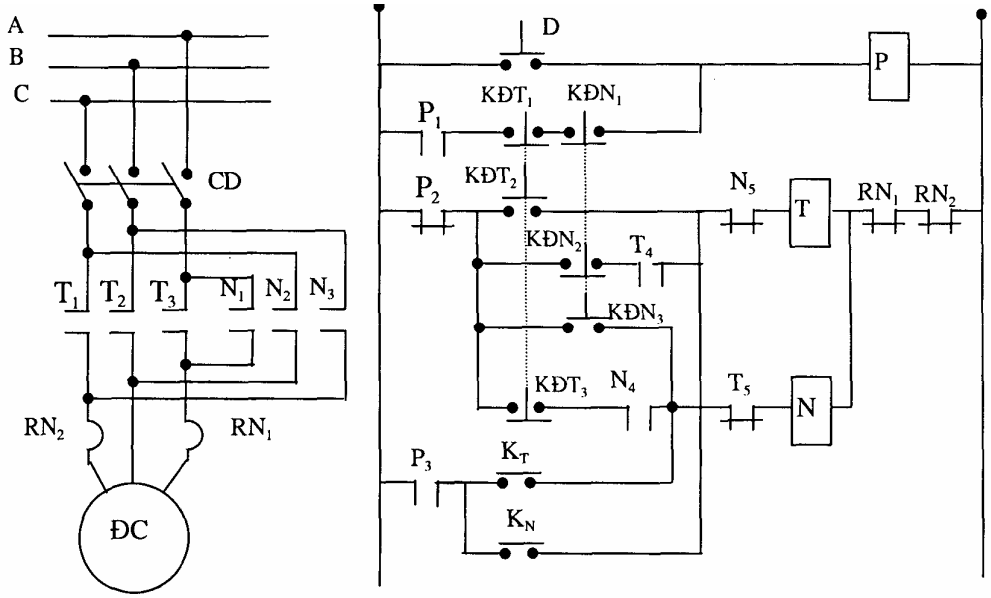
Để dừng động cơ ấn nút dừng D, các cuộn hút mất điện, cắt điện động cơ khỏi lưới điện, động cơ tự dừng.

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

2. Mạch khống chê đảo chiều có giám sát tốc độ

Xét sơ đồ khống chế động cơ rôto 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 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 mô men 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 được nhiệm vụ khởi động, đảo chiều. Trong sơ đồ có thêm rơle trung gian, hai rơle tốc độ (gắn với động cơ), rơle tốc độ thuận có tiếp điểm K_T và rơle tốc độ ngược có tiếp điểm K_N các rơle này khi tốc độ cao thì các tiếp điểm rơle kín, tốc độ thấp thì tiếp điểm rơle hở.



Hình 2.2. Mạch khống chế đảo chiều có giám sát tốc độ

Khi khởi động chạy thuận ấn nút khởi động thuận KĐT, tiếp điểm $KĐT_1$ hở ngăn không cho P có điện, $KĐT_3$ hở ngăn không cho cuộn hút N 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.

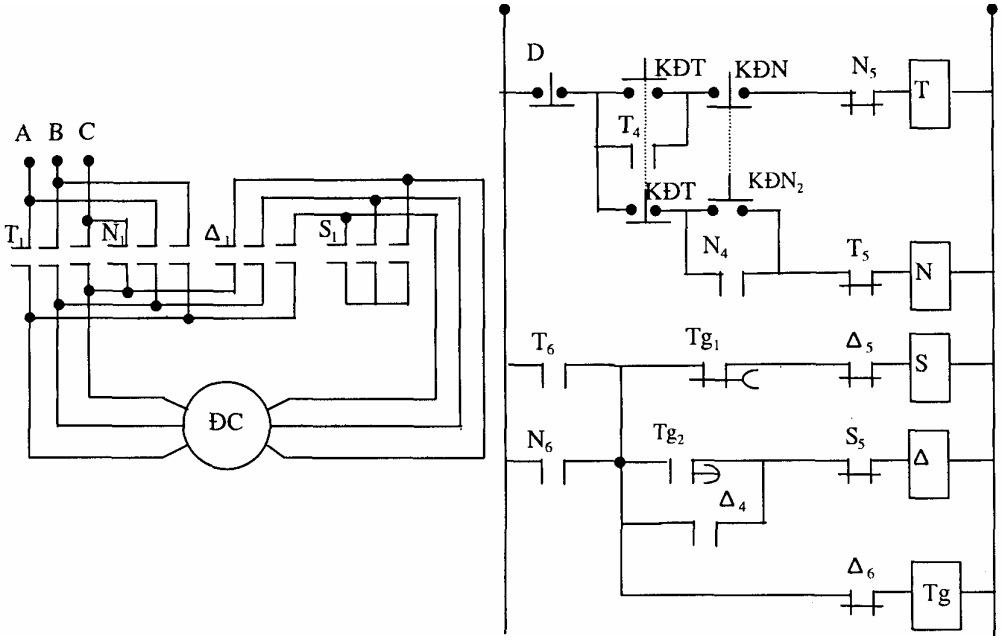
Khi đang chạy thuận cần chạy ngược ấn nút khởi động ngược KĐN, tiếp điểm $KĐN_1$ hở không cho P có điện, tiếp điểm $KĐN_2$ hở cắt điện cuộn hút T làm mất điện chế độ chạy thuận, tiếp điểm $KĐN_3$ kín cấp điện cho cuộn hút N để cấp điện cho chế độ chạy ngược, khi N hút tiếp điểm N_4 kín để tự duy trì.

Nếu muốn dừng 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 K_T 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 K_T mở cắt điện cuộn hút N, động cơ dừng hẳn.

Khi các rơle nhiệt tác động thì động cơ dừng tự do.

3. Khống chế động cơ lồng sóc kiểu đổi nối γ/Δ có đảo chiều

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



Hình 2.3. Khống chế động cơ lồng sóc kiểu đổi nối γ/Δ có đảo

Trong sơ đồ có khởi động từ T đóng điện cho chế độ chạy thuận, khởi động từ N đóng điện cho chế độ chạy ngược, khởi động từ S đóng điện cho chế độ khởi động hình sao, khởi động từ ỗ đóng điện cho chế độ chạy tam giác. Rơle thời gian Tg để duy trì thời gian khởi động, 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 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, T có điện đóng các tiếp điểm T_1...T_3 đưa điện áp thuận vào động cơ, T_4 đóng để tự duy trì, T_5 mở ngăn không cho N có điện, T_6 đóng cấp điện cho rơle thời gian Tg, đồng thời cấp điện ngay cho cuộn hút S, động cơ khởi động kiểu nối sao, tiếp điểm S_5 mở chưa cho cuộn Δ có điện. Khi Tg có điện, 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 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

Khi cần đảo chiều (nếu đang chạy thuận) 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 đứng 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.

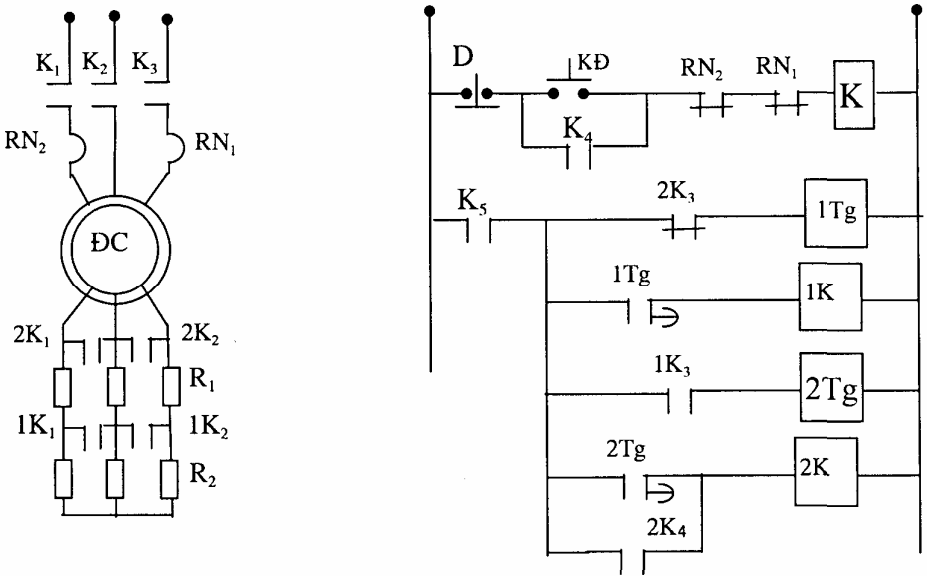
1. Khởi động động cơ rôto dây quấn theo nguyên tắc thời gian

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

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

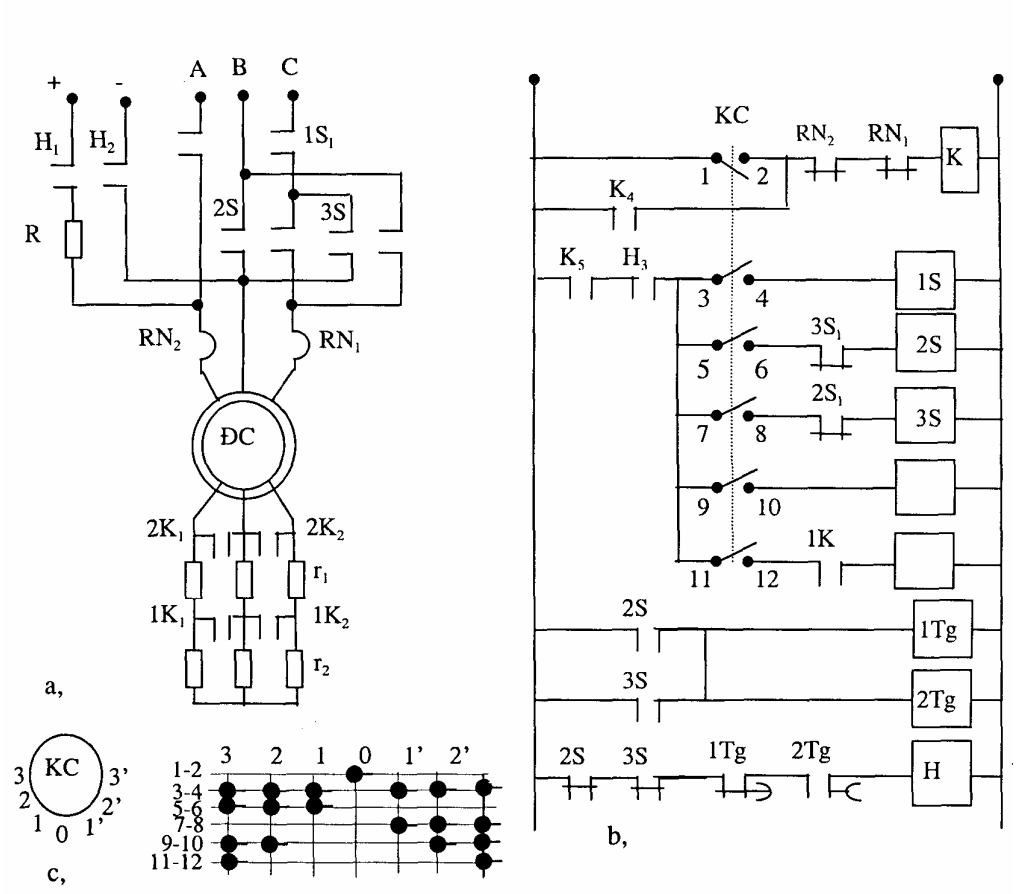


Hình 2.4. Khởi động động cơ rôto dây quấn theo nguyên tắc thời gian

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 mô men hãm. Sơ đồ nguyên lý của hệ thống như 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 môm hãm.



Hình 2.5. Thay đổi tốc độ động cơ rôto dây quấn

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- 1 0 sẽ đóng ở các vị trí 2, 3 bên trái và $2^{'}$, $3^{'}$ bên phải.

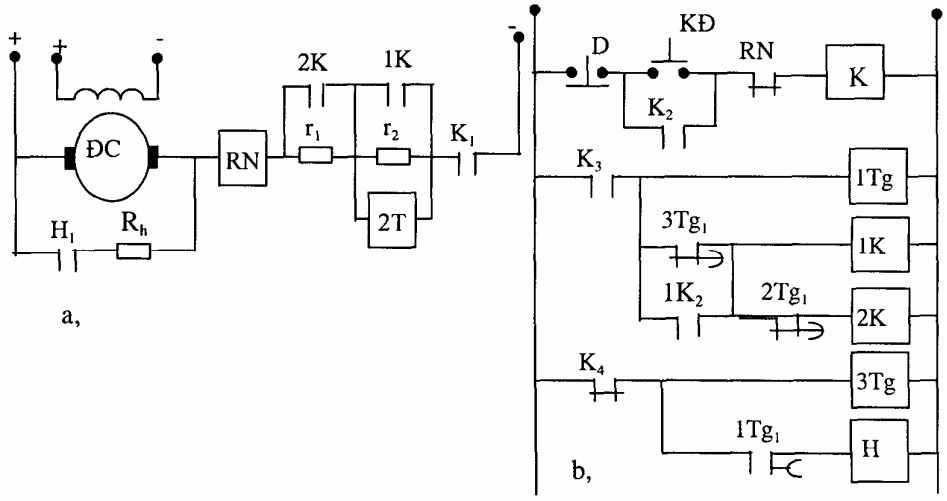
Hoạt động của bộ khống chế như sau: khi đã đóng điện cấp nguồn cho hệ thống. Ban đầu bộ khống chế được đặt ở vị trí 0 công tắc tơ K có điện, các tiếp điểm K ở mạch khống chế đóng lại, chuẩn bị cho hệ thống làm việc. Nếu muốn động cơ quay theo chiều thuận thì quay bộ KC về phía trái, nếu muốn động cơ quay ngược thì quay bộ KC về phía phải. Giả thiết 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 stato 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 toàn 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 stato động cơ và động cơ được hãm động năng. Sau thời gian chỉnh định $\Delta t2$ 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 stato, 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.

Để khởi động động cơ ấn nút khởi động KĐ lúc đó công tắc tơ K có điện, các tiếp điểm thường mở K, 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 3Tg, 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.



Hình 2.6. Khống chế động cơ điện một chiều

Để dừng động cơ 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 1 Tg 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 R_h vào để thực hiện quá trình hãm. Sau thời gian chỉnh định tiếp điểm thường mở mở chậm 1 Tg_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.

PHẦN 2: ĐIỀU KHIỂN LOGIC CÓ LẬP TRÌNH (PLC)

CHƯƠNG 3: LÝ LUẬN CHUNG VỀ ĐIỀU KHIỂN LOGIC LẬP TRÌNH PLC

§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 giai 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 cho các kỹ sư, không yêu cầu cao về kiến thức máy tính và ngôn ngữ máy tính, có thể vận hành. Chúng được thiết kế cho các nhà kỹ thuật 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 sẽ giám sát các tín hiệu vào và các tín hiệu ra theo chương trình này 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.

Riêng đối với máy công cụ và người máy công nghiệp thì bộ PLC có thể liên kết với bộ điều khiển số NC hoặc CNC hình thành bộ điều khiển thích nghi. Trong hệ thống của các 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

1. Cấu hình phần cứng

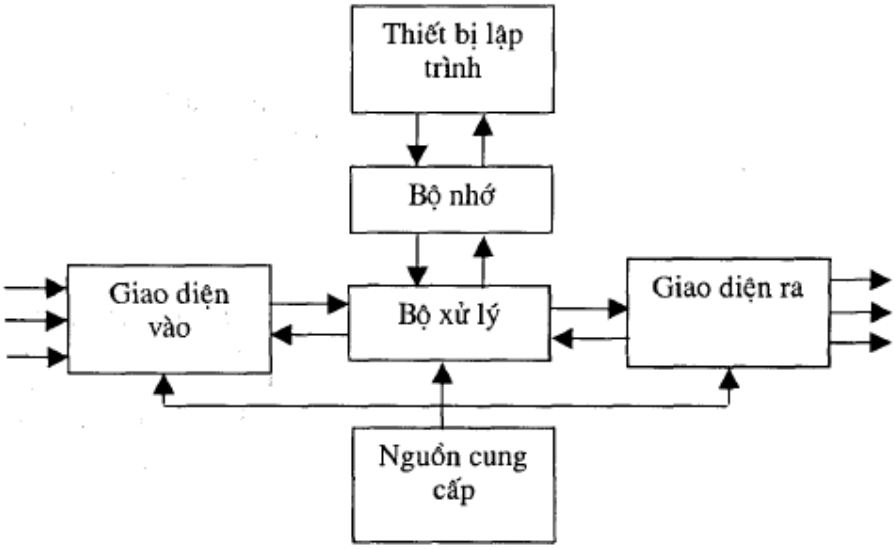
Bộ PLC thông dụng có năm bộ phận cơ bản gồm: bộ xử lý, bộ nhớ, bộ nguồn, giao diện vào/ra và thiết bị lập trình. Sơ đồ hệ thống như hình 3.1 .

1.1 Bộ xử lý

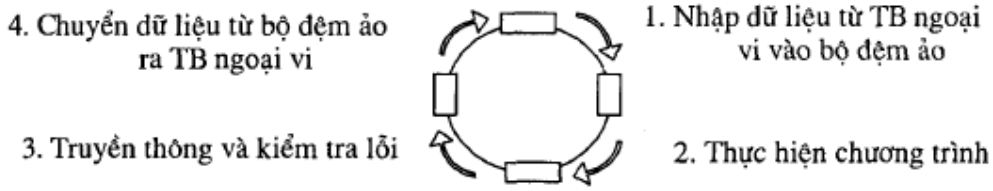
Bộ xử lý còn gọi là bộ xử lý trung tâm (CPU), là linh kiện chứa bộ vi xử lý. Bộ xử lý biên dịch các tín hiệu vào và thực hiện các hoạt động điều khiển theo chương trình được lưu trong bộ nhớ của CPU, truyền các quyết định dưới dạng tín hiệu hoạt động đến các thiết bị ra.

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 điều khiển tới đầu ra. Chu kỳ thời gian này gọi là thời gian quét (scan). Thời gian một vòng quét phụ thuộc vào dung lượng của bộ nhớ, vào tốc độ của CPU. Nói chung chu kỳ một vòng quét như hình 3.2.

Sự thao tác tuần tự của chương trình dẫn dấn 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.



Hình 3.1. Sơ đồ nguyên lý bộ PLC



Hình 3.2. Vòng quét

Để đánh giá thời gian trễ người ta đo thời gian quét của một chương trình dài 1K byte và coi đó là chỉ tiêu để so sánh các PLC. Với nhiều loại PLC 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.

1. 2. 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 đầu ra hoặc các module còn lại (thường là 24V).

1.3. 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 lập trình chuyên dụng, có thể là thiết bị lập trình cầm tay gọn nhẹ, có thể là phần mềm được cài đặt trên máy tính cá nhân.

1.4. Bộ nhớ

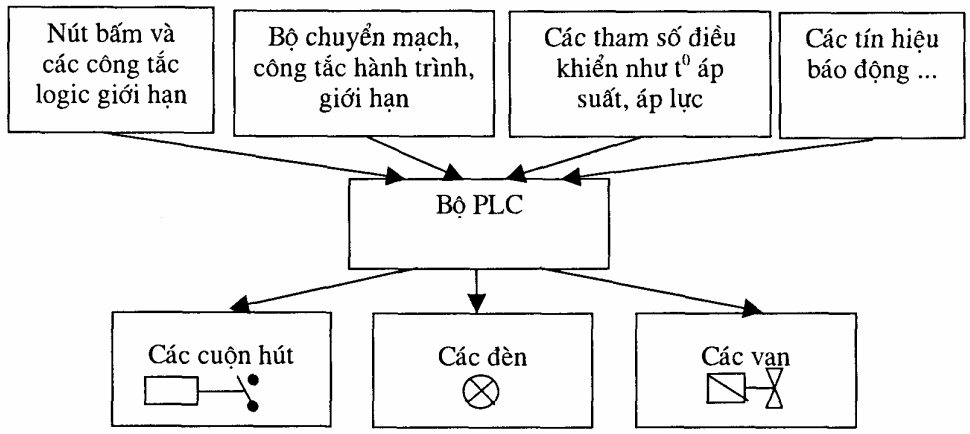
Bộ nhớ là nơi lưu giữ chương trình sử dụng cho các hoạt động điều khiển. Các

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.

1.5. 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.

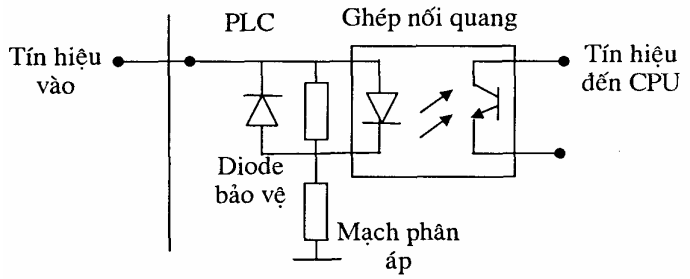
Mỗi điểm vào ra có một địa chỉ duy nhất được PLC sử dụng.



Hình 3.3: Giao diện vào/ra

Các kênh vào/ra đã có các chức năng cách ly và điều hoà 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.

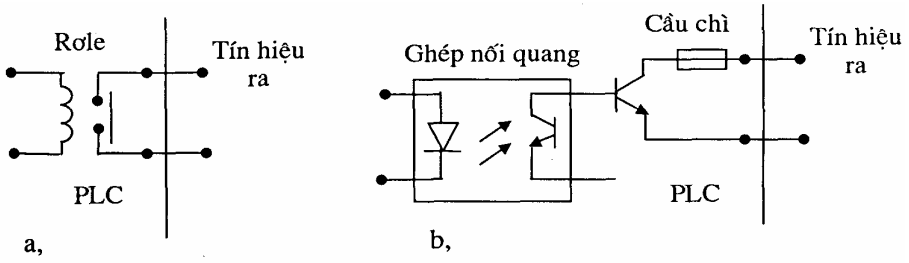
Tín hiệu vào thường được ghép cách điện (cách ly) nhờ linh kiện quang như hình 3.4. Dải tín hiệu nhận vào cho các PLC cỡ lớn có thể là 5v, 24v, 110v, 220v. Các PLC cỡ nhỏ thường chỉ nhập tín hiệu 24v.



Hình 3.4. Cách ly tín hiệu vào

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. 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.

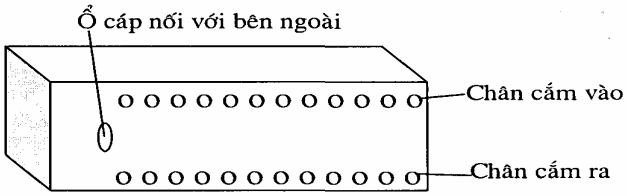


Hình 3.5. Cách ly tín hiệu ra

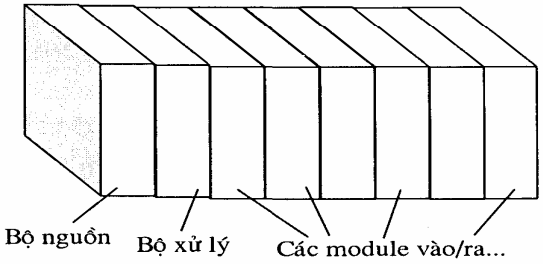
2. Cấu tạo chung của PLC

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 hộp đơn



Hình 3.7: Kiểu module

Kiểu module ghép nối 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 dược kết nối với nhau. 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.

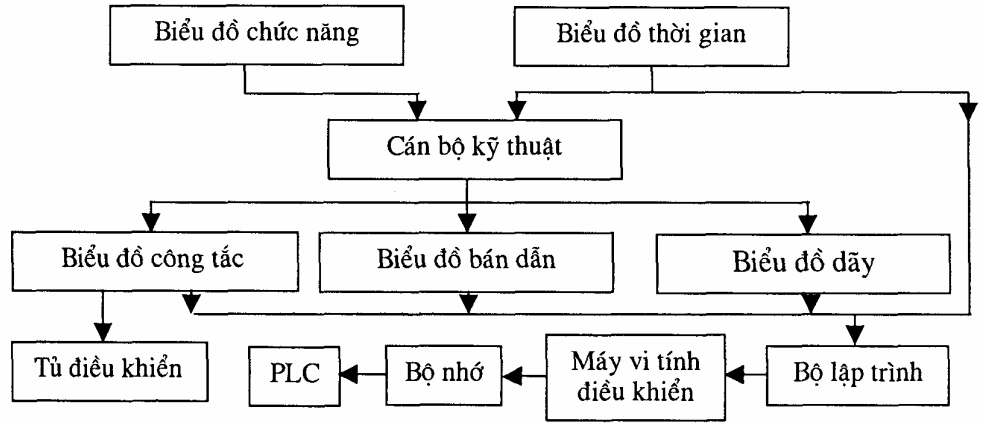
§3.3. Các vấn đề về lập trình

1 Khái niệm chung

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 quy trình lập trình theo giản đổ hình 3.8.

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



Hình 3.8. Quy trình lập trình

+ 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 PLC 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.

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 đã 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.

2.1. Một số ký hiệu chung

Cấu trúc lệnh

Một lệnh thường có ba phần chính và thường viết như hình 3.9 (có loại PLC có cách viết hơi khác):

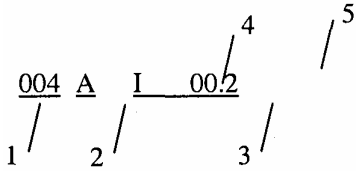
1. Địa chỉ tương đối của lệnh (thường khi tập trình thiết bị lập trình tự đưa ra).

2. Phần lệnh là nội dung thao tác mà PLC phải tác động lên đối tượng của lệnh, trong lập trình LAD thì 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 ":", có các 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.



Hình 3.9: Lệnh STL

Ký hiệu thường có trong mỗi lệnh:

Các ký hiệu trong lệnh, quy 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.

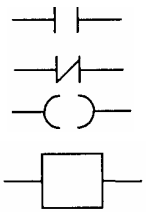
Ví dụ: A I 1.0; = Q 0.7.

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 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.

2.2. 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)

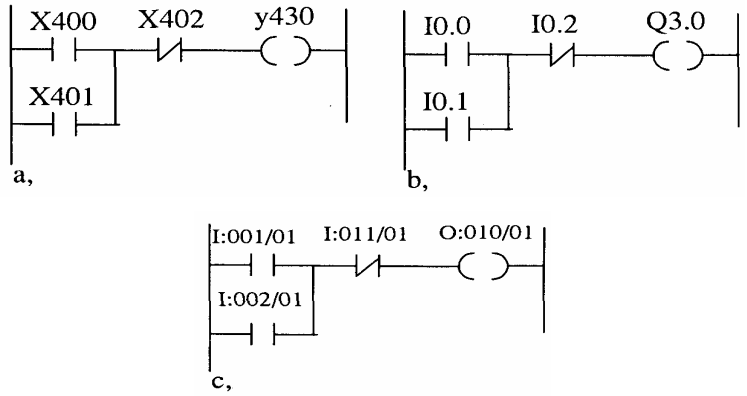
Bảng 3.1

IEC 1131-3	Misubishi	OMRON	Siemens	Telemec-anique	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

IEC 1131-3	Misubishi	OMRON	Siemens	Telemec-anique	Spreher và Schuh	Chú thích
ANDN	ANI	AND NOT	AN	AN	AND NOT	Phần tử nối tiếp có tiếp điểm kín
O	OR	OR	O	O	OR	Phần tử song song có tiêu điểm mở
ORN	ORI	OR NOT	ON	ON	OR NOT	Phần tử song song có tiếp điểm kín
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.



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)

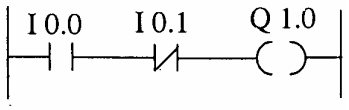
Hình 3.10c là ký hiệu của Allen Bradley

2.3. Phương pháp liệt kê 1ệnh 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) khi lập luôn sử dụng các lệnh khởi đầu như LD, L, A, O... (bảng 3.l). 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)

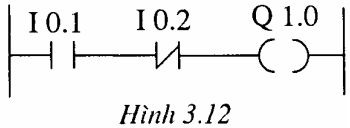
```
0   A   I 0.0
1   A   I 0.1
2   =   Q 1.0
```



Hình 3.11

Một đoạn STL của PLC S7-200 (Siemens)

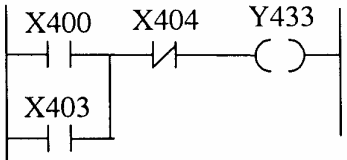
```
0 LD I 0.1
1 A  I 0.2
3 =  Q 1.0
```



Hình 3.12

Một đoạn STL của PLC MELSEC Fl (Nhật)

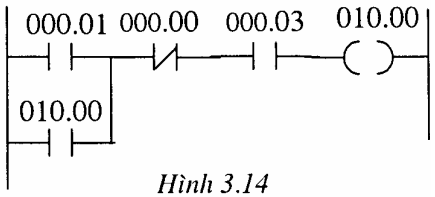
```
0 LD  X 400
1 O   X 403
2 ANI X 404
3 OUT Y 433
```



Hình 3.13

Một đoạn STL của CPM1A (OMRON)

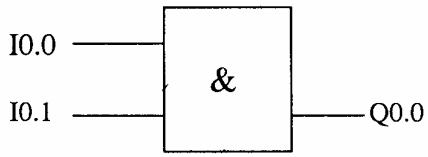
```
0 LD     000.01
1 OR     010.00
2 AND NOT 000.00
3 AND      000.03
4 OUT    010.00
```



Hình 3.14

2.4. 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 Boo1e.



Hình 3.15. Phương pháp lập trình CSF

3. Các rơle nội

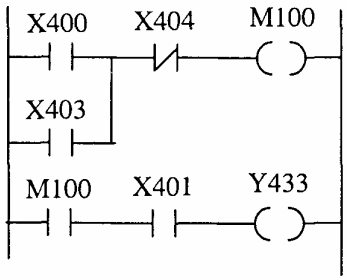
Trong các loại PLC có nhiều thuật ngữ dùng để chỉ các linh kiện loại này, ví dụ: rơle phụ, bộ vạch dấu, cờ hiệu, lưu trữ bít, bít nhớ... Đây là linh kiện cung cấp các chức năng đặc biệt gắn liền với PLC và được dùng phổ biết trong lập trình. Rơle nội này tương tự như các rơle trung gian trong sơ đồ rơle 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	Cuốn dây	C	C001; C002
TelemecaniQue	Bít	B	B0; B1
Toshiba	Rơle nội	R	R000; R001
Bradley	Lưu trữ bít	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  OUTY  433
```



Hình 3.16

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.

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 cơ bản là bộ đếm tiến và bộ đếm lùi. Các nhà sản xuất PLC cũng sử dụng các bộ đếm theo những cách 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 quy 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.

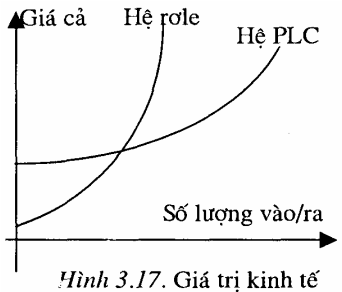
+ Đánh giá nhu cầu đơn giản: Khi biết các đầu vào và các đầu ra thì có thể đánh giá được kích cỡ yêu cầu của bộ nhớ hay độ dài chương trình. Do đó, có thể dễ dàng và nhanh chóng lựa chọn PLC phù hợp với các yêu cầu công nghệ đặt ra.

+ Khả năng tái tạo: Nếu dùng nhiều PLC với quy 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 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. Trên hình 3.17 thể hiện, 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.



Hình 3.17. Giá trị kinh tế

Khi tính đến giá cả của PLC thì không thể không kể đến giá của các bộ phận 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 quy trình điều khiển.
  + Kích thước nhỏ.
  + Có thể nối với mạng máy tính.
    - Giá thành cao

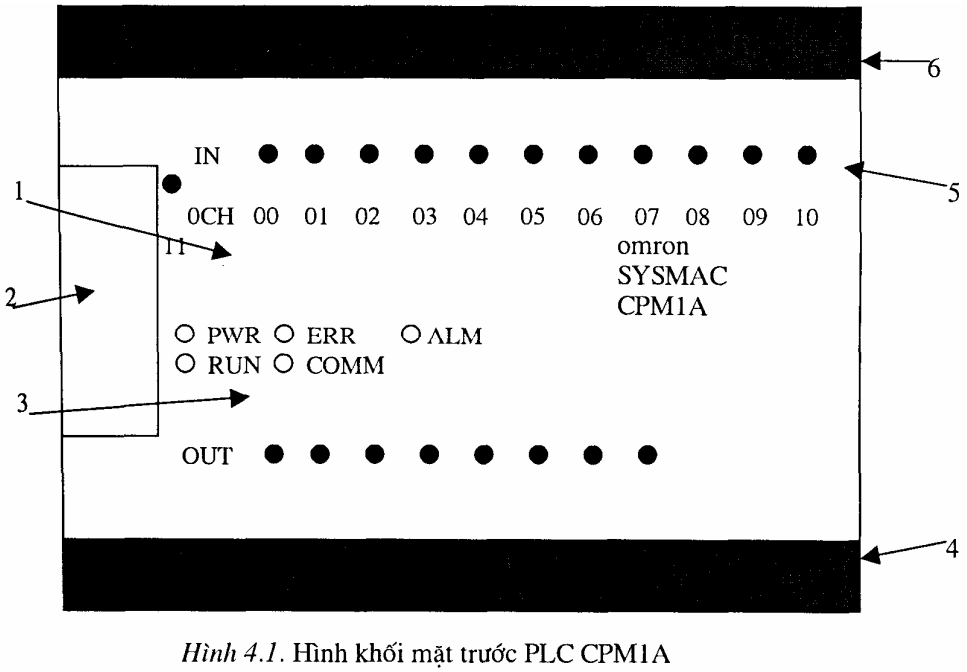
Bộ thiết bị lập trình thường đắt, sử dụng ít.

CHƯƠNG 4: BỘ ĐIỀU KHIỂN PLC – CPM1A

§4.l. Cấu hình cứng

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,

+ Nháp 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).

2. Các thông số kỹ thuật

2.1. 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	AD			
CPM1A-40CDR-A	AC	24	16	40
CPM1A-40CDR-D	DC			

2.2. Thông số chung

Mục		**10-đầu I/O**	**20-đầu I/O**	**30-đầu I/O**	**40-đầu I/O**
Điên 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 điên		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/s^2 (20G) ba lần mỗi chiều X, Y và Z			
Nhiệt độ môi trường		Nhiệt độ làm việc: 0 đến 55C^o Nhiệt đô bảo quản: -20 đến 75C^o			
Đô 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)			
Trong 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

2.3 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 μs Lệnh đặc biệt: 12,375 μs (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 input 16 output
	Có module mở rộng	----	----	54 input 36 output	60 input 40 output

Mục	10 - đầu I/O	20 - đầu I/O	30 - đầu I/O	40 - đầu I/O
Vào dạng bít	00000 đến 00915 (Words 0 đến 9)			
Ra dạng bít	01000 đến 01915 (Words 10 to 19)			
Từ bít (vùng IR)	5 1 2 bíts : IR20000 to 23115 (words IR 200 to IR 231)			
Bít đặc biệt (vùng SR)	384 bíts: SR 23200 to 25515 (words SR 232 to IR 255)			
Bít nhớ tạm thời (vùng TR)	8 bíts (TR0 to TR7)			
Bít giữ (vùng HR)	320 bíts: HR 0000 to HR 1915 (words HR 00 to HR 19)			
Bít bổ trơ (Vùng AR)	256 bíts:AR 0000 to AR 1515 (words AR 00 to AR 15)			
Bít liên kết (vùng LR)	256 bíts : LR 0000 to LR 1515 (words LR 00 to LR 15)			
Timers/Cunters	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), bít giữ, bít nhớ bổ trợ, bộ đếm (20 ngày trong điều kiện nhiệt độ 25^{o}C)			
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 bíts) Kiểu tăng/giảm: -32.767 đến 32.767 (16 bíts)			
Nhập hằng số thời gian	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 đến 200 BCD)			

2.4. Cấu trúc vùng nhớ

Dữ liệu		Từ (words)	Bít	Chức năng
IR	vào	IR 000 đến IR 009 (10 words)	IR 00000 đến IR 00915 (160 bíts)	Các bít này có thể làm việc ở vùng vào ra mở rộng
	Ra	IR 010 đến IR 019 (10 words)	IR 01000 đến IR 01915 (160 bíts)	
	làm việc	Ir 200 đến IR 231 (32 words)	Ir 20000 đến IR to 23 115 (5 2 bíts)	Các từ bít này có thể sử dụng tuỳ ý trong chương trình
SR		SR 232 đến SR 255 (24 words)	SR 23200 đến 25515 (384 bíts)	Những bít này phục vụ cho chức năng đặc biệt như cờ và bít điều khiển.
TR		---	TR 0 đến TR 7 (8 bíts)	Bít này được sử dụng ở trạng thái đóng mở trong chương trình phân nhánh

Dữ liệu		Từ (words)	Bít	Chức năng
HR		HR 00 đến HR 19 (20 words)	HR 0000 dấn HR 1915 (320 bíts)	Những bít này lưu giữ trạng thái đóng mở khi mất nguồn ngoài.
Ar		AR 00 đến HR 15 (1 6 words)	AR 0000 đến HR 1515 (256 bíts)	Những bít này phục vụ cho chức năng đặc biệt như cờ và bít điều khiển.
LR		LR 00 đến LR 15 (16 words)	LR 00000 đến LR 1515 (256 bíts)	Sử dụng để kết nối với PC khác
Timer/ couter		TC 000 đến TC 127 (timer/counter)		Số giống nhau sử dụng cho cả thuế và couter.
DM	Đọc /viết	DM 0000 ÷ DM 0999 DM 1022 ÷ DM 1023 (1,002 words)	---	DM là dữ liệu chỉ truy cập dạng từ (words). Các dữ liệu dạng từ (words) được cất giũ khi mất nguồn.
	Ghi lỗi	DM 1000 đến 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 đến DM 6599 (456 words)	- -	Không thể ghi đè lên chương trình
	Cài đặt PC	Dài 6600 đến DM 6655 (%6 words)	-	sử dụng đến nhiều vùng tham số để điều khiển làm việc của PC

Chú ý:

1. Bít IR và LR khi chưa sử dụng cho các chức năng chính thì có thể sử dụng như bít 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 25^{o}C, với thời gian 20 ngày.

3. Khi truy nhập các số PV, TC thì dữ liệu dạng từ (words), khi truy cấp vào cờ thì dữ liệu dạng bít.

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".

2.5. Cực vào ra - các bít vùng IR cho vào ra mở rộng

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

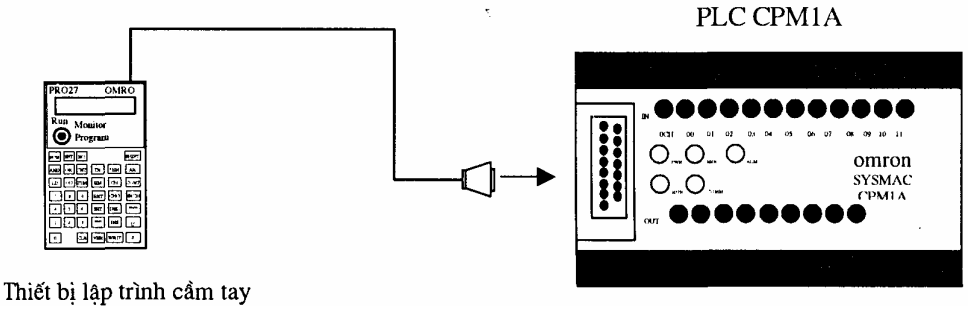
Số vào/ra của CPU	Điểm nối CPU (địa chỉ)		Điểm nối vùng mở rộng (địa chỉ)		Nguồn	Số module
	Vào	Ra	Vào	Ra		
10	6 điểm: 00000 ÷ 00005	4 điểm: 01000 ÷ 01003	---	---	AC	CPM1A-10CDR-A
					DC	CPM1A-10CDR-D
20	12 điểm: 00000 ÷ 00011	8 điểm : 01000 ÷ 01007	---	---	AC	CPM1A-20CDR-A
					DC	CPM1A-20CDR-D

Số vào/ra của CPU	Điểm nối CPU (địa chỉ)		Điểm nối vùng mở rộng (địa chỉ)		Nguồn	Số module
	Vào	Ra	Vào	Ra		
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 ÷ 11007 01100 ÷ 01107			AC	CPM1A-40CDR-A
					DC	CPM1A-40CDR-D

§4.2. Ghép nối

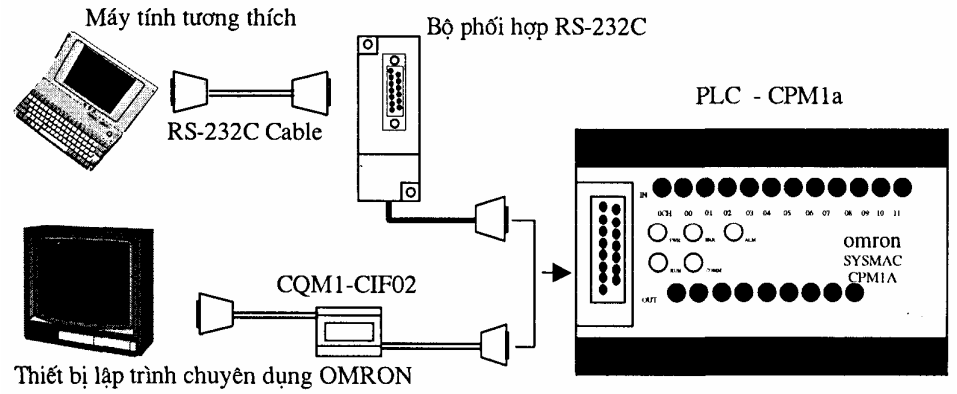
PLC CPM1A có thể ghép nối với 32 bộ PLC cùng loại thành hệ thống. Để lập trình cho PLC thì có thể ghép nối nó với thiết bị lập trình cầm tay, bộ lập trình chuyên dụng hoặc máy tính tương thích.

1. Ghép nối với thiết bị lập trình cầm tay: 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

2. Ghép nối với thiết bị lập trình chuyên dụng hoặc máy tính tương thích



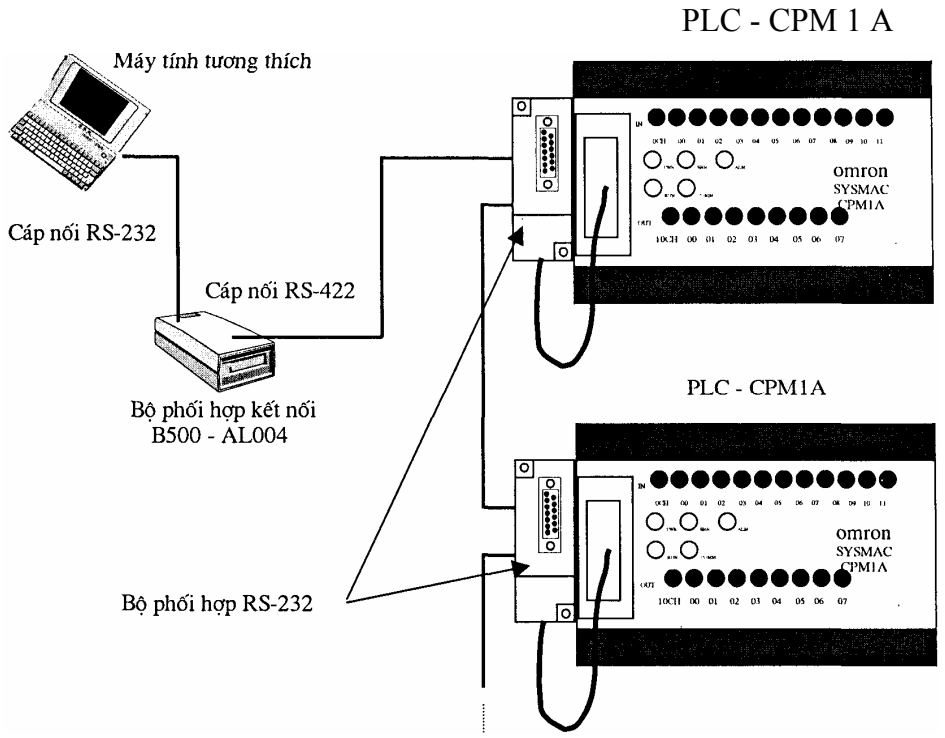
Hình 4.3. Ghép nối với lập trình chuyên dụng hoặc

Khi ghép nối với máy tính tương thích người ta dùng cáp nối chuẩn RS-232C và

bộ phối hợp RS-232 (hoặc RS-422) hoặc cáp chuyển đổi loại CQMI-CIF02. 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.

3. Ghép nối nhiều PLC và máy tính

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 500 m.



Hình 4.4. Ghép nối nhiều PLC

§4.3. Ngôn ngữ lập trình

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 OBI. 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.

2. Bảng lệnh của PLC – PCM1A

Xem phần "Bảng lệnh" phụ lục 2

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.

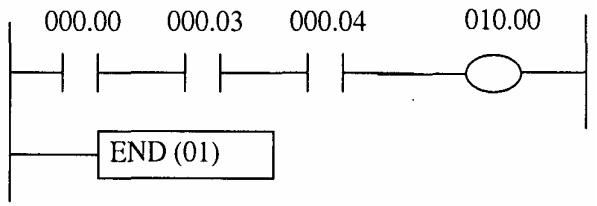
Khi lập trình phần mềm lập trình đã tự hiểu các địa chỉ trên, không cần đưa khái niệm để phân biệt vào/ra. Nếu đưa thêm khái niệm vào/ra (X/Y) phần mềm sẽ không chấp nhận.

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

3.1. 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)

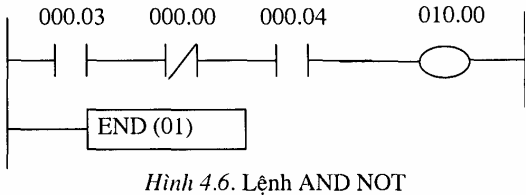
+ Chọn trạng thái MONITOR hoặc trạng thái PROGRAM (STOP/PRG) nhờ Shift + F10 hoặc biểu tượng "PLC Mode". Đổ chương trình sang PLC từ biểu tượng hoặc từ đường dẫn (như phụ lục l).

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

3.2. Lệnh AND NOT

Dạng STL

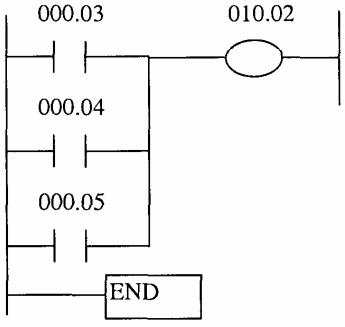
```
LD          000.03
AND NOT     000.00
AND         000.04
OUT         010.00
END
```



Hình 4.6. Lệnh AND NOT

3.3. Lệnh OR: Dạng SLT

```
LD      000.03
OR      000.04
OR      000.05
```



Hình 4.7. Lệnh OR

OUT 010.02

END

3.4. Lệnh OR NOT

Dạng STL

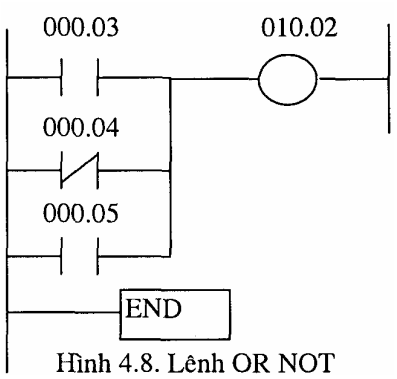
LD 00.03

OR NOT 00.04

OR 000.05

OUT 010.02

END



Hình 4.8. Lênh OR NOT

3. 5. Lệnh OR giữa hai 1ệnh AND

Dạng STL

LD 000.03

AND 000.04

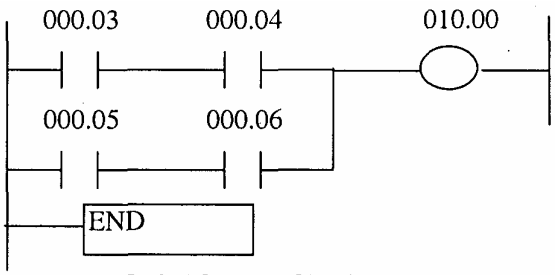
LD 000.05

AND 000.06

OR LD

OUT 010.00

END



Hình 4.9. Lệnh OR và AND

3.6. Lệnh thời gian trễ

Dạng STL

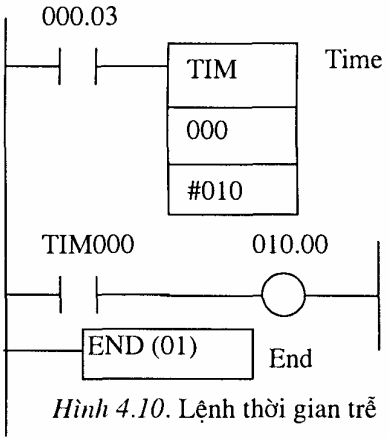
LD 000.03

TIM 000 #010

LD TIM000

OUT 010.00

END



Hình 4.10. Lệnh thời gian trễ

Chú ý:

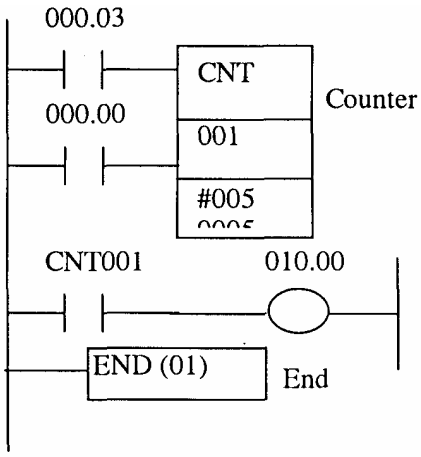
+ Trong lệnh (TIM 000 #010) loạt số đầu chỉ số hiệu của rơle thời gian (rơle thời gian số 0), loạt số thứ hai chỉ thời gian đặt (10s)

+ 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.

3.7. Bộ đếm

LD 000.03

LD 000.00



Hình 4.11. Bộ đếm

```
CNT000          #005
LD        CNT000
OUT010.00
END
```

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 #0051 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 vào 000.03 đạt 5 lần giá trị 1 thì bộ đếm cho giá trị ra, tức đầu ra 010.00 có giá trị 1.

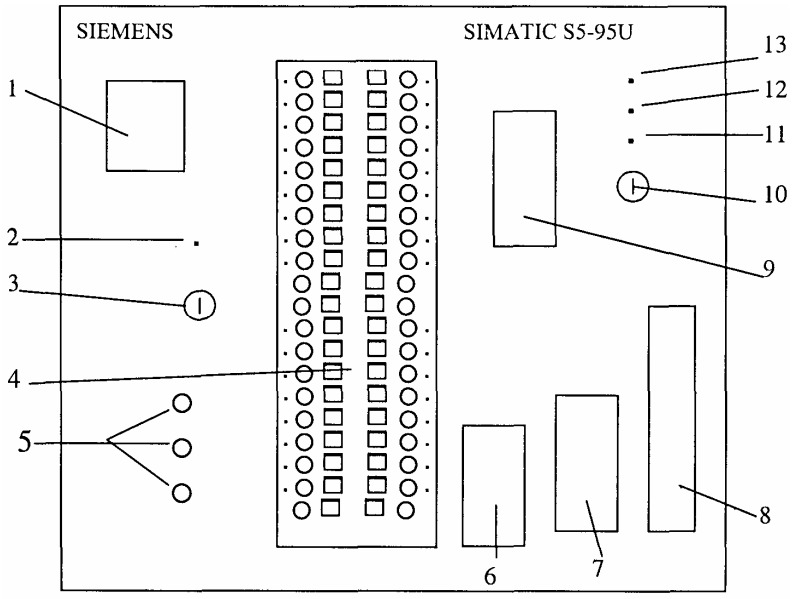
CHƯƠNG 5: BỘ ĐIỀU KHIỂN PLC - S5

§5.l. 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 module 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 là hợp lại cho phù hợp với những nhiệm vụ kỹ thuật cụ thể.

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

Trong đó:

1. Ngăn để ắc quy,

2. Công tắc mở điện ắc quy,

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.

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 8 module vào ra qua 8 vị trí có sẵn trên panen về phía phải. Thường Step 5 sử dụng các module mở rộng:

+ Module vào, ra số duy trì,

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

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

+ Module thông tin không duy trì CCP.

* Quy ước các chân của module mở rộng như hình 5.2.

+ Chân l: Dương nguồn (L+),

+ Chân 2: Âm nguồn (M),

+ Chân 4: Kênh số 0,

+ Chân 3: Kênh số 1,

+ Chân 6: Kênh số 2,

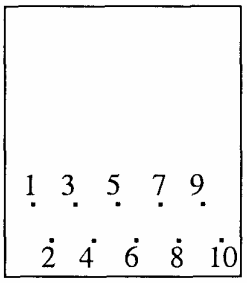
+ Chân 5 : Kênh số 3,

+ Chân 8: Kênh số 4,

+ Chân 7: Kênh số 5,

+ Chân 1 0 : Kênh số 6 +

Chân 9: Kênh số 7.



Hình 5.2. Sơ đồ chân module mở rộng

§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.

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. 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.

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 .

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

Khe số: 0 1 2 3 ...

Đơn vị cơ bản	0 1 : 7	0 1 : 7	0 1 : 7	0 1 : 7	

Hình 5.3. Số hiệu khe và kênh trên module số

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 bít số liệu, vì vậy địa chỉ của kênh trên module số còn được gọi là địa chỉ bít, mỗi module mang nhiều kênh tức là chứa nhiều bít, thường là 8 bít 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.

2. Địa chỉ vào ra trên module tương tự

Để diễn tả một giá trị tương tự phải cần nhiều bít. Trong PLC S5 người ta dùng 16 bít (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 lừ 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 địa chỉ kênh là đã biết nó nằm ở khe nào.

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

Khe số: 0 1 2 3 4 5 6

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+l01 102+103	l04+l05 106+107 l08+l09 110+111	112+113 114+115 116+117 118+119	120+121 122+123 124+125 126+127

Hình 5.4. Địa chỉ module tương tự

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 quy 1	
2	ACCUM2	Ắc quy 2	
3	BN	Hằng số byte	-127 đến 127
4	C	Bộ đếm - Có nhớ - Không nhớ	0 đến 7 8 đến 127
5	CC0/CC1	Mã điều kiện 1 và mã điều kiện 2	
6	D	Số liệu dạng bít	0.0 đến 255.15
7	DB	Khối số liệu	2 đến 255
8	DL	Từ (word) dữ liệu trái	0 đến 255
9	DR	Từ (word) dữ liệu phải	0 đến 225
10	DW	Từ (word) 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 (word) cờ - Có nhớ - Không nhớ	0 đến 62 64 đến 254
14	FY	Từ (word) byte - Có nhớ - Không nhớ	0 đến 63 64 đến 255
15	I	Đầu vào bít	0.0 đến 127.7
16	IB	Đầu vào byte	0 đến 127
17	Iw	Đầu vào từ (word)	0 đến 126
18	KB	Hằng số 1 byte	0 đến 255
19	KC	Hằng số đếm	0 đến 999
20	KF	Hằng số	-32768 đến 32677
21	KH	Hằng số dạng cơ số 16	0000 đến FFFF
22	KM	Hằng số bít dạng byte	Mỗi byte 16 bít
23	KS	Hằng số cho ký tự	2 ký tự ASCII
24	KT	Hằng số cho thời gian	0.0 dấn 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	

TT	Tên tham số	Diễn giải	Vùng tham số
31	PW	Đệm ngoại vi dạng từ (word)	0 đến 125
32	Q	Đấu ra bít	0.0 đến 127.7
33	QB	Đầu ra dạng byte	0 đến 127
34	QW	Đầu ra dạng từ (word)	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

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 OBI. 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.

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 quá trình 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 các 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 (Dâm 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:

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.

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.

3. Nhóm lệnh hệ thống

Các lệnh hệ thống được phép thâm nhập trực liếp 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" phụ lục 2.

§5.6. Cú pháp một số lệnh cơ bản của S5

1. Nhóm lệnh logic cơ bản

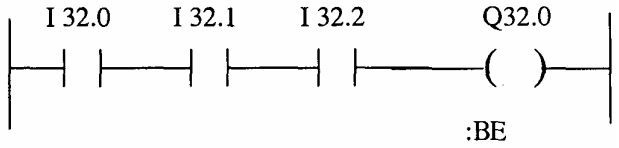
Khi thực hiện lệnh đầu tiên của một loạt phép toán logic thì nội dung của đối tượng lệnh được lấy vào sẽ được nạp ngay vào RLO (kết quả của phép toán logic) mà không cần thực hiện phép toán.

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

1.1 Lệnh A

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

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



Hình 5.6. Lệnh A

+ Ấn Enter để trở về màn hình Output.

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

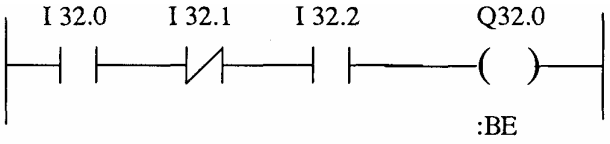
+ Ấn Shift-F7 để cất chương trình và đổ chương trình sang PLC, chọn yes để xác nhận việc đổ đè chương trình lên chương trình cũ trong PLC (khi cất thì PLC phải để ở chế độ STOP).

+ Bật công tắc của CPU về chế độ RUN để chạy chương trình.

1.2. Lệnh AN

Lập trình dạng STL

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

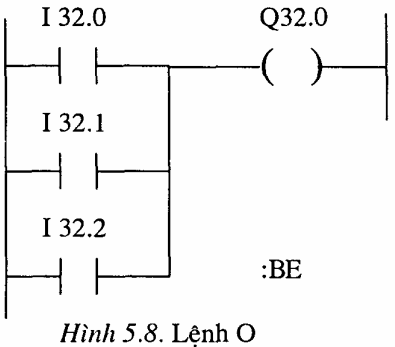


Hình 5.7. Lệnh AN

1.3. Lệnh O

Lập trình dạng STL

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



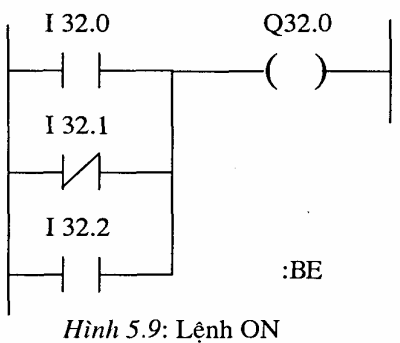
Hình 5.8. Lệnh O

1.4. Lệnh ON

Lập trình dạng STL

```
O    I    32.0
```

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

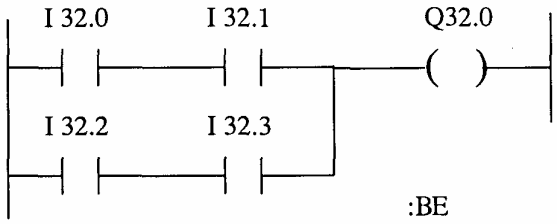


Hình 5.9: Lệnh ON

1.5. 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
```

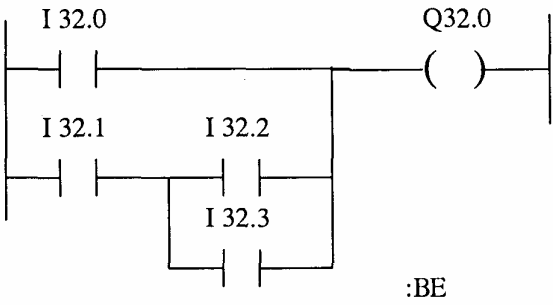


Hình 5.10. Lệnh O giữa hai lệnh A

1.6. 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 ")"

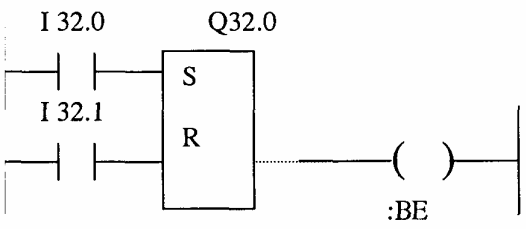
2. Nhóm lệnh set và reset

Các lệnh set và reset để lưu giữ hoặc xoá bỏ 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 l:

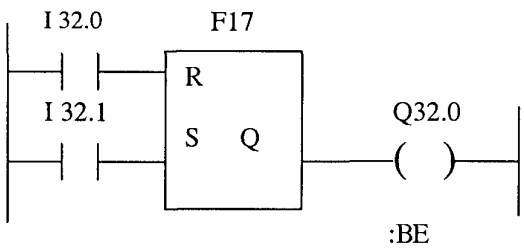
```
A     I    32.0
S     Q    32.0
A     I    32.1
R     Q    32.0
NOP0
```



Hình 5.12. Lệnh set /reset

Khi đầu vào I32.0 có thì đầu ra Q32.0 có và được giữ lại cho dù I32.0 mất, chỉ khi I32.l 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.



Hình 5.13. Lệnh set /reset

Ví dụ 2:

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

Đây là ví dụ về lệnh sét 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.l cùng có trạng thái 1 thì cờ sẽ có trạng thái 1 vì lệnh sét ở sau, gọi là ưu tiên sét.

3. Nhóm lệnh nạp và truyền

Lệnh nạp và truyền để trao đổi thông tin giữa các vùng đối tượng lệnh khác nhau.

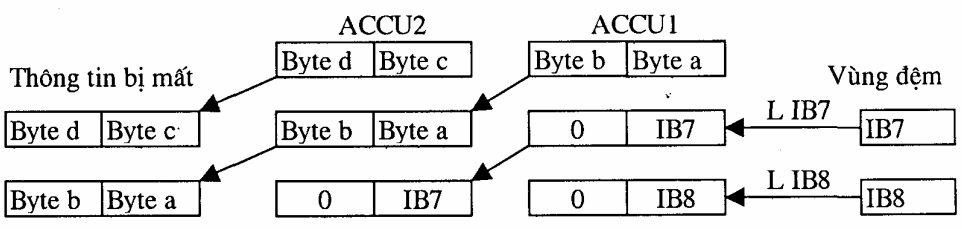
Lệnh nạp và truyền để chuẩn bị giá trị thời gian và giá trị đếm cho các lệnh thời gian và lệnh đếm, nạp hằng số phục vụ việc xử lý chương trình.

Lượng 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 trữ tạm thời các thông tin. Mỗi thanh ghi có độ dài 16 bít.

Có thể nạp hoặc truyền các đối tượng theo byte hoặc từ (word). Để trao đổi theo byte, thông tin lưu trữ trong byte phải tức là byte thấp của thanh ghi, số bít còn thừa (ngoài 8 bít) 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:

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.

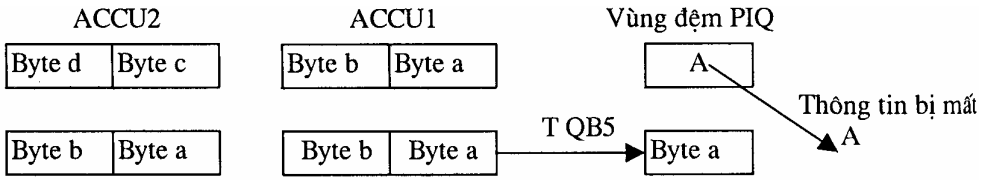


Hình 5.14. Lệnh nạp

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

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.

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.



Hình 5.15. Lệnh truyền

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.

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.

4.1. Nạp giá trị thời gian

Khi một bộ thời gian được khởi phát thì nội dung trong ACCU1 (dạng từ 16 bít) 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 $\Delta t = 40 \times 1s = 40s$.

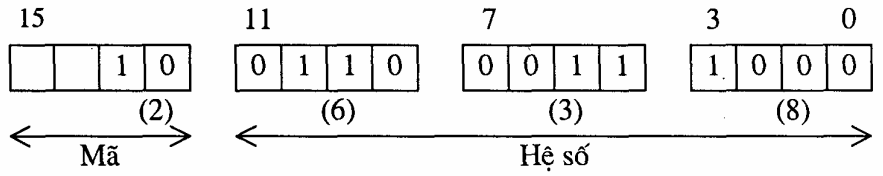
Mã càng nhỏ thì giá trị thời gian càng chính xác, vì vậy nên dùng mã nhỏ.

• *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, 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. Hằng số thời gian dạng mã BCD

Mã thời gian cũng được sử dụng như trên.

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

Vậy, trước khi dùng lệnh nạp trên 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
```

4.2. 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     Tl    % đọc giá trị thời gian dạng nhị phân.
LD    Tl    % 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.

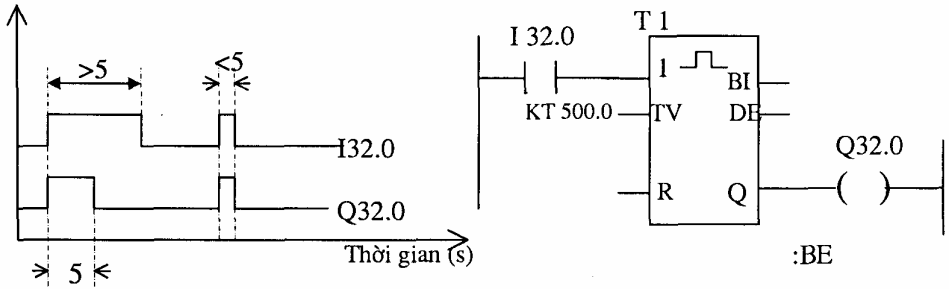
4.3. Các lệnh

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



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

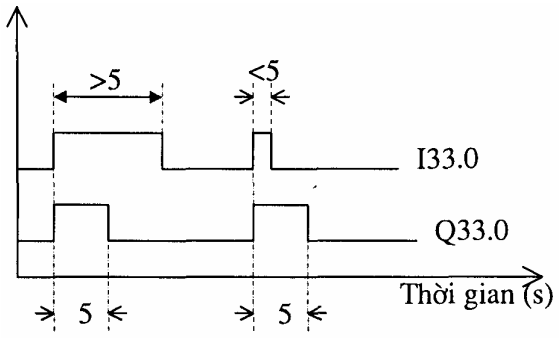
Khi lập trình còn ba chân R, BI và DE chưa sử dụng phải dùng lệnh NOP để giữ chỗ. Chân R là chân để xoá giá trị thời gian hiện hành, chân BI là chân để lấy giá trị thời gian hiện thời dạng nhị phân, chân DE là chân để lấy giá trị thời gian hiện thời dạng mã BCD, có thể dùng lệnh L hoặc LD để đọc các giá trị thời gian.

2. 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
NOP0
NOP0
```



Hình 5.18. Giản đồ thời gian lệnh SE

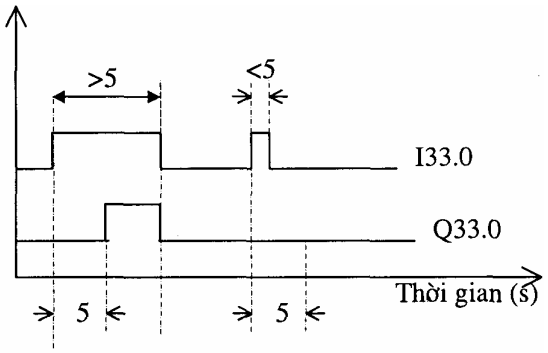
```
NOP0
A     T2
=     Q     33.0
BE
```

3. Bộ thời gian bắt đầu trễ SD

Thời gian bắt đầu chậm hơn so với sườn lên của RLO một khoảng 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     F     W16
NOP0
NOP0
NOP0
=     Q     33.0
BE
```

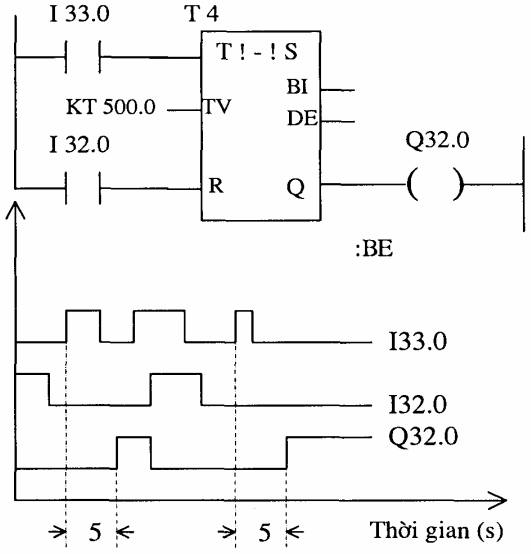


Hình 5.19. Giản đồ thời gian lệnh SD

4. 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
```

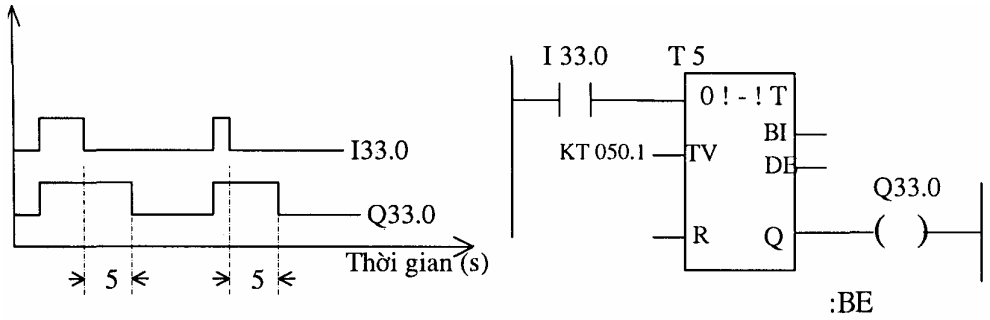


Hình 5.20. Giản đồ thời gian và dạng LAD lệnh SS

5. 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
```



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

5. Nhóm lệnh đếm

5.1. Nạp giá trị đếm

Cũng như bộ thời gian khi một bộ đếm được khởi phát thì nội dung trong ACCU1 (dạng từ 16 bít) 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

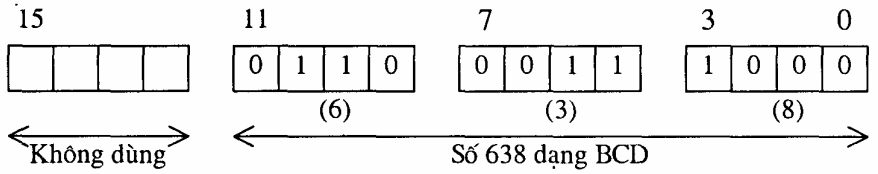
  Số đếm từ 0 đến 999

- *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, viết lệnh sau:

  L DW2

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:



Hình 5.22. Nạp giá trị đếm dạng mã BCD

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.

5.2. Chuẩn bị thực hiện các lệnh đếm

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

+ *Xoá bộ đếm*: Khi đã đếm tới một giá trị nào đó 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*: 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 .

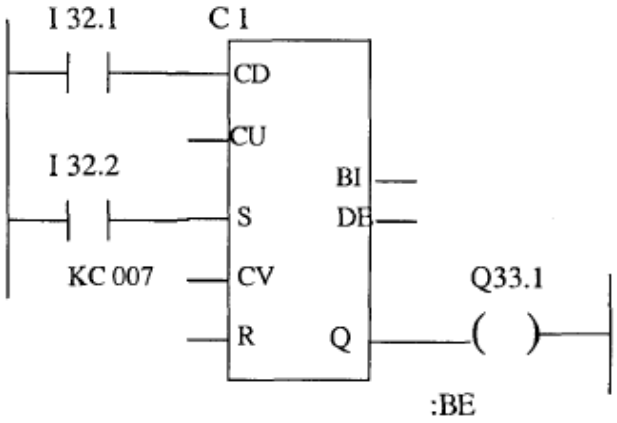
+ *Xuất ra trạng thái bộ đếm hiện hành*: Có thể dùng lệnh L và LD để đưa trạng thái bộ đếm hiện hành vào ACCU1 để xử lý sau này, lệnh L dùng cho số nhị phân, lệnh LD dùng cho số BCD.

4.3. Các lệnh

1. 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     CK   7
S     C    1
NOP        0
```



Hình 5.23. Lệnh đếm xuống CD

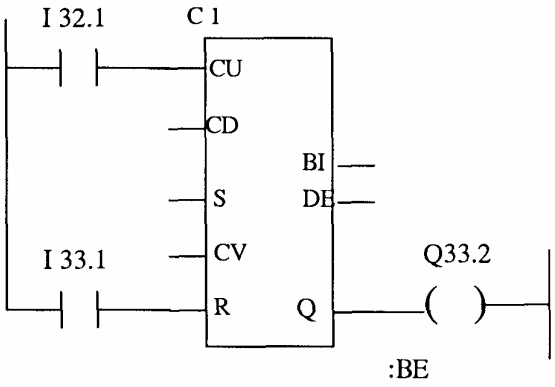
```
NOP     0
NOP     0
A   C   1
BE
```

Chân BI là chân để lấy giá trị đếm hiện thời dạng nhị phân, chân DE là chân để lấy giá trị đếm hiện thời dạng mã BCD, có thể dùng lệnh L hoặc LD để đọc các giá trị đếm.

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 bi ảnh hưởng.

```
A   I   32.1
CU  C   1
NOP0
NOP0
NOP0
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 - S7-20

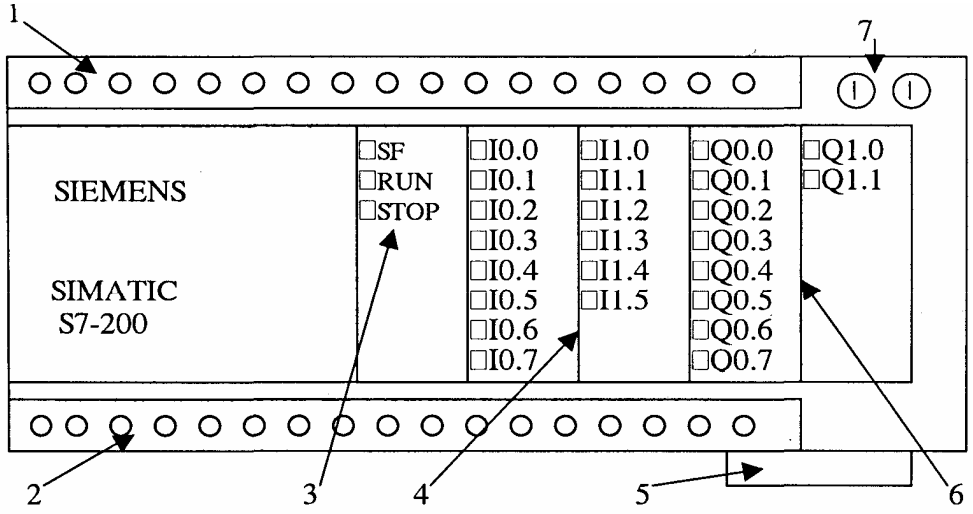
§6.1. Cấu hình cứng

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.

1. Đơn vị cơ bản

1.1. Cấu trúc đơn vị có bản

Đơn vị cơ bản của PLC S7-200 (CPU 3 14) 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

+ RUN: cho phép PLC thực hiện chương trình trong bộ nhớ. PLC sẽ tự 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 đến 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.

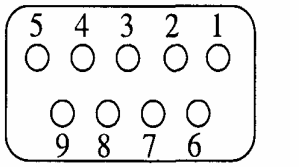
+ TERM: cho phép PLC tự quyết định một chế độ làm việc (hoặc RUN hoặc STOP)

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 với góc quay được 270^o.

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 nguồn 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 (1 20 ma)
8. truyền và nhận dữ liệu
9. không dùng.



Hình 6.2. Cổng truyền thông

1.2. 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 (4 Kbyte) thuộc miền nhớ đọc/ghi không đổi để lưu chương trình (vùng nhớ giao diện với EFROM),

+ 2048 từ đơn (4 Kbyte) 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 10 ms và 108 bộ 100 ms,

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

+ 688 bít 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 2 KHZ và 7 KHZ,

+ 2 bộ phát xung nhanh cho dãy xung kiểu I7ro hoặc kiểu PWM,

+ 2 bộ điều chỉnh tương tự,

+ Toàn bộ vùng nhớ không bị mất dữ liệu trong khoảng thời gian 190^h 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 (lkbyte) 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 1 ms, 8 bộ 10 ms và 54 bộ 100 ms,

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

+ 368 bít 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.

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 (3vào,1 ra)
IO.0 QO.O	I2.0	I3.0	AIW0	Q3.0	AIW8
IO.1 QO.1	I2.1	I3.1	AIW2	Q3.1	AIW10
IO.2 QO.2	I2.2	I3.2	AIW3	Q3.2	AIW 12
IO.3 QO.3	I2.3	I3.3	AIW4	Q3.3	
IO.4 QO.4		I3.4		Q3.4	AQW4
IO.5 QO.5	Q2.0	I3.5	AQWO	Q3.5	
IO.6 QO.6	Q2.1	I3.6		Q3.6	
IO.7 QO.7	Q2.2	I3.7		Q3.7	
I1.0 Q1.0	Q2.3				
I1.1 Ql.l					
I1.2					
I1.3					
I1.4					
I1.5					

Hình 6.3. Địa chỉ các module mở rộng của S7-200

§6.2. Cấu trúc bộ nhớ

Bộ nhớ của PLC S7-200 được chia thành 4 vùng chính đó là:

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.

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.

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 bít, byte, từ (word) hoặc từ kép.

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

STT	Tên tham số	Diễn giải	Tham số	
			CPU 212	CPU214
1	V	Là 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 đọc/ghi	Vùng nhớ đặc biệt	30.0 ÷ 45.7	30.0 ÷ 85.7

Địa chỉ truy nhập được quy ước với công thức:

* Truy nhập theo bít:

Tên miền + địa chỉ byte . chỉ số bít.

Ví dụ : V 150.4 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 tử 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). Quy 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 có các ví dụ:

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

+ 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 bít 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	ACO	ắc quy 0 (không có khả năng làm con trỏ)		
2	AC	ắc quy	1 ÷ 3	1 ÷ 3
3	C	Bộ đếm	0 ÷ 63	0 đến 127
4	HSC	Bô đếm tốc độ cao		0 đến 2
5	AW	Bộ đệm cổng vào tương tự	0 ÷ 30	0 đến 30
6	AQW	Bộ đệm cổng ra tương tự	0 ÷ 30	0 đến 30
7	T	Bộ thời gian	0 ÷ 63	0 đến 127

§6.3. Chương trình của S7-200

1. Cấu trúc chương trình S7-200

Các chương trình điều khiển 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 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.

```
Main Program
⋮
MEND

SBR 0    Chương trình con thứ nhất
⋮
RET

SBR n    Chương trình con thứ n +1
⋮
RET

INT 0    Chương trình xử lý ngắt thứ nhất
⋮
RETI

INT n    Chương trình xử lý ngắt thứ n +1
⋮
RETI
```

Hình 6.4. Cấu trúc chương trình của S7-200

- 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.

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.

2. Bảng lệnh của S7-200

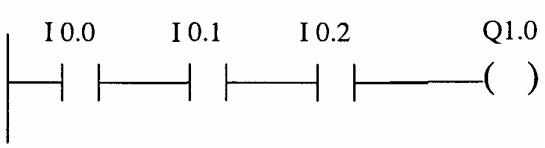
Xem phần phụ lục 2.

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

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
```

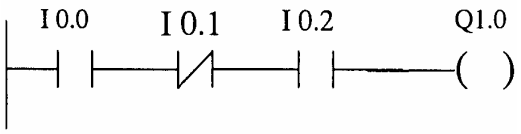


Hình 6.5. Lệnh LD và A

2. Lệnh AN

Lập trình dạng STL

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

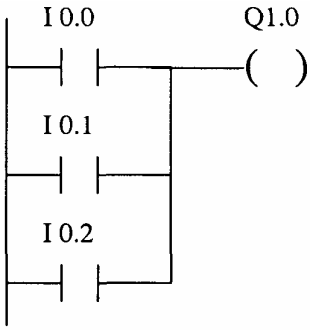


Hình 6.6. Lệnh AN

3. Lệnh O

Lập trình dạng STL

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

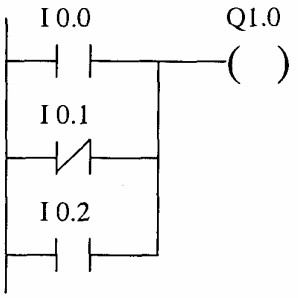


Hình 6.7. Lệnh O

4. Lệnh ON

Lập trình dạng STL

```
LD    I    0.0
ON    I    0. 1
O     I    0.2
=     Q    1.0
```

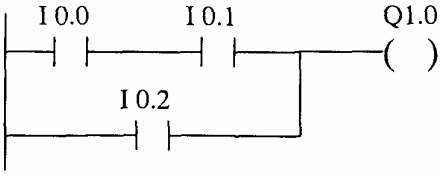


Hình 6.8. Lệnh ON

5. Lệnh OLD

Lập trình dạng STL

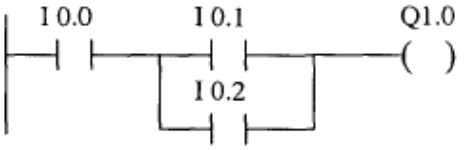
```
LD    I    0.0
A     I    0.1
LD    I    0.2
OLD
=     Q    1.0
```



Hình 6.9: Lệnh OLD

6. Lệnh ALD

Lập trình dạng STL



Hình 6.10. ALD

LD I 0.0

LD I 0.1

O I 0.2

ALD

= Q 1.0

7. Lệnh LPS, LRD,LPP

Lập trình dạng STL

LD I 0.0

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

AI 0.5 = Q 0.2

8. Lệnh TON

NETWORK 1

LD I0.0

AN I0.1

ION T32, VW0

NETWORK 2

LD T32 = Q0

9. Lệnh TONR

NETWORK 1

LD I0.0

AN I0.1

TONR T32, VW0

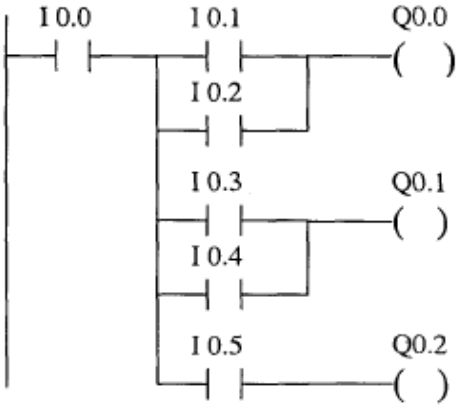
NETWORK 2

LD T32 = Q0.0

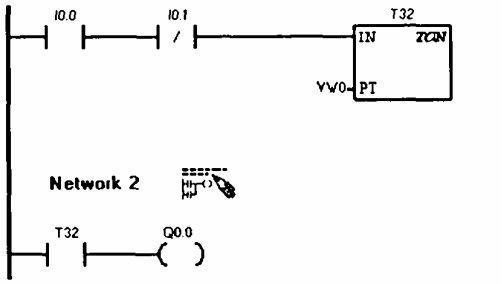
10. Lệnh CTU

NETWORK 1

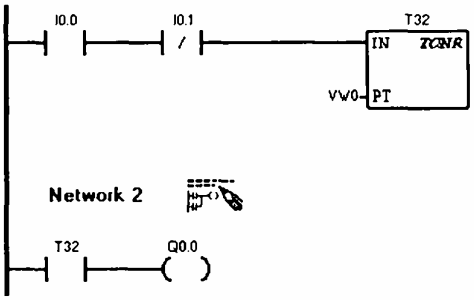
LD I0.0



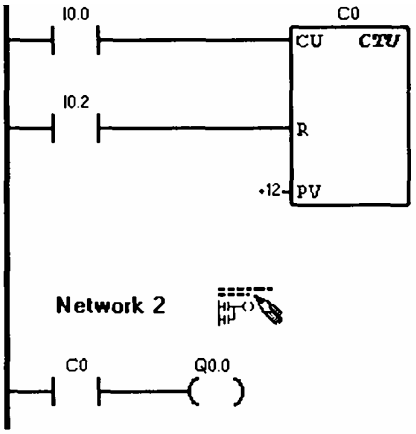
Hình 6.11. Lệnh LPS, LRD, LPP



Hình 6.12. Lệnh TON



Hình 6.13. Lệnh TONR



Hình 6.14. Lệnh CTU

LD I0.2

CTU C0, +12

NETWORK 2

LD C0 = Q0.0

11. Lệnh CTUD

NETWORK 1

LD I0.0

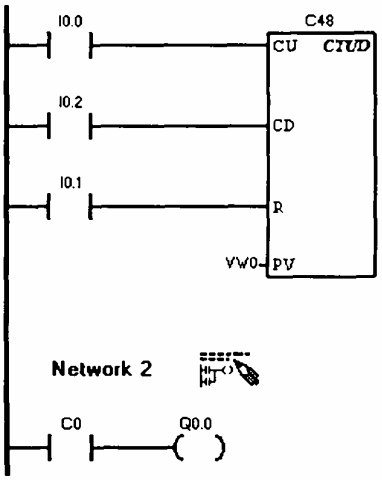
LD I0.2

LD I0.1

CTUD C48, VW0

NETWORK 2

LD C0 = Q0.0



Hình 6.15. Lệnh CTUD

CHƯƠNG 7: BỘ ĐIỀU KHIỂN PLC - S7-300

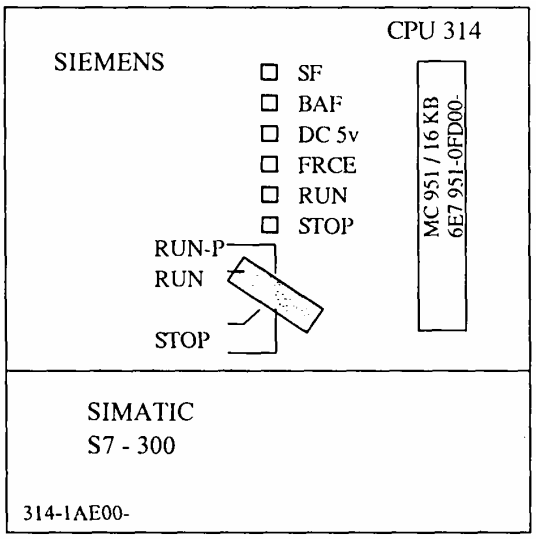
§7.l. Cấu hình cứng

1. Cấu tạo của họ PLC- S7-300

PLC Step S7-300 thuộc họ Simatic do hãng Siemcns 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 mở rộng này bao gồm những đơn vị chức năng mà có thể là hợp lại cho phù hợp với những nhiệm vụ kỹ thuật cụ thể.

1.1. Đơn vị cơ bản

Đơn vị cơ bản của PLC S7-300 như hình 7. 1.



Hình 7.1. Hình khối mặt trước CPU-314

Trong đó:

Các đèn báo:

+ Đèn SF: báo lỗi CPU,

+ Đèn BAF: báo nguồn ắc quy,

+ Đèn DC 5v: Báo nguồn 5v,

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

+ Đèn STOP: Báo PLC dang 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 trong PLC thì giữ nút bấm về vị trí MRES để đèn STOP nhấp nháy, khi thôi không nhấp nháy thì nhả nhanh 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.

1.2. Các kiểu module

Tuỳ theo quá trình tự động hoá đòi hỏi số lượng đầu vào và đầu ra mà 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 các loại: 8 kênh, 16 kênh, 32 kênh,
    - Ra số các loại: 8 kênh, 16 kênh, 32 kênh,
    - Vào ra số các loại: 8 kênh vào 8 kênh ra, 16 kênh vào 16 kênh ra,
    - Vào tương tự các loại: 2 kênh, 4 kênh, 8 kênh,
    - Ra tương tự các loại: 2 kênh, 4 kênh, 8 kênh,
    - Vào, ra tương tự các loại: 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 bốt,
    - Module điều khiển động cơ bước,
    - Module điều khiển động cơ servo.

2. Địa chỉ và gán địa chỉ

Trong PLC các bộ phận con 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...

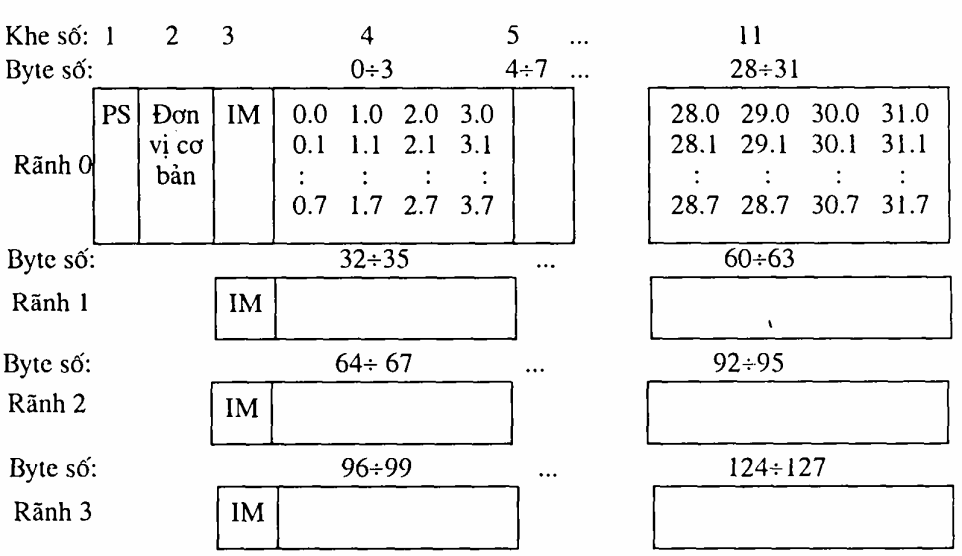
Các địa chỉ đầu vào và đầu ra cùng với các module chức năng có cách gán địa chỉ giống nhau. Địa chỉ phụ thuộc vào vị trí gá của module trên panen. Chỗ gá module

trên pancn gọi là khe (Slot), các khe đều có đánh số, khe số 1 là khe đầu tiên của và cứ thế tiếp tục.

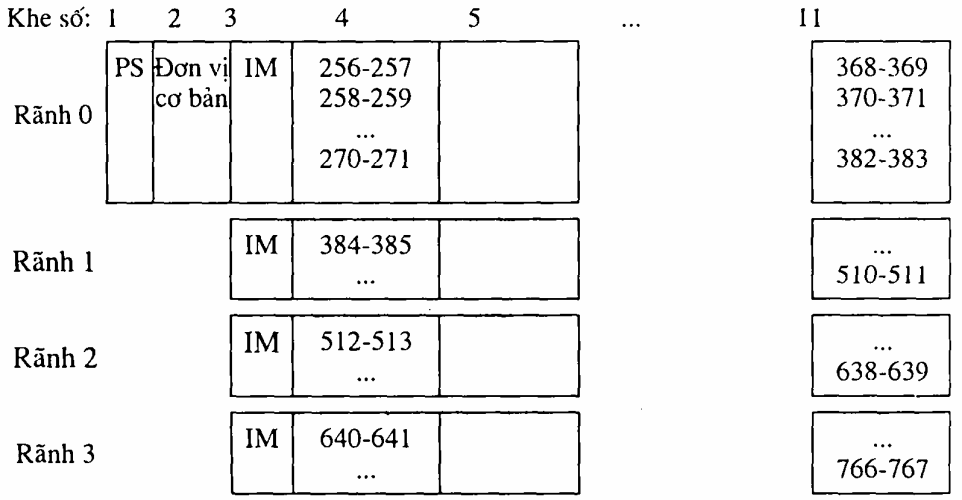
①. Địa chỉ vào ra trên module số:

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.

Trên mỗi module thì mỗi đầu vào, ra là một kênh, các kênh đều có địa chỉ bít 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 lính tử 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 7.2.



Hình 7.2. Địa chỉ khe và kênh trên module số



Hình 7.3. Địa chỉ của module tương tự

Ví dụ: Module 2 dấu vào, 2 đầu ra số gá vào khe số 5 rãnh 0 có địa chỉ là 14.0, I4.1 và Q4.0, Q4.1.

Module số có thể được gá trên bất kỳ khe nào trên panen của PLC.

②. Địa chỉ vào ra trên module tương tự

Để diễn tả một giá trị tương tự phải cân nhiều bít. Trong PLC S7-300 người ta dùng 16 bít (một word) cho một kênh. Một khe có 8 kênh với địa chỉ đầu liên là PIW256 hoặc PQW256 (byte 256 và 257) cho đến PIW766 hoặc PQW766 như hình 7.3

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

1. Các vùng nhớ

Bảng 7.1

TT	Tên tham số	Diễn giải	vùng tham số
1	I	Đầu vào bít	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 tử kép	0.0 đến 65532
5	Q	Đầu ra bít	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 bít	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 byle	0 đến 65535
14	PIW	Vùng đệm đầu vào dạng tử	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 tử 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 bít	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 tử	0 đến 65534
24	DBD	Khối dữ liệu kiểu BD dạng từ kép	0 đến 65532

TT	Tên tham số	Diễn giải	vùng tham số
25	DIX	Khối dữ liệu kiểu BI dạng bít	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 tử kép	0 đến 65532
29	L	Vùng dữ liệu tạm thời dạng bít	0.0 đến 65535.7
30	LB	Vùng dữ liệu tạm thời dạng byte	0 đến 65535
31	LW	Vùng dữ liệu lạm thời dạng lử	0 đến 65534
32	LD	Vùng dữ liệu tạm thời dạng từ kép	0 đến 65532

2. Nhập các hằng số

Các hằng số được viết gồm phần đầu và tham số di 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	Bít	Cơ số	Phần dầu	Phạm vi tham số
Byte	8	16	B#16#...	0 đến rF
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	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 ± l.175495e - 38
Thời gian	16 32	giờ_phút_ giây_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_tháng_ngày	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

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 qua 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.

* *Khối 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 (Sllared 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 trong 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).

2. Bảng lệnh của S7-300

Xem phần phụ lục 2.

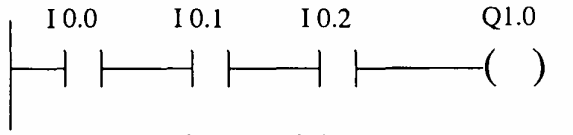
§7.4. Lập trình một số lệnh cơ bản

1. Nhóm lệnh 1ogic

1.1 Lệnh LD và lệnh A

Lập trình dạng STL

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

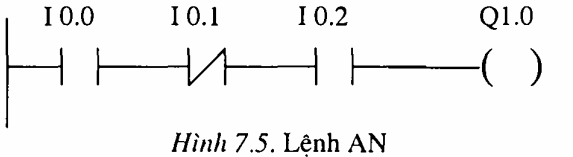


Hình 7.4. Lệnh LD và A

1 2. Lệnh AN

Lập trình dạng STL

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

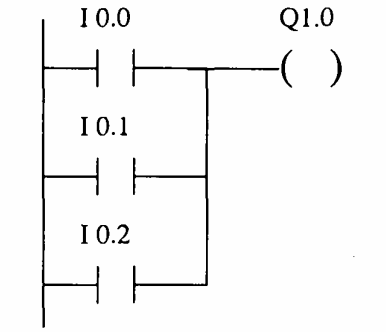


Hình 7.5. Lệnh AN

1.3. Lệnh O

Lập trình dạng STL

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

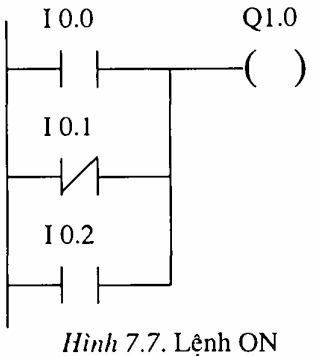


Hình 7.6: Lệnh O

1.4. Lệnh ON

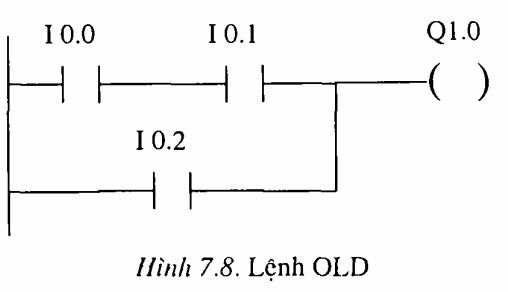
Lập trình dạng STL.

```
O    I    0.0
ON   I    0.1
```



Hình 7.7. Lệnh ON

```
O     I     0.2
=     Q     1.0
```

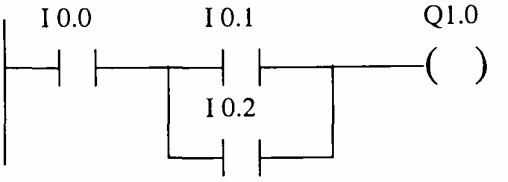


Hình 7.8. Lệnh OLD

1.5. Lệnh A và lệnh O

Lập trình dạng STL

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



Hình 7.9. Lệnh *"(" và lệnh ")"*

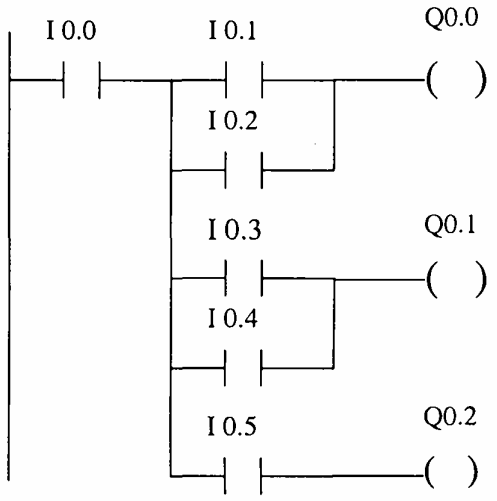
1.6. Lệnh "(" và lệnh ")"

Lập trình dạng STL

```
A     I     0.0
A(
O     I     0.1
O     I     0 2
)
=     Q     1.0
```

1 7. Lập trình với vùng dữ liệu tạm thời L

```
A     I     0.0
=     L     20.0
A     L     20.0
A(
O     I     0.1
O     I     0.2
)
=     Q     0.0
A     L     20.0
A(
O     I     0.3
O     I     0.4
)
=     Q     0.1
A     L     20.0
A     I     0.5
=     Q     0.2
```



Hình 7.10. Lập trình với vùng dữ liệu tạm thời

1.8. Lập trình với bít nhớ nội M

Nework 1 :

A I 0.0

= M 10.0

Nework 2:

A I 0.1

= M 10.1

A M 10.1

= Q 0.0

Network 3:

A(

O I 1.0

O Q 1.0

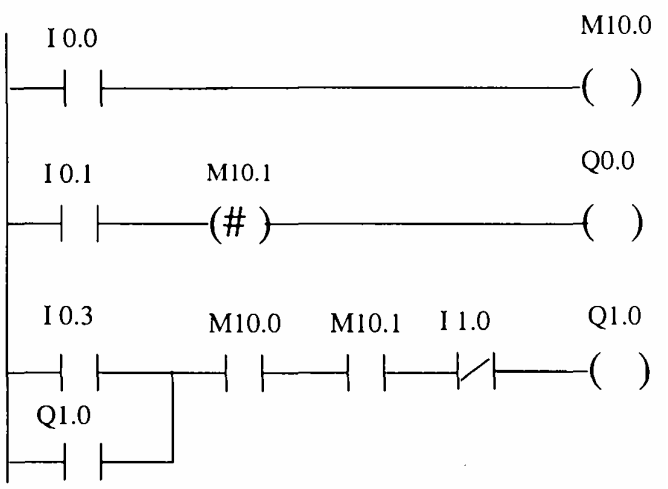
)

A M 10.0

A M 10.1

AN I 1.0

= Q 1.0



Hình 7.11. Lập trình với bít nhớ nội M

2. 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.

Khi một bộ thời gian được khởi phát thì giá trị thời gian cần được nạp vào thanh ghi CV (Current value). Do đó, muốn dùng các lệnh thời gian phải nạp giá trị thời gian cần đặt vào thanh ghi CV trước khi bộ thời gian hoạt động.

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

+ Dữ liệu thời gian thực: S5T#H_M_S_MS

+ Dạng số nguyên 16 bít: W#16#.... (ở dạng mã BCD)

- *Nạp thời gian thực:* L S5T#10s

Với lệnh trên giá trị thời gian được nạp là 10s

- *Nạp thời gian dạng mã BCD:*

Ví dụ*:* L W#16#2127

Số trên sẽ được nạp vào thanh ghi CV dạng mã BCD như hình 7. 12.

Trong thanh ghi CV thì:

Ba số cuối chỉ hệ số: Số 127 (có thể gán từ 0 đến 999)

Số đầu chỉ mã số: Số 2. có 4 mã:

0 tương ứng 0,01 s

1 tương ứng 0,1 s

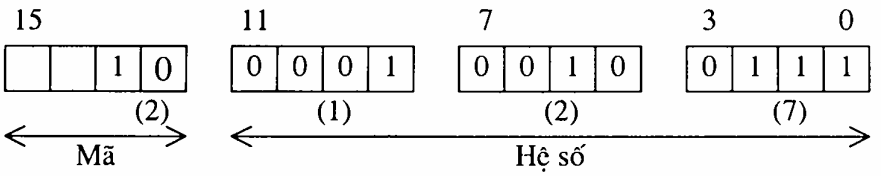
2 tương ứng 1s

3 tương ứng 10s

Với số đã vào thanh ghi CV như trên thì thời gian được tính là

$$\Delta t : 127 \times 1s = 127s.$$

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



Hình 7.12. Nạp hằng số thời gian dạng mã BCD

Trong các bộ thời gian của S7-300 ngoài tín hiệu kích thích chính (bắt đầu) như các bộ thời gian của các PLC khác, còn có tín hiệu kích thích cưỡng bức, tín hiệu kích thích cưỡng bức cho phép tính lại thời gian từ đầu khi có sườn lên của tín hiệu này, tuy nhiên tín hiệu kích thích cưỡng bức chỉ có giá trị khi tín hiệu kích thích chính có giá trị 1. Lệnh thực hiện kích thích cưỡng bức (có điều kiện) là: FR.

Lệnh FR chỉ có ở dạng lập trình STL. Bộ thời gian cũng có thể dùng lệnh R để xoá.

2.1. Bộ thời gian xung SP

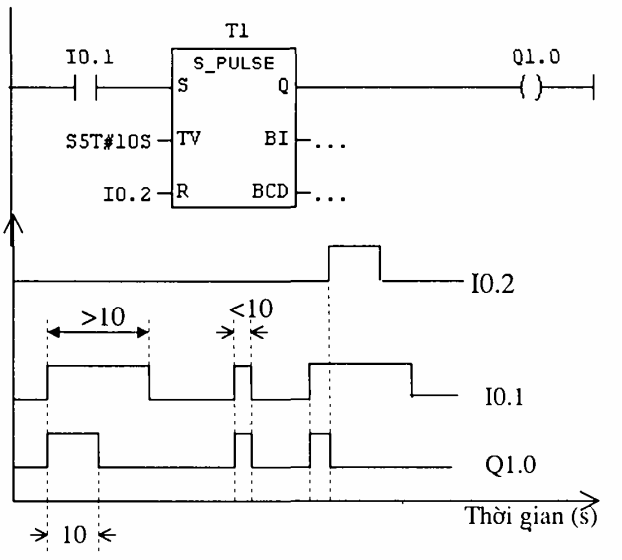
Bộ thời gian được khởi phát lên 1 tại sườn lên của RLO khi RLO là 1 thì bộ thời gian vẫn duy trì trạng thái 1 cho đến khi đạt giá trị đặt mới xuống. Nhưng khi RLO về không thì bộ thời gian về không ngay.

Có hai kiểu lập trình:

Kiểu thứ nhất có lệnh NOP:

```
A     I     0.1
L     S5T#10S
SP    T     1
A     I     0.2
R     T     1
NOP   0
NOP   0
A     T     1
=     Q     1.0
```

Dạng LA D hình 7. 1 3.

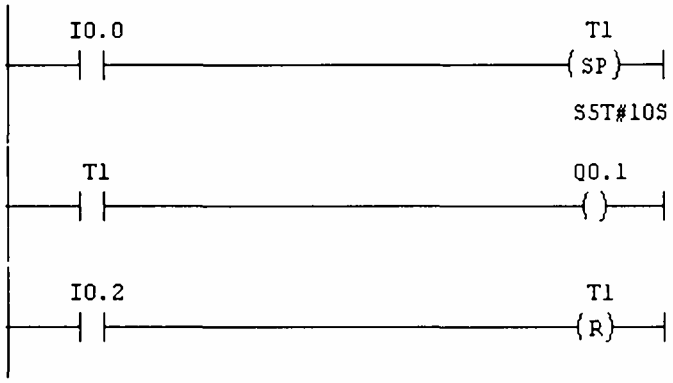


Hình 7.13. Dạng LAD và giản đồ thời gian lệnh SP kiểu 1

Trong lập trình trên còn hai chân BI và BCD chưa sử dụng phải dùng lệnh NOP để giữ chỗ. Chân BI là chân để lấy giá trị thời gian hiện thời dạng nhị phân, chân BCD là chân để lấy giá trị thời gian hiện thời dạng mã BCD, có thể dùng lệnh L hoặc LC để đọc các giá trị thời gian.

Kiểu thứ hai (không dùng lệnh NOP)

```
Nework 1:
      A   I    0.0
      L   S5T#10S
      SP  T    1
Nework 2:
      A   T    1
      =   Q    0.1
Nework 3:
      A   I    0.2
      R   T    1
```



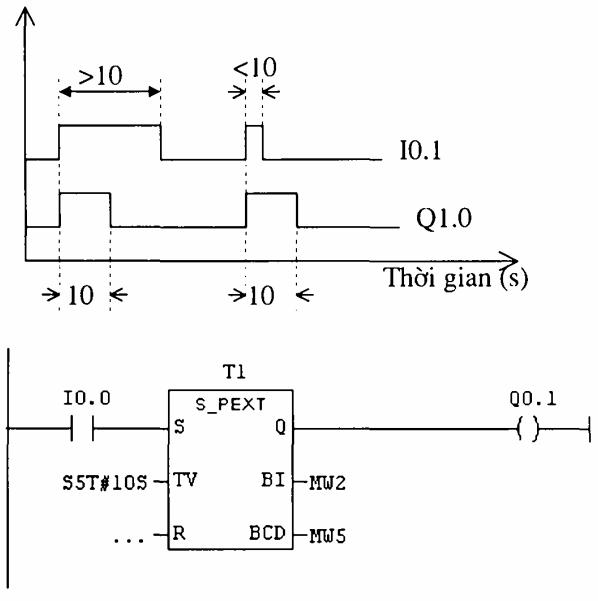
Hình 7.14. Dạng LAD lệnh SP kiểu 2

Dạng LAD hình 7.14.

2.2. Bộ thời gian mở rộng SE

Bộ thời gian xung mở rộng SE được khởi phát lên 1 lại sườn lên của RLO sau đó không phụ thuộc RLO nữa cho đến khi đủ thời gian đặt mới về không. Cũng tương tự như bộ thời gian SP, ở các bộ thời gian khác cũng luôn có hai kiểu lập trình.

```
A     I     0.0
L     S51 # 10S
SE    T     1
NOP   0
L     T     1
T     MW    2
LC    T     1
T     MW    5
A     T     1
=     Q     0.1
```



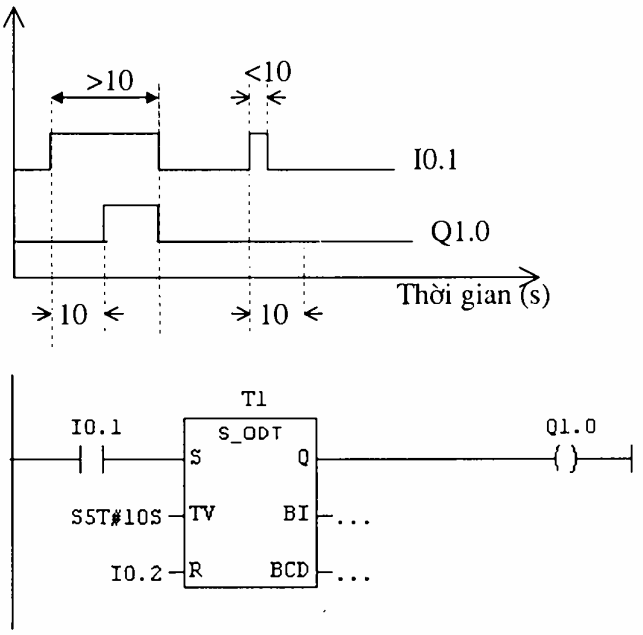
Hình 7.15. Lệnh SE

2.3 Bộ thời gian bắt đầu trễ SD

Thời gian bắt đầu chậm hơn so với sườn của RLO một khoảng bằng thời gian đặt trong lệnh. Khi RLO về không thì bộ thời gian cũng bị đặt ngay về

không.

```
A   I   0.1
L   S5T#10S
SD  T   1
A   I   0.2
R   T   1
NOP 0
NOP 0
A   T   1
=   Q   1.0
```

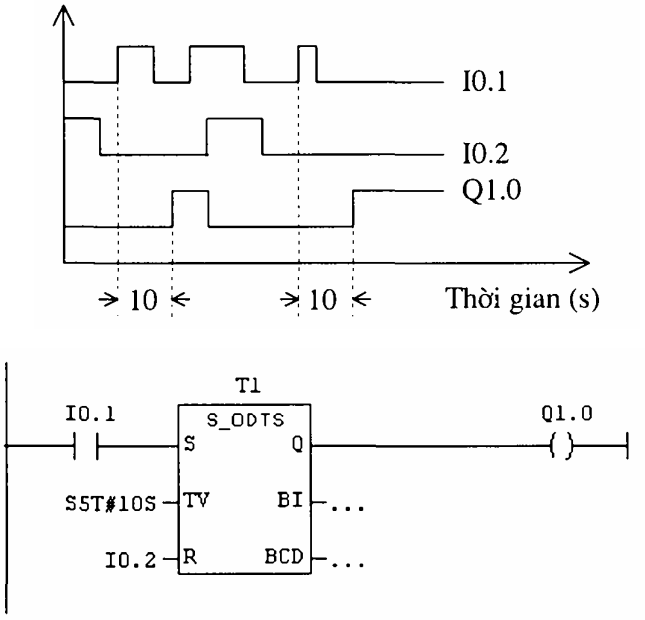


Hình 7.16. Lệnh SD

2.4.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     0.1
L     S5T#10S
SS    T     1
A     I     0.2
R     T     1
NOP   0
NOP   0
A     T     1
=     Q     1.0
```



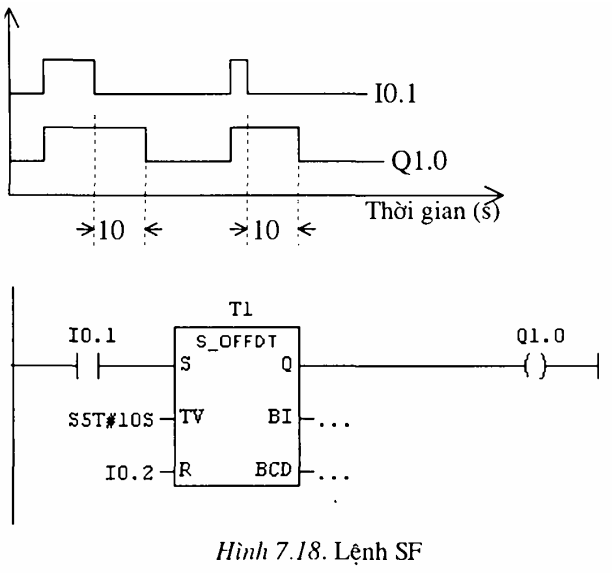
Hình 7.17. Lệnh SS

2.5. 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     0.1
L     ST5#10S
SF    T     1
A     I     0.2
NOP   0
NOP   0
A     T     1
=     Q     1.0
```



Hình 7.18. Lệnh SF

3. Nhóm lệnh đếm

Giá trị trong thanh ghi CV (current value) là giá trị đếm tức thời của bộ đếm, CV luôn không âm, do đó lệnh đếm lùi sẽ không dẫn khi CV = 0.

Giá trị đếm PV có thể được đặt trước bằng lệnh L, ví dụ L C#4 (đặt giá trị đếm bằng 4). Tuy nhiên, khác với bộ thời gian, giá trị đếm chỉ được nạp vào CV khi có lệnh đặt bộ đếm (S). Nếu không đặt giá trị đếm thì bộ đếm có thể vẫn tiến hành đếm (chỉ khi CV = 0 thì không đếm lùi).

Giá trị đầu ra của bộ đếm sẽ là 1 nếu CV ≠ 0, bằng 0 nếu CV = 0.

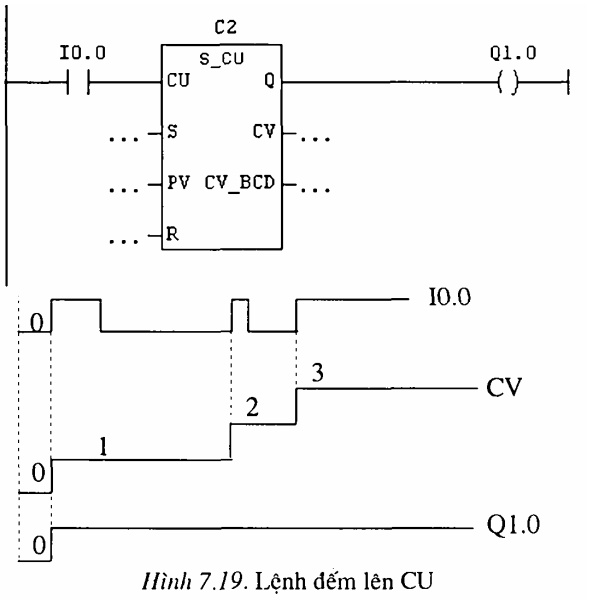
Bộ đếm có thể được xoá chủ động bằng tín hiệu xoá R.

Cũng tương tự như bộ thời gian, bộ đếm cũng có thể dùng lệnh kích đếm (đếm cưỡng bức) FR (lệnh có điều kiện), bộ đếm cũng đếm xung khi điều kiện của FR đảm bảo. Lệnh FR chỉ có ở dạng lập trình STL.

Có thể dùng lệnh L hoặc LD để đọc giá trị tức thời của bộ đếm vào ACCU1 để xử lý. Lệnh L đọc số dạng cơ số 2, lệnh LD đọc số dạng BCD.

3.1. Lệnh đếm lên CU

```
A     I     0.0
CD    C     2
BLD   101
NOP 0
NOP 0
NOP 0
NOP 0
NOP 0
A     C     2
=     Q     1.0
```



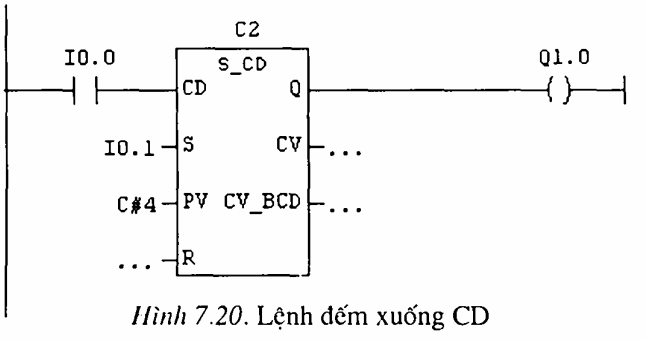
Hình 7.19. Lệnh đếm lên CU

Lệnh BLD để hiển thị dạng LAD. Với các lệnh trên khi đầu vào IO 0 có sườn lên thì giá trị bộ đếm CV tăng thêm 1 đơn vị, tức là khi đã có chỉ một lần sườn lên của 10.0 thì đầu ra Qui luôn là 1 (không xoá).

Chân CV là chân để lấy giá trị đếm dạng nhị phân, chân CV_BCD là chân để lấy giá trị đếm dạng mã BCD, có thể dùng lệnh L hoặc LC để đọc các giá trị đếm.

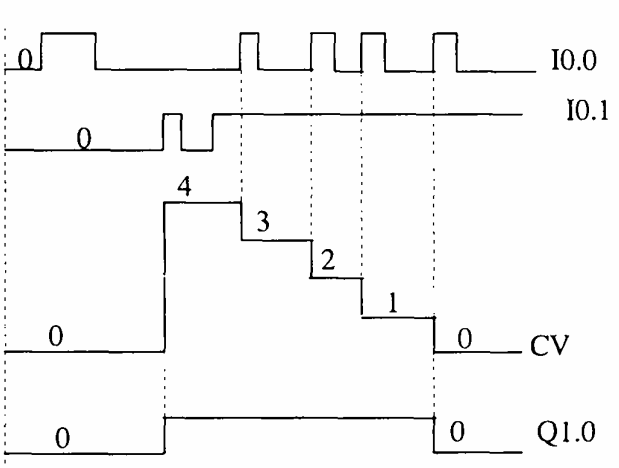
3.2. Lệnh đếm xuống CD

```
A     I     0.0
CD    C     2
BLD   101
A     I     0.1
L     C#4
NOP   0
NOP   0
NOP   0
A     C     2
=     Q 1.0
```



Hình 7.20. Lệnh đếm xuống CD

Trong phần lập trình trên có: Lệnh L C#4 là nạp số đếm bằng 4. Trên hình 7.20 khi I0.0 có trước, bộ đếm vẫn không làm việc vì khi đó CV = 0, cho đến khi có lệnh đặt bộ đếm, I0.1 có thì bộ đếm bắt đầu được nạp giá trị đếm, CV = 4. Từ khi này mỗi lần I0.0 có thì giá trị đếm giảm một đơn vị, sau 4 xung vào giá trị đếm CV = 0 Khi CV≠ 0 đầu ra Q1.0 có, khi CV = 0 đầu ra Q1.0 mất.



Hình 7.21. Xung đếm lệnh đếm xuống CD

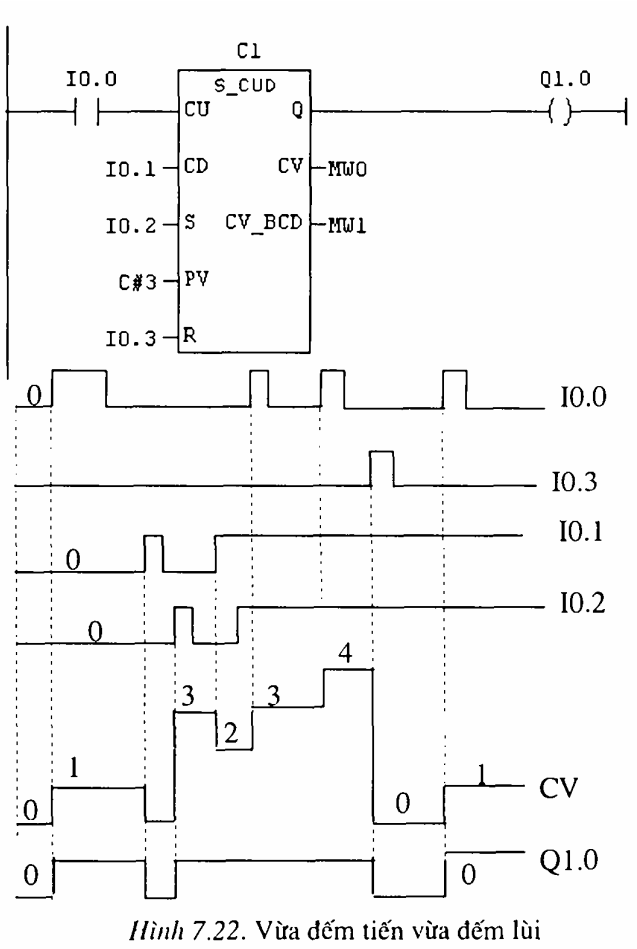
3.3. Lệnh đếm vừa tiến vừa lùi

```
A     I     0.0
CU    C     1
A     I     0.1
CD    C     1
A     I     0.2
L     C#3
A     I     0.3
R     C     1
```

```
L     C     1
T     MW    0
LC    C     1
T     MW    1
A     C     1
=     Q 1.0
```

Từ giản đồ nhận thấy: khi đầu vào đếm tiến có lập tức bộ đếm làm việc, giá trị đếm tăng 1 đơn vị, CV ≠ 0, đầu ra Q 1.0 có. Tiếp đó đầu vào đếm lùi có, do do bộ đếm lại giảm 1 đơn vị (CV = O) đầu ra Q1.0 lại mất.

Tuy nhiên, nếu đầu vào đếm lùi có trước thì bộ đếm không đếm vì khi đó CV = 0 Tiếp đó dấu vào đặt bộ đếm SET có làm giá trị đếm được nạp vào CV (CV = 3), từ đó nếu có đầu đếm tiến thì giá trị đếm tăng 1 đơn vị, có đầu đếm lùi giá trị đếm giảm 1 đơn vị, đầu ra Q1.0 có. Khi có đầu RESET giá trị đếm lập tức về 0, đầu ra Q1.0 về 0.



Hình 7.22. Vừa đếm tiến vừa đếm lùi

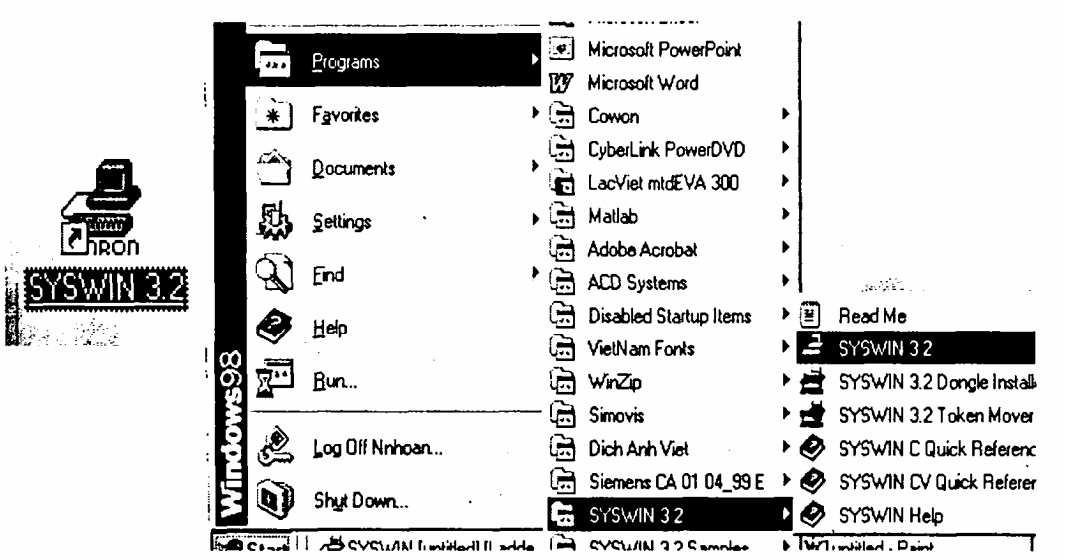
PHỤ LỤC 1
CÁC PHẦN MỀM LẬP TRÌNH PLC

1. Tập trình cho OMRON

1. Phần mềm SYSWIN (cho OMRON)

Phần hướng dẫn được thực hiện trên version 3.2.

1.1. Khởi động

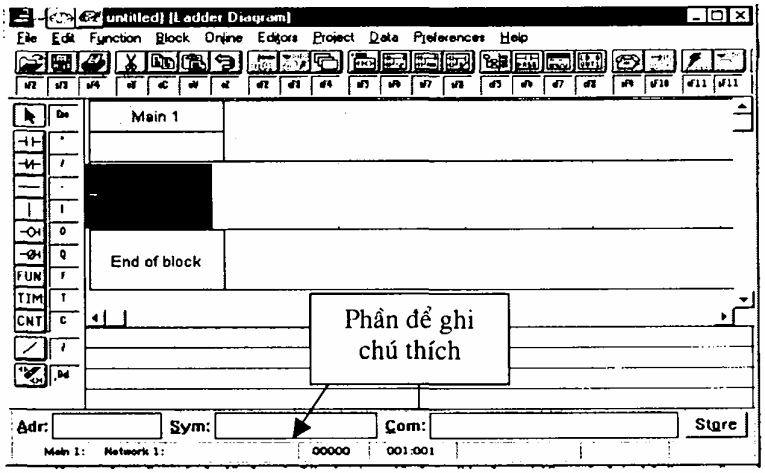


Hình P.1. Khởi động phần mềm SYSWIN

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

2. Khởi động phần mềm SYSWIN từ biểu tượng hoặc từ file chương trình như hình P. 1. Cửa sổ màn hình ban đầu có dạng như hình P.2. Trong cửa sổ có 2 thanh công cụ hỗ trợ cho quá trình soạn thảo chính là:

- 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 P.3.
- Thanh dọc: Lần lượt tử 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 (RUN), bộ thời gian (TIM), bộ đếm (CNT).



Hình P.2. Màn hình ban đầu

3. Kiểm tra một số điều kiện trước khi tập 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 P.4. (đường dẫn Project \ Communications).

1.2. 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).

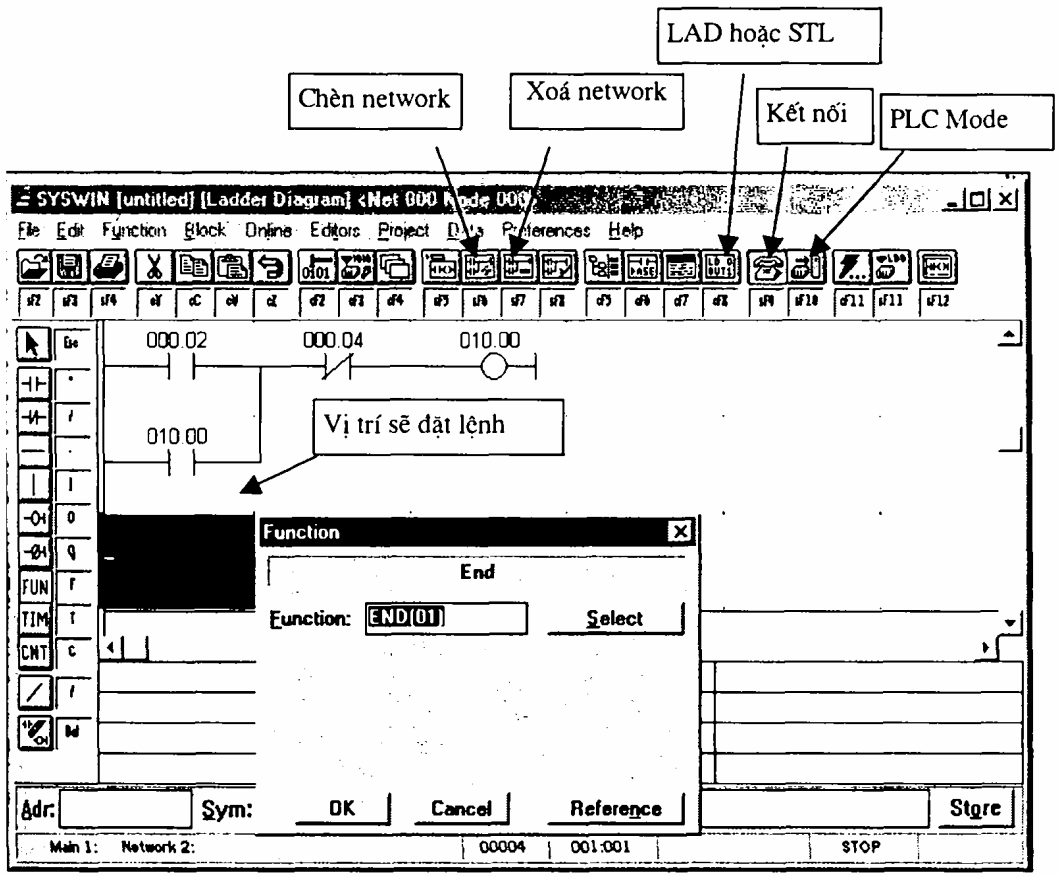
2. Nháy chuột trái vào khối muốn chọn (tiếp điểm, cuộn dây. khối hàm....).

3. Đưa con cho 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 chỉ ra trên hình P.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 một từ biểu tượng như chỉ ra trên hình P.3.

6. Nếu soạn sai Network nào thì đánh dấu và xoá Network đó từ biểu tượng hình P.3.



Hình P.3. Một số chức năng chính

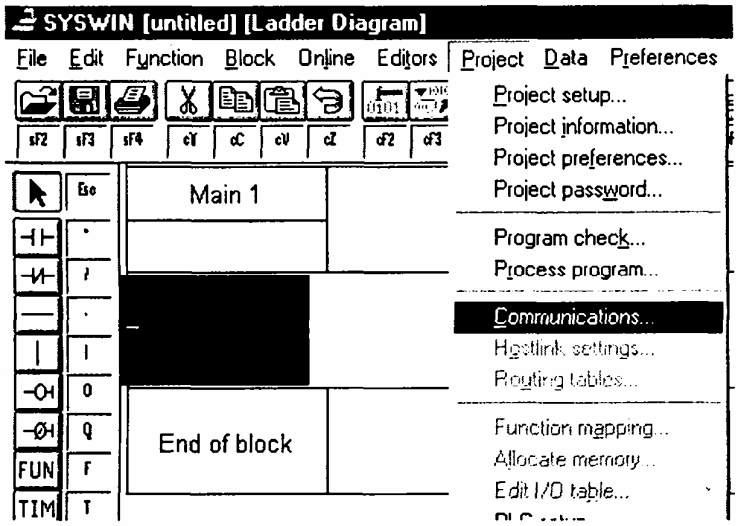
7. Tiến hành soạn thảo hết các Network.

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 P.3, hoặc chọn các khối ở mục Select sau đó chọn OK.

9. Để đổ chương trình sang PLC chọn Online \ Download program to PLC như trên hình P.5.

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 P.3.

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



Hình P.4. Khai báo cổng ghép nối

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

2.1. 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,

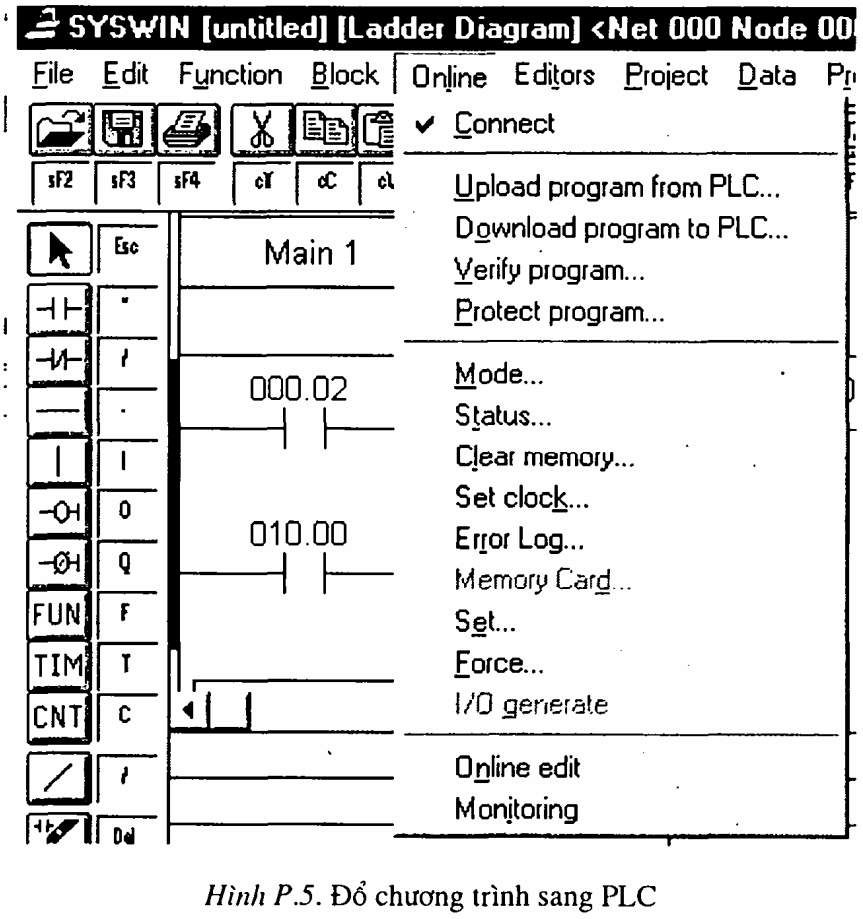
* **MONITOR**: Chế độ này để thay đổi các giá trị của bộ đếm và thời gian trong khi PLC vẫn đang vận hành,

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

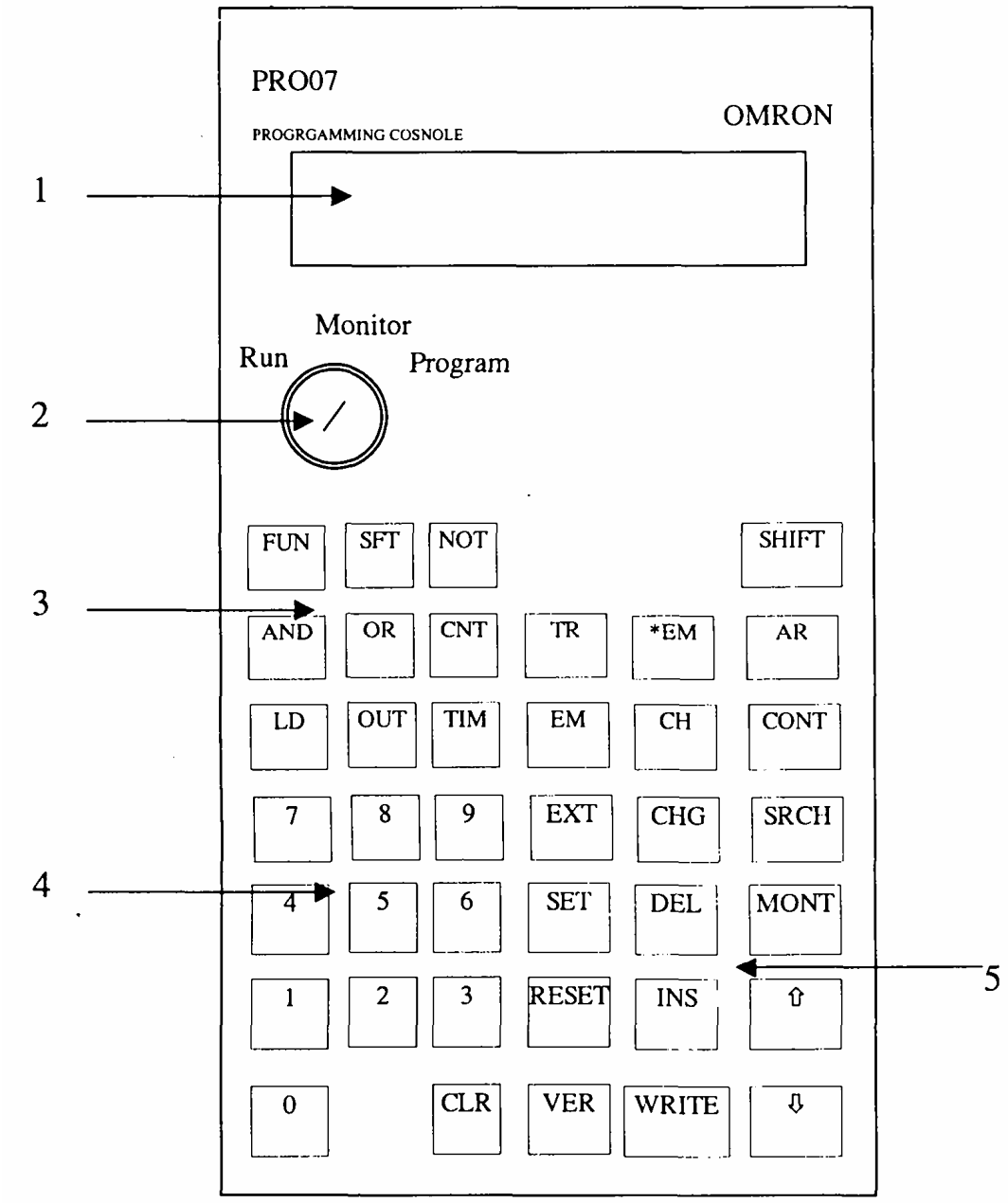
3. Các phím lệnh,

4. Các phím số,

5. Các phím hàm.



Hình P.5. Đổ chương trình sang PLC



Hình P.6. Ghép nối PLC với thiết bị lập trình cầm tay

2.2. 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

2.3. Thủ tục vào lệnh

1. Khởi động bộ tập 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.

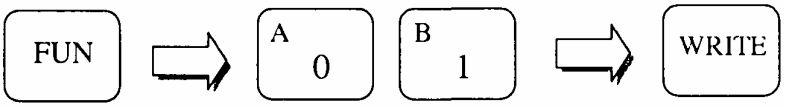
3. Các lệnh được vào theo thứ tự:

+ Tên lệnh (các lệnh bát 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).

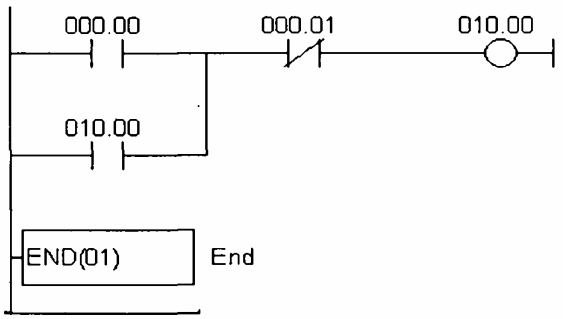
4. Kết thúc một chương trình phải có lệnh kết thúc. Lệnh kết thúc vào theo thứ tự:



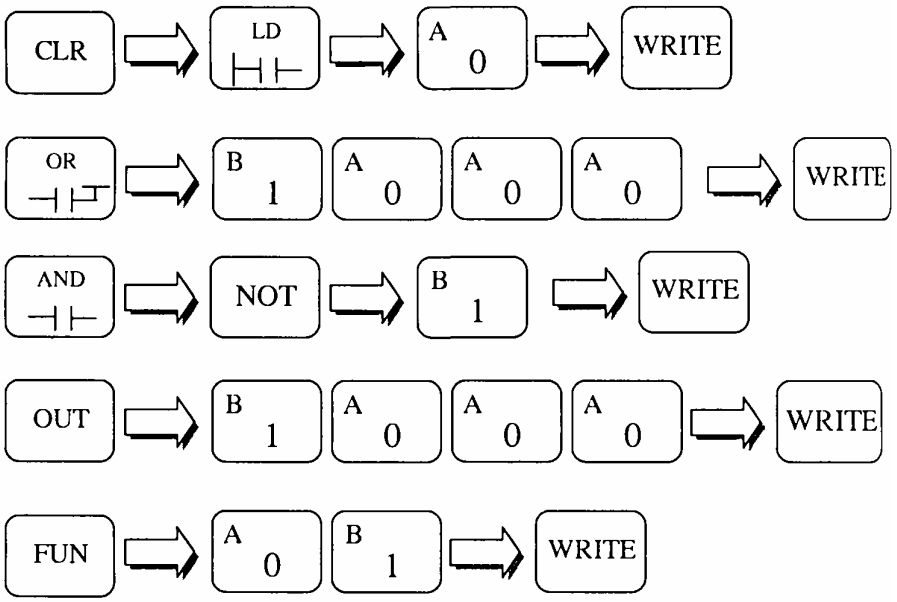
Ví dụ: Chương trình của một mạch tự duy trì dạng LAD và STL như hình P.7:

LD 000.00
OR 010.00
AND NOT 000.01
OUT 010.00
END.

Cách vào chương trình hình P.7 như sau:



Hình P.7. Mạch tự duy trì

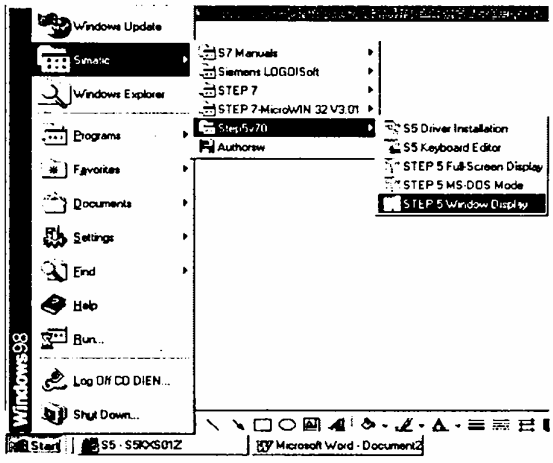


6. Để chạy chương trình chuyển công tắc chọn chế độ sang RUN.

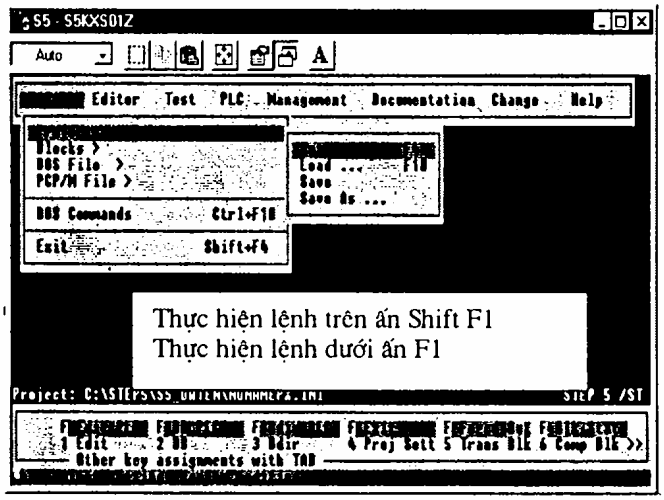
2. Lập trình cho PLC - S5

Sử dụng phần mềm Step5 for Win.

1. Trình tự thao tác



Hình P.8. Khởi động Step 5



Hình P.9. Màn hình ban đầu

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

2. chạy trình Step5 từ file chương trình như hình P.8.

Màn hình chế độ bắt đầu có dạng như hình P.9.

3. Vào lúc \ Project \ Set (phần này có thể đái nhiều tham số, xem phần đặt tham số hình P.12 đến P. 17). 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.

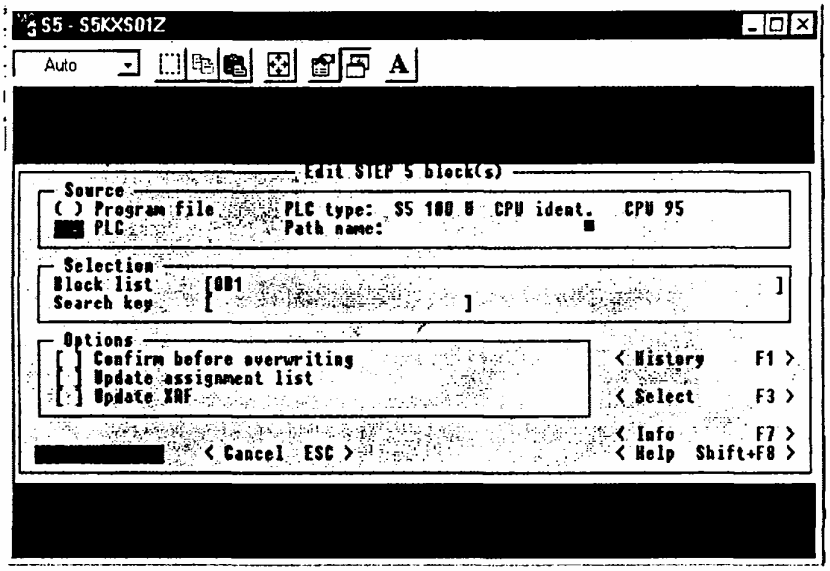
+ Chọn Blocks \ Program Eile để tạo file mới, (nếu cần mở một file đã có thì vào đường dẫn và lên file, nếu sử dụng file ngay buổi làm việc trước và chương trình trước đây đã kết nối với PLC thì bỏ qua bước này) sau đó ấn Enter.

4. Vào chế độ soạn thảo từ Editor \ Step 5 Block...., hoặc ấn F1 (Edit). Màn hình trước soạn thảo có dạng như hình P.10.

Trong đó:

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

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

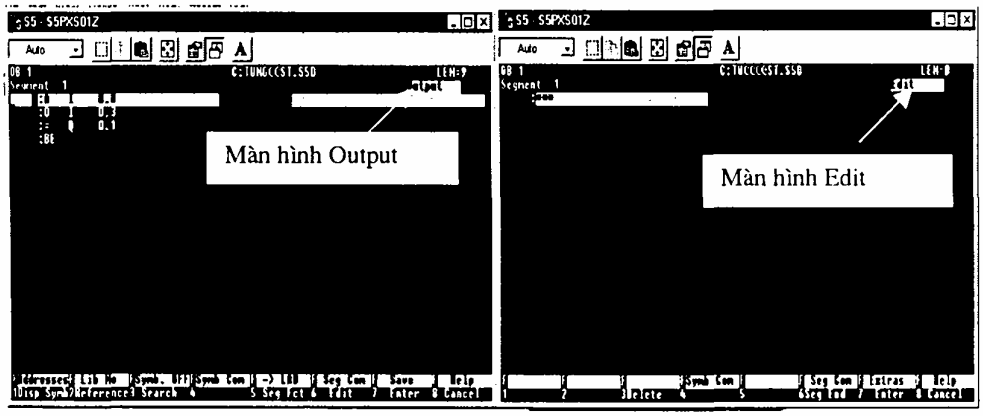


Hình P.10. Màn hình trước soạn thảo

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

Update XRF: Nếu được chọn thì file *XR.INI chứa tham chiếu chéo được điều chỉnh hoặc được tạo nếu chưa tồn tại trước đó, nếu không chọn thì file *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 dễ soạn thảo (có thể vào các khối khác nếu cần), trong mục Options không chọn như hình P.10 sau đó chọn Edit (ấn Enter), nếu làm việc với file mới thì máy tự động vào luôn màn hình Edit như hình P.11b nếu làm việc với file cũ thì máy vào màn hình Output như hình P.11a.



a, b,

Hình P.11. Màn hình soạn thảo

Trong đó: hình P.11a

Fl (Disp Symbb) 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 dổi hoặc trở về menu chính.

F8 (Cancet): Trở về menu chính.

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

Shift F2 (Liu no): Cho phép vào sổ 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 hình P.13 *(Btocks).*

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 Fl (- 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 thường không cần quan tâm đến cấu trúc và 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 Fl (-l) hoặc F2 (+l) 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).

2. Đặt tham số cho việc soạn thảo chương trình

Vào File \ Project \ Set để 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 hình P.12

đến P. 1 7, 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.

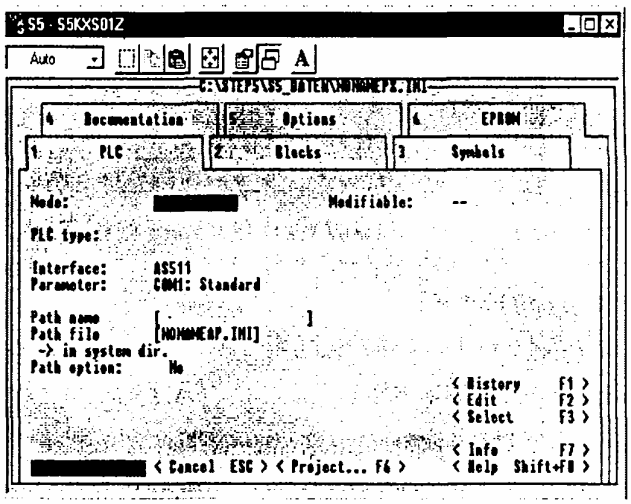
+ *Project... F6:* Cất tham số dã thay đổi.

+ *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 P.12. Trang 1

* Trang 1 (PLC): như hình P.12

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

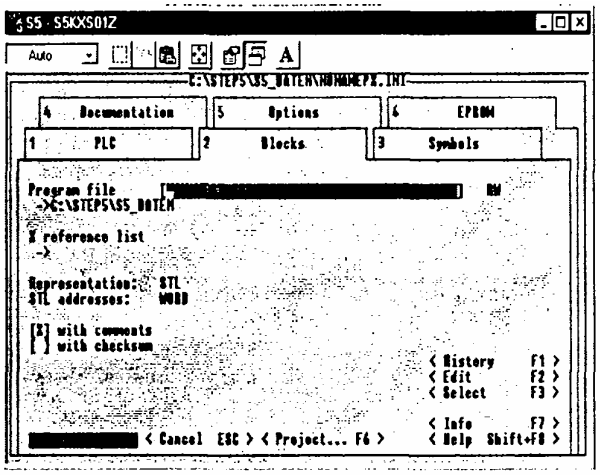
+ *PLC type:* Loại PLC

+ *Interface:* Chọn giao diện.

+ *Parameter:* Địa chỉ cổng giao diện.

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

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



Hình P.13. Trang 2

* Trang 2 (Blocks): như hình P. 1 3

+ *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.

* Trang 3.(Symbols): như hình P.14

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

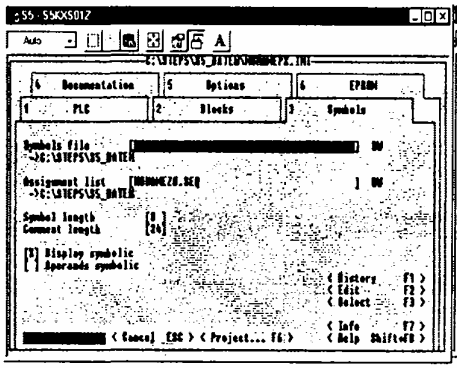
+ *Assigment lisl:* Đặt tên của file danh sách (ZO.SEQ).

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

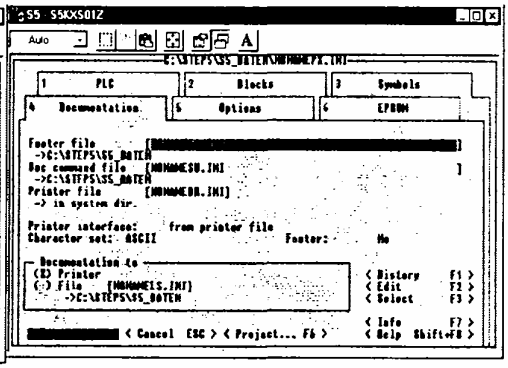
+ *Comment length:* Đạt độ dài dòng chú thích, cho phép nhiều nhất là 40 ký tự.

+ *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 P.14. Trang 3



Hình P.15. Trang 4

* Trang 4 (Documetation): như hình P.15.

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

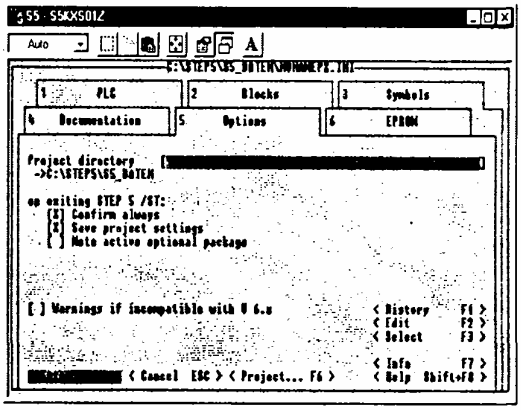
được tạo ra trong Documentation.

+ *Dọc 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.



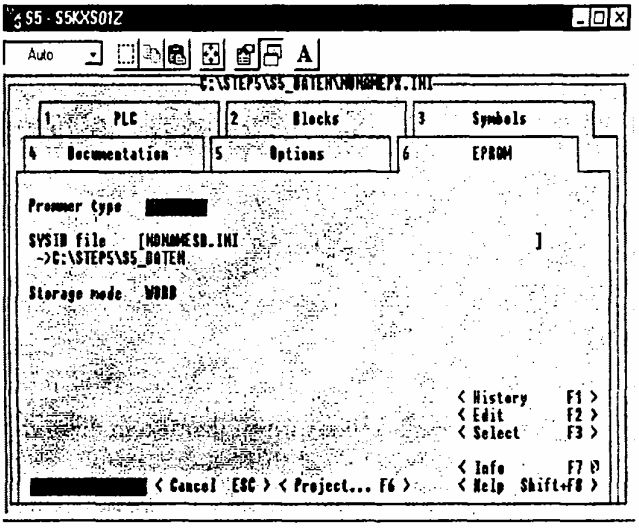
Hình P.16. Trang 5

* Trang 5 (Options): hình P.16

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

* Trang 6 (EFROM) : như hình P. 1 7

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



Hình P.17. Trang 6

3. Lập trình cho PLC - S7200

1 Sử dụng phần mềm Step7-200 for Win

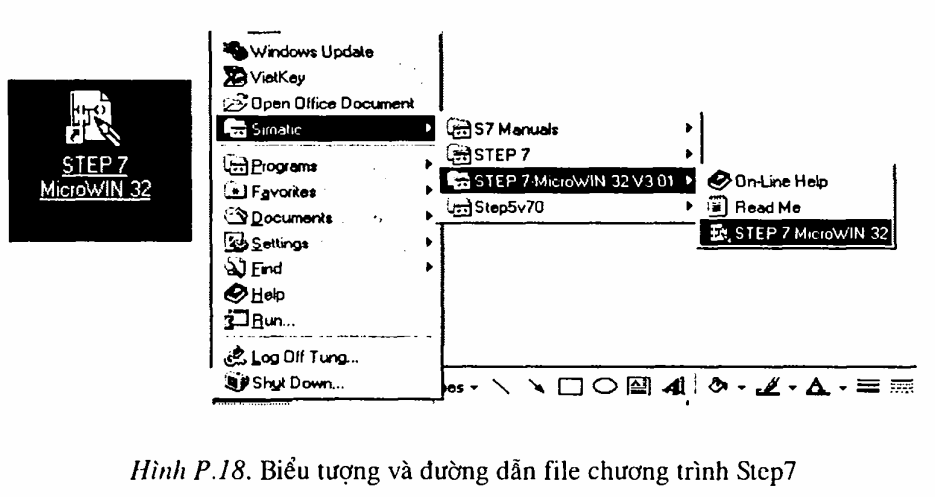
Thao tác chuẩn bị (phần hướng dẫn viết theo Version 3.2)

1. Khởi động máy tính ở chế độ Windows, 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 file chương trình như hình P.18.

Màn hình chế độ bắt đầu có dạng như hình P. 1 9.

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



Hình P.18. Biểu tượng và đường dẫn file chương trình Step7

4. Vào File để mở một file mới hoặc file đã có.

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

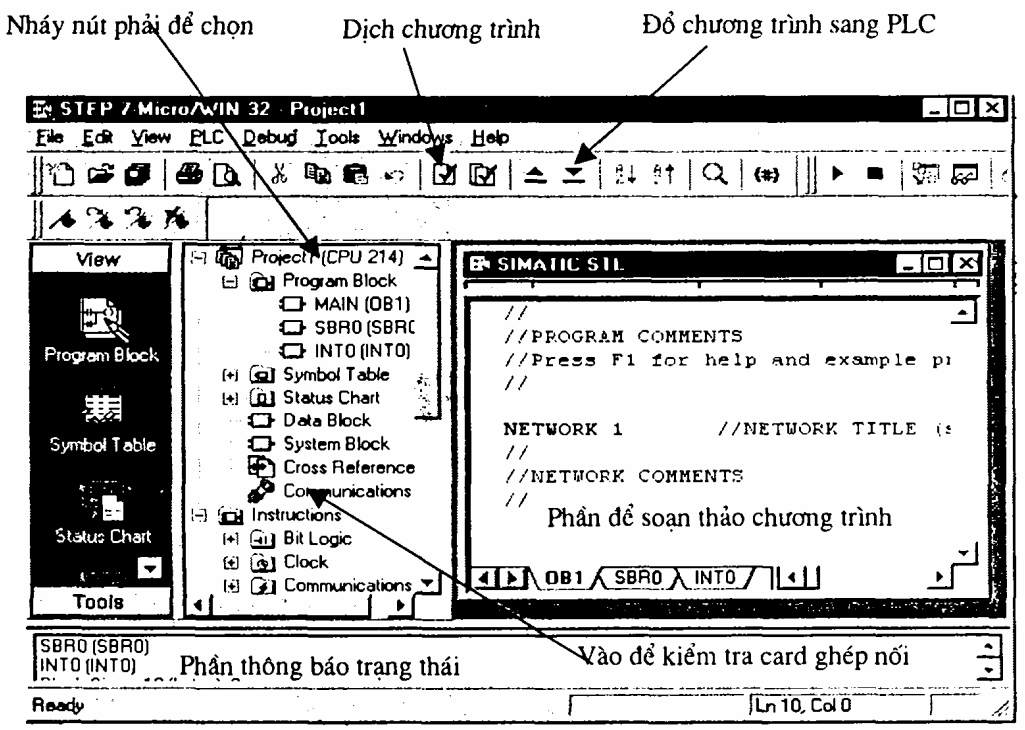
6. Tiến hành soạn thảo chương trình theo STL (nếu soạn thảo chương trình theo LAD thì có thể sử dụng các khâu, khối phía trái màn hình soạn thảo). Khi soạn thảo chỉ cần cách lệnh và đối tượng lệnh một nhịp (dấu cách), không cần chú ý chữ in và chữ thường, máy sẽ tự dịch và chỉnh chữ cho phù hợp. Trong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

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

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 cảm ghép nối cho phù hợp tử 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 P.19. Màn hình soạn thảo

2. Sử dụng phần mềm Step7-200 for Dos

Thao lác chuẩn bị:

1. Khởi động máy tính ở chế độ Windows.

2. Chạy trình S7-200 từ biểu tượng hoặc từ file chương trình, màn hình chế độ bắt đầu có dạng như hình P.20.

Trong hình P.20:

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*

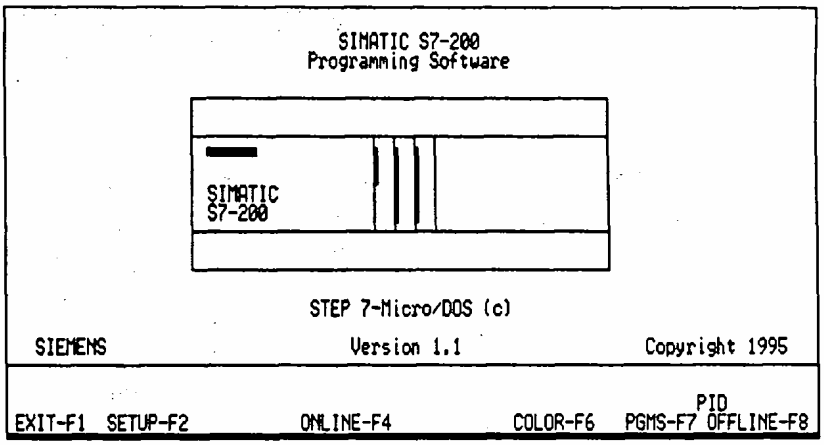
ONLENE-F4: Khi máy tính có nối với PLC.

COLOR-F6: Chọn mầu.

PGMS-F7: Chương trình quản lý file.

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

Chữ PID chỉ tên file đang sử dụng.



Hình P.20. Màn hình bắt đầu của STEP7-Micro/Dos

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 file để mở file mới hoặc file đã có. Để mở file 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 file và chọn SELECT-F8 để xác nhận, chọn ABORT-F1 để về màn hình ban đầu, ấn file 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 P.2 1 :

EXIT-F1 INSNW-F2 DELLN-F4 INSLN-F5 DELFLD-F6 UNDO-F8

Hình P.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,

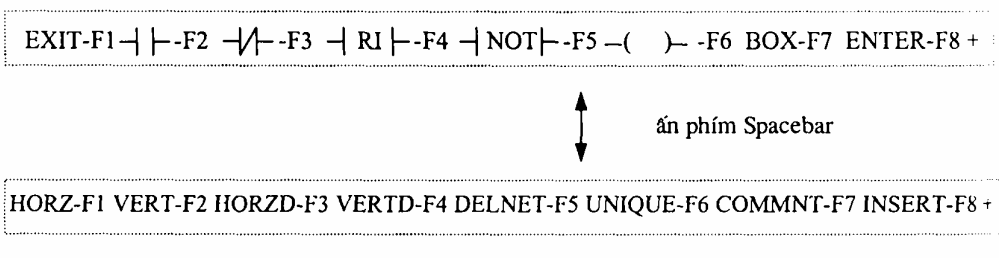
DELLN-F4: Xóa một dòng có con trỏ,

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 P.22: dấu cộng ở cuối dòng thể hiện thư mục vẫn còn cần ấn phím Spacebar để chuyển đổi.



Hình P.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 kể 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.

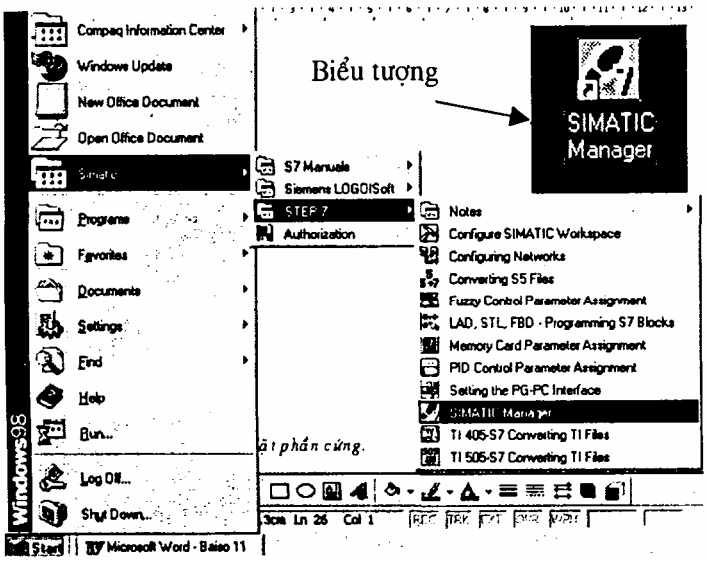
10 Chọn WRITDK-F8 để đổ chương trình sang PLC.

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.

4. Lập trình cho PLC - S7-300

Sử dụng phần mềm S7-300.

1. Khởi động



Hình P.23. Đường dẫn khởi động Step 7

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

2. Khởi động phần mềm Step7 tử biểu tượng hoặc từ file chương trình như hình P.23.

2. Cài đặt phần cứng

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

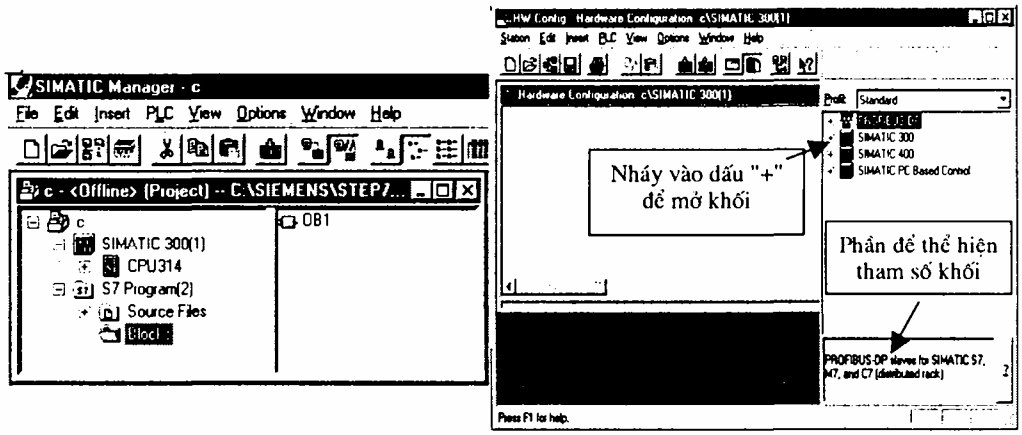
2. Vào File để tạo một thư mục chương trình mới (hoặc mở một thư mục chương trình đã có) *(vì một chương trình của S7-300 là cả thột thư mục "Project").* Một chương trình của S7-300 sẽ có dạng như hình P.24 (khi đã tạo đủ). Nếu mở một thư mục chương trình đã có sẵn chương trình thì có thể bỏ qua một số bước sau.

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

4. Mở thư mục Simatic 300(l) 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 P.25.

6. Nháy vào dấu "+" của SIMATIC 300 để chọn lần lượt các khối của cấu hình cứng. Nên chọn các khối thực trên PLC như trên hình P.26.



Hình P.24. Cấu trúc chương trình Step 7 *Hình P.25*. Hướng dẫn cài đặt phần cứng

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 nguồn đã có.

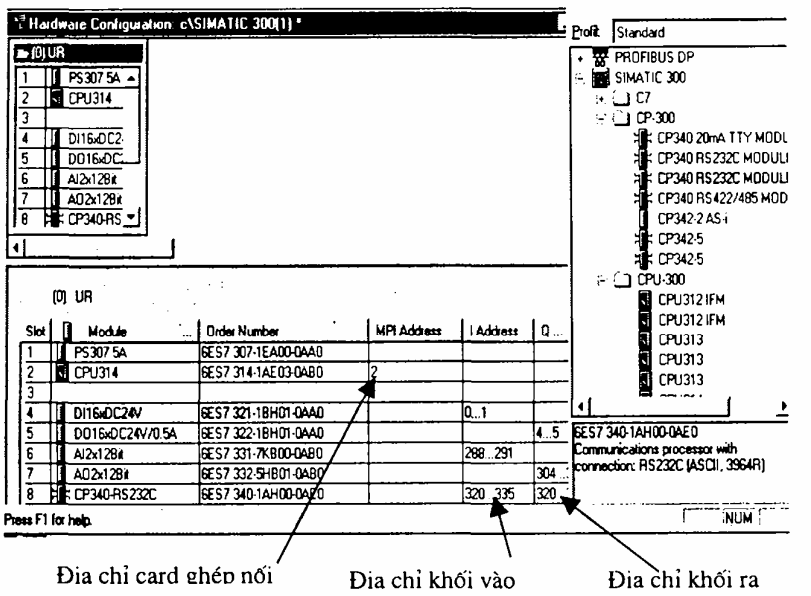
+ *Chọn khối CPU:* Chọn CPU-300 và chọn CPU 314, chọn loại có tham số (được chỉ ra ở phần thể hiện tham số hình P.26) như tham số của CPU hiện có (được chỉ ra ở dòng trên cùng và dòng dưới cùng của CPU.

+ *Chọn khối giao diện:* IM (Interfare), khi cần khối ghép nối thì chọn khối ghép nối, nếu không có có thể bỏ qua. Khi bỏ qua khối ghép nối phải để trống vị trí của khối ghép nối (vị trí 3 của Rail hình P.26).

+ *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 P.26.

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



Hình P.26. Các khối đã được chon

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 nếu 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 P.27.

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

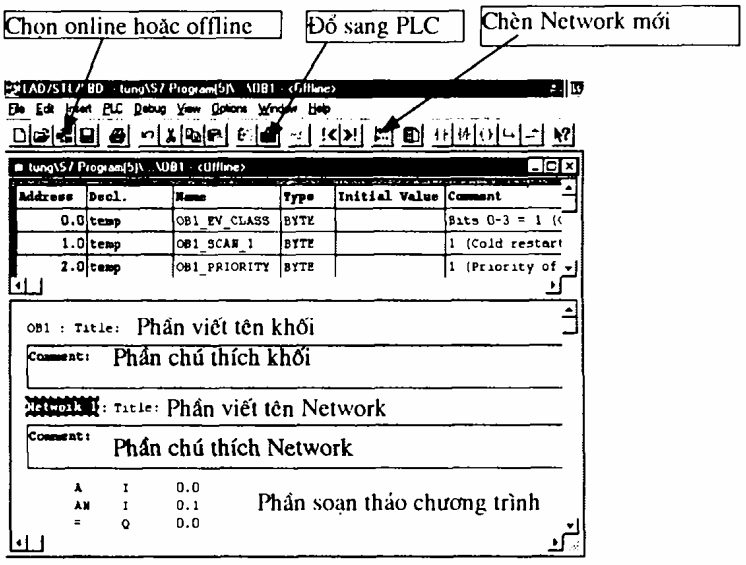
7. Có thể đặt tên cho khối, tên cho đoạn (Networks) và các chú thích nếu cần.

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.



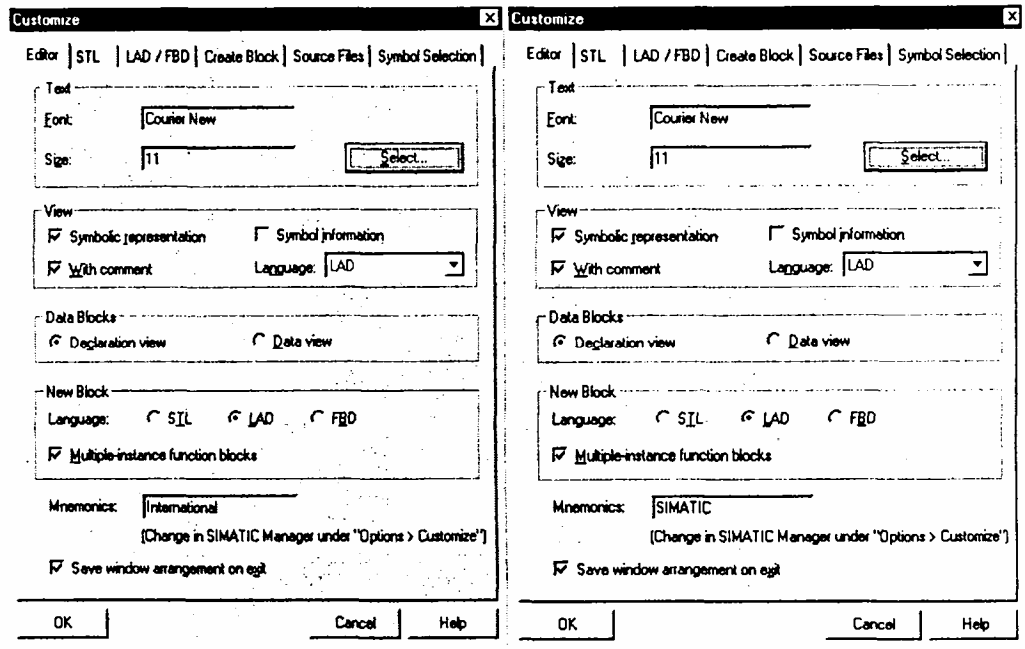
Hình P.27. Màn hình soạn thảo

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:

+ Tử màn hình soạn thảo như hình P.27 chọn **Options/customize**... được cửa sổ như hình P.28.

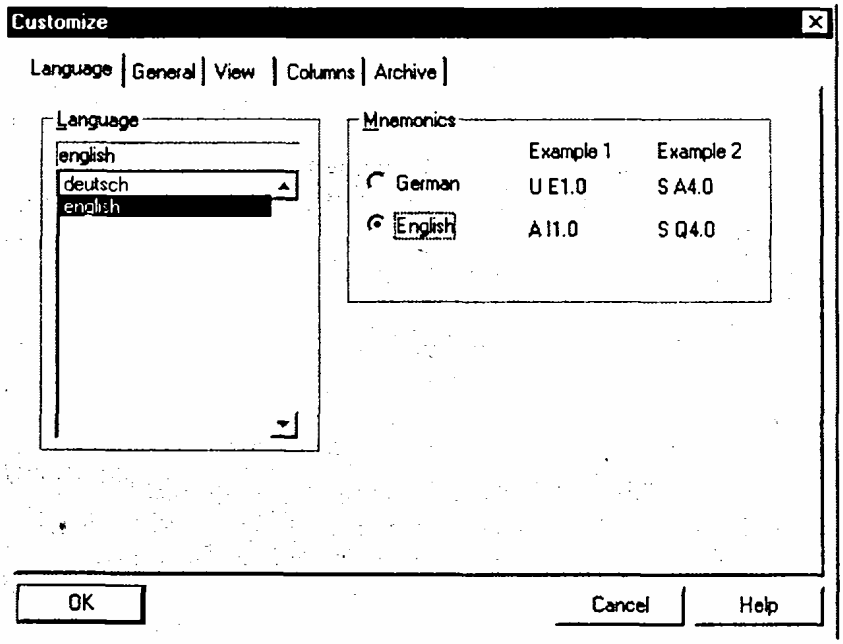
+ Trong cửa sổ Editor hình P.28, hộp kiểm Mnemonics phải là Internectiona như hình P.28a. Nếu trong hộp kiểm Mnemonics là SMATIC như hình P.28b là sai ngôn ngữ (dùng tiếng Đức). Muốn đổi ngôn ngữ để có thể lập trình được phải quay lại màn hình ban đầu như hình P.24 và tiến hành các bước:

+ Từ màn hình P.24 chọn **Options/customize**... được cửa sổ của màn hình Customize như hình P.29. Trong màn hình Customize ở cửa sổ Language tại hộp kiểm Language phải chọn English, lại hộp kiểm Mnemonics phải chọn English như hình P.29 sau đó nhấn OK.



a, b,

Hình P.28. Kiểm tra ngôn ngữ



Hình P.29

PHỤ LỤC 2
BẢNG LỆNH CỦA CÁC PHẦN MỀM PLC

1. BẢNG LỆNH CỦA PLC CPM1A

TT	Tên lệnh	Mô tả
1	AND	Nhân logic trạng thái của bít 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 bít xác định với điều kiện thực hiện.
4	CNT	Đếm lùi.
5	LD	Khởi động một dãy lệnh với trạng thái của bít xác định hoặc dễ định nghĩa một khối logic được dùng với ANDLD hoặc ORLD.
6	LD NOT	Khởi động một dãy lệnh với nghịch đảo của bít xác định.
7	OR	Cộng logic trạng thái của bít 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 bít xác định với điều kiện thực hiện.
10	OUT	Đưa ra cổng ra giá trị của bít thực hiện.
11	OUT NOT	Đưa ra cổng ra giá trị nghịch đảo của bít 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 FAL vào bộ lập trình cầm tay.
20	FALS	Phát một lỗi tiền định và cho ra FALS vào bộ lập trình cầm tay.
21	STEP	Khi dùng với bít điều khiển sẽ xác định điểm bắt đầu một bước mới và phục hồi (R) bước trước đó. Khi không dùng với bít điều khiển sẽ xác định điểm cuối của việc thực hiện bước.
22	SNXT	Dùng với một bít điều khiển để chỉ ra kết thúc bước, phục hồi bước và bắt đầu bước tiếp theo.
23	SET	Tạo ra bộ ghi dịch bít.
24	KEEP	xác định một bít 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.
26	DIFU	Đặt bít xác định cho một chu kỳ tại sườn trước của xung vào.
27	DIFD	Nhân logic trạng thái của bít 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 ở các cờ GR, EQ, LE.
31	MOV	Chép dữ liệu nguồn (từ hoặc hằng số) vào từ dịch.
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 bít và dưa dữ liệu dã được chuyển vào từ kết quả.

TT	Tên lệnh	Mô tả
34	BCD	Chuyển dữ liệu nhị phân trong từ nguồn thành BCD sau đó dưa 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 bít 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 bít và đặt bít tương ứng vào từ kết quả nếu các bít tương ứng trong các từ vào đều ON.
45	ORW	Cộng logic 2 từ vào 16 bít và đặt bít tương ứng vào từ kết quả nếu các bít tương ứng trong dữ liệu vào là ON.
46	XORW	Cộng đảo (EXNOR) 2 từ 16 bít và đặt bít vào từ kết quả khi các bít tương ứng trong các từ vào có trạng thái khác nhau.
47	XNRW	Cộng đảo (EXNOR) 2 từ 16 bít và đặt bít vào tử kết quả khi các bít 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 dữ liệu, không dùng với CQM 1 -CPU 11/21 -E.
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 Ilexa 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 tử 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 dưa dữ liệu dã chuyển đổi đến 2 từ kết quả liên tiếp.
64	XFER	Chuyển 1 số 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.

TT	Tên lệnh	Mô tả
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 tử định trước.
67	XCIIG	Trao đổi nội dung của hai từ khác nhau.
68	@COLM	Chép 16 bít của một từ xác định vào một cột bít của các tử 16 bít liên tiếp.
69	CPS	So sánh hai giá trị nhị phân 16 bít (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 bít (8 số) dã đánh dấu và đưa kết quả đến các cờ GR, EQ, LE.
71	@DBS	Chia 1 giá trị nhị phân 16 bít đã đánh dấu cho một giá trị khác và dưa kết quả nhị phân 32 bít đã đánh dấu vào từ R đến R+l.
72	@DBSL	Chia 1 giá trị nhị phân 32 bít đã đánh dấu cho một giá trị khác và đưa kết quả nhị phân 64 bít đã đánh dấu vào từ R+3 đến R.
73	@FCS	Kiểm tra lỗi trong dữ liệu truyền bởi lệnh Host linh.
74	@FPD	Tim 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).
78	@XE	Chép một bít của cụm 16 lừ liền liếp 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 bít đã đánh dấu vào R+l và R.
81	@MBSL	Nhân hai giá trị nhị phân 32 bít (8 số) đã đánh dấu và đưa kết quả nhị phân 16 bít 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+l.
85	PID	(Chỉ có CQM1-CPV43E) thể hiện điều khiển PID dựa trên các thông số xác định.
86	@PLS2	(Chỉ có CQM 1 -CPV43E) Tăng tốc độ xung ra từ 0 tới tần số đích.
87	@PWM	(Chỉ có CQM 1 -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 CQM 1-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 CQM 1 -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 thành dữ liệu giây (s).
92	@SBBL	Trừ đi một giá trị nhị phân 8 chữ số (bình thường hoặc đánh dấu) trả 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.

TT	Tên lệnh	Mô tả
95	@XFRB	Chép trạng thái của nhiều nhất là 255 bít 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 dưa kết quả đến các cờ GR, EQ, LE.
98	SLD	Dịch trái dữ liệu giữa các từ đầu và cuối một chữ số (4 bít) về lề bên trái.
99	SRD	Dịch phải dữ liệu giữa các từ đầu và cuối một chữ số (4 bít) về bên phải.
100	MLPX	Chuyển đổi 4 chữ số hexa trong từ nguồn thành giá trị thập phân tử 0 đến 15 và ghi vào các từ hoặc các bít kết quả có vị trí tương ứng với giá trị được chuyển đổi.
101	DMPX	Xác định vị trí ON cao nhất trong từ nguồn và chuyền các bít tương ứng vào tử kết quả.
102	SDEC	Chuyển giá trị hexa từ nguồn đến dữ liệu cho hiện thị 7 thanh.
103	DIST	Chuyển một tử của dữ liệu nguồn đến từ cuối mà địa chỉ của nó được cho bởi từ cuối cộng với OFF SET.
104	CON	Lỗi dữ liệu từ nguồn và viết nó vào từ cuối.
105	MOVB	Truyền bít xác định của tử hoặc bằng số nguồn đến bít xác định của từ cuối.
106	MOVD	Chuyển nội dung hexa của các chữ số nguồn 4 bít xác định đặt 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 hên 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 bít 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	IOFF	Làm tươi tất cả đầu vào và ra giữa từ đầu và tử 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 (10 Hz – 50 kHz trong các bộ 10 Hz) 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.
126	@BCNT	Đếm tổng số các bít đang chạy (ON) trong cụm từ xác định.

TT	Tên lệnh	Mô tả
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 bít ngắt cho các ngắt vào ra.
132	@ACC	Cho (CQM 1-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 lính.
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 - Tây Đức)*

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		
1	)	Dùng để đóng ngoặc biểu thức đã mở ngoặc trước đó, lệnh này không có đối tượng.
2	A n	Thực hiện lệnh AND giữa nội dung của RLO với giá trị của điểm n (đơn vị bít) chỉ dẫn trong lệnh, kết quả ghi vào RLO.
3	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.
4	AN n	Thực hiện lệnh AND giữa nội dung của RLO với giá trị nghịch đảo của điểm n (đơn vị bít) chỉ dẫn trong lệnh, kết quả ghi vào RLO.
5	O n	Thực hiện lệnh OR giữa nội dung của RLO với giá trị của điểm n (đơn vị bít) chỉ dẫn trong lệnh, kết quả ghi vào RLO.
6	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.
7	ON n	Thực hiện lệnh OR giữa nội dung của RLO với giá trị nghịch đảo của điểm n (đơn vị bít) chỉ dẫn trong lệnh, kết quả ghi vào RLO.
2.1.2. Lệnh set, reset		
8	= n	Nội dung của RLO hiện hành được gán cho đối tượng n.
9	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
10	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		
11	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.
12	LD n	Nạp nội dung đối tượng n (dạng mã BCD) vào ACCU1 không phụ thuộc RLO.
13	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.

TT	Tên lệnh	Mô tả
2.1.4 lệnh về thời gian		
14	R T	Xoá bộ thời gian nếu RLO = 1
15	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.
16	SE	Bộ thời gian lên 1 khi RLO chuyển từ 0 lên 1 (sườn lên) và duy trì dữ thời gian đặt, không phụ thuộc RLO nữa.
17	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.
18	SP	Bộ thời gian lên 1 khi RLO chuyển tử 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.
19	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		
20	CD	Số đếm giảm 1 đơn vị tại sườn lên của RLO sau đó không phụ thuộc RLO nữa.
21	CU	Số đếm tăng 1 đơn vị tại sườn lên của RLO sau đó không phụ thuộc RLO nữa.
22	R C	Xoá bộ đếm nếu RLO = 1
23	S C	Đặt bộ đếm nếu RLO = 1
2.1.6. Các lệnh toán học		
24	!=F	So sánh bằng nhau của hai thanh ghi ACCU1 và ACCU2 (dạng bít)
25	+F	Cộng nội dung hai thanh ghi ACCU1 và ACCU2, kết quả nạp vào ACCU1 (lệnh này chỉ có ở STL).
26	<=F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có nhỏ hơn hay bằng ở ACCU1 không ?
27	<F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có nhỏ hơn ở ACCU1 không?
28	><F	So sánh đối tượng lệnh trong hai thanh ghi ACCU1 và ACCU2 xem có khác nhau không ?
29	>=F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có lớn hơn hay bằng ở ACCU1 không ?
30	>F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có lớn hơn ở ACCU1 không?
31	-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.		
32	C n	Gọi khối dữ liệu DB, không phụ thuộc vào RLO, quét chương trình không bị gián đoạn, RLO không bị ảnh hưởng.
33	G	Tạo lập hoặc xoá khối dữ liệu độc lập với RLO.
34	JC n	Nhảy sang làm việc ở khối n nếu RLO = 1.
35	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.		
36	BE	Lệnh kết thúc khối.
37	BEC	Lệnh kết thúc có điều kiện giữa khối (RLO = 1)
38	BEU	Lệnh kết thúc không điều kiện giữa khối, không phụ thuộc RLO.

TT	Tên lệnh	Mô tả
2.1.9. Các 1ệnh thống.		
39	NOP 0	Mã lệnh 16 bít trong RAM đều bằng 0 (để giữ chỗ).
40	NOP 1	Mã lệnh 16 bít trong RAM đều bằng 1 (để giữ chỗ).
2.1.10. Lệnh dừng		
41	STP	Lệnh dừng cuối chương trình, bộ PLC đi vào trạng thái nghỉ.
2.2. Các 1ệnh thay thế (chỉ dùng với khối FB)		
2.2.1. Các lệnh đại số logic Bool thay thế.		
42	A=	Lệnh AND thay thế.
43	AN=	Lệnh AND đảo thay thế.
44	AW	Tổ hợp từng bít theo luật logic AND.
45	DO=	Lệnh DO thay thế.
46	O=	Lệnh OR thay thế.
47	ON=	Lệnh OR đảo thay thế.
48	OW	Tổ hợp từng bít theo luật logic OR.
49	XOR	Tổ hợp từng bít theo luật logic OR đặc biệt.
2.2.2. Các lệnh về bít.		
50	RU	Lệnh xoá bít không điều kiện.
51	SU	Đặt một bít vô điều kiện.
52	TB	Trắc nghiệm bít cho trạng thái tín hiệu 1
53	TBN	Trắc nghiệm bít cho trạng thái tín hiệu 0.
2.2.3. Lệnh sét, reset thay thế.		
54	= =	Lệnh gán thay thế.
55	RB=	Lệnh xoá đối tượng lệnh hình thức.
56	RD=	Lệnh xoá đối tượng lệnh hình thức dạng số.
57	S=	Lệnh đặt đối tượng lệnh hình thức.
2.2.4. Các lệnh về thời gian và đếm		
58	FR=	Lệnh khả thi thay thế.
59	SD=	Lệnh khởi động bộ thời gian bắt đầu trễ hình thức.
60	SEC=	Khởi động bộ thời gian mở rộng hoặc bộ đếm.
61	SFD=	Lệnh khởi động bộ thời gian tắt trễ hoặc bộ đếm xuống.
62	SP=	Lệnh khởi động bộ thời gian xung hình thức.
63	SSU=	Lệnh khởi động bộ thời gian bắt đầu trễ.
2.2.5. Các lệnh nạp là truyền.		
64	L=	Lệnh nạp thay thế.
65	LD=	Lệnh nạp đối tượng hình thức dạng cơ số BCD.
66	LW=	Lệnh nạp mẫu bít của đối tượng lệnh hình thức.
67	T=	Lệnh truyền đối tượng lệnh hình thức.
2.2.6. Các lệnh chuyển đổi.		
68	CTW	Nội dung ACCU1 được chuyển đổi từng bít một.
69	CSW	Bổ sung cho 2.
2.2.7. Các 1ệnh dịch chuyển.		
70	SLW	Dãy bít trong ACCU1 dịch sang trái.
71	SRW	Dãy bít trong ACCU1 dịch sang phải.

TT	Tên lệnh	Mô tả
2.2.8. Các lệnh nhảy.		
72	JC=	Nhảy có điều kiện (RLO = l)
73	JM=	Nhảy nếu kết quả là âm (CC1 = 0, CC0 = 1).
74	JN:	Nhảy nếu kết quả là (0,0) (CC1 = l, CC0 = 0).
75	JO=	Nhảy khi cờ tràn.
76	JP=	Nhảy nếu kết quả là dương (CC1 = l, CC0 = 0).
77	JU=	Nhảy không điều kiện.
78	JZ=	Nhảy nếu kết quả là 0 (CC1 = 0, CC0 = 0)
2.2.9. Các lệnh khác.		
79	D	Giảm nội dung trong ACCU1.
80	DO	Xử lý từ cờ hoặc từ dữ liệu.
81	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.
82	I	Tăng nội dung trong ACCU1.
83	IA	Lệnh cấm ngắt.
84	LRS	Nạp miền dữ liệu hệ thống (nạp miền RS... vào ACCU1).
85	RA	Cho phép ngắt.
2.2.10. Nhóm lệnh hệ thống.		
86	ADD	Lệnh cộng một hằng số.
87	JC n	Nhảy sang làm việc ở khối n nêu RLO = 1.
88	JU n	Nhảy sang làm việc ở khối n, không phụ thuộc RLO và RLO không bị ảnh hưởng.
89	LIR	Lệnh nạp gián tiếp thanh ghi.
90	RU	Lệnh xoá bít không điều kiện.
91	STS	lệnh dừng tức khắc.
92	SU	Đặt một bít vô điều kiện.
93	TAK	Lệnh trao đổi nội dung thanh ghi.
94	TIR	Lệnh truyền gián tiếp thanh ghi.
95	TNB	Lệnh truyền một trường dữ liệu.

3. BẢNG LỆNH CỦA PLC - S7-200 *(Siemens - Tây Đức)*

TT	Tên lệnh	Mô tả
3.1. Các lệnh thực hiện vô điều kiện		
1	= N	Giá trị bít đầu tiên trong ngăn xếp được sao chép sang điểm n chỉ dẫn trong lệnh.
2	=I N	Giá trị bít 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<= n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu giá trị byte n1 không lớn hơn giá trị của byte n2. Kết quả được ghi lại vào bít đầu tiên của ngăn xếp.
5	AB= n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu giá trị của hai byte n1 và n2 thoả mãn n1 = n2. Kết quả được ghi lại vào bít đầu tiên của ngăn xếp.

TT	Tên lệnh	Mô tả
6	AB>= n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu giá trị của hai byte n1 và n2 thoả mãn n1 ≥ n2. Kết quả được ghi lại vào bít đầu tiên của ngăn xếp.
7	AD<= n1,n2	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 bít đầu tiên của ngăn xếp.
8	AD>= n1,n2	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 bít đầu tiên của ngăn xếp.
9	A D = n1,n2	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 bít 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 bít đầu của ngăn xếp.
11	ALD	Thực hiện lệnh AND giữa giá trị của bít đầu tiên và của bít thứ hai trong ngăn xếp. Kết quả được ghi lại vào bít đầu tiên của ngăn xếp, các giá trị còn lại trong ngăn xếp được kéo lên một bít.
12	AN N	Thực hiện lệnh AND giữa giá trị của bít dấu tiên trong ngăn xếp với giá trị nghịch đảo của điểm n trong chỉ dẫn. Kết quả được ghi lại vào bít 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 bít đầu tiên của ngăn xếp.
14	AR<= n1,n2(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 n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi lại vào bít đầu tiên của ngăn xếp.
15	AR= n1,n2(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 n1 và n2 thoả mãn n1 = n2. Kết quả được ghi lại vào bít đầu tiên của ngăn xếp.
16	AR>= n1,n2(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 n1 và n2 thoả mãn n1 ≥ n2. Kết quả được ghi lại vào bít đầu tiên của ngăn xếp.
17	AW<= n 1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi lại vào bít đầu tiên của ngăn xếp.
18	AW= n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai tử n1 và n2 thoả mãn n1 = n2. Kết quả được ghi lại vào bít đầu tiên của ngăn xếp.
19	AW>= n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ n1 và n2 thoả mãn n1 ≥ n2. Kết quả được ghi lại vào bít 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	CTDU 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) nêu đầu vào R của bộ đếm được kích.
22	ED	Đặt giá trị logic 1 vào bít đầu tiên của ngăn xếp khi xuất hiện sườn xuống của tín hiệu.

TT	Tên lệnh	Mô tả
23	EU	Đặt giá trị logic 1 vào bít đầu tiên của ngăn xếp khi xuất hiện sườn lên của tín hiệu.
24	LD n	Nạp giá trị logic của điểm n chỉ dẫn trong lệnh vào bít đầu tiên của ngăn xếp.
25	LDB<= n1,n2	Bít đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1 ≤ n2.
26	LDB= n1,n2	Bít đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội đung hai byte n1 và n2 thoả mãn n1 = n2.
27	LDB>= n1,n2	Bít đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1 ≥ n2.
28	LDD= n1,n2	Bít đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ kép n1 và n2 thoả mãn n1 = n2.
29	LDD>= n1,n2	Bít đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ kép n1 và n2 thoả mãn n1 ≥ n2.
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 n1 và n2 thoả mãn n1 ≤ n2.
34	LDR= n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai số thực n1 và n2 thoả mãn n1 = n2.
35	LDR>= n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai số thực n1 và n2 thoả mãn n1 ≥ n2.
36	LDW<=n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ n1 và n2 thoả mãn n1 ≤ n2.
37	LDW= n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ n1 và n2 thoả mãn n1 = n2.
38	LDW>=n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ n1 và n2 thoả mãn n1 ≥ n2.
39	LPP	Kéo nội dung của ngăn xếp lên một bít. Giá trị mới của bít trên là giá trị cũ của bít dưới, độ sâu của ngăn xếp giảm đi một bít.
40	LPS	Sao chép giá trị bít đầu tiên trong ngăn xếp vào bít thứ hai. Nội dung còn lại của ngăn xếp bị đẩy xuống một bít.
41	LRD	Sao chép giá trị của bít thứ hai vào bít đầu tiên trong ngăn xếp. Giá trị còn lại của ngăn xếp giữ nguyên.
42	MEND (1)(2)	Kết thúc phần chương trình trong một vòng qua.
43	NOT	Đảo giá trị của bít đầu tiên trong ngăn xếp.
44	O n	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với điểm n chỉ dẫn trong lệnh. Kết quả được ghi vào bít đầu tiên trong ngăn xếp.
45	OB<= n1, n2	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi vào bít đầu tiên trong ngăn xếp.
46	OB= n1, n2	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1 = n2. Kết quả được ghi vào bít đầu tiên trong ngăn xếp.

TT	Tên lệnh	Mô tả
47	OB>= n1, n2	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1 ≥ n2. Kết quả được ghi vào bít đầu tiên trong ngăn xếp.
48	OD<= n1, n2	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai từ kép n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi vào bít đầu tiên trong ngăn xếp.
49	OD= n1, n2	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai từ kép n1 và n2 thoả mãn n1 = n2. Kết quả được ghi vào bít đầu tiên trong ngăn xếp.
50	OD>= n1, n2	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai từ kép n1 và n2 thoả mãn n1 ≥ n2. Kết quả được ghi vào bít đầu tiên trong ngăn xếp.
51	OI n	Thực hiện tức thời toán tử OR giữa bít đầu tiên của ngăn xếp với điểm n chỉ dẫn trong lệnh. Kết quả được ghi vào bít đầu tiên trong ngăn xếp.
52	OLD	Thực hiện toán tử OR giữa bít đầu và bít thứ hai trong ngăn xếp. Kết quả được ghi vào bít đầu tiên trong ngăn xếp, các giá trị còn lại của ngăn xếp được chuyển lên một bít.
53	ON n	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic nghịch đảo của điểm n chỉ dẫn trong lệnh. Kết quả được ghi vào bít đầu tiên trong ngăn xếp
54	ONI n	Thực hiện tức thời toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic nghịch đảo của điểm n chỉ dẫn trong lệnh. Kết quả được ghi vào bít đầu tiên trong ngăn xếp.
55	OR<= n1,n2(5)	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu hai số thực n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi lại vào bít đầu trong ngăn xếp
56	OR= n1,n2(5)	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu hai số thực n1 và n2 thoả mãn n1 = n2. Kết quả ghi vào bít đầu trong ngăn xếp.
57	OR>= n1,n2(5)	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu hai số thực n1 và n2 thoả mãn n1 > n2. Kết quả ghi lại vào bít đầu trong ngăn xếp.
58	OW<= n1,n2(5)	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu hai từ n1 và n2 thoả mãn n1 < n2. Kết quả được ghi lại vào bít đầu trong ngăn xếp.
59	OW= n1,n2(5)	Thực hiện toán tử OR giữa bít đầu liên của ngăn xếp với giá trị logic 1 nếu hai từ n1 và n2 thoả mãn n1 = n2. Kết quả được ghi lại vào bít đầu trong ngăn xếp.
60	OW>= n1,n2(5)	Thực hiện toán tử OR giữa bít đầu tiên của ngăn xếp với giá trị logic 1 nếu hai từ n1 và n2 thoả mãn n1 ≥ n2. Kết quả được ghi lại vào bít 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	RET (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 (32bít) IN1 và IN2. Kết quả được ghi lại vào IN2.
64	/R IN1, IN2(5)	Thực hiện phép chia hai số thực (32bít) 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 (32bít) lại 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	AICH	INT, EVENT	Khai báo chương trình xử lý ngắt INT theo kiểu EVENT
71	ATH	INT, OUT, LEN	Biến đổi một sâu ký tự tử 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 dầu IN (2 byte) vào mảng có vị trí đầu OUT.
76	CALL	n[1][6]	Gọi chương trình con được đánh nhãn n.
77	CRET	[l][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 bít của từ OUT (2 byte) có chỉ số là IN.
82	DECW	IN	Giảm giá trị của từ IN đi một đơn vị.
83	DSIS	[1]	vô hiệu hoá tất cả các ngắt (interrupt).
84	DIV	IN1, IN2	Chia số nguyên 16 bít, được xác định là từ thấp của IN2 (kiểu từ kép), cho IN1 kiểu lừ. 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 bít IN có dấu sang thành một số thực 32 bít OUT
87	ENCO	IN,OUT	Chuyển đổi chỉ số của bít thấp nhất có giá trị logic 1 trong từ IN sang thành một số nguyên và ghi vào bít cuối của byte OUT.
88	ENI	[l]	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.
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).

TT	Tên lệnh		Mô tả
91	FND<	SRC, PATRRI NDX(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	END<>	SRC, PATRRI, NDX(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, PATRRI, NDX(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, PATRRI, NDX(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 là (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 bít) 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.
111	MOVW	IN, OUT	Sao giá trị của từ IN sang từ OUT.
112	MUL	IN1, IN2	Nhân hai số nguyên 16 bít IN1 với hai byte thấp của số nguyên 32 bít IN2 sau đó ghi lại kết quả vào IN2.
113	NETR	TABLE, PORT(5)	Khởi tạo truyền thông để đọc dữ liệu tử ngoại vi qua cổng loét vào bảng TABLE.
114	NETW	TABLE, PORT(5)	Khởi tạo truyền thông để ghi dữ liệu của bảng TABLE ra ngoai vi qua cổng PORT.
115	NEXT	(l)(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.

TT	Tên lệnh		Mô tả
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_BíT,n	Xoá một mảng gồm n bít kể từ địa chỉ S_BíT (kiểu bít).
121	-R	IN1, NT2(5)	Thực hiện phép trừ hai số thực (32bít) IN1 và IN2. Kết quả được ghi lại vào IN2.
122	Ri	S_BíT,n	Xoá tức thời một mảng gồm n bít kể từ địa chỉ S_BíT.
123	RLD	IN, n	Quay tròn từ kép IN sang trái n bít.
124	RLW	IN, n	Quay tròn tử IN sang trái n bít.
125	RRD	IN, n	Quay tròn từ kép IN sang phải n bít.
126	RRW	IN, n	Quay tròn từ IN sang phải n bít.
127	S	S_BíT,n	Đặt giá trị logic 1 vào một mảng n bít kể từ địa chỉ S_BíT.
128	SBR	N(1)(2)(4)	Khai báo nhãn n cho chương trình con.
129	SEG	IN, OUT	Chuyển đổi giá trị của 4 bít thấp trong byte IN sang thành mã tương ứng cho thanh ghi 7 nét và ghi vào OUT
130	SHRB	DATA, S_BíT,n	Dịch thanh ghi gồm |n|0 bít có bít thấp nhất là S_BíT sang trái nếu n>0. hoặc sang phải nếu n<0. Giá trị của bít DATA được đưa vào bít trống của thanh ghi sau khi dịch (bít S_BíT nếu n>0, hoặc bít S_BíT nếu n<0)
131	SI	S_BíT,n	Đặt tức thời giá trị logic 1 vào mảng n bít kể từ bít S_BíT.
132	SLD	IN,n	Dịch từ kép IN sang trái một bít.
133	SLW	IN,n	Dịch từ IN sang trái một bít.
134	SQRT	IN, OUT	Lấy căn bậc hai của số thực 32 bít IN và ghi kết quả vào OUT (32bít).
135	SRD	IN,n	Dịch từ kép IN sang phải một bít.
136	SRW	IN,n	Dịch từ IN sang phải một bít.
137	STOP		Dừng "mềm" chương trình.
138	SWAP	IN	Đổi chỗ hai bít 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 bộ thời gian Txx được 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 bít IN thành một số nguyên 32 bít 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 bít 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 bít của hai từ IN1 và IN2. Kết quả được ghi lại vào IN2.
(l)	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 bít còn lại của ngăn xếp.
(7)	Đặt TOS lên 1.

4. BẢNG LỆNH CỦA PLC S7-300 *(SIEMENS - Tây đức)*

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.
6	+D	Cộng 2 số nguyên 32 bít ở ACCU1 và ACCU2, kết quả để ở ACCU1.
7	-D	Trừ số nguyên 32 bít ở ACCU2 cho số nguyên 32 bít ở ACCU1, kết quả để ở ACCU1.
8	*D	Nhân 2 số nguyên 32 bít ở ACCU1 và ACCU2, kết quả để ở ACCU1.
9	/D	Chia số nguyên 32 bít ở ACCU2 cho số nguyên 32 bít ở ACCU1, kết quả để ở ACCU1.
10	= =D	So sánh hai số nguyên 32 bít ở ACCU1 và ACCU2 có bằng nhau không.
11	<>D	So sánh hai số nguyên 32 bít ở ACCU1 và ACCU2 xem có khác nhau không.
12	>D	So sánh số nguyên 32 bít ở ACCU2 có lớn hơn số nguyên 32 bít ở ACCU1 không.
13	<D	So sánh số nguyên 32 bít ở ACCU2 có nhỏ hơn số nguyên 32 bít ở ACCU1 không.
14	>=D	So sánh số nguyên 32 bít ở ACCU2 có lớn hơn hay bằng số nguyên 32 bít ở ACCU1 không.
15	<=D	So sánh số nguyên 32 bít ở ACCU2 có nhỏ hơn hay bằng số nguyên 32 bít ở ACCU1 không.
16	+I	Cộng 2 số nguyên 16 bít ở ACCU1 và ACCU2, kết quả để ở ACCU1.
17	-I	Trừ số nguyên 16 bít ở ACCU2 cho số nguyên 16 bít ở ACCU1, kết quả để ở ACCU1.
18	*I	Nhân 2 số nguyên 1 6 bít ở ACCU1 và ACCU2, kết quả để ở ACCU1.
19	/I	Chia số nguyên 16 bít ở ACCU2 cho số nguyên 16 bít ở ACCU1, kết quả để ở ACCU1.
20	= =I	So sánh hai số nguyên 16 bít ở ACCU1 và ACCU2 có bằng nhau không.
21	<>I	So sánh hai số nguyên 16 bít ở ACCU1 và ACCU2 xem có khác nhau không.
22	>I	So sánh số nguyên 16 bít ở ACCU2 có lớn hơn số nguyên 16 bít ở ACCU1 không.

TT	Tên lệnh	Mô tả
23	<I	So sánh số nguyên 16 bít ở ACCU2 có nhỏ hơn số nguyên 16 bít ở ACCU1 không.
24	>=I	So sánh số nguyên 16 bít ở ACCU2 có lớn hơn hay bằng số nguyên 16 bít ở ACCU1 không.
25	<=I	So sánh số nguyên 16 bít ở ACCU2 có nhỏ hơn hay bằng số nguyên 16 bít ở ACCU1 không.
26	+R	Cộng 2 số thực 32 bít ở ACCU1 và ACCU2, kết quả để ở ACCU1.
27	-R	Trừ số thực 32 bít ở ACCU2 cho số thực 32 bít ở ACCU1, kết quả để ở ACCU1.
28	*R	Nhân 2 số thực 32 bít ở ACCU1 và ACCU2, kết quả để ở ACCU1.
29	/R	Chia số thực 32 bít ở ACCU2 cho số thực 32 bít ở ACCU1, kết quả để ở ACCU1.
30	= =R	So sánh hai số thực 32 bít ở ACCU1 và ACCU2 có bằng nhau không.
31	<>R	So sánh hai số thực 32 bít ở ACCU1 và ACCU2 xem có khác nhau không.
32	>R	So sánh số thực 32 bít ở ACCU2 có lớn hơn số thực 32 bít ở ACCU1 không.
33	<R	So sánh số thực 32 bít ở ACCU2 có nhỏ hơn số thực 32 bít ở ACCU1 không.
34	>=R	So sánh số thực 32 bít ở ACCU2 có lớn hơn hay bằng số thực 32 bít ở ACCU1 không.
35	<=R	So sánh số thực 32 bít ở ACCU2 có nhỏ hơn hay bằng số thực 32 bít ở ACCU1 không.
36	A n	Thực hiện lệnh AND giữa nội dung của RLO với giá trị của điểm n (đơn vị bít) 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 bít.
39	AD	Thực hiện lệnh AND giữa nội dung trong ACCU1 và ACCU2, kết quả để ở ACCU1 (32 bít).
40	AN n	Thực hiện lệnh AND giữa nội dung của RLO với giá trị nghịch đảo của điểm n (đơn vị bít) 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 bít).
43	BEC	Lệnh kết thúc có điều kiện giữa khối (RLO:l)
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 bít.
47	BTI	Chuyển số dạng mã BCD sang số nguyên 16 bít.
48	CAD	Đổi thứ tự byte trong ACCU1 (32 bít).
49	CAR	Chuyển nội dung thanh ghi 1 với nội dung thanh ghi 2.
50	CAW	Đổi thứ tự byte trong ACCU1 (16 bít)
51	CALL	Lệnh gọi khối.
52	CC	Lệnh gọi khối có điều kiện.

TT	Tên lệnh		Mô tả
53	CD		Số đếm giảm 1 đơn vị tại sườn lên của RLO sau đó không phụ thuộc RLO nữa.
54	CDB		Chuyên 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 bít thành số dạng mã BCD.
59	DTR		Đổi số nguyên 32 bít thành số thực.
60	IN		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 bít.
66	INVI		Lấy phần bù một của số nguyên 16 bít.
67	ITB		Đổi số nguyên 16 bít thành số dạng mã BCD.
68	ITD		Đổi số nguyên 16 bít thành số nguyên 32 bít.
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 = l)
75	JMZ		Nhảy nếu kết quả là âm hoặc bằng không (CC1 = 0 hoặc 0, CC0 = 0 hoặc l).
76	JN		Nhảy nếu kết quả là khác không (CC1 = 1 hoặc 0, CC0 = 0 hoặc l).
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 ACCU 1.
89	L DBLG		Nạp độ dài của khối dữ liệu DB vào ACCU1.

TT	Tên lệnh		Mô tả
90	L	DBNO	Nạp số của khối dữ liệu DB vào ACCU1.
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	LAR 1		Nạp địa chỉ vào thanh ghi 1 từ ACCU1.
95	LAR 1	n	Nạp địa chỉ vào thanh ghi 1 từ vị trí n ghi trong lệnh.
96	LAR 1	AR2	Nạp địa chỉ vào thanh ghi 1 từ thanh ghi 2.
97	LAR 1	P#	Nạp vào thanh ghi 1 tử 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 đệm 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 bít ở ACCU2 cho số nguyên 32 bít ở ACCU1, kết quả để ở ACCU1.
109	NEGD		Lấy số bù hai của số nguyên 32 bít.
110	NEGI		Lấy số bù hai của số nguyên 16 bít.
111	NEGR		Lấy dấu âm cho số thực 32 bít.
112	NOP	0	Mã lệnh 16 bít trong RAM đều bằng 0 (để giữ chỗ).
113	NOP	1	Mã lệnh 16 bít 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 bít) 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 bít).
118	ON	n	Thực hiện lệnh OR giữa nội dung của RLO với giá trị nghịch đảo của điểm n (đơn vị bít) 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 ngoắc), 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 bít).
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

TT	Tên lệnh		Mô tả
127	RLD	n	Quay tròn tử kép ở ACCU1 sang trái n bít.
128	RLDA		Quay tròn từ kép ở ACCU1 sang trái 1 bít qua CC 1.
129	RND		Đổi số thực 32 bít thành số nguyên 32 bít (bỏ phần thập phân).
130	RND+		Đổi số thực 32 bít thành số nguyên 32 bít, nếu là số dương thì làm tròn tăng, là số âm thì bỏ phần thập phân.
131	RND-		Đổi số thực 32 bít thành số nguyên 32 bít, 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 bít.
133	RRDA		Quay tròn từ kép ở ACCU1 sang phải 1 bít 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 đêm 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 dặt, 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 tử kép trong ACCU1 sang trái n bít hoặc số bít dịch được nạp vào ACCU2.
142	SLW	n	Dịch từ đơn trong ACCU1 sang trái n bít hoặc số bít dịch được nạp vào ACCU2.
143	SP		Bộ thời gian lên 1 khi RLO chuyển tử 0 lên 1 (sườn lem 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 tử kép trong ACCU1 sang phải n bít hoặc số bít dịch được nạp vào ACCU2.
145	SRW	n	Dịch tử đơn trong ACCU1 sang phải n bít hoặc số bít 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 bít trong ACCU1 sang phải n bít hoặc số bít dịch được nạp vào ACCU2. các bít trống được chèn bít dấu của số nguyên.
148	SSI	n	Dịch số nguyên 16 bít trong ACCU1 sang phải n bít hoặc số bít dịch được nạp vào ACCU2, các bít trống được chèn bít dấu của số nguyên.
149	T	n	Nội dung của ACCU1 truyền cho đối lượng 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 bít trong ACCU1 thành số nguyên 32 bít có dấu.

TT	Tên lệnh	Mô tả
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 bít) 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 bít của hai từ kép.
164	XOW	Thực hiện lệnh OR (đặc biệt) giữa các bít của hai từ đơn

TÀI LIỆU THAM KHẢO

1. Nguyễn Trọng Thuần, *Điều khiển logic và ứng dụng*, Nhà xuất bản Khoa học và kỹ thuật, 2000.
2. Nguyễn Doãn Phước, Phan Xuân Minh, Vũ Văn Hà. *Tự động hoá tới Simatic S7-300*, Nhà xuất bản Khoa học và kỹ thuật, 2000.
3. Tăng Văn Mùi. Nguyễn Tiến Dũng, *Điều khiển logic lập trình PLC*, Nhà xuất bản thống kê, 2003.
4. Nguyễn Doãn Phước, Phan Xuân Minh, *Tự động hoá với Simatic S7-200*, Nhà xuất bản 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.

Giáo trình PLC

MỤC LỤC

Nội dung Trang

Phần 1: LOGIC HAI TRẠNG THÁI VÀ ỨNG DỤNG

Chương 1: LÍ THUYẾT CƠ SỞ

§1.1. Những khái niệm cơ bản

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 của nó 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.

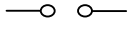
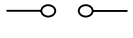
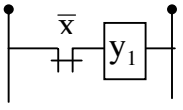
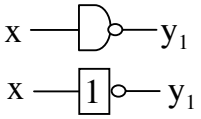
2. Các hàm logic cơ bản

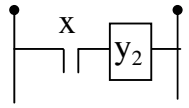
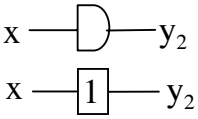
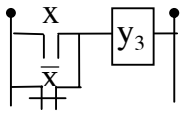
Một hàm $y = f(x_1, x_2, ..., x_n)$ với các biến x_1, x_2, ... x_n chỉ nhận hai giá trị: 0 hoặc 1 và hàm y cũng chỉ nhận hai giá trị: 0 hoặc 1 thì gọi là hàm logic.

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	$y_1 = \bar{x}$			

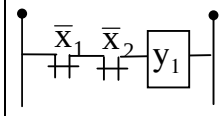
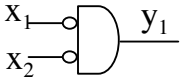
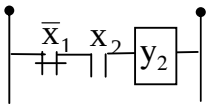
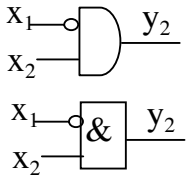
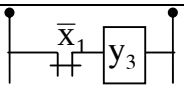
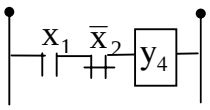
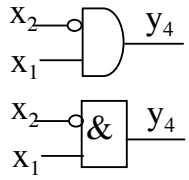
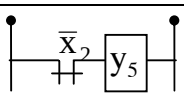
Hàm lặp (YES)	y_2	0	1	$y_2 = x$			
Hàm đơn vị	y_3	1	1	$y_3 = 3$ $y_3 = x + \overline{x}$			

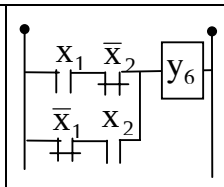
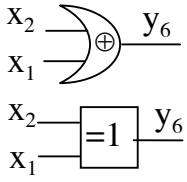
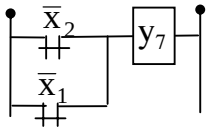
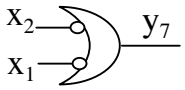
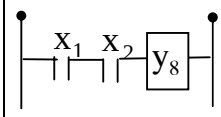
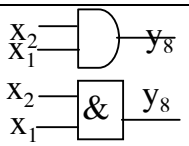
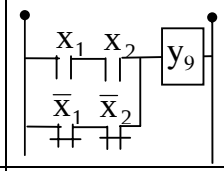
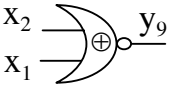
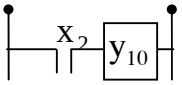
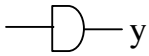
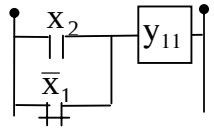
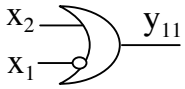
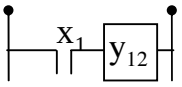
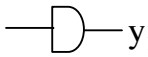
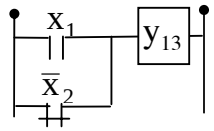
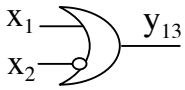
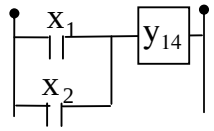
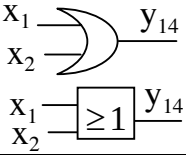
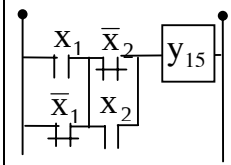
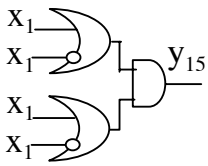
Trong các hàm trên hai hàm $y_0$và y_3 luôn có giá trị không đổi nên ít được quan tâm, thường chỉ xét hai hàm y_1 và y_2.

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 bảng1.2

Bảng 1.2

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

Hàm hoặc loại trừ XOR	y_6	0	1	1	0	$y_6 = x_1\bar{x}_2 + \bar{x}_1x_2$			Cộng mod ule
Hàm Chef-fer	y_7	0	1	1	1	$y_7 = \overline{\bar{x}_1 + \bar{x}_2} = \overline{x_1x_2}$			
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 x_2
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 x_1
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 $y_0 = \bar{y}_{15}$, $y_1 = \bar{y}_{14}$...

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.

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)

4. Tính chất và một số hệ thức cơ bản

4.1. 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:

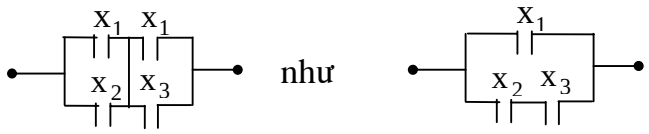
$$(x_1 + x_2).x_3 = x_1.x_3 + x_2.x_3$$
$$x_1 + x_2.x_3 = (x_1 + x_2).(x_1 + x_3)$$

Ta có thể minh hoạ để kiểm chứng tính đũng 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:

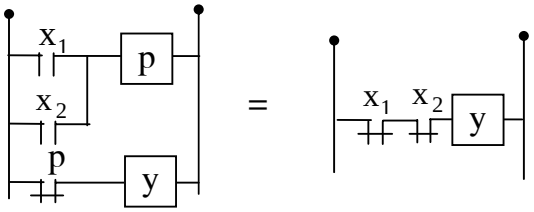
$$\overline{x_1.x_2} = \overline{x}_1 + \overline{x}_2; \qquad \overline{x_1 + x_2} = \overline{x}_1.\overline{x}_2$$

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	$\overline{x}_1$	$\overline{x}_2$	$\overline{x_1 + x_2}$	$\overline{x}_1.\overline{x}_2$	$\overline{x}_1 + \overline{x}_2$	$\overline{x_1.x_2}$
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:

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

4.2. Các hệ thức cơ bản

Một số hệ thức cơ bản thường dùng trong đại số logic được cho ở bảng 1.5:

Bảng 1.5

1	$x + 0 = x$	10	$x_1.x_2 = x_2.x_1$
2	$x.1 = x$	11	$x_1 + x_1x_2 = x_1$
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. Phương pháp biểu diễn bằng bảng trạng thái:

Ở phương pháp này các giá trị của hàm được trình bày trong một bảng. Nếu hàm có n biến thì bảng có $n+1$ cột (n cột cho biến và 1 cột cho hàm) và 2^n hàng tương ứng với 2^n 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.

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.

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 $\overline{x}_i$. 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 trên ta có hàm ở dạng tổng chuẩn đầy đủ là:

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

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 $\overline{x}_i$. 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 trên ta có hàm ở dạng tích chuẩn đầy đủ là:

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

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

Nguyên tắc xây dựng bảng Karnaugh là:

- Để biểu diễn hàm logic n biến cần thành lập một bảng có 2^n 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

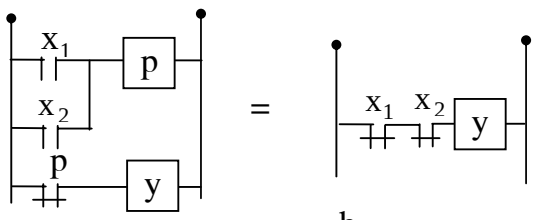
Ví dụ 2: bảng Karnaugh cho hàm bốn biến như bảng 1.8 sau:

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. 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

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

2. Phương pháp tối thiểu hoá hàm logic dùng thuật toán

Phương pháp dùng bảng Karnaugh

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

Qui tắc của phương pháp là: nếu có 2^n ô có giá trị 1 nằm kề nhau hợp thành một khối vuông hay chữ nhật thì có thể thay 2^n 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).

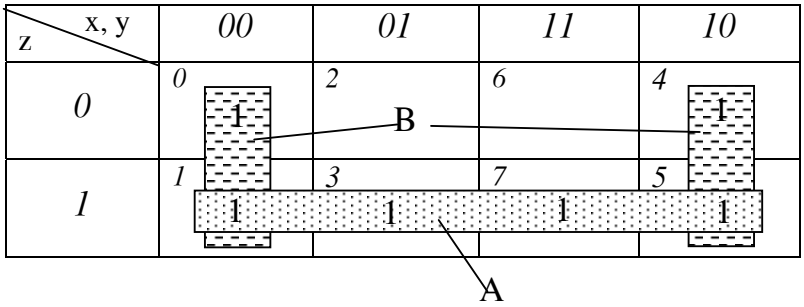
Qui tắc này áp dụng theo thứ tự giảm dần độ lớn các ô, sao cho cuối cùng toàn bộ các ô chưa giá trị 1 đều được bao phủ. Cũng có thể tiến hành tối thiểu theo giá trị 0 của hàm nếu số lượng của nó ít hơn nhiều so với giá trị 1, lúc bấy giờ hàm là hàm phủ định.

Ví dụ: Tối thiểu hàm

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

+ Lập bảng Karnaugh được như bảng 1.9. Bảng Karnaugh có 3 biến với 6 mintec có giá trị 1.

Bảng 1.9



+ 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 $\bar{y}$ không đổi vậy mintec mới B chỉ còn biến $\bar{y}$: $B = \bar{y}$.

Kết quả tối thiểu hoá là: $f = A + B = z + \bar{y}$

Phương pháp Quine Mc. Cluskey

Đây là phương pháp có tính tổng quát, cho phép tối thiểu hoá mọi hàm logic với số lượng biến vào lớn.

a, Một số định nghĩa

+ Đỉnh: là một tích chứa đầy đủ các biến của hàm, nếu hàm có n biến thì đỉnh là tích của n biến.

Đỉnh 1 là đỉnh mà hàm có giá trị bằng 1.

Đỉnh 0 là đỉnh mà hàm có giá trị bằng 0.

Đỉnh không xác định là đỉnh mà tại đó hàm có thể lấy một trong hai giá trị 0 hoặc 1.

+ Tích cực tiểu: là tích có số biến là cực tiểu để hàm có giá trị bằng 1 hoặc không xác định.

+ Tích quan trọng: là tích cực tiểu mà giá trị hàm chỉ duy nhất bằng 1 ở tích này.

b, Tối thiểu hoá bằng phương pháp Quine Mc. Cluskey

Để rõ phương pháp ta xét ví dụ minh hoạ, tối thiểu hoá hàm $f(x_1,x_2,x_3,x_4)$ với các đỉnh bằng 1 là L = 2, 3, 7, 12, 14, 15 và các đỉnh có giá trị hàm không xác định là N = 6, 13. Các bước tiến hành như sau:

Bước 1: Tìm các tích cực tiểu

- Lập bảng biểu diễn các giá trị hàm bằng 1 và các giá trị không xác định ứng với mã nhị phân của các biến theo thứ tự số số 1 tăng dần (bảng 1.10a).
- Xếp thành từng nhóm theo số lượng chữ số 1 với thứ tự tăng dần. (bảng 1.10b ta có 4 nhóm: nhóm 1 có 1 số chứa 1 chữ số 1; nhóm 2 gồm 3 số chứa 2 chữ số 1; nhóm 3 gồm 3 số chứa 3 chữ số 1, nhóm 4 có 1 số chứa 1 chữ số 1).
- So sánh mỗi tổ hợp thứ i với tổ hợp thứ i +1, nếu hai tổ hợp chỉ khác nhau ở một cột thì kết hợp 2 tổ hợp đó thành một tổ hợp mới, đồng thời thay cột số khác nhau của 2 tổ hợp cũ bằng một gạch ngang (-) và đánh dấu v vào hai tổ hợp cũ (bảng 1.10c). Về cơ sở toán học, ở đây để thu gọn các tổ hợp ta đã dùng tính chất:

$$xy + x\bar{y} = x$$

- Cứ tiếp tục công việc. Từ bảng 1.10c ta chọn ra các tổ hợp chỉ khác nhau 1 chữ số 1 và có cùng gạch ngang (-) trong một cột, nghĩa là có cùng biến vừa được giản ước ở bảng 1.10c, như vậy ta có bảng 1.10d.

Bảng 1.10

a		b			c		d	
Số thập phân	Cơ số 2 $x_1x_2x_3x_4$	Số chữ số 1	Số thập phân	Cơ số 2 $x_1x_2x_3x_4$	Liên kết	$x_1x_2x_3x_4$	Liên kết	$x_1x_2x_3x_4$
2	0010	1	2	0010v	2,3	001-v	2,3,6,7 2,6,3,7	0-1-
3	0011	2	3	0011v	2,6	0-10v	6,7,14,15 6,14,7,15	-11-
6 *	0110		6	0110v	3,7	0-11v	12,13,14,15	11- -
12	1100		12	1100v	6,7	011-v		
7	0111	3	7	0111v	6,14	-110v		
13 *	1101		13	1101v	12,13	110-v		
14	1110		14	1110v	12,14	11-0v		
15	1111	4	15	1111v	7,15	-111v		
					13,15	11-1v		
					14,15	111-v		

Các tổ hợp tìm được ở bảng 1.10d là tổ hợp cuối cùng, các tổ hợp này không còn khả năng kết hợp nữa, đây chính là các tích cực tiểu của hàm đã cho. Theo thứ tự $x_1x_2x_3x_4$, chỗ có dấu (-) được lược bỏ, các tích cực tiểu được viết như sau:

0-1- (phủ các đỉnh 2,3,6,7) ứng với: $\overline{x}_1x_3$

-11- (phủ các đỉnh 6,7,14,15) ứng với: x_2x_3

11- - (phủ các đỉnh 12,13,14,15) ứng với: x_1x_2

Bước 2: Tìm các tích quan trọng

Việc tìm các tích quan trọng cũng được tiến hành theo các bước nhỏ.

Gọi L_i là tập các đỉnh 1 đang xét ở bước nhỏ thứ i, lúc này không quan tâm đến các đỉnh có giá trị không xác định nữa.

Z_i là tập các tích cực tiểu đang ở bước nhỏ thứ i.

E_i là tập các tích quan trọng ở bước nhỏ thứ i.

- Với i = 0

$$L_0 = (2,3,7,12,14,15)$$

$$Z_0 = (\overline{x}_1x_3, x_2x_3, x_1x_2)$$

Xác định các tích quan trọng E_0 từ tập L_0 và Z_0 như sau:

+ Lập bảng trong đó mỗi hàng ứng với một tích cực tiểu thuộc Z_0, mỗi cột ứng với một đỉnh thuộc L_0. Đánh dấu "x" vào các ô trong bảng ứng với tích cực tiểu bảng 1.11 (tích $\overline{x}_1x_3$ ứng với các đỉnh 2,3,7; tích x_2x_3 ứng với các đỉnh 7,14,15; tích x_1x_2 ứng với các đỉnh 12,14,15 bảng 1.10)

Bảng 1.11

Z_0 \ L_0	*2*	*3*	*7*	*12*	*14*	*15*
$\overline{x}_1x_3$	(x)	(x)	x			
x_2x_3			x		x	x
x_1x_2				(x)	x	x

Xét từng cột, cột nào chỉ có một dấu "x" thì tích cực tiểu (hàng) ứng với nó là tích quan trọng, ta đổi thành dấu "(x)'. Vậy tập các tích quan trọng ở bước này là:

$$E_0 = (\overline{x}_1x_3, x_1x_2)$$

- Với i = 1

Tìm L_1 từ L_0 bằng cách loại khỏi L_0 các đỉnh 1 của E_0.

Tìm Z_1 từ Z_0 bằng cách loại khỏi Z_0 các tích trong E_0 và các tích đã nằm trong hàng đã được chọn từ E_0.

Khi đã tìm được L_1 và Z_1, làm lại như bước i = 0 ta sẽ tìm được tích quan trọng E_1.

Công việc cứ tiếp tục cho đến khi $L_k = 0$.

Trong ví dụ này vì $E_0 = (\overline{x}_1x_3, x_1x_2)$ mà các đỉnh 1 của $\overline{x}_1x_3$ là 2,3,7; các đỉnh 1 của x_1x_2 là 12,14,15 (bỏ qua đỉnh 6, 13 là các đỉnh không xác định); do đó $L_1 = 0$, quá trình kết thúc. Kết quả dạng hàm tối thiểu chính là tổng của các tích cực tiểu. Vậy hàm cực tiểu là:

$$f = \overline{x}_1x_3 + x_1x_2$$

§1.4. Các hệ mạch logic

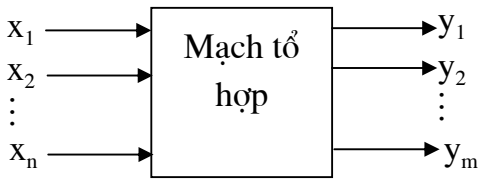
Các phép toán và định lý của đại số Boole giúp cho thao tác các biểu thức logic. Trong kỹ thuật thực tế là bằng cách nối cổng logic của các mạch logic với nhau (theo kết cấu đã tối giản nếu có). Để thực hiện một bài toán điều khiển phức tạp, số mạch logic sẽ phụ thuộc vào số lượng đầu vào và cách giải 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. 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.



Hình 1.4

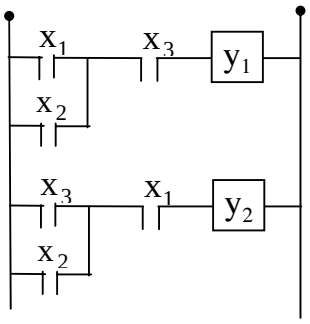
Sơ đồ mạch logic tổ hợp như hình 1.4

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.

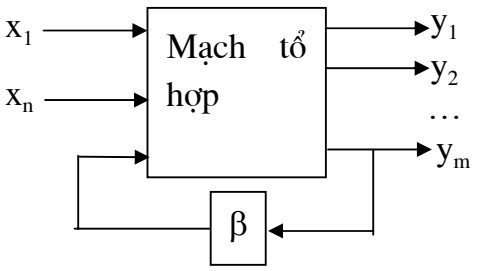


Hình 1.5

Ví dụ: về mạch logic tổ hợp như hình 1.5

2. Mạch logic trình tự

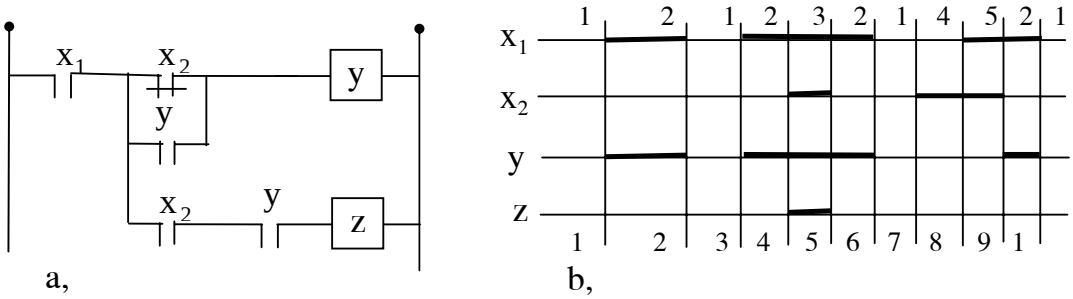
Mạch trình tự hay còn gọi là mạch dãy (sequential circuits) là mạch trong đó trạng thái của tín hiệu ra không những phụ thuộc tín hiệu vào mà còn phụ thuộc cả trình tự tác động của tín hiệu vào, nghĩa là có nhớ các trạng thái. Như vậy, về mặt thiết bị thì ở mạch trình tự không những chỉ có các phần tử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



Hình 1.6

Sơ đồ nguyên lý mạch logic trình tự như 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.



Hình 1.7

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.

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

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.

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.

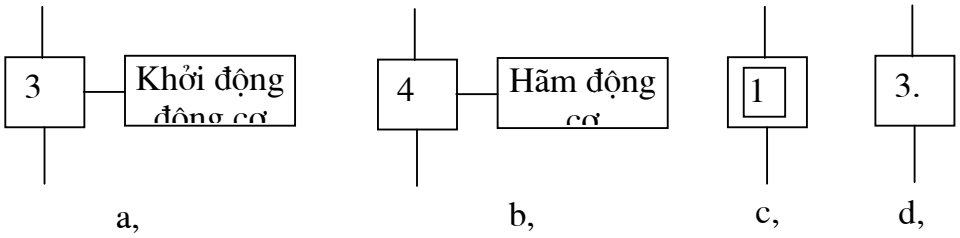
Mạng grafcet cho một quá trình sản xuất luôn luôn là một đồ hình khép kín từ trạng thái đầu đến trạng thái cuối và từ trạng thái cuối về trạng thái đầu.

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.

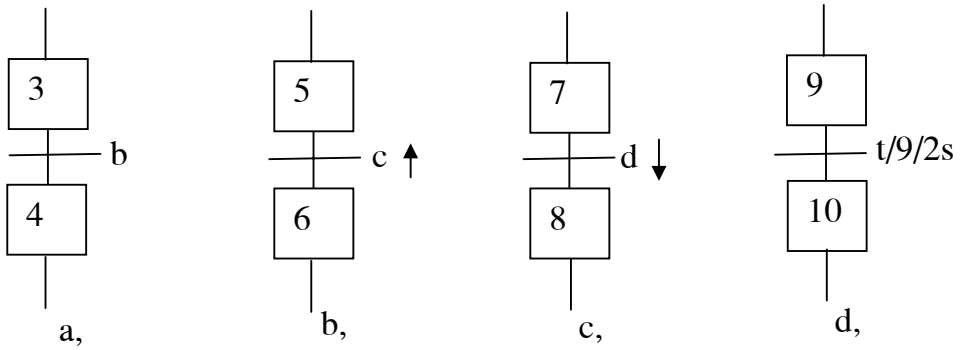
- Trạng thái khởi động được thể hiện bằng 2 hình vuông lồng vào nhau, thứ tự thường là 1 hình 1.8c.

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



Hình 1.8

- 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



Hình 1.9

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.

- 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.

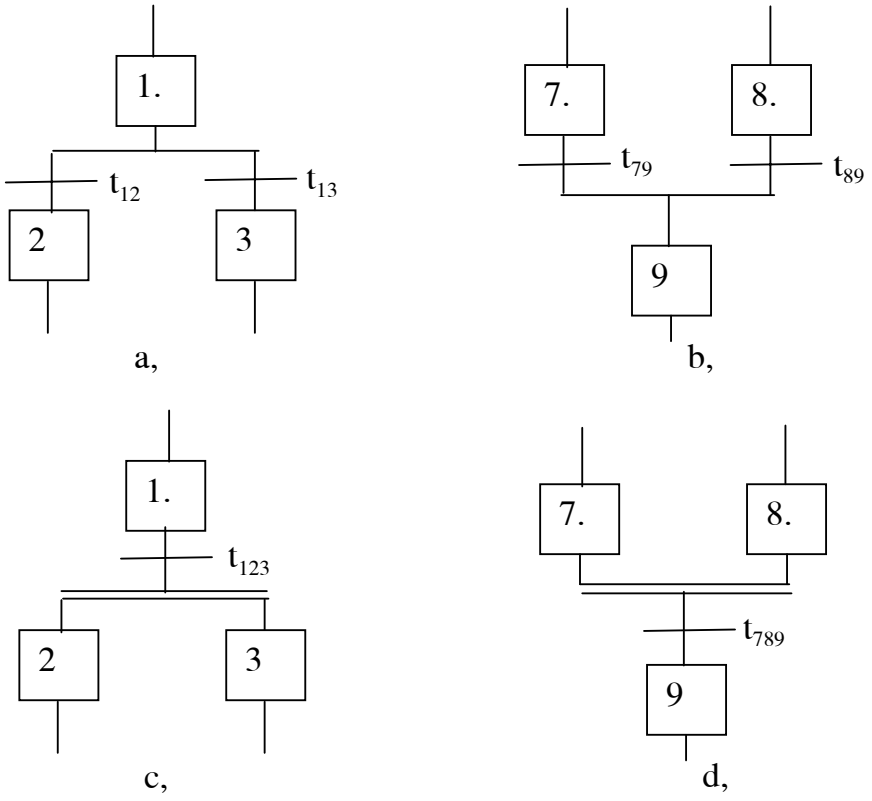
Sơ đồ song song là sơ đồ chỉ có một điều kiện liên hệ giữa 3 trạng thái như hình 1.10c và d.

Ở hình 1.10a , khi trạng thái 1 đang hoạt động, nếu chuyển tiếp t_{12} thoả mãn thì trạng thái 2 hoạt động; nếu chuyển tiếp t_{13} thoả mãn thì trạng thái 3 hoạt động.

Ở hình 1.10b nếu trạng thái 7 đang hoạt động và có t_{79} thì trạng thái 9 hoạt động, nếu trạng thái 8 đang hoạt động và có t_{89} thì trạng thái 9 hoạt động.

Ở hình 1.10c nếu trạng thái 1 đang hoạt động và có t_{123} 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 t_{789} thì trạng thái 9 hoạt động.

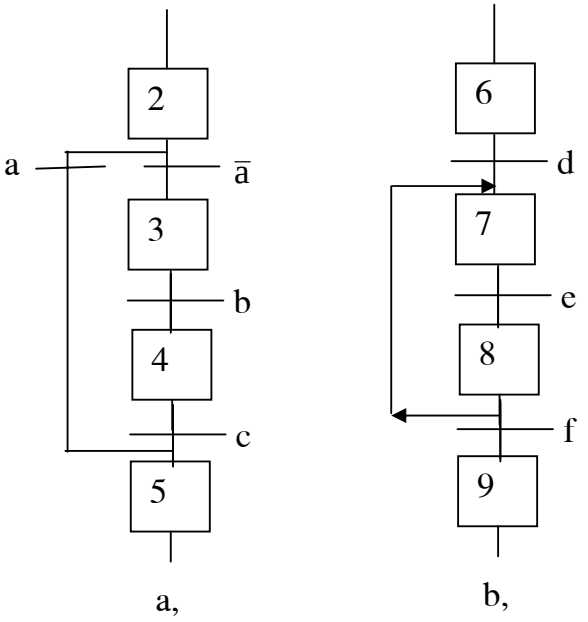


Hình 1.10

- 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 1.11b 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.

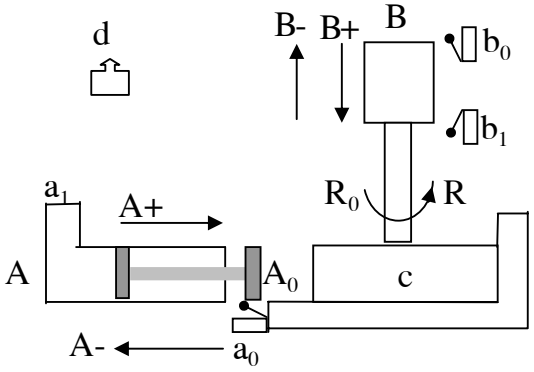


Hình 1.11

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

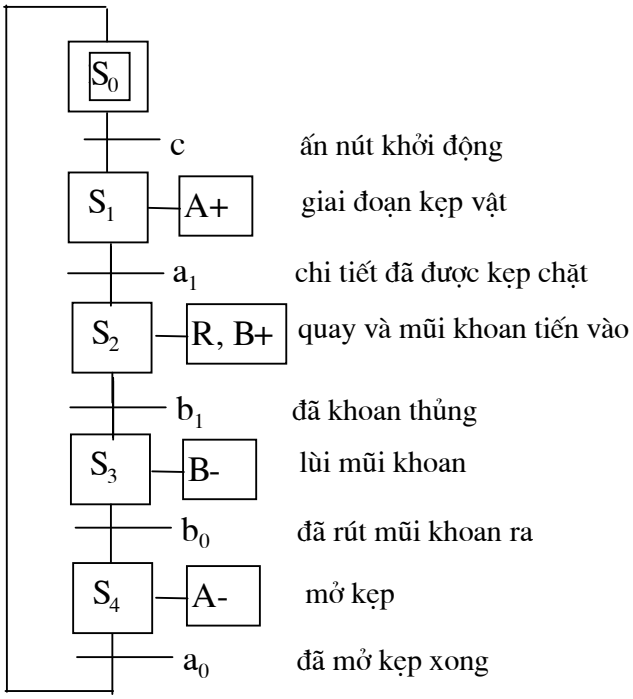
+ Giai đoạn 1: S_1 píttông A chuyển động theo chiều A+ để kẹp chặt chi tiết c. Khi lực kẹp đạt yêu cầu được xác định bởi cảm biến áp suất a_1 thì chuyển sang giai đoạn 2.

+ 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

5. Phân tích mạng grafcet

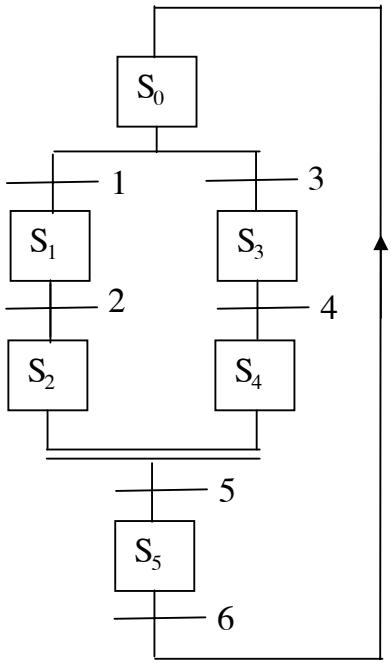
5.1. Qui tắc vượt qua, chuyển tiếp

- 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.

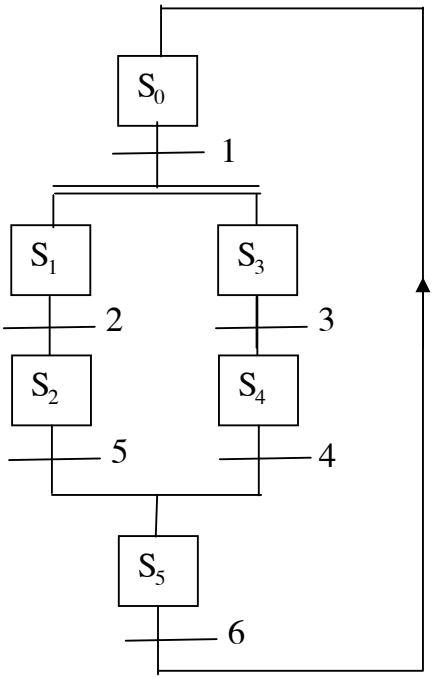


Hình 1.14

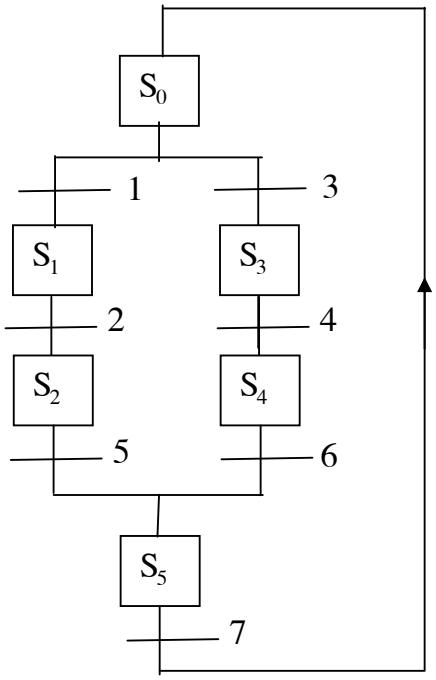
Ví dụ 1: Sơ đồ hình 1.14 là sơ đồ có nhánh chết. Sơ đồ này không thể làm việc được do S_2 và S_4 không thể cùng tích cực vì giả sử hệ đang ở trạng thái ban đầu S_0 nếu có điều kiện 3 thì S_0 hết tích cực và chuyển sang S_3 tích cực. Sau đó nếu có điều kiện 4 thì S_3 hết tích cực và S_4 tích cực. Nếu 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.

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



Hình 1.15



Hình 1.16

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.

5.2. 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.

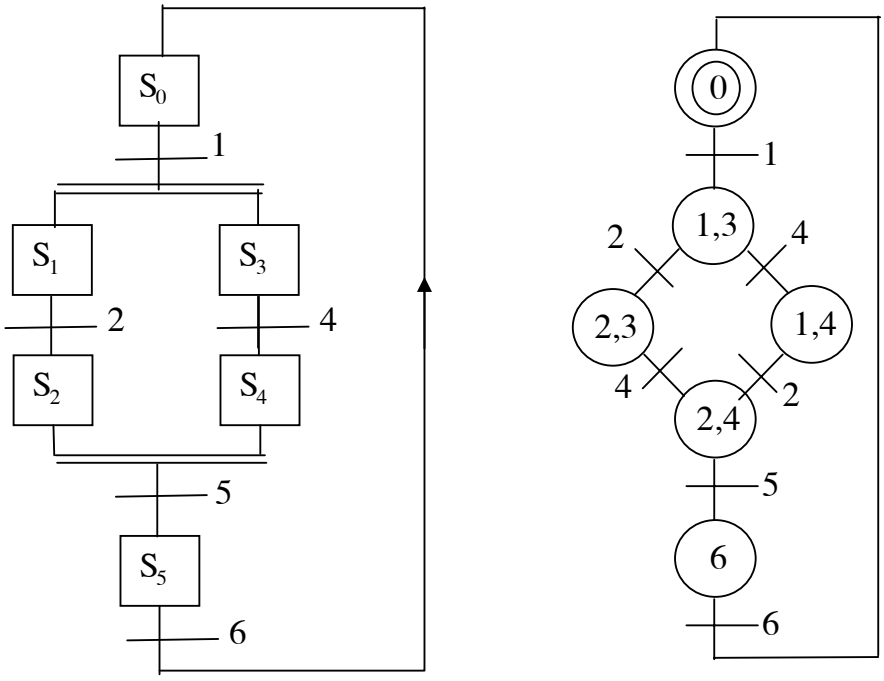
+ Quá trình cứ như vậy tiếp tục, ta có thể vẽ hoàn thiện được giản đồ điểm (sơ đồ tạo thành mạch liên tục, sau khi kết thúc lại trở về điểm xuất phát) hoặc không vẽ hoàn thiện được. Nhìn vào giản đồ điểm ta sẽ có các kết luận sau:

- 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.



a, b,

Hình 1.17

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.

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.

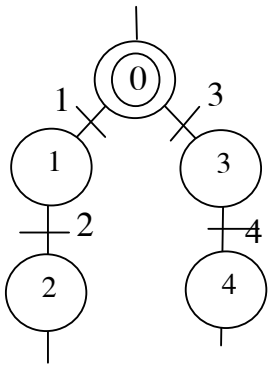
Khi hệ đang ở 2,4 nếu điều kiện 5 được thực hiện thì giai đoạn 5 tích cực (thêm dấu "."), giai đoạn 2 và 4 hết tích cực (mất dấu "."). Vậy sau điều kiện 5 tạo ô mới (nối với ô 2,4), ô này ghi trạng thái tích cực còn lại trên grafcet là 5.

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

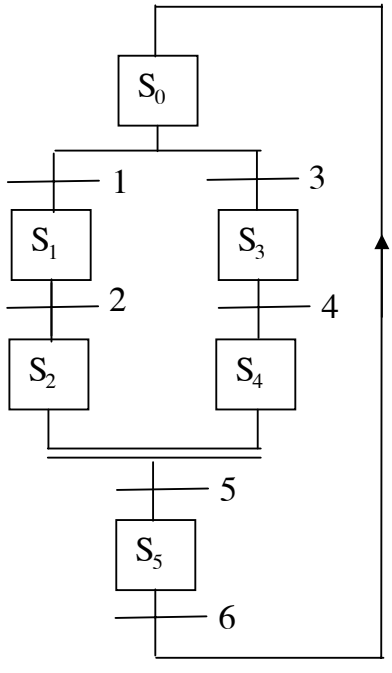
Giản đồ điểm như hình 1.18. Trong trường hợp này ta không thể vẽ tiếp được nữa vì để S_5 tích cực phải có cả S_2 và S_4 cùng tích cực cùng điều kiện 5. Nhưng không có ô nào có 2,4.



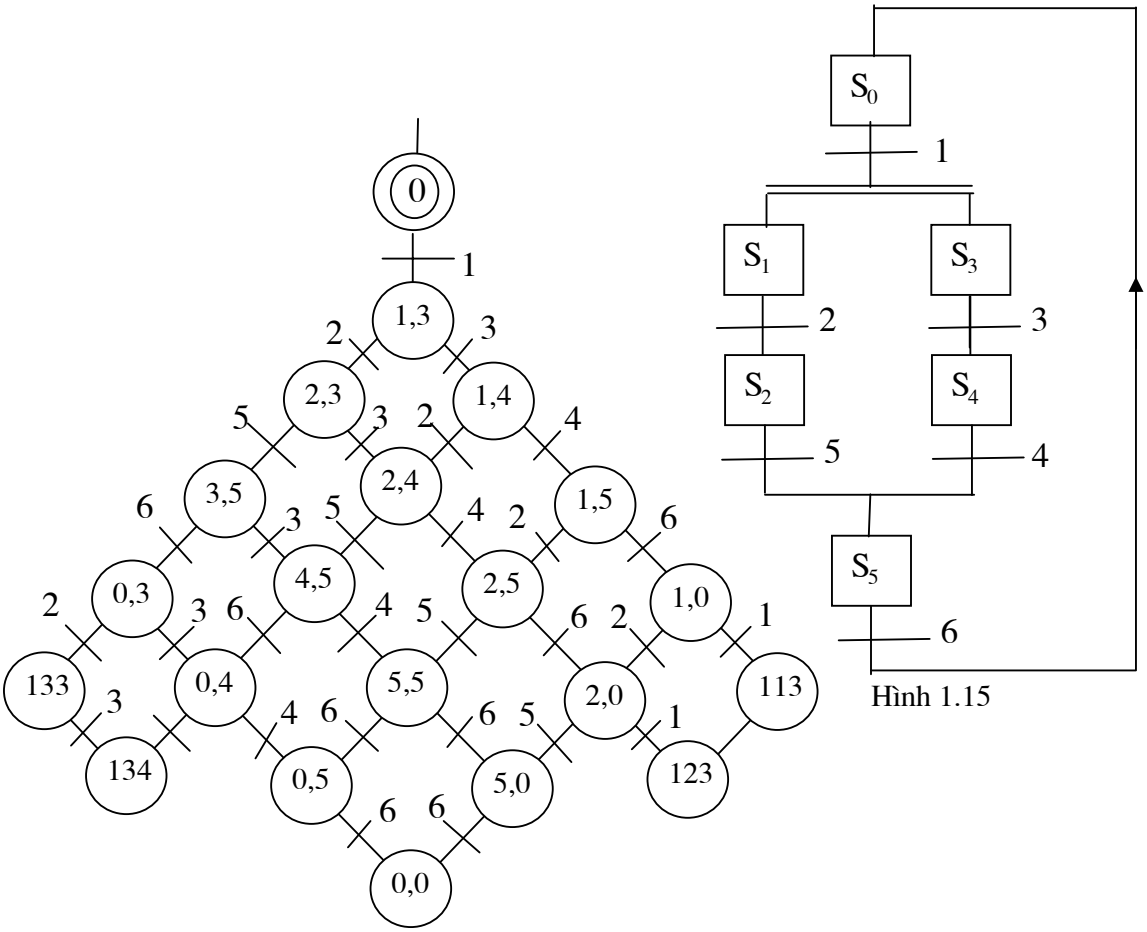
Hình 1.18

Ví dụ 3: Vẽ giản đồ điểm cho sơ đồ không sạch hình 1.5

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 1.14



Hình 1.15

Hình 1.19

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: MỘT SỐ ỨNG DỤNG MẠCH LOGIC TRONG ĐIỀU KHIỂN

§2.1. Các thiết bị điều khiển

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. 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.

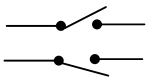
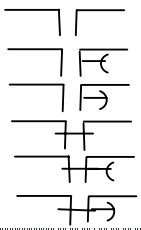
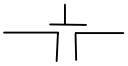
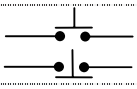
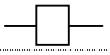
Phần tử cơ bản của rơle nhiệt là bản lưỡng kim gồm hai miếng kim loại có độ dãn nở nhiệt khác nhau dán lại với nhau. Khi bản lưỡng kim bị đốt nóng (thường là bằng dòng điện cần bảo vệ) sẽ bị biến dạng (cong), độ biến dạng tới ngưỡng thì sẽ tác động vào các bộ phận khác để cắt mạch điện.

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

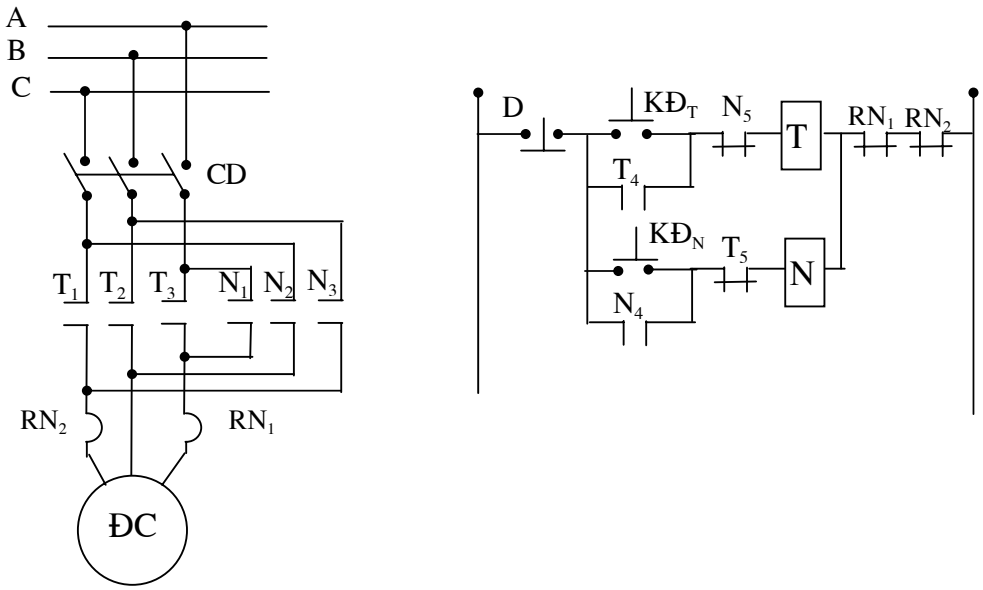
Tuỳ theo công suất và yêu cầu công nghệ mà động cơ không đồng bộ rôto lồng sóc có thể được nối trực tiếp vào lưới điện, dùng đổi nối sao-tam giác, qua điện kháng, qua biến áp tự ngẫu, ngày nay thường dùng các bộ khởi động mềm để khởi động động cơ.

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

Cầu dao 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 $KĐ_T$ (hoặc $KĐ_N$), 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.

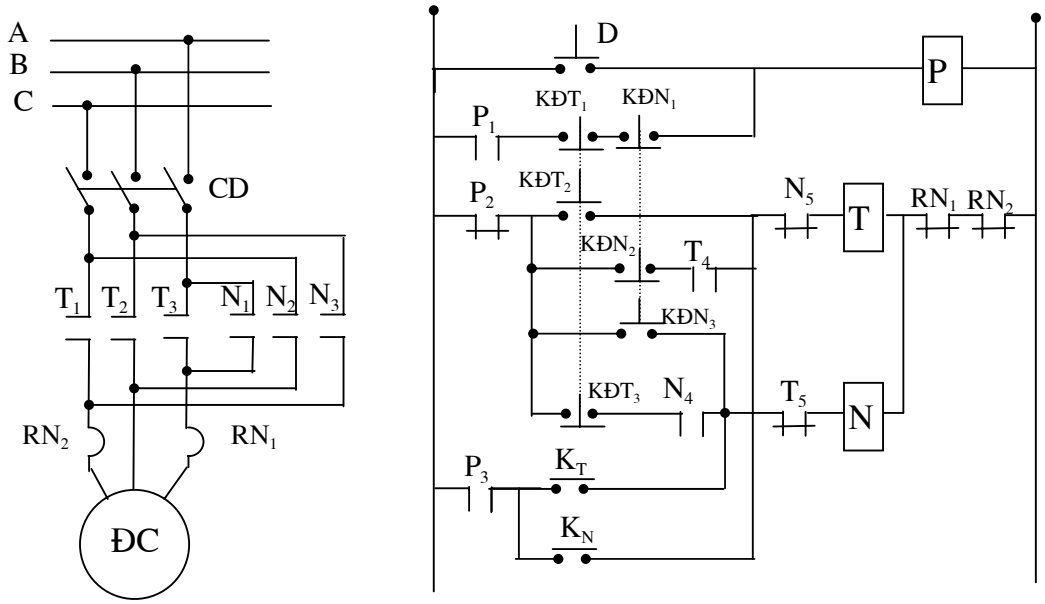
Để dừng động có ta ấn nút dừng D, các cuộn hút mất điện, cắt điện động cơ, động cơ tự dừng.

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. 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 K_T và rơle tốc độ ngược có tiếp điểm K_N, 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 $KĐN_1$ 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.

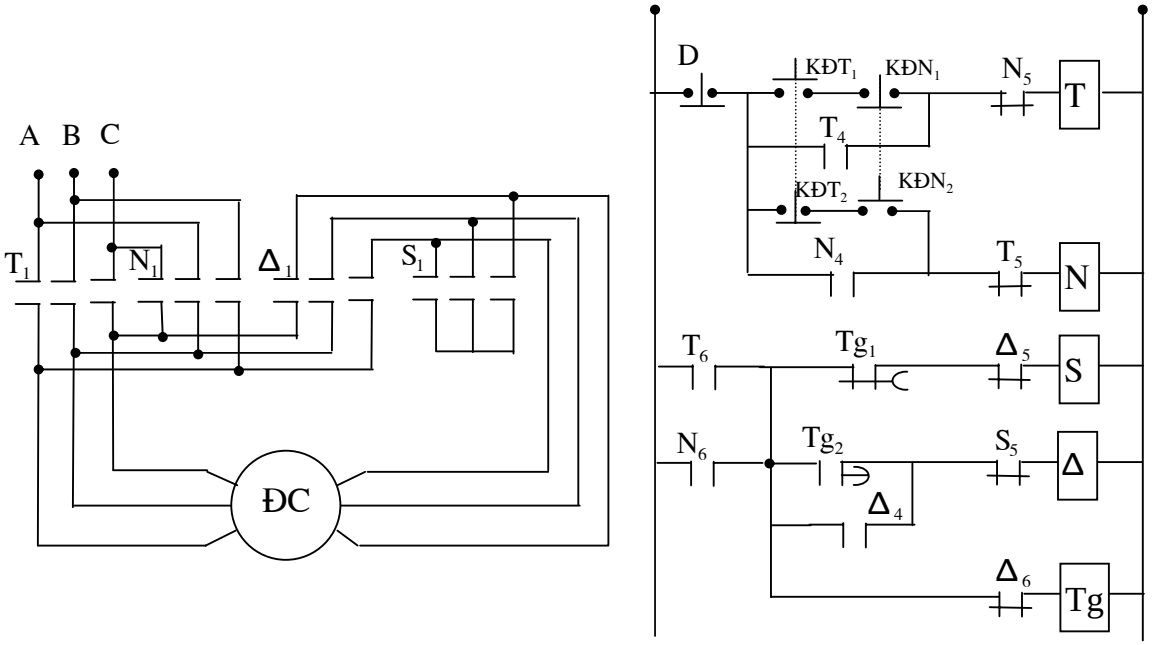
Nếu muốn dừng ta ấn nút dừng D, cấp điện cho cuộn hút P, cuộn hút P đóng tiếp điểm P_1 để tự duy trì, hở P_2 cắt đường nguồn đang cấp cho cuộn hút T hoặc N, nhưng lập tức P_3 kín cuộn hút N hoặc T lại được cấp điện, nếu khi trước động cơ đang chạy thuận (cuộn T làm việc) tốc độ đang lớn thì K_T kín, cuộn N được

cấp điện đóng điện cho chế độ chạy ngược làm động cơ dừng nhanh, khi tốc độ đã giảm thấp thì K_T mở cắt điện cuộn hút N, động cơ dừng hẳn.

Khi các rơle nhiệt tác động thì động cơ dừng tự do.

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 $\sqrt{3}$ do đó dòng điện khởi động giảm. Sơ đồ hình 2.3 cho phép thực hiện đổi nối Y/Δ có đảo chiều.



Hình 2.3

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.

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.

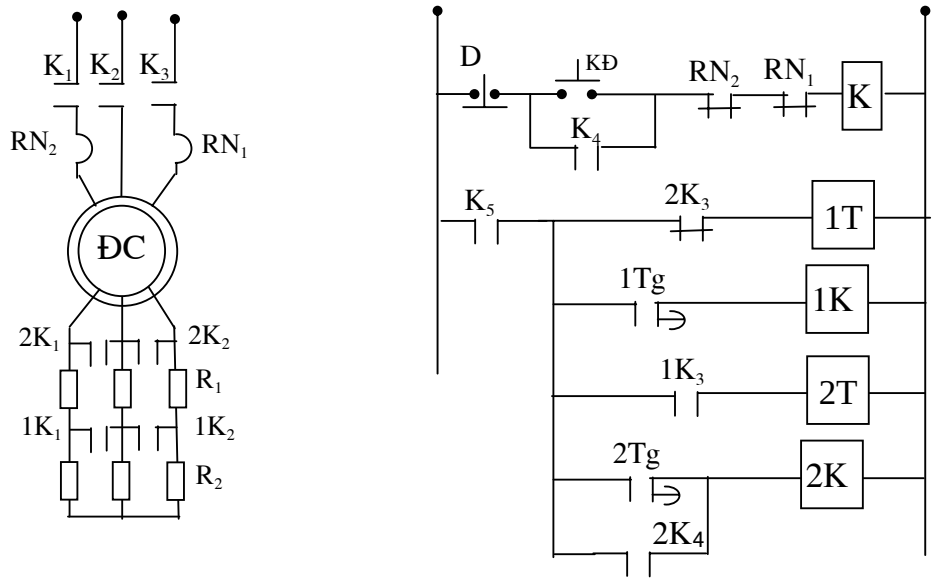
Khi muốn dứng ta ấn nút dừng D, động cơ dừng tự do.

§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.

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.

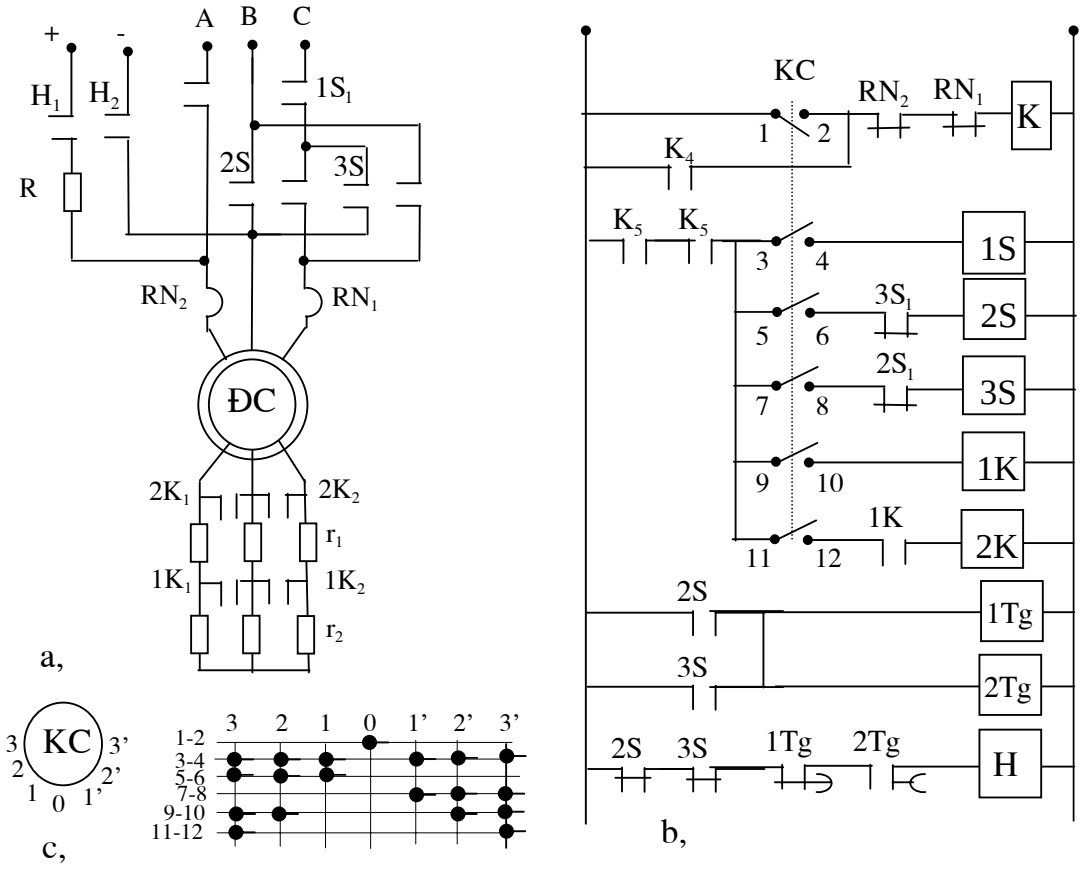
Để khởi động ta ấn nút khởi động KĐ cấp điện cho cuộn hút K các tiếp điểm K_1, K_2, K_3 đóng cấp điện cho động cơ, động cơ khởi động với hai cấp điện trở phụ, tiếp điểm K_4 để tự duy trì, tiếp điểm K_5 để cấp điện cho các rơle thời gian. Sau khoảng thời gian chỉnh định tiếp điểm thường mở đóng chậm 1Tg 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. 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 xtato 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

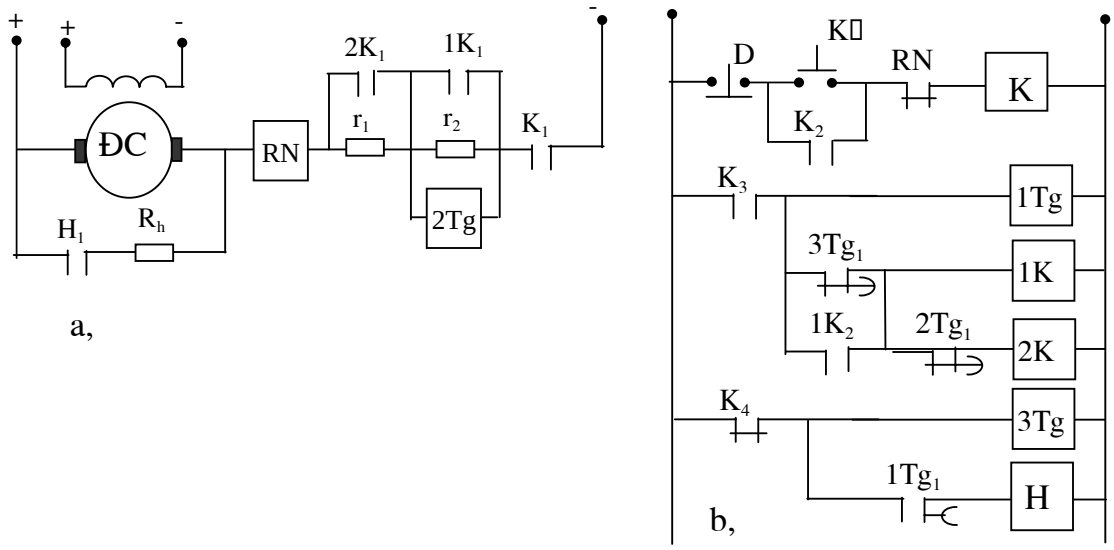
có gắn các tiếp điểm động và kết hợp với các tiếp điểm tĩnh tạo thành các cặp tiếp điểm được đóng cắt tuỳ thuộc vào vị trí quay của trụ. Đồ thị đóng mở tiếp điểm của bộ khống chế KC được thể hiện trên hình 2.5c. Ví dụ ở vị trí 0 của bộ khống chế chỉ có tiếp điểm 1-2 đóng, tất cả các vị trí còn lại của 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

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 $3Tg_1$ 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



Hình 2.6

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 R_h 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.

Phần 2: ĐIỀU KHIỂN LOGIC CÓ LẬP TRÌNH (PLC)

Chương 3: LÝ LUẬN CHUNG VỀ ĐIỀU KHIỂN LOGIC LẬP TRÌNH PLC

§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 cho các kỹ sư, không yêu cầu cao kiến thức về máy tính và ngôn ngữ máy tính, có thể vận hành. Chúng được thiết kế cho 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 yêu 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 sẽ giám sát các tín hiệu vào và các tín hiệu ra theo chương trình này 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.

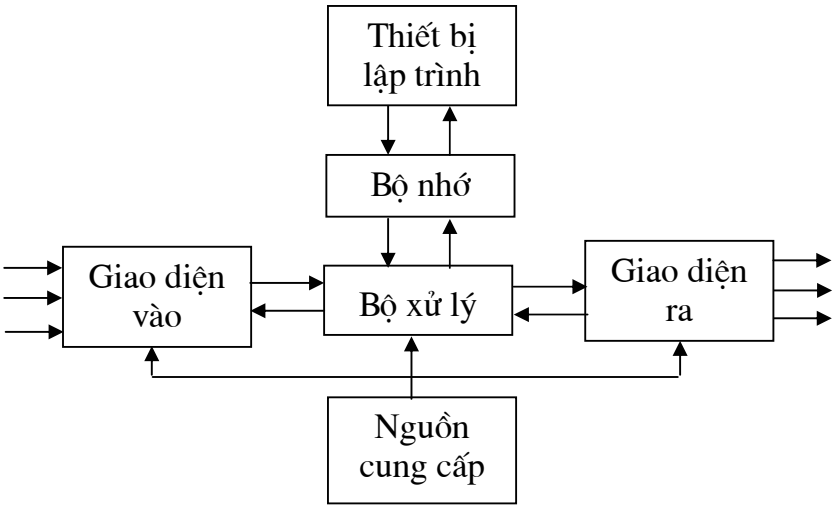
+ Phân phát các lệnh điều khiển đến các địa chỉ thích hợp.

Riêng đối với máy công cụ và người máy công nghiệp thì bộ PLC có thể liên kết với bộ điều khiển số NC hoặc CNC hình thành bộ điều khiển thích nghi. 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

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

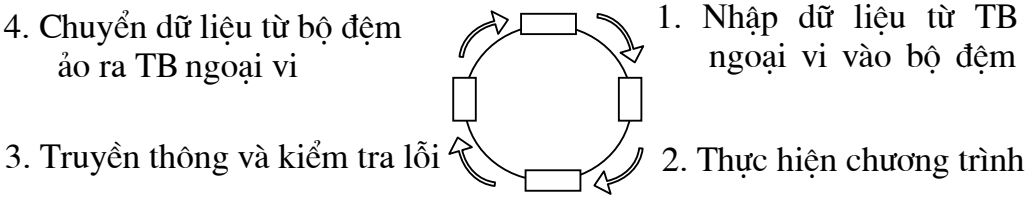


Hình 3.1

1.1. 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



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.

1.2. 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).

1.3. Thiết bị lập trình

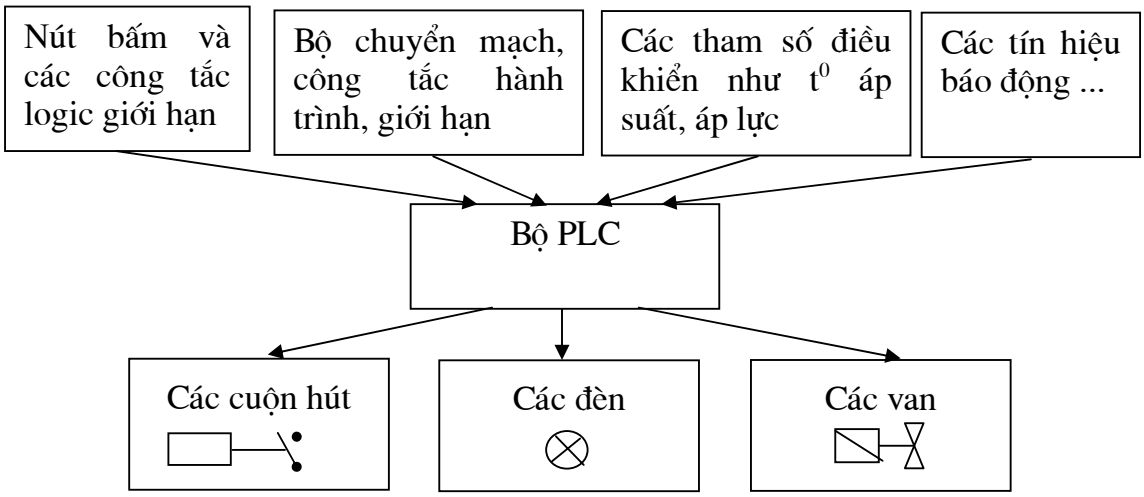
Thiết bị lập trình được sử dụng để lập các chương trình điều khiển cần thiết sau đó được chuyển cho PLC. Thiết bị lập trình có thể là thiết bị lập trình chuyên dụng, có thể là thiết bị lập trình cầm tay gọn nhẹ, có thể là phần mềm được cài đặt trên máy tính cá nhân.

1.4. 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.

1.5. 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



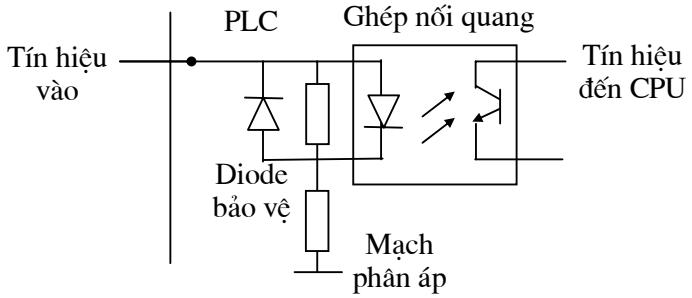
Hình 3.3

vào/ra có thể thể hiện như hình 3.3.

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

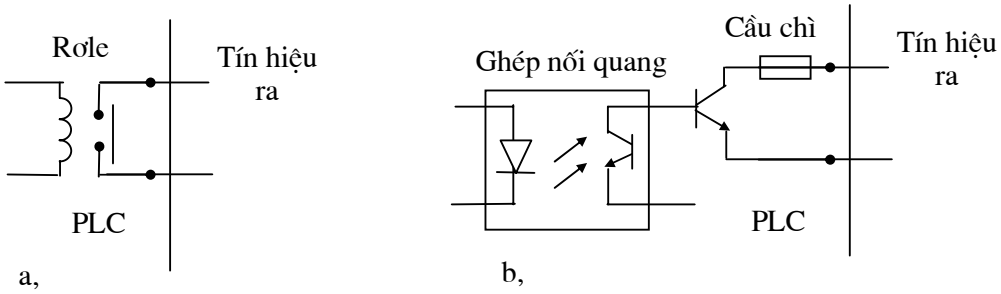
Các kênh vào/ra đã có các chức năng cách ly và điều hoá tín hiệu sao cho các bộ cảm biến và các bộ tác động có thể nối trực tiếp với chúng mà không cần thêm mạch điện khác.

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



Hình 3.4

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. 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.

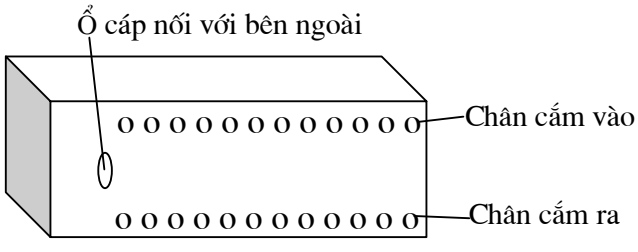


Hình 3.5

2. Cấu tạo chung của PLC

Các PLC có hai kiểu cấu tạo cơ bản là: kiểu hộp đơn và kiểu modulle nối ghép.

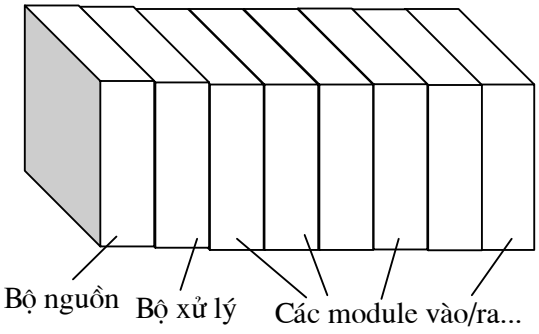
Kiểu hộp đơn thường dùng cho các PLC cỡ nhỏ và được cung cấp dưới dạng nguyên chiếc hoàn chỉnh gồm bộ nguồn, bộ xử lý, bộ nhớ và các giao diện vào/ra. Kiểu hộp đơn thường vẫn có khả năng ghép nối được với các module ngoài để mở rộng khả năng của PLC. Kiểu hộp đơn như hình 3.6.



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. 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.



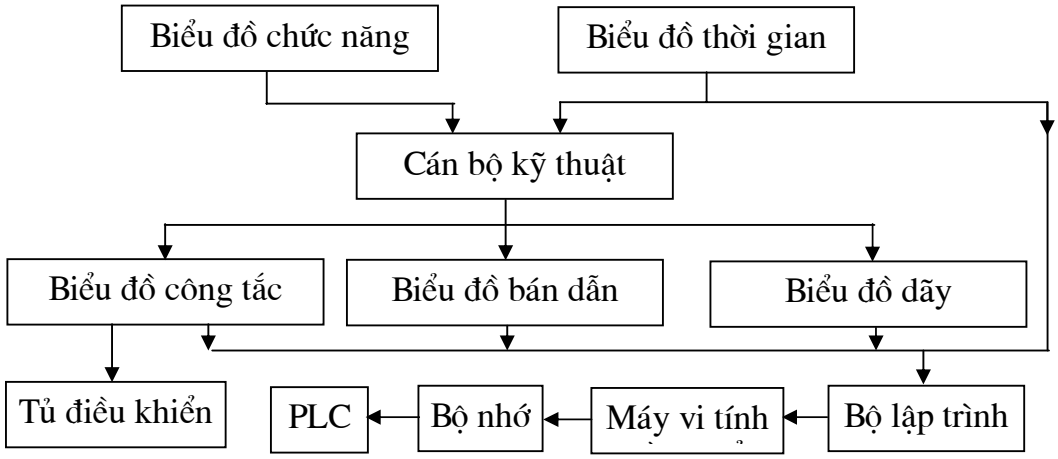
Hình 3.7

§3.3. Các vấn đề về lập trình

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.

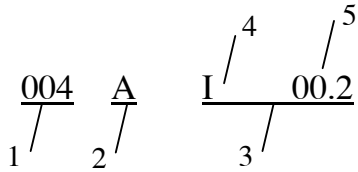
2. Các phương pháp lập trình

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 đã 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.

2.1. Một số ký hiệu chung

Cấu trúc lệnh:

Một lệnh thường có ba phần chính và thường viết như hình 3.9 (có loại PLC có cách viết hơi khác):



Hình 3.9

1. Địa chỉ tương đối của lệnh (thường 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 tượng của lệnh, trong lập trình LAD thì 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

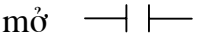
  Ví dụ: A I 1.0; = Q 0.7

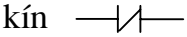
Ngoài các ký hiệu khá chung như trên thì mỗi hãng còn có các ký hiệu riêng, có bộ lệnh riêng. Ngay cùng một hãng ở các thời chế tạo khác nhau cũng có đặc điểm khác nhau với bộ lệnh khác nhau. Do đó, khi sử dụng PLC thì mỗi loại PLC ta phải tìm hiểu cụ thể hướng dẫn sử dụng của nó.

Một số ký hiệu khác nhau với các lệnh cơ bản được thể hiện rõ trên bảng 3.1

2.2. Phương pháp hình thang LAD (Ladder Logic)

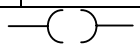
Phương pháp hình thang có dạng của biểu đồ nút bấm. Các phần tử cơ bản của phương pháp hình thang là:

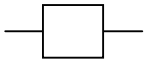
+ Tiếp điểm: thường mở 

Thương kín 

Bảng 3.1

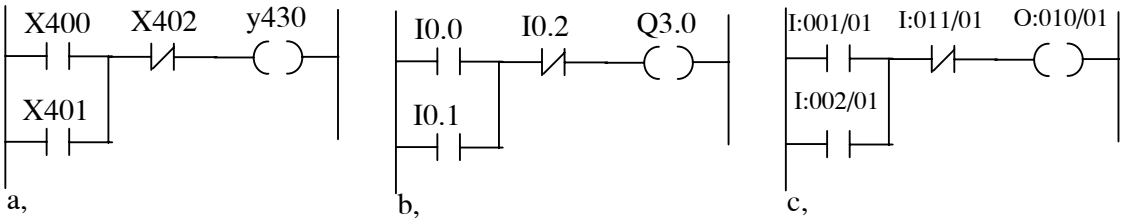
IEC 1131-3	Misubishi	OMRON	Siemens	Telemec-anique	Spreher và Schuh	Chú thích
LD	LD	LD	A	L	STR	Khởi đầu với tiếp điểm thường mở
LDN	LDI	LD NOT	AN	LN	STR NOT	Khởi đầu với tiếp điểm thường kín
AND	AND	AND	A	A	AND	Phần tử nối tiếp có tiếp điểm mở
ANDN	ANI	AND NOT	AN	AN	AND NOT	Phần tử nối tiếp có tiếp điểm 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

+ Cuộn dây (mô tả các rơle) 

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

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 biếu 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



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

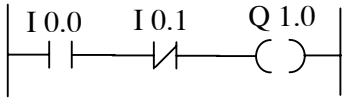
2.3. Phương pháp liệt kê lệnh STL (Statement List)

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)

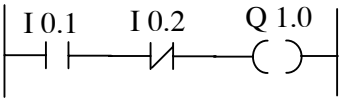
```
0    A    I  0.0
1    A    I  0.1
2    =    Q  1.0
```



Hình 3.11

Một đoạn STL của PLC S7-200 (Siemens)

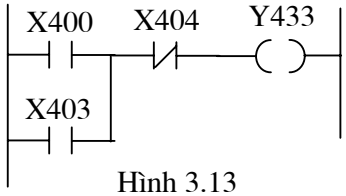
```
0    LD   I  0.1
1    A    I  0.2
3    =    Q  1.0
```



Hình 3.12

Một đoạn STL của PLC MELSEC F1 (Nhật)

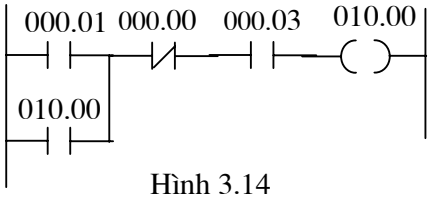
```
0    LD   X  400
1    O    X  403
2    ANI  X  404
3    OUT  Y  433
```



Hình 3.13

Một đoạn STL của CPM1A (OMRON)

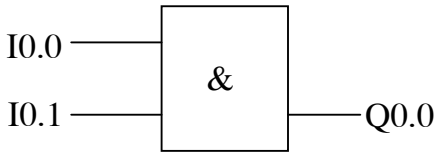
```
0    LD          000.01
1    OR          010.00
2    AND NOT     000.00
3    AND         000.03
4    OUT         010.00
```



Hình 3.14

2.4. 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: Phương pháp biểu diễn CSF

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

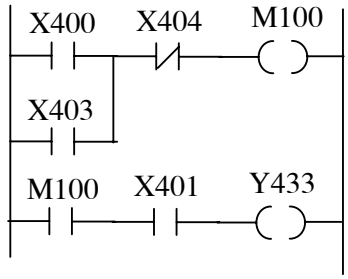
cấp các chức năng đặc biệt gắn liền với PLC và được dùng phổ biết trong lập trình. Rơle nội này tương tự như các rơle trung gian trong sơ đồ rơle công tắc tơ. Rơle nội cũng được coi là các đầu ra để nhận các lệnh gán đầu ra, nhưng thực chất đầu ra này không đưa ra ngoài (không phải thiết bị ngoại vi) 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

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.

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.

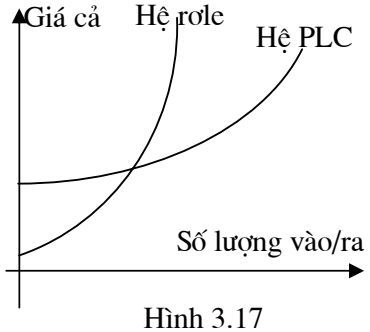
+ Dễ dàng thay đổi chương trình: Những thay đổi chương trình được tiến hành đơn giản. Để sửa đổi hệ thống điều khiển và các quy tắc điều khiển đang được sử dụng, người vận hành chỉ cần nhập tập lệnh khác, gần như không cần mắc nối lại dây (tuy nhiên, có thể vẫn phải nối lại nếu cần thiết). Nhờ đó hệ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

+ 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.



Hình 3.17

+ 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

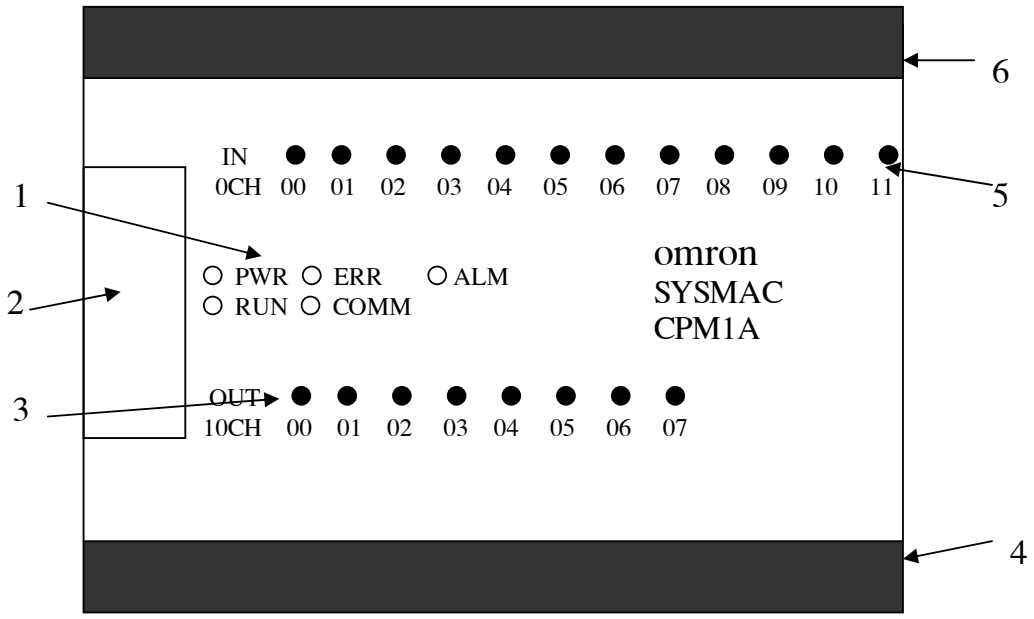
Bộ thiết bị lập trình thường đắt, sử dụng ít.

Chương 4: BỘ ĐIỀU KHIỂN PLC - CPM1A

§4.1. Cấu hình cứng

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).

2. Các thông số kỹ thuật

2.1. 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	AD			
CPM1A-40CDR-A	AC	24	16	40
CPM1A-40CDR-D	DC			

2.2. 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 bên xung lực		147m/s^2 (20G) ba lần mỗi chiều X, Y và Z			
Nhiệt độ môi trường		Nhiệt độ làm việc: 0 đến 55C^0 Nhiệt độ bảo quản:-20 đến 75C^0			
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

2.3. 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 μs Lệnh đặc biệt: 12.375 μs (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 (vùng TR)		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 25^0C)			
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 thời gian		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)			

2.4. Cấu trúc vùng nhớ

Dữ liệu		Từ	Bit	Chức năng
IR	Vào	IR 000 đến IR 009 (10 words)	IR 00000 đến IR 00915 (160 bits)	Các bit này có thể làm việc ở vùng vào ra mở rộng
	Ra	IR 010 đến IR 019 (10 words)	IR 01000 đến IR 01915 (160 bits)	
	làm việc	Ir 200 đến IR 231 (32 words)	Ir 20000 đến IR to 23115 (512 bits)	Các từ bit này có thể sử dụng tuỳ ý trong chương trình
SR		SR 232 đến SR 255 (24 words)	SR 23200 đến 25515 (384 bits)	Những bit này phục vụ cho chức năng đặc biệt như cờ và bit điều khiển.
TR		---	TR 0 đến TR 7 (8 bits)	Bit này được sử dụng ở trạng thái đóng mở trong chương trình phần nhánh
HR		HR 00 đến HR 19 (20 words)	HR 0000 đến HR 1915 (320 bits)	Những bít này lưu giữ trạng thái đóng mở khi mất nguồn ngoài.
Ar		AR 00 đến HR 15 (16 words)	AR 0000 đến HR 1515 (256 bits)	Những bit này phục vụ cho chức năng đặc biệt như cờ và bit điều khiển.
LR		LR 00 đến LR 15 (16 words)	LR 00000 đến LR 1515 (256 bits)	Sử dụng để kết nối 1:2 với PC khác.
Timer/ couter		TC 000 đến 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 đến 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 đến DM 6599 (456 words)	---	Không thể ghi đè lên chương trình
	Cài đặt PC	Dm 6600 đến DM 6655 (%6 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 25^0C, 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".

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

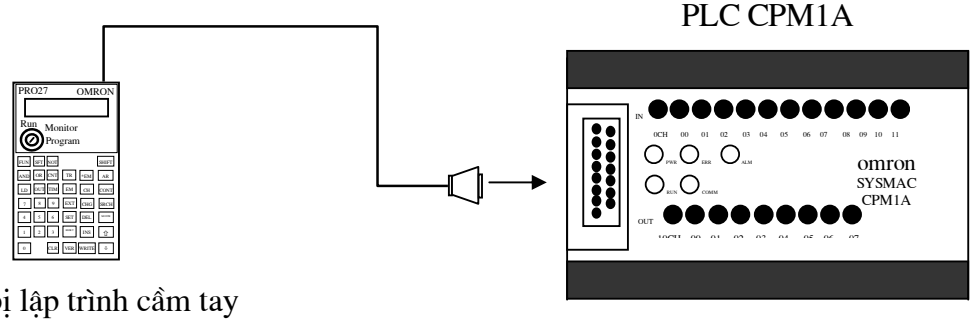
Bảng sau cho biết các bit vùng IR dùng cho module vào ra mở rộng của CPM1A và các loại module mở rộng.

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.

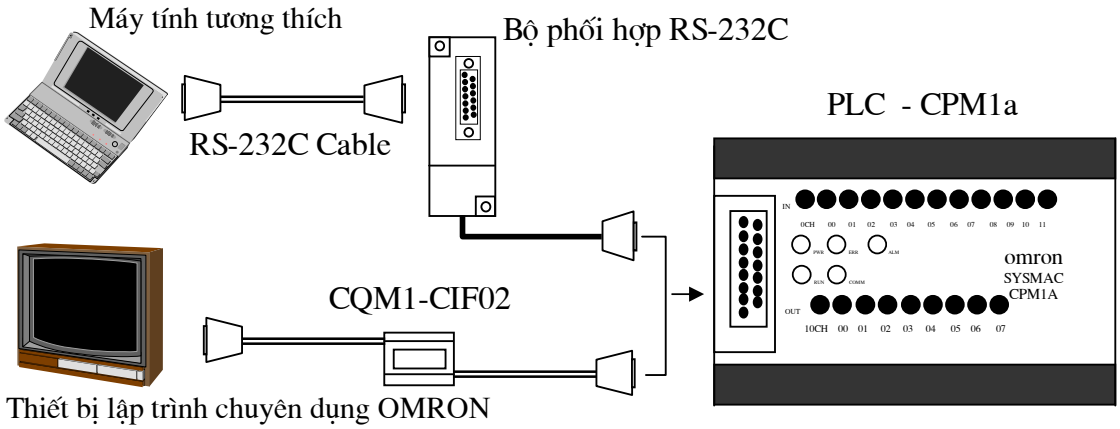
1. Kết nối với thiết bị lập trình cầm tay: Ta nối trực tiếp cáp của thiết bị cầm tay vào PLC như hình 4.2



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

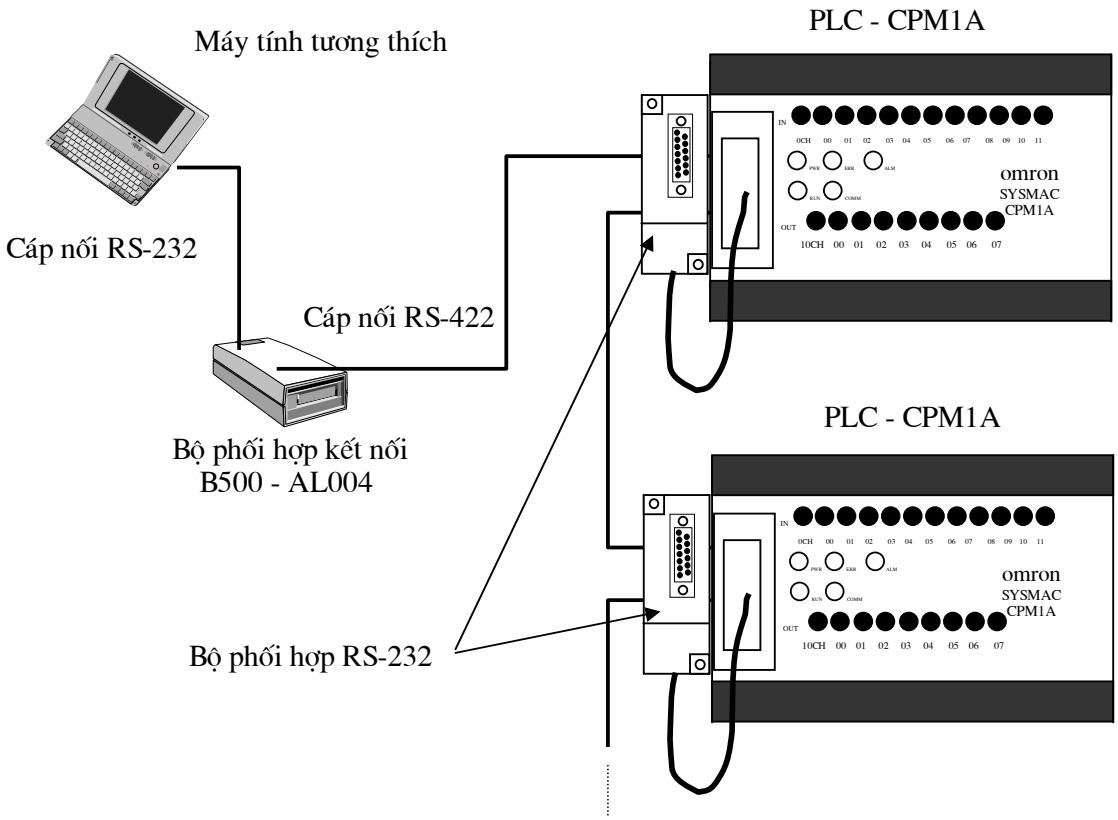
2. Kết nối với thiết bị lập trình chuyên dụng hoặc máy tính tương thích

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.



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

3. Kết nối nhiều PLC và máy tính



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

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.

2. Bảng lệnh của PLC - PCM1A

Xem phần "Bảng lệnh"

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

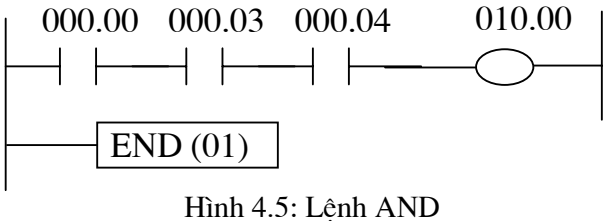
Khi lập trình phần mềm lập trình đã tự hiểu các địa chỉ trên, không cần đưa khái niệm để phân biệt vào/ra. Nếu đưa thêm khái niệm vào/ra (X/Y) máy sẽ không chấp nhận.

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

3.1. 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).

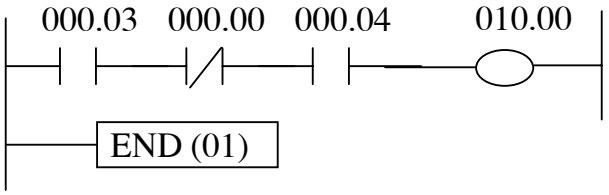
+ Chọn trạng thái MONITOR hoặc trạng thái RUN nhờ Shift + F10 hoặc biểu tượng "PLC Mode" để chạy chương trình.

Quan sát các kết quả.

3.2. Lệnh AND NOT

Dạng STL

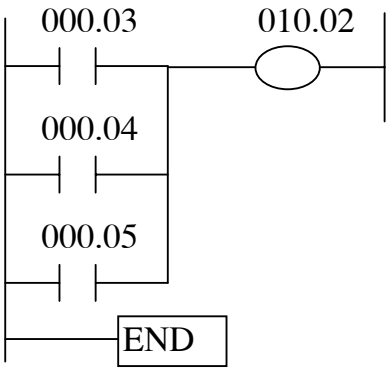
```
LD     000.03
AND NOT   000.00
AND  000.04
OUT  010.01
END
```



Hình 4.6: Lệnh AND NOT

3.3. Lệnh OR: Dạng STL

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

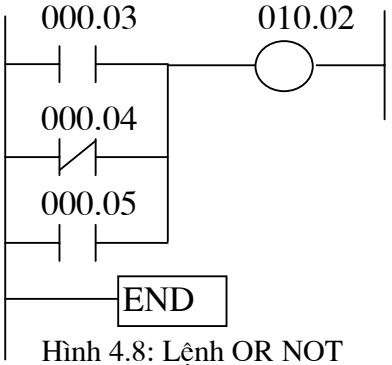


Hình 4.7: Lệnh OR

3. 4. Lệnh OR NOT

Dạng STL

```
LD     00.03
OR NOT     00.04
OR     000.05
OUT  010.02
END
```

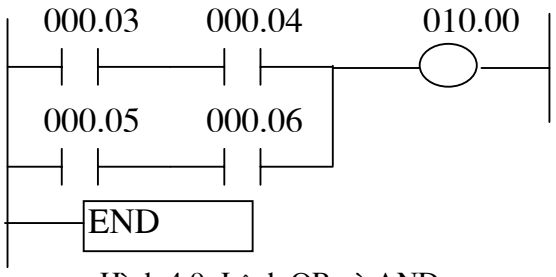


Hình 4.8: Lệnh OR NOT

3. 5. 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
```

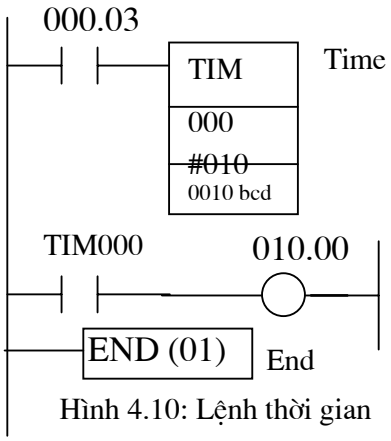


Hình 4.9: Lệnh OR và AND

3. 6. Lệnh thời gian trễ

Dạng STL

```
LD    000.03
TIM   000   #010
LD    TIM000
OUT   010.00
END
```



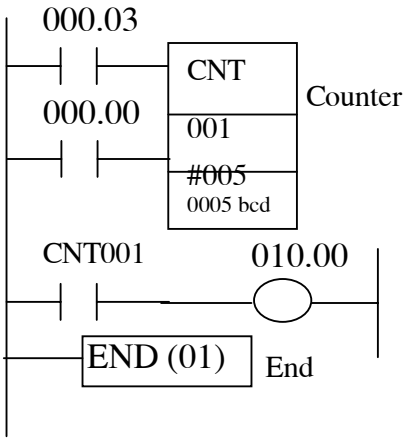
Hình 4.10: Lệnh thời gian

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.

3.7. 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 vào 000.03 đạt năm lần giá trị 1 thì bộ đếm cho giá trị ra, tức đầu ra 010.00 có giá trị 1.

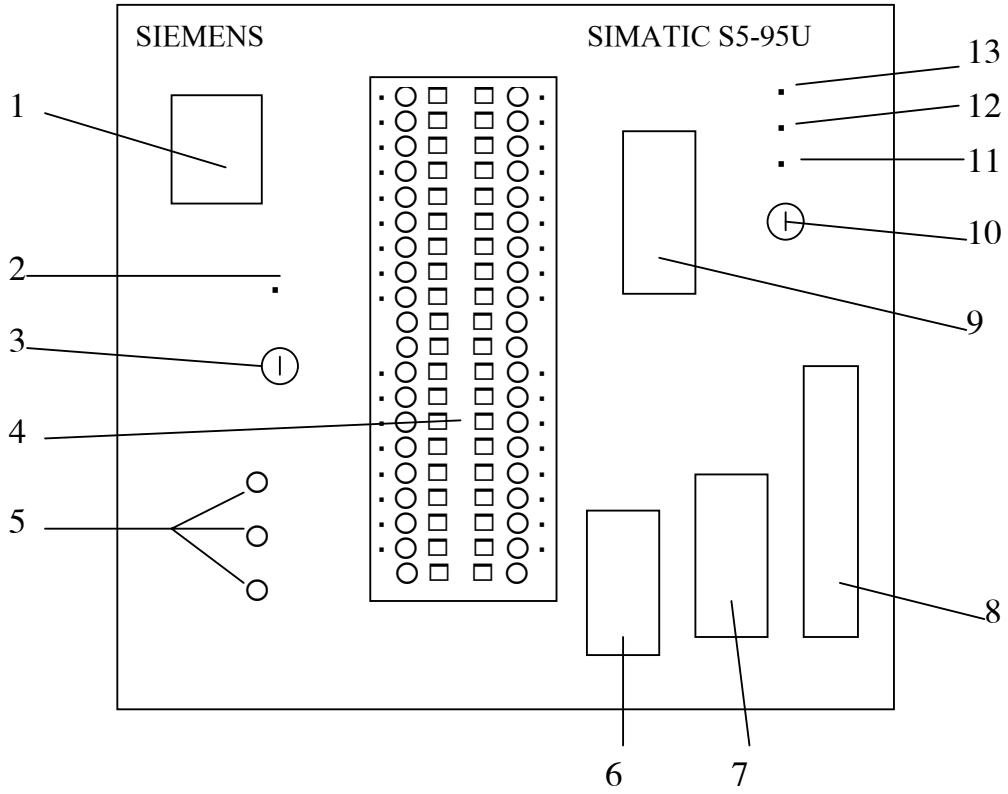
Chương 5: BỘ ĐIỀU KHIỂN PLC - S5

§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.

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

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.

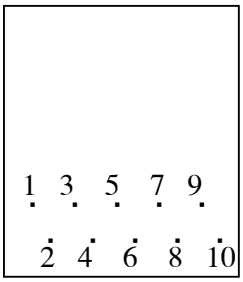
2. Các module vào ra mở rộng

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



Hình 5.2: Sơ đồ chân module mở rộng

§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.

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.

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.

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

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

Hình 5.3: Số hiệu khe và kênh trên module

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.

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 địa chỉ kênh là đã biết nó nằm ở khe nào.

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

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 module tương tự

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	ACCUM2	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	Hằng số đếm	0 đến 999
20	KF	Hằng số	-32768 đến 32677
21	KH	Hằng số dạng cơ số 16	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,	0 đến 255

		13, 21, 31, 34, 251)	
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

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.

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:

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.

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.

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

1. Nhóm lệnh logic cơ bản

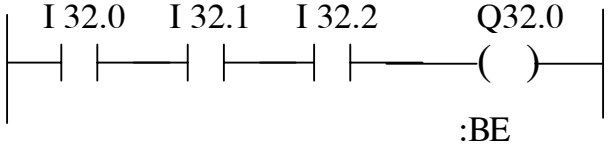
Khi thực hiện lệnh đầu tiên của một loạt phép toán logic thì nội dung của đối tượng lệnh được lấy vào sẽ được nạp ngay vào RLO (Kết quả của phép toán logic) mà không cần thực hiện phép toán.

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

1.1. Lệnh A

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

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



Hình 5.6: Lệnh A

+ Ấn Enter để trở về màn hình Output.

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

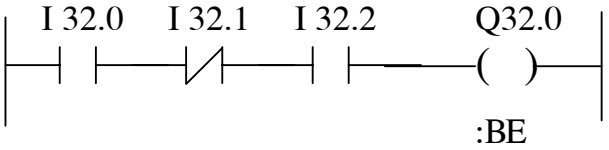
+ Ấn Shift-F7 để cất chương trình và đổ chương trình sang PLC, chọn yes để xác nhận việc đổ đè chương trình lên chương trình cũ trong PLC (khi cất thì PLC phải để ở chế độ STOP).

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

1.2. Lệnh AN

Lập trình dạng STL.

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

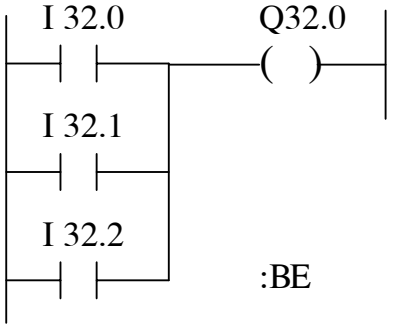


Hình 5.7: Lệnh AN

1. 3. Lệnh O

Lập trình dạng STL.

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

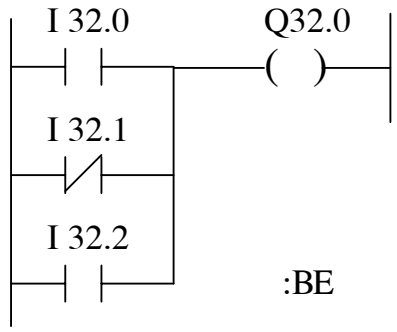


Hình 5.8: Lệnh O

1. 4. Lệnh ON

Lập trình dạng STL.

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

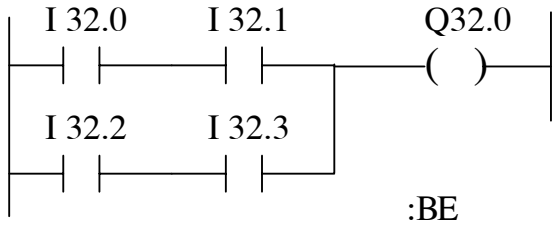


Hình 5.9: Lệnh ON

1. 5. 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
```

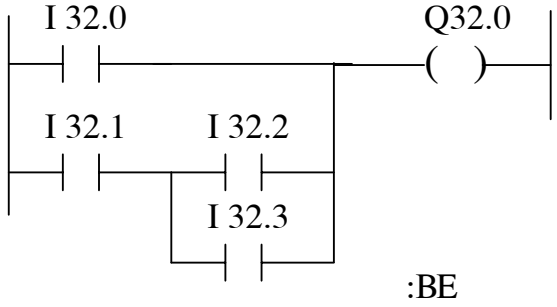


Hình 5.10: Lệnh O giữa hai lệnh A

1.6. 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 ")"

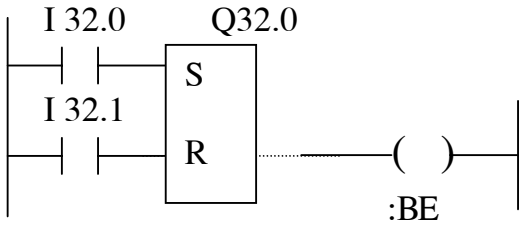
2. Nhóm lệnh set và reset

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
R     Q     32.0
NOP   0
```



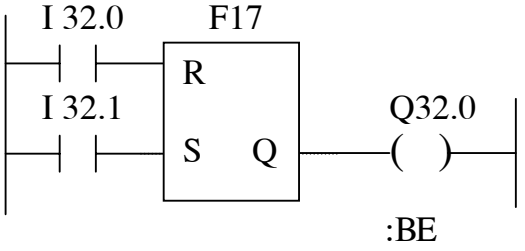
Hình 5.12: Lệnh set /reset

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.

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.

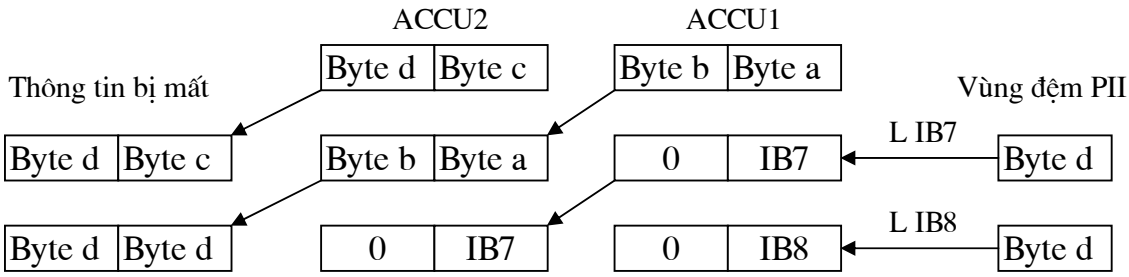
Nạp hằng số phục vụ việc xử lý chương trình.

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 trữ tạm 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:

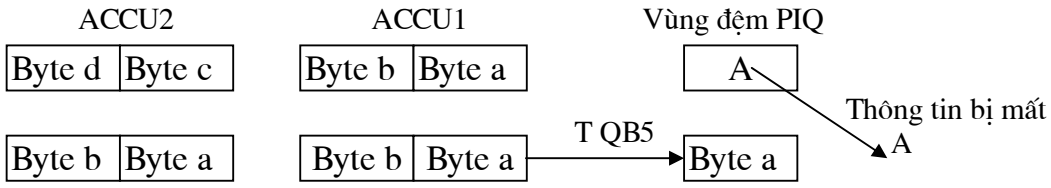
3.1. Lệnh nạp L: Nội dung của đối tượng (đơn vị byte) được chép vào ACCU1 không phụ thuộc vào RLO và RLO cũng không bị ảnh hưởng. Nội dung trước đó của ACCU1 được chuyển dịch sang ACCU2, nội dung cũ của ACCU2 sẽ bị mất.



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.

3.2.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

3.3. 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.

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.

4.1. 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 $\Delta t = 40 \times 1s = 40s$.

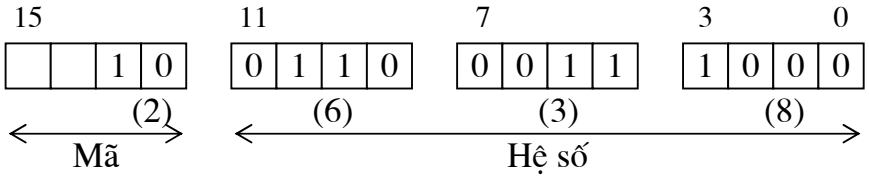
Với mã càng nhỏ thì giá trị thời gian càng chính xác, vì vậy nên dùng mã nhỏ.

- *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
```

4.2. 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.

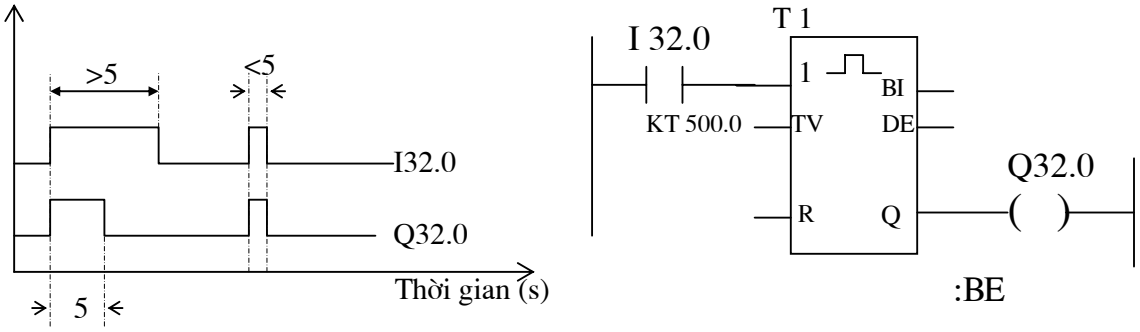
4.3. Các lệnh

1. Bô 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
```



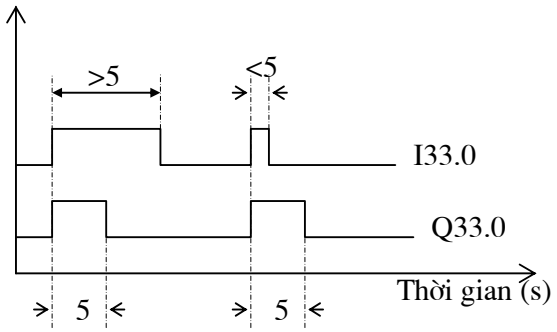
Hình 5.17: Giản đồ thời gian và dạng LAD Lệnh SP

2. Bô thời gian mở rông 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     T2
=     Q     33.0
BE
```



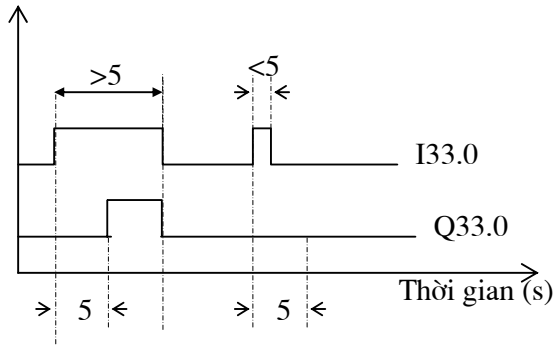
Hình 5.18: Giản đồ thời gian lệnh SE

3. Bô thời gian bắt đầu trễ SD

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

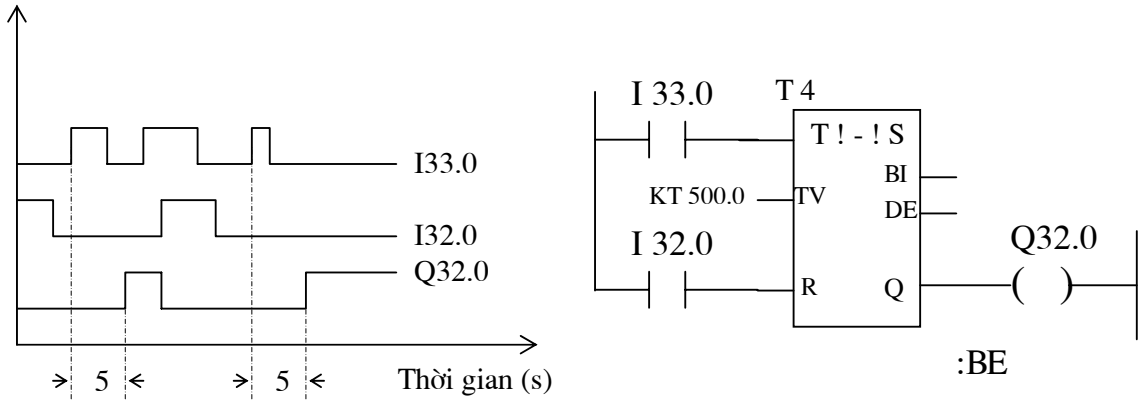


Hình 5.19: Giản đồ thời gian lệnh SD

4. <u>Bộ thời gian bắt đầu trễ lưu trữ SS</u>

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
```

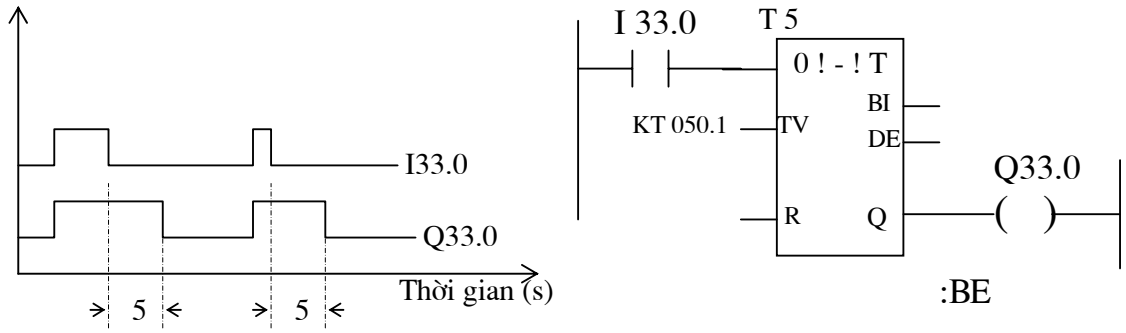


Hình 5.20: Giản đồ thời gian và dạng LAD lệnh SS

5. 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
```



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

5. Nhóm lệnh đếm

5.1. 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

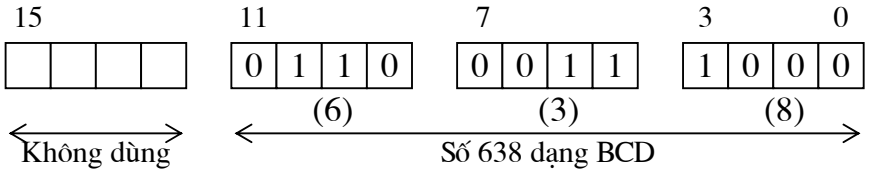
Số đếm từ 0 đến 999

- *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 DW2

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:



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.

5.2. 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.

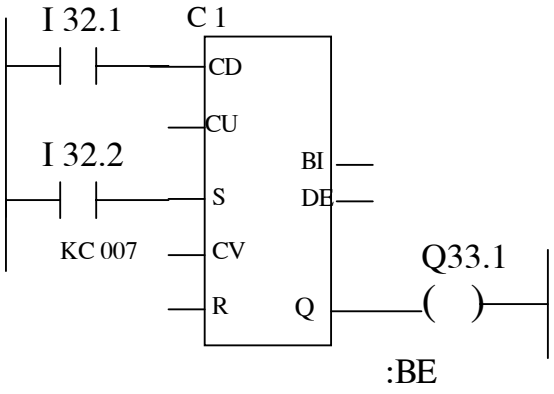
+ *Xuất ra trạng thái bộ đếm hiện hành*: Có thể dùng lệnh L và LD để đưa trạng thái bộ đếm hiện hành vào ACCU1 để xử lý sau này, lệnh L dùng cho số nhị phân, lệnh LD dùng cho số BCD.

4.3. Các lệnh

1. <u>Lệnh đếm xuống CD</u>

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
```



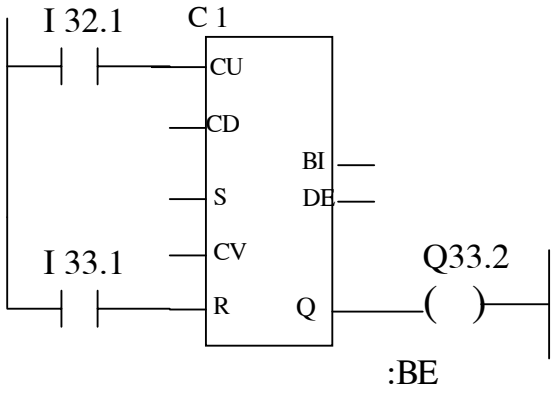
Hình 5.23: Lệnh đếm xuống CD

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

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

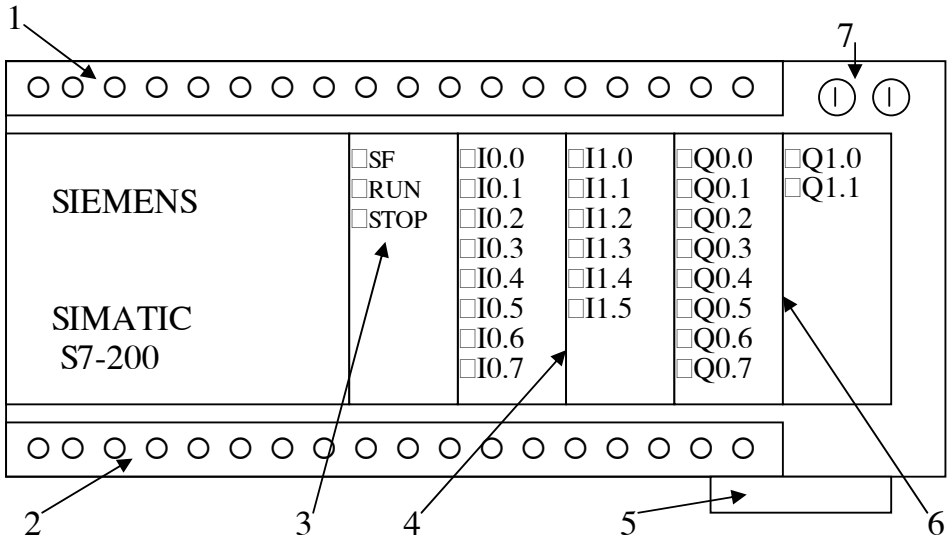
Chương 6: BỘ ĐIỀU KHIỂN PLC - S7-200

§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.

1. Đơn vị cơ bản

1.1. 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

+ RUN: cho phép PLC thực hiện chương trình trong bộ nhớ. PLC sẽ tự 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.

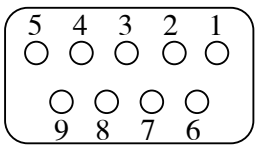
+ TERM: cho phép PLC tự quyết định một chế độ làm việc (hoặc RUN hoặc STOP).

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 $270^{0.}$.

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

1.2. 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 190^h 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 50^h khi PLC bị mất nguồn cung cấp.

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	I2.0	I3.0	AIW0	Q3.0	AIW8
I0.1	Q0.1	I2.1	I3.1	AIW2	Q3.1	AIW10
I0.2	Q0.2	I2.2	I3.2	AIW3	Q3.2	AIW12
I0.3	Q0.3	I2.3	I3.3	AIW4	Q3.3	
I0.4	Q0.4		I3.4		Q3.4	AQW4
I0.5	Q0.5	Q2.0	I3.5	AQW0	Q3.5	
I0.6	Q0.6	Q2.1	I3.6		Q3.6	
I0.7	Q0.7	Q2.2	I3.7		Q3.7	
I1.0	Q1.0	Q2.3				
I1.1	Q1.1					
I1.2						
I1.3						
I1.4						
I1.5						

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:

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.

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.

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	Là 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 đến 127
4	HSC	Bộ đếm tốc độ cao		0 đến 2
5	AW	Bộ đệm cổng vào tương tự	0 ÷30	0 đến 30
6	AQW	Bộ đệm cổng ra tương tự	0 ÷ 30	0 đến 30
7	T	Bộ thời gian	0 ÷63	0 đến 127

§6.3. Chương trình của S7-200

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

```
Main Program
⋮
MEND
```

```
SBR 0     Chương trình con thứ nhất
⋮
RET
```

```
SBR n     Chương trình con thứ n +1
⋮
RET
```

```
INT 0     Chương trình xử lý ngắt thứ nhất
⋮
RETI
```

```
INT n     Chương trình xử lý ngắt thứ n +1
⋮
RETI
```

Hình 6.4: Cấu trúc chương trình của S7-200

kết thúc chương trình chính MEND.

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.

2. Bảng lệnh của S7-200

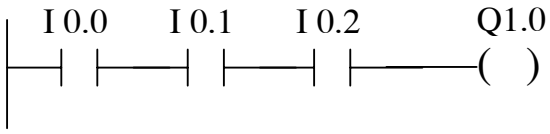
Xem phần phụ lục.

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

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
```

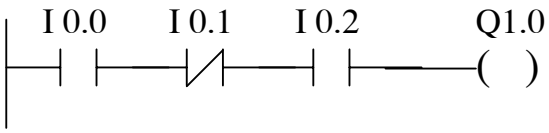


Hình 6.5: Lệnh LD và A

2. Lệnh AN

Lập trình dạng STL.

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

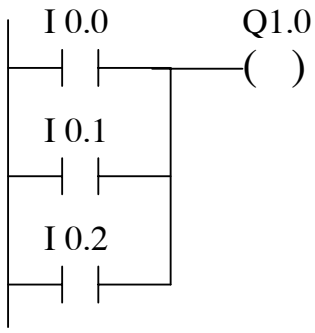


Hình 6.6: Lệnh AN

3. Lệnh O

Lập trình dạng STL.

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

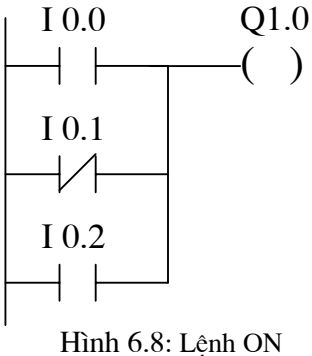


Hình 6.7: Lệnh O

4. Lệnh ON:

Lập trình dạng STL.

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

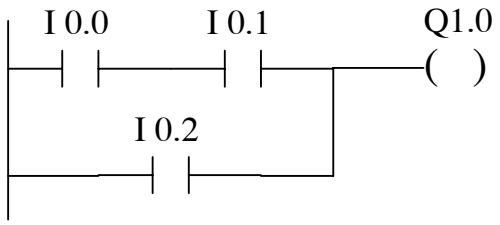


Hình 6.8: Lệnh ON

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
```

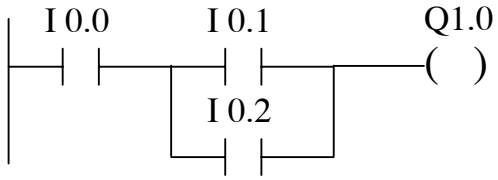


Hình 6.9: Lệnh OLD

6. Lệnh ALD

Lập trình dạng STL.

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

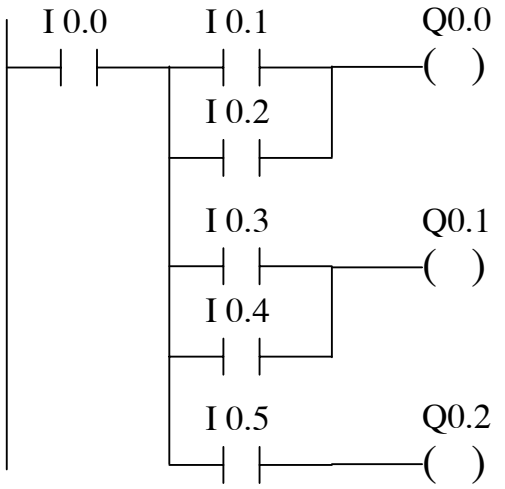


Hình 6.10: ALD

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: LPS, LRD, LPP

Chương 7: BỘ ĐIỀU KHIỂN PLC - S7-300

§7.1. Cấu hình cứng

1. Cấu tạo của họ PLC- S7-300

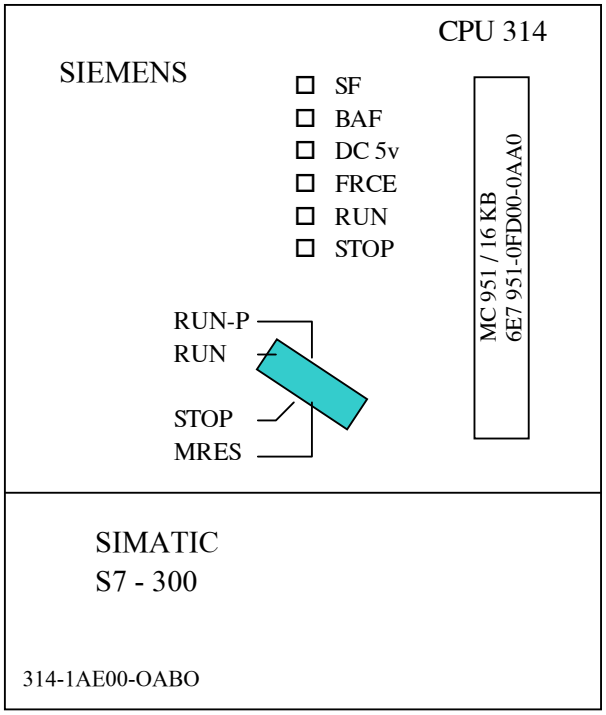
PLC Step S7-300 thuộc họ Simatic do hãng Siemens sản xuất. Đây là loại PLC đa khối. Cấu tạo cơ bản của loại PLC này là một đơn vị cơ bản (chỉ để xử lý) sau đó ghép thêm các module mở rộng về phía bên phải, có các module mở rộng tiêu chuẩn. Những module ngoài này bao gồm những đơn vị chức năng mà có thể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

1.1. 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.



Hình 7.1: Hình khối mặt trước CPU-314

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.

1.2. 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.

2. Địa chỉ và gán địa chỉ

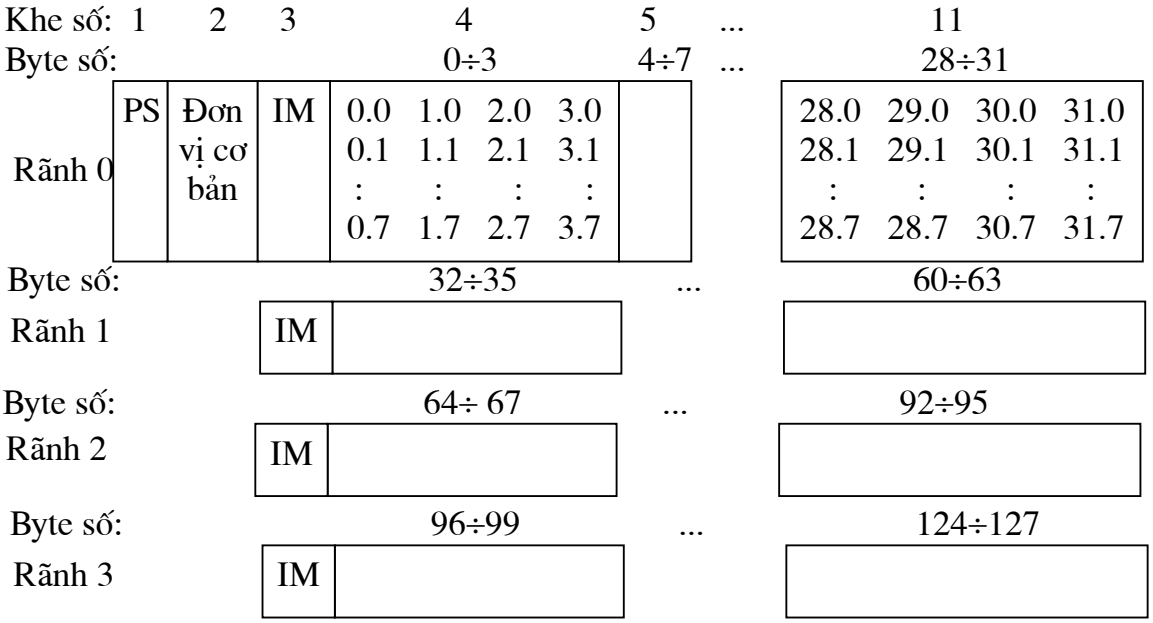
Trong PLC các bộ phận cần gửi thông tin đến hoặc lấy thông tin đi đều phải có địa chỉ để liên lạc. Địa chỉ là con số hoặc tổ hợp các con số đi theo sau chữ cái. Chữ cái chỉ loại địa chỉ, con số hoặc tổ hợp con số chỉ số hiệu địa chỉ.

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.

①, Địa chỉ vào ra trên 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.

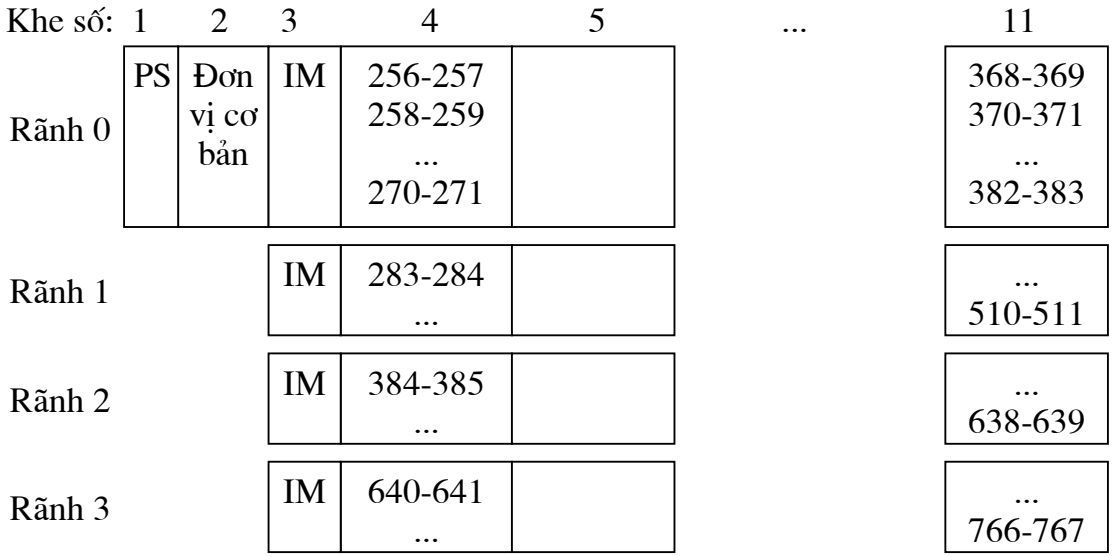


Hình 7.2: Địa chỉ khe và kênh trên module số

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 7.2.

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.



Hình 7.3: Địa chỉ của module tương tự

②, Địa chỉ vào ra trên 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.

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

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

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-tháng-ngày	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

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).

2. Bảng lệnh của S7-300

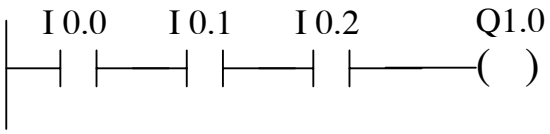
Xem phần phụ lục

§7.4. Lập trình một số lệnh cơ bản

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
```

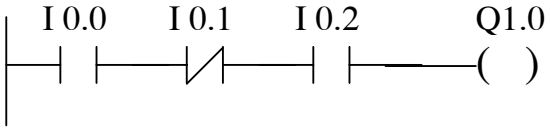


Hình 7.4: Lệnh LD và A

2. Lệnh AN

Lập trình dạng STL.

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

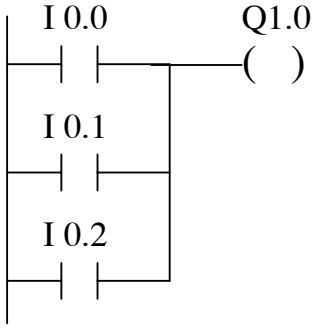


Hình 7.5: Lệnh AN

3. Lệnh O

Lập trình dạng STL.

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

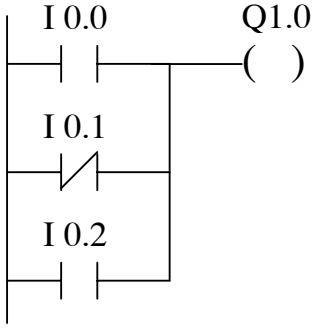


Hình 7.6: Lệnh O

4. Lệnh ON

Lập trình dạng STL.

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

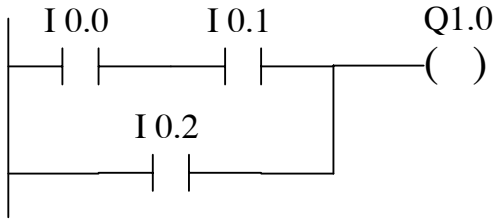


Hình 7.7: Lệnh ON

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
```

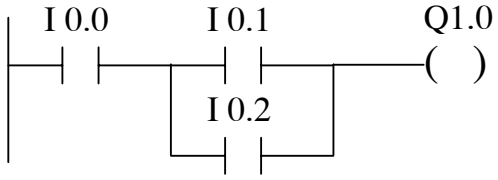


Hình 7.8: Lệnh OLD

6. Lệnh ALD

Lập trình dạng STL.

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

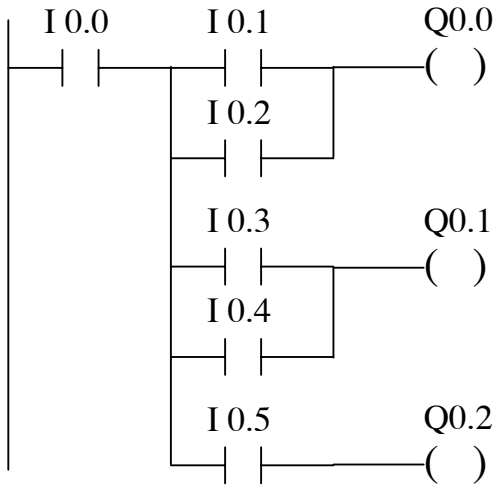


Hình 7.9: ALD

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 7.10: LPS, LRD, LPP

Phụ lục 1

CÁC PHẦN MỀM LẬP TRÌNH PLC

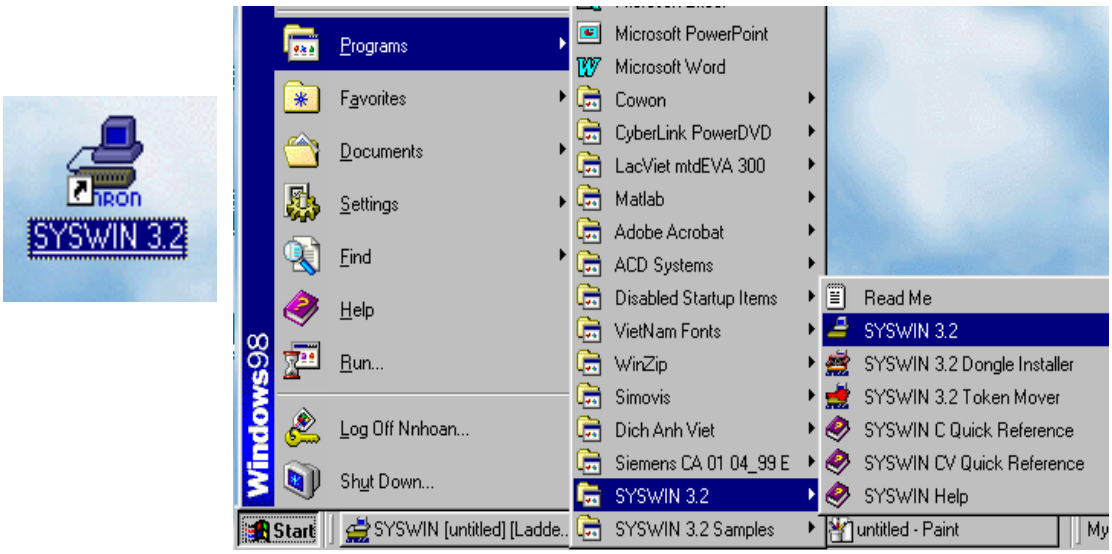
I. Lập trình cho OMRON

1. Phần mềm SYSWIN (cho OMRON)

1.1. Khởi động

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

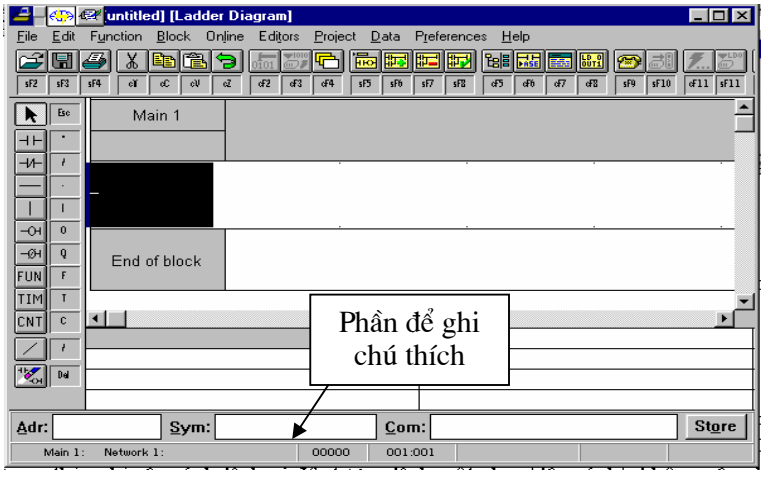
2. Khởi động phần mềm SYSWIN từ biểu tượng hoặc từ file chương trình như hình P.1. Cửa sổ màn hình ban đầu có dạng như hình P.2. Trong cửa sổ có 2 thanh công cụ hỗ trợ cho quá trình soạn thảo chính là:



Hình P.1: Khởi động phần mềm SYSWIN

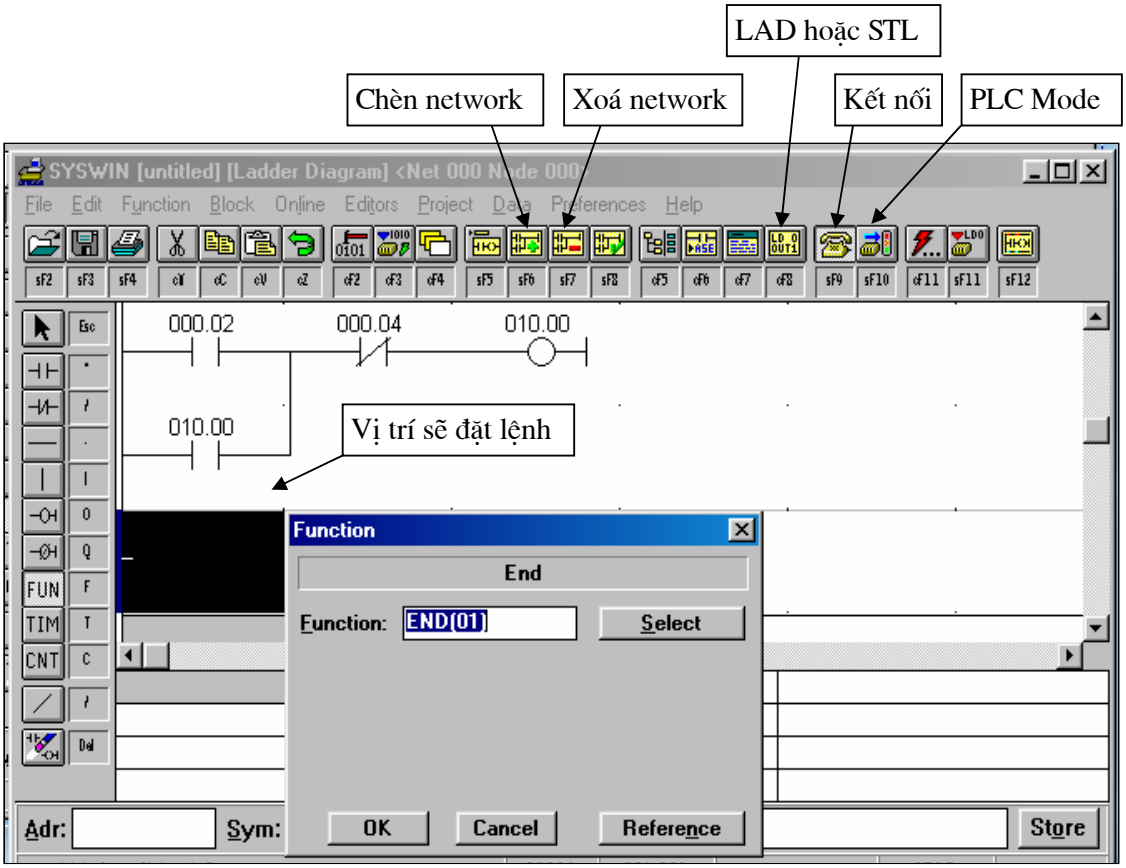
- Thanh trên: ngoài một số chức năng như soạn thảo văn bản bình thường còn một số chức năng để soạn thảo lệnh như chỉ ra trên hình P.3.

- 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 P.2: Màn hình ban đầu

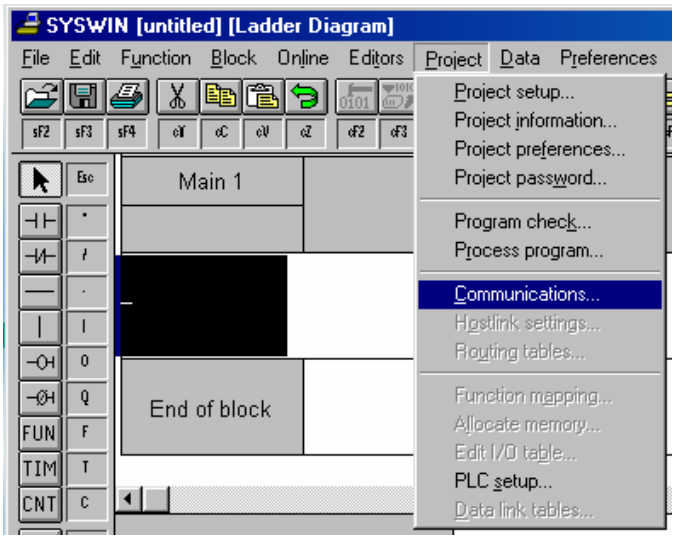
3. Kiểm tra một số điều kiện trước khi lập trình:



Hình P.3: Một số chức năng chính

+ 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 P.4. (đường dẫn Project \ Communications).

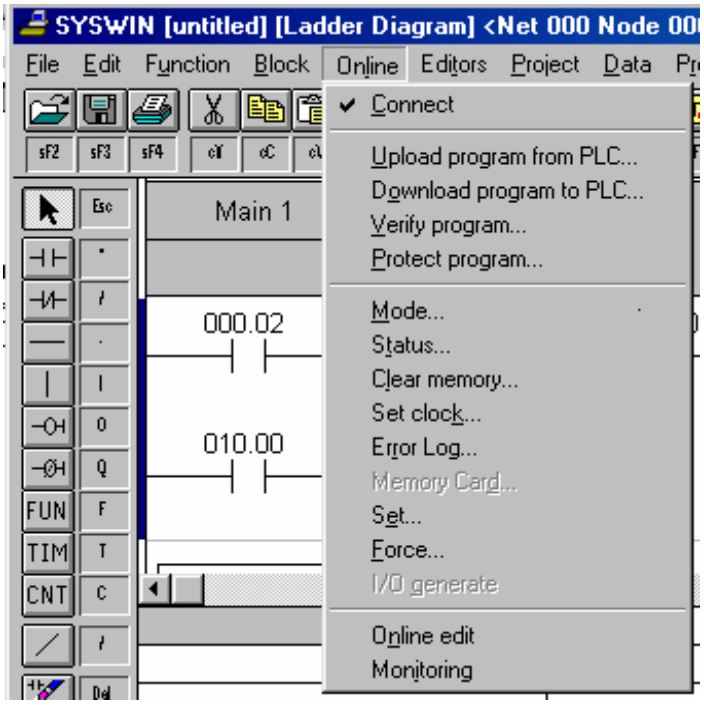


Hình P.4: Khai báo cổng ghép nối

1.2. 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).

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 chỉ ra trên hình P.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 mới từ biểu tượng như chỉ ra trên hình P.3.
6. Nếu soạn sai Network nào thì đánh dấu và xoá Network đó từ biểu tượng hình P.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 P.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 P.5.



Hình P.5: Đổ chương trình sang PLC

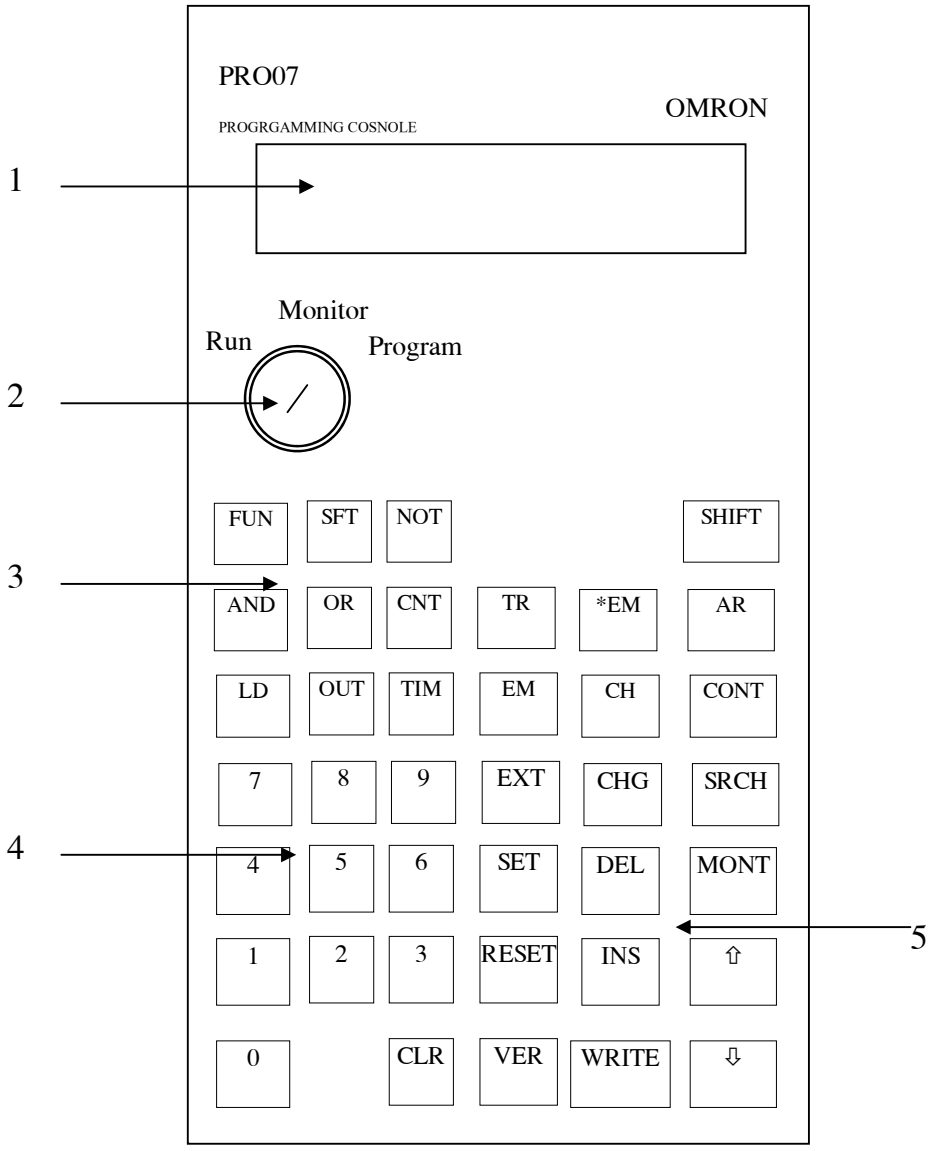
<u>Chú ý:</u> Khi đổ chương trình sang PLC thì PLC phải đang ở trạng thái MONITOR hoặc trạng thái PROGRAM (STOP/PRG). Muốn chuyển đổi các trạng thái trên thì chọn Shift + F10 hoặc biểu tượng "PLC Mode" như hình P.3.

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

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

2.1. Cấu tạo thiết bị lập trình cầm tay

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



Hình P.6: Ghép nối PLC với thiết bị lập trình cầm tay

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).

3. Các phím lệnh

4. Các phím số.

5. Các phím hàm.

2.2. 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

2.3. 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.

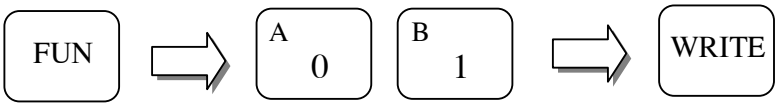
3. Các lệnh được vào theo thứ tự:

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

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

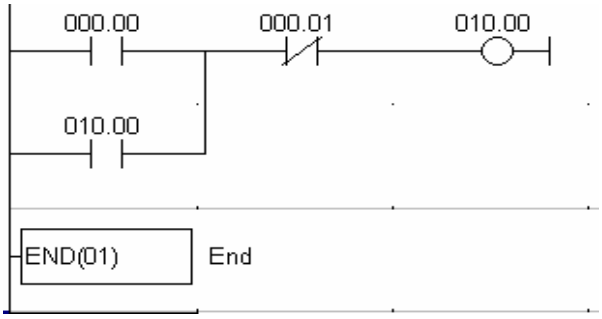
+ Kết thúc một lệnh là WRITE (viết vào PLC).

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



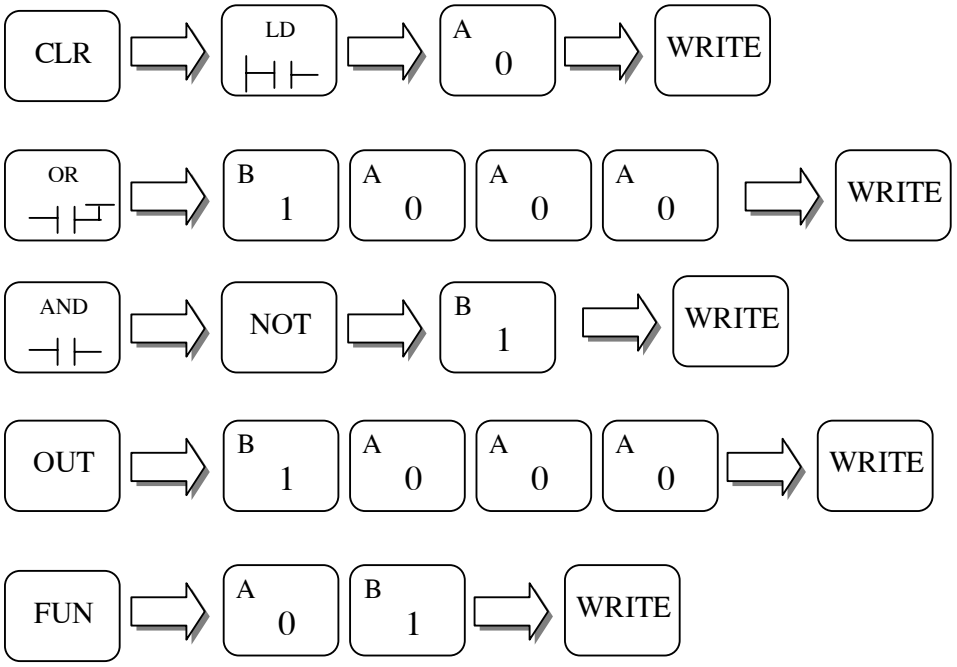
Ví dụ: Chương trình của một mạch tự duy trì dạng LAD và STL như hình P.7:

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



Hình P.7: Mạch tự duy trì

Cách vào chương trình hình P.7 như sau:



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

II. Lập trình cho PLC - S5

Sử dụng phần mềm Step5 for Win.

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.

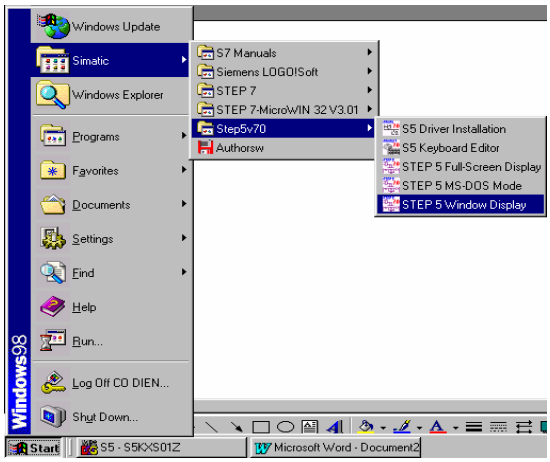
2. Chạy trình Step5 từ fite chương trình như hình P.8.

Màn hình chế độ bắt đầu có dạng như hình P.9.

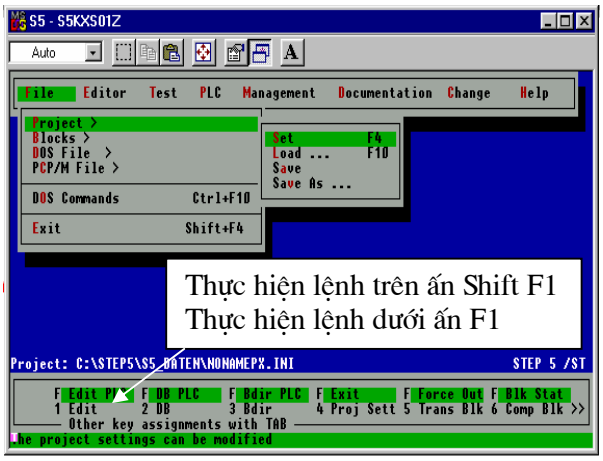
3. Vào File \ Project \ Set (phần này có thể đặt nhiều tham số, xem phần đặt tham số trang 94). Cần đặt 3 tham số cơ bản.

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

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



Hình P.8: Khởi động Step 5



Hình P.9: Màn hình ban đầu

+ 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 P.10.

Trong đó:

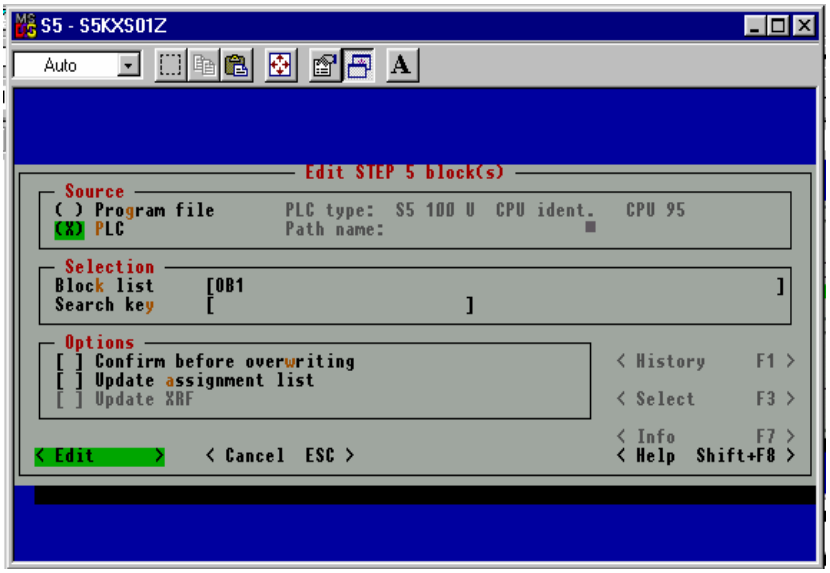
Block list: Vào tên của khối hoặc nhiều khối để soạn thảo.

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.

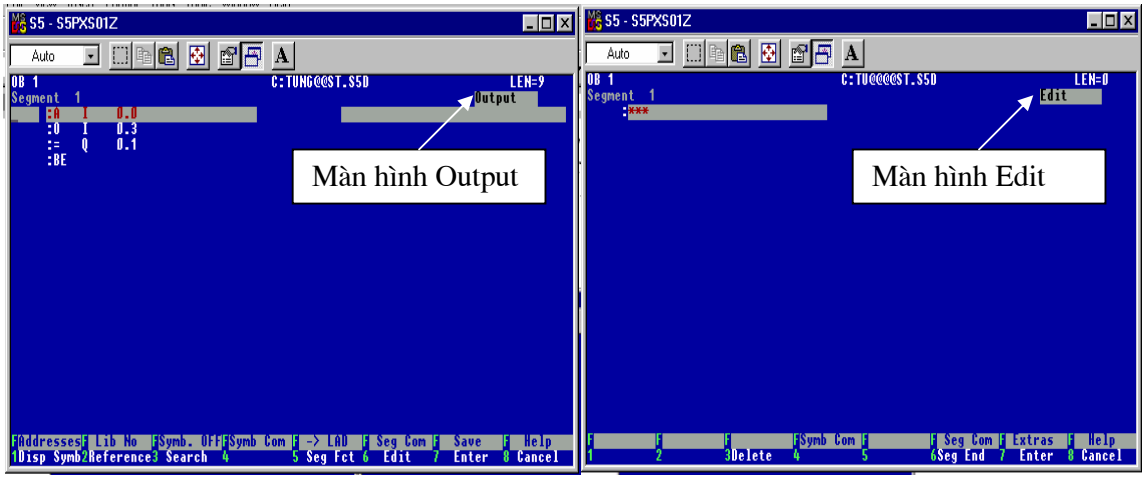
Update assignment: Nếu được chọn thì fite biểu tượng *ZO.INI thay đổi thì fite nguồn *ZO.SEQ cũng được điều chỉnh, nếu không chọn thì fite nguồn *ZO.SEQ không được điều chỉnh.

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 P.10 sau đó chọn Edit (ấn Enter), nếu làm việc với file mới thì máy tự động vào luôn màn hình Edit như hình P.11b, nếu làm việc với file cũ thì máy vào màn hình Output như hình P.11a.



Hình P.10: Màn hình trước soạn thảo



a, b,

Hình P.11: Màn hình soạn thảo

Trong đó: hình P.11a

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).

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.

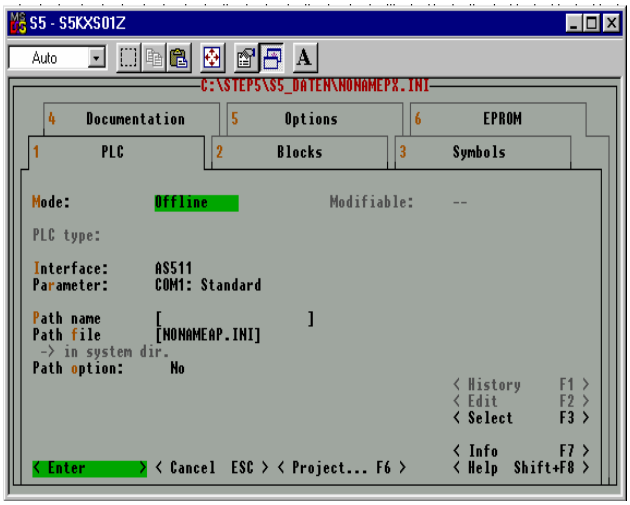
+ *Info F7*: Hiện thông tin về vùng hiện tại mà tại đó có con trỏ.

+ *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.

*Trang 1 (PLC): như hình P.12



Hình P.12: Trang 1

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

+ *PLC type*: Loại PLC

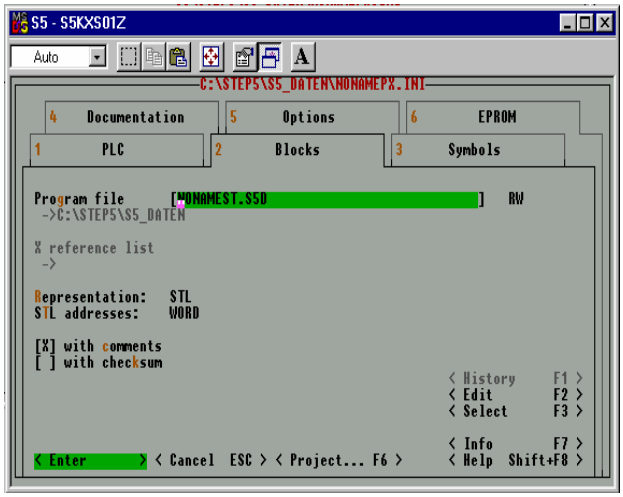
+ *Interface*: Chọn giao diện.

+ *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.

*Trang 2 (Blocks): như hình P.13



Hình P.13: Trang 2

+ *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.

*Trang 3 (Symbols): như hình P.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 từ 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.

*Trang 4 (Documetation): như hình P.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.

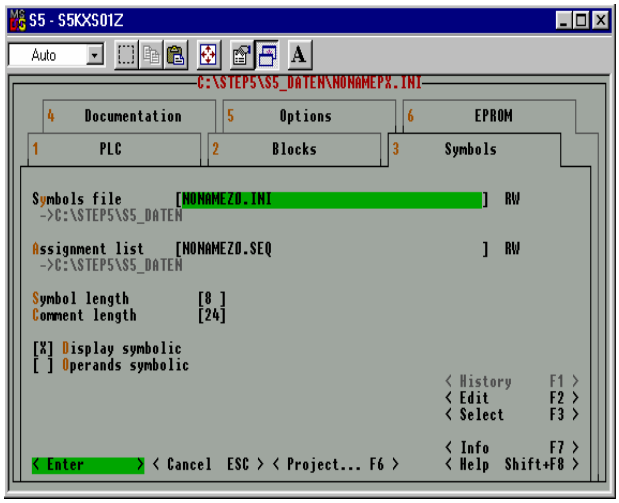
+ *Printer file*: Đặt tên file chưa thông tin về tham số in được chọn trong menu Documentation như kích cỡ giấy, số dòng trong mỗi trang in, cổng giao tiếp với máy in...

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

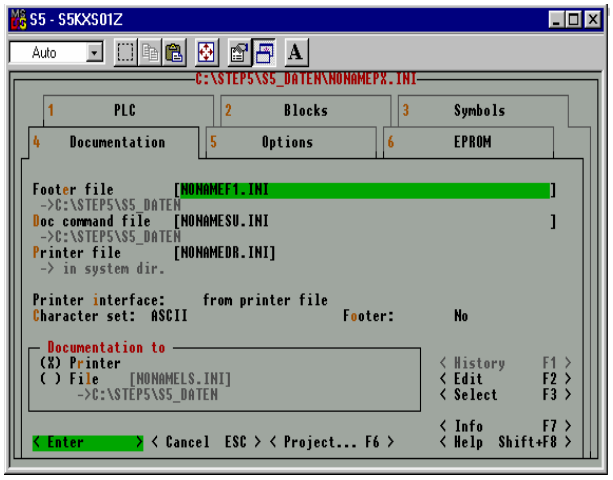
+ *Documetation to*: Đặt chế độ làm việc cho phép in tài liệu.

*Trang 5 (Options): hình P.16

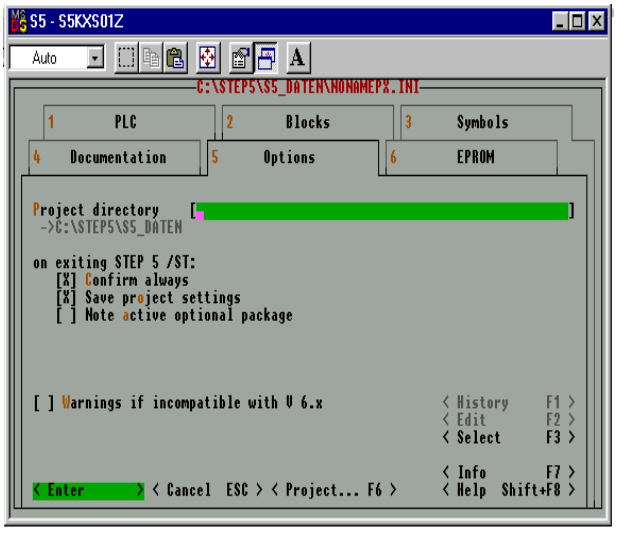
+ *Project directory*: Định thư mục làm việc.



Hình P.14: Trang 3



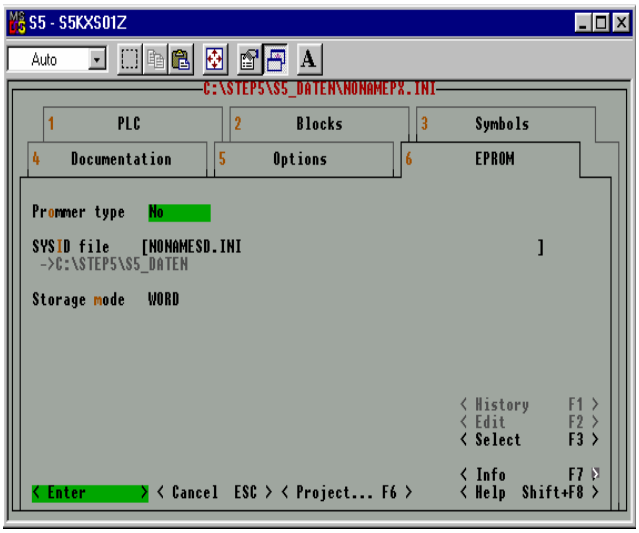
Hình P.15: Trang 4



Hình P.16: Trang 5

*Trang 6 (EFROM): như hình P.17

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



Hình P.17: Trang 6

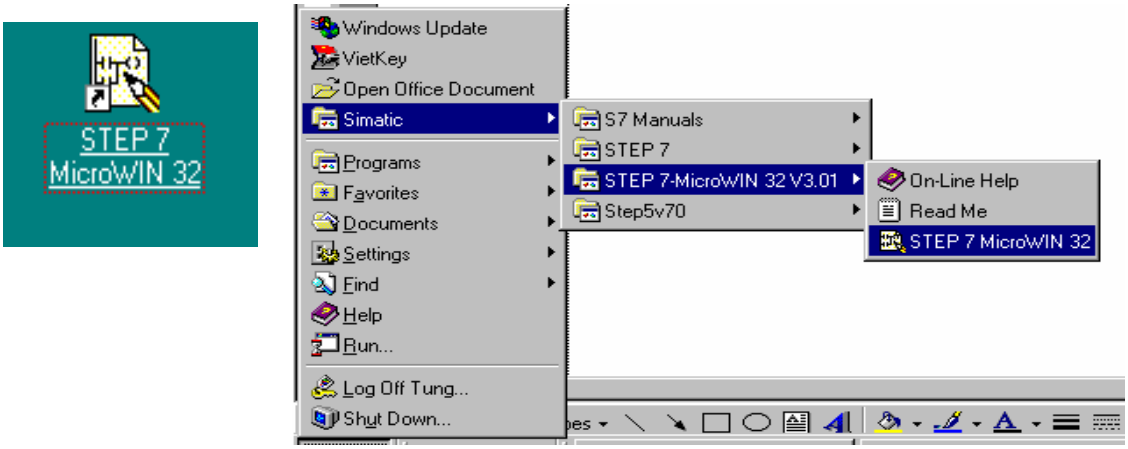
III. Lập trình cho PLC - S7-200

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 P.18.



Hình P.18: Biểu tượng và đường dẫn file chương trình Step7

màn hình chế độ bắt đầu có dạng như hình P.19.

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).

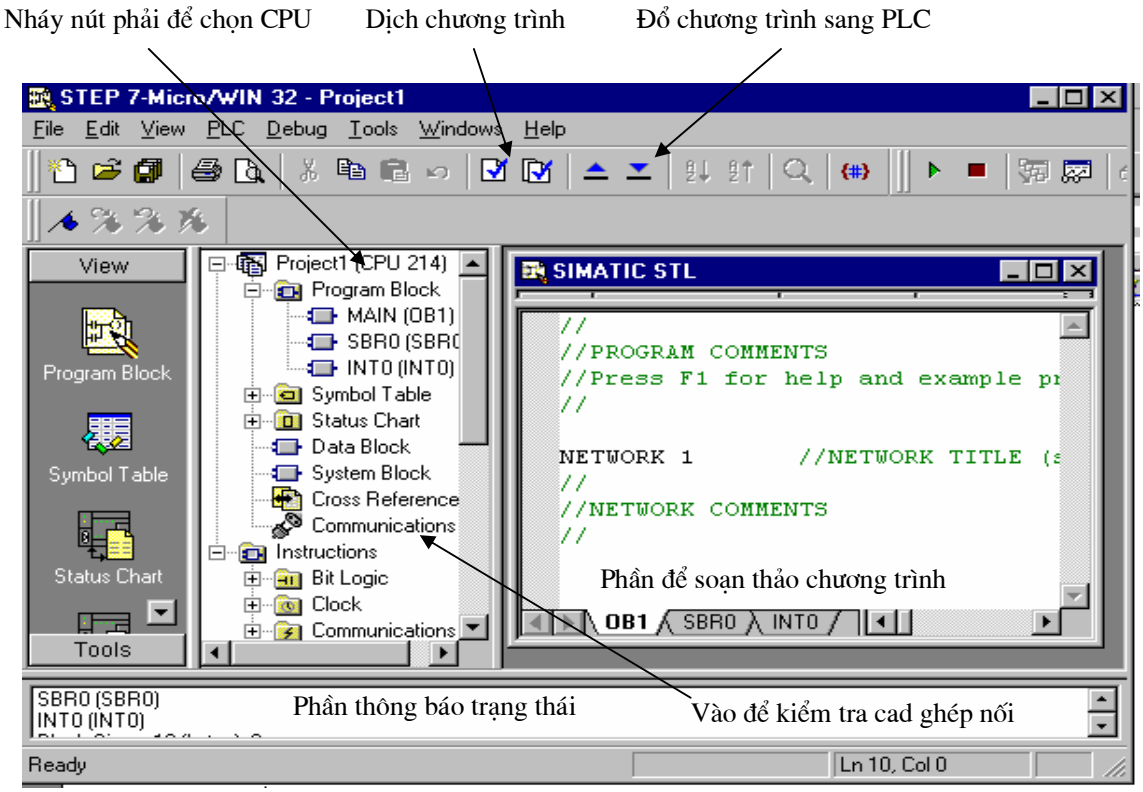
6. Tiến hành soạn thảo chương trình theo STL (nếu soạn thảo chương trình theo LAD thì có thể sử dụng các khâu, khối phía trái màn hình soạn thảo). Khi soạn thảo chỉ cần cách lệnh và đối tượng lệnh một nhịp (dấu cách), không cần chú ý chữ in và chữ thường, máy sẽ tự dịch và chỉnh chữ cho phù hợp. Trong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



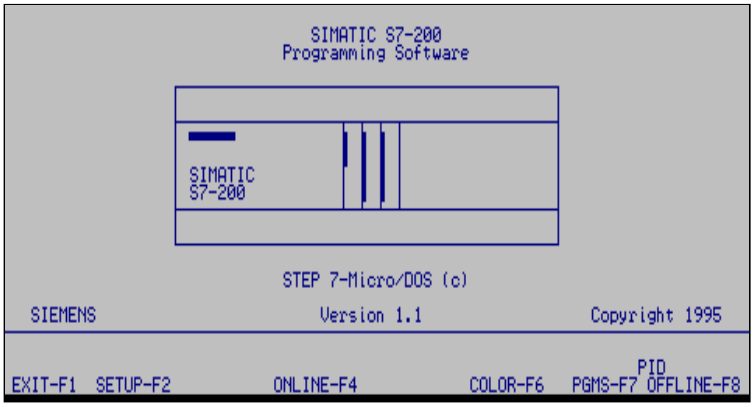
Hình P.19: Màn hình soạn thảo

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 P.20.



Hình P.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 P.21:

EXIT-F1	INSNW-F2	DELLN-F4	INSLN-F5	DELFLD-F6	UNDO-F8

Hình P.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.

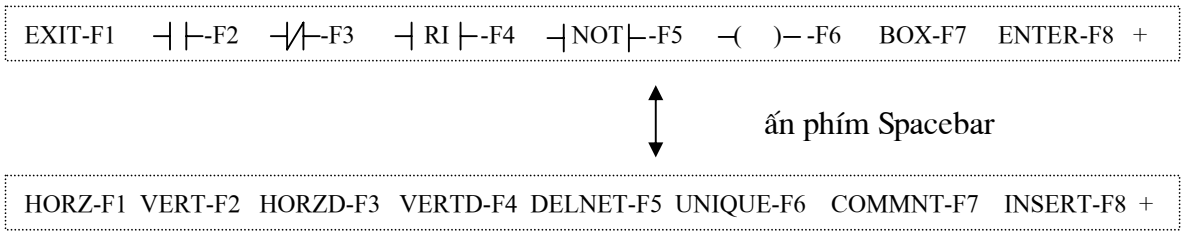
DELLN-F4: Xóa một dòng có con trỏ.

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 P.22: dấu cộng ở cuối dòng thể hiện thư mục vẫn còn cần ấn phím Spacebar để chuyển đổi.



Hình P.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 kể 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.

IV. Lập trình cho PLC - S7-300

Sử dụng phần mềm S7-300.

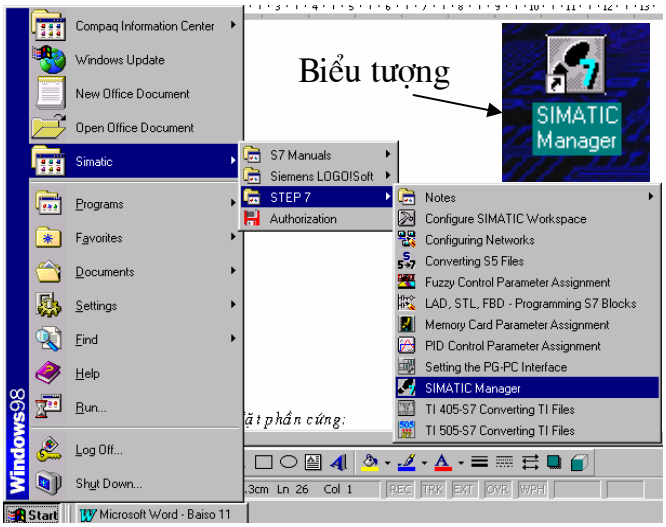
1. Khởi động:

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.

2. Khởi động phần mềm Step7 từ biểu tượng hoặc từ file chương trình như hình P.23.

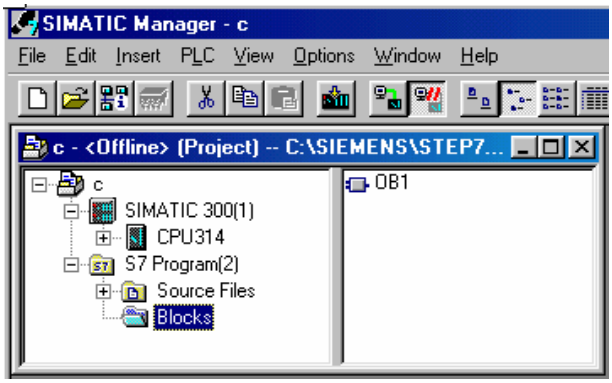
1.2. Cài đặt phần cứng:

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



Hình P.23: Đường dẫn khởi động Step 7

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 P.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.



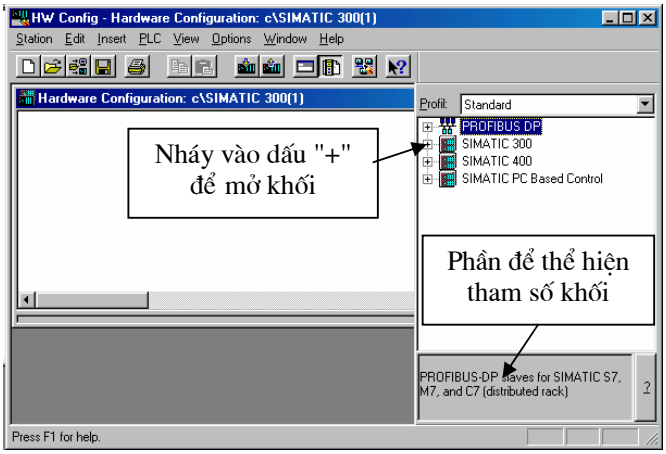
Hình P.24: Cấu trúc chương trình Step 7

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 P.25.

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



Hình P.25: Hướng dẫn cài đặt phần cứng

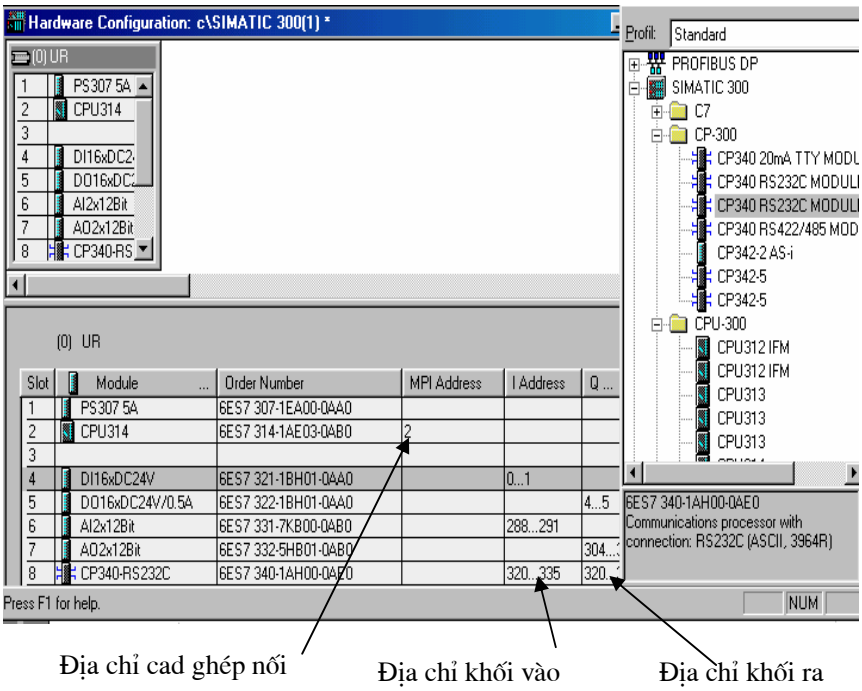
khối thí nghiệm). Các khối thực trên PLC như trên hình P.26.

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 P.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.



Hình P.26: Các khối đã được chọn

+ *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 P.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.

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.

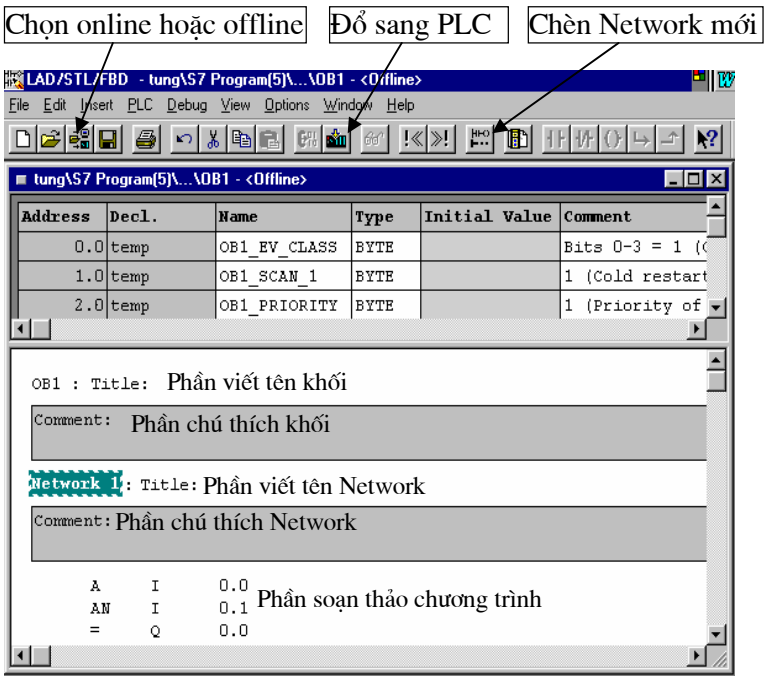
4. Mở thư mục Blocks, nếu cần thì chèn thêm các khối (Blocks) cần thiết khác cho chương trình từ Insert / S7 Blocks.

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 P.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 nếu cần.

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.



Hình P.27: Màn hình soạn thảo

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

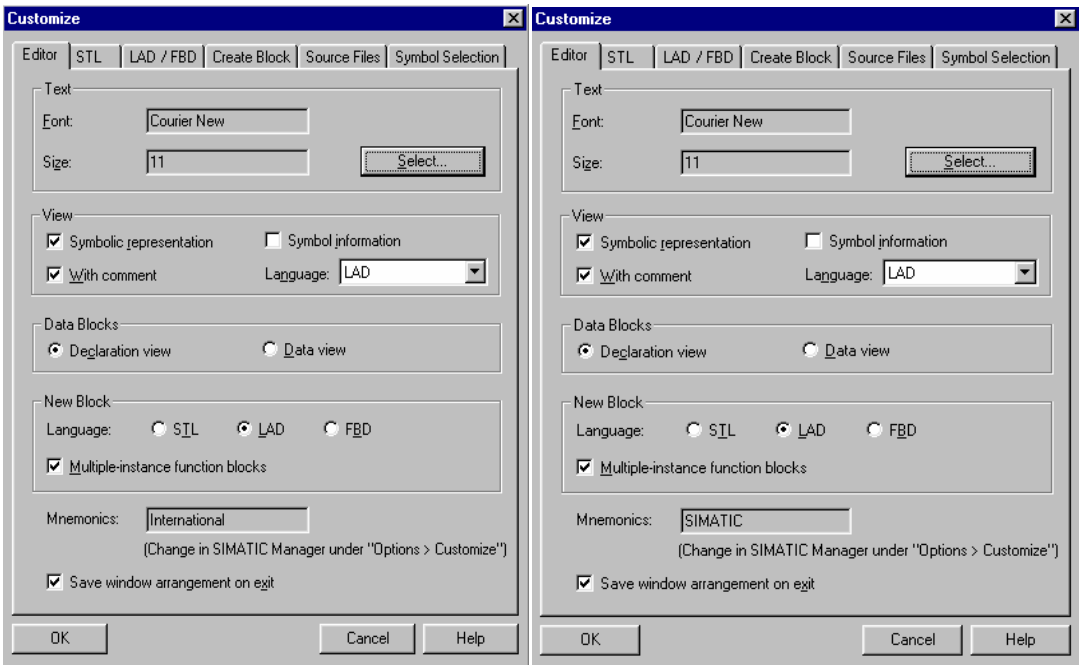
10. Xem lại dạng LAD hoặc FBD từ View / LAD hoặc View / FBD.

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:

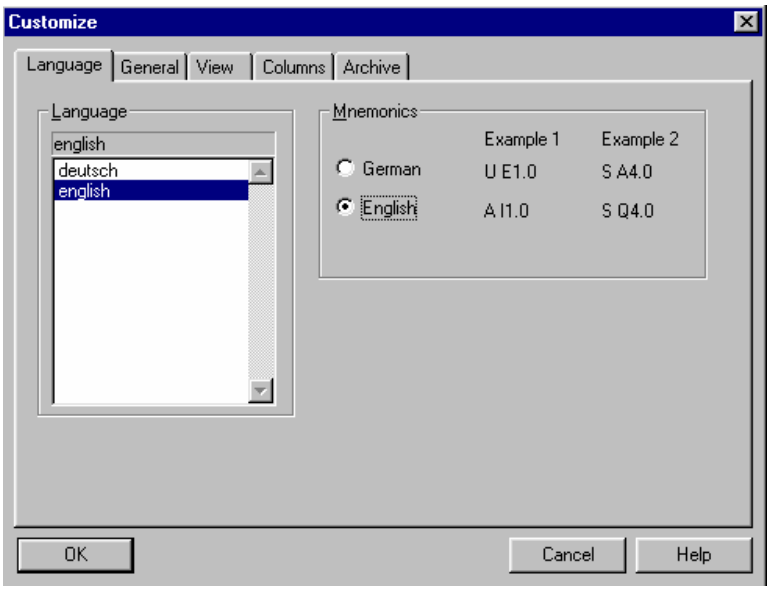
+ Từ màn hình soạn thảo như hình P.27 chọn **Options/Customize**... ta được cửa sổ như hình P.28.

+ Trong cửa số Editor hình P.28, hộp kiểm Mnemonics phải là Internectiona như hình P.28a. nếu trong hộp kiểm Mnemonics là SMATIC như hình P.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 P.24 và tiến hành các bước:



a, Hình P.28 b,

+ Từ màn hình P.24 chọn **Options/Customize**... ta được cửa sổ của màn hình Customize như hình P.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 P.29 sau đó nhấn OK.



Hình P.29

Phụ lục 2

BẢNG LỆNH CỦA CÁC PHẦN MỀM PLC

I. 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.

TT	Tên lệnh	Mô tả
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.
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

TT	Tên lệnh	Mô tả
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.
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.

TT	Tên lệnh	Mô tả
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).
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.

TT	Tên lệnh	Mô tả
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 với giá trị được chuyển đổi.

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.

TT	Tên lệnh	Mô tả
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.
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 - Tây đức)

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		
1	)	Dùng để đóng ngoặc biểu thức đã mở ngoặc trước đó, lệnh này không có đối tượng.
2	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.
3	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.
4	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.
5	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.
6	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.
7	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		
8	= n	Nội dung của RLO hiện hành được gán cho đối tượng n.
9	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.
10	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		
11	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.
12	LD n	Nạp nội dung đối tượng n (dạng mã BCD) vào ACCU1 không phụ thuộc RLO.
13	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.

TT	Tên lệnh	Mô tả
2.1.4 Lệnh về thời gian		
14	R T	Xoá bộ thời gian nếu RLO = 1
15	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.
16	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.
17	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.
18	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.
19	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		
20	CD	Số đếm giảm 1 đơn vị tại sườn lên của RLO sau đó không phụ thuộc RLO nữa.
21	CU	Số đếm tăng 1 đơn vị tại sườn lên của RLO sau đó không phụ thuộc RLO nữa.
22	R C	Xoá bộ đếm nếu RLO = 1
23	S C	Đặt bộ đếm nếu RLO = 1
2.1.6. Các lệnh toán học		
24	!=F	So sánh bằng nhau của hai thanh ghi ACCU1 và ACCU2 (dạng bit)
25	+F	Cộng nội dung hai thanh ghi ACCU1 và ACCU2, kết quả nạp vào ACCU1 (lệnh này chỉ có ở STL).
26	<=F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có nhỏ hơn hay bằng ở ACCU1 không ?
27	<F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có nhỏ hơn ở ACCU1 không ?
28	><F	So sánh đối tượng lệnh trong hai thanh ghi ACCU1 và ACCU2 xem có khác nhau không ?
29	>=F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có lớn hơn hay bằng ở ACCU1 không ?

TT	Tên lệnh	Mô tả
30	>F	So sánh đối tượng lệnh trong thanh ghi ACCU2 có lớn hơn ở ACCU1 không ?
31	-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.		
32	C n	Gọi khối dữ liệu DB, không phụ thuộc vào RLO, quét chương trình không bị gián đoạn, RLO không bị ảnh hưởng.
33	G	Tạo lập hoặc xoá khối dữ liệu độc lập với RLO.
34	JC n	Nhảy sang làm việc ở khối n nếu RLO =1.
35	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.		
36	BE	Lệnh kết thúc khối.
37	BEC	Lệnh kết thúc có điều kiện giữa khối (RLO=1)
38	BEU	Lệnh kết thúc không điều kiện giữa khối, không phụ thuộc RLO.
2.1.9. Các lệnh khống.		
39	NOP 0	Mã lệnh 16 bit trong RAM đều bằng 0 (để giữ chỗ).
40	NOP 1	Mã lệnh 16 bit trong RAM đều bằng 1 (để giữ chỗ).
2.1.10. Lệnh dừng		
41	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.		
42	A=	Lệnh AND thay thế.
43	AN=	Lệnh AND đảo thay thế.
44	AW	Tổ hợp từng bit theo luật logic AND.
45	DO=	Lệnh DO thay thế.
46	O=	Lệnh OR thay thế.
47	ON=	Lệnh OR đảo thay thế.
48	OW	Tổ hợp từng bit theo luật logic OR.
49	XOR	Tổ hợp từng bit theo luật logic OR đặc biệt.

TT	Tên lệnh	Mô tả
2.2.2. Các lệnh về bit.		
50	RU	Lệnh xoá bít không điều kiện.
51	SU	Đặt một bit vô điều kiện.
52	TB	Trắc nghiệm bit cho trạng thái tín hiệu 1
53	TBN	Trắc nghiệm bit cho trạng thái tín hiệu 0.
2.2.3. Lệnh set, reset thay thế.		
54	==	Lệnh gán thay thế.
55	RB=	Lệnh xoá đối tượng lệnh hình thức.
56	RD=	Lệnh xoá đối tượng lệnh hình thức dạng số.
57	S=	Lệnh đặt đối tượng lệnh hình thức.
2.2.4. Các lệnh về thời gian và đếm.		
58	FR=	Lệnh khả thi thay thế.
59	SD=	Lệnh khởi động bộ thời gian bắt đầu trễ hình thức.
60	SEC=	Khởi động bộ thời gian mở rộng hoặc bộ đếm.
61	SFD=	Lệnh khởi động bộ thời gian tắt trễ hoặc bộ đếm xuống.
62	SP=	Lệnh khởi động bộ thời gian xung hình thức.
63	SSU=	Lệnh khởi động bộ thời gian bắt đầu trễ.
2.2.5. Các lệnh nạp và truyền.		
64	L=	Lệnh nạp thay thế.
65	LD=	Lệnh nạp đối tượng hình thức dạng cơ số BCD.
66	LW=	Lệnh nạp mẫu bit của đối tượng lệnh hình thức.
67	T=	Lệnh truyền đối tượng lệnh hình thức.
2.2.6. Các lệnh chuyển đổi.		
68	CFW	Nội dung ACCU1 được chuyển đổi từng bit một.
69	CSW	Bổ sung cho 2.
2.2.7. Các lệnh dịch chuyển.		
70	SLW	Dãy bit trong ACCU1 dịch sang trái.
71	SRW	Dãy bit trong ACCU1 dịch sang phải.
2.2.8. Các lệnh nhảy.		
72	JC=	Nhảy có điều kiện (RLO=1)
73	JM=	Nhảy nếu kết quả là âm (CC1=0, CC0=1).

TT	Tên lệnh	Mô tả
74	JN=	Nhảy nếu kết quả là (0,0) (CC1=1, CC0=0).
75	JO=	Nhảy khi cờ tràn.
76	JP=	Nhảy nếu kết quả là dương (CC1=1, CC0=0).
77	JU=	Nhảy không điều kiện.
78	JZ=	Nhảy nếu kết quả là 0 (CC1=0, CC0=0)
2.2.9. Các lệnh khác.		
79	D	Giảm nội dung trong ACCU1.
80	DO	Xử lý từ cờ hoặc từ dữ liệu.
81	FR T C	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.
82	I	Tăng nội dung trong ACCU1.
83	IA	Lệnh cấm ngắt.
84	LRS	Nạp miền dữ liệu hệ thống (nạp miền RS... vào ACCU1).
85	RA	Cho phép ngắt.
2.2.10. Nhóm lệnh hệ thống.		
86	ADD	Lệnh cộng một hằng số.
87	JC n	Nhảy sang làm việc ở khối n nếu RLO =1.
88	JU n	Nhảy sang làm việc ở khối n, không phụ thuộc RLO và RLO không bị ảnh hưởng.
89	LIR	Lệnh nạp gián tiếp thanh ghi.
90	RU	Lệnh xoá bít không điều kiện.
91	STS	Lệnh dừng tức khắc.
92	SU	Đặt một bit vô điều kiện.
93	TAK	Lệnh trao đổi nội dung thanh ghi.
94	TIR	Lệnh truyền gián tiếp thanh ghi.
95	TNB	Lệnh truyền một trường dữ liệu.

3. BẢNG LỆNH CỦA PLC - S7-200 (Siemens - Tây đức)

TT	Tên lệnh		Mô tả
3.1. Các lệnh thực hiện vô điều kiện			
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<=	n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu giá trị byte n1 không lớn hơn giá trị của byte n2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp.
5	AB=	n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu giá trị của hai byte n1 và n2 thoả mãn n1 = n2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp.
6	AB>=	n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu giá trị của hai byte n1 và n2 thoả mãn n1 ≥ n2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp.
7	AD<=	n1,n2	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>=	n1,n2	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.
9	AD=	n1,n2	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.
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.

TT	Tên lệnh	Mô tả
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<= n1,n2(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 n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp.
15	AR= n1,n2(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 n1 và n2 thoả mãn n1 = n2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp.
16	AR>= n1,n2(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 n1 và n2 thoả mãn n1 ≥ n2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp.
17	AW<= n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp.
18	AW= n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ n1 và n2 thoả mãn n1 = n2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp.
19	AW>= n1,n2	Thực hiện lệnh AND giữa giá trị của bít đầu tiên trong ngăn xếp với giá trị 1 nếu nội dung của hai từ n1 và n2 thoả mãn n1 ≥ n2. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp.

TT	Tên lệnh	Mô tả
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.
24	LD n	Nạp giá trị logic của điểm n chỉ dẫn trong lệnh vào bit đầu tiên của ngăn xếp.
25	LDB<= n1,n2	Bit đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1≤ n2.
26	LDB= n1,n2	Bit đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1= n2.
27	LDB>= n1,n2	Bit đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1≥ n2.
28	LDD= n1,n2	Bit đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ kép n1 và n2 thoả mãn n1= n2.
29	LDD>= n1,n2	Bit đầu tiên của ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ kép n1 và n2 thoả mãn n1≥ n2.
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 n1 và n2 thoả mãn n1 ≤ n2.
34	LDR= n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai số thực n1 và n2 thoả mãn n1 = n2.
35	LDR>= n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai số thực n1 và n2 thoả mãn n1 ≥ n2.

TT	Tên lệnh	Mô tả
36	LDW<= n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ n1 và n2 thoả mãn n1 ≤ n2.
37	LDW= n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ n1 và n2 thoả mãn n1 = n2.
38	LDW>= n1,n2(5)	Bít đầu tiên trong ngăn xếp nhận giá trị logic 1 nếu nội dung hai từ n1 và n2 thoả mãn n1 ≥ n2.
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<= n1,n2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi vào bit đầu tiên trong ngăn xếp.
46	OB= n1,n2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1 = n2. Kết quả được ghi vào bit đầu tiên trong ngăn xếp.
47	OB>= n1,n2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai byte n1 và n2 thoả mãn n1 ≥ n2. Kết quả được ghi vào bit đầu tiên trong ngăn xếp.
48	OD<= n1,n2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai từ kép n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi vào bit đầu tiên trong ngăn xếp.

TT	Tên lệnh	Mô tả
49	OD= n1,n2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai từ kép n1 và n2 thoả mãn n1 = n2. Kết quả được ghi vào bit đầu tiên trong ngăn xếp.
50	OD>= n1,n2	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu nội dung hai từ kép n1 và n2 thoả mãn n1 ≥ n2. 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<= n1,n2(5)	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai số thực n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi lại vào bit đầu trong ngăn xếp.
56	OR= n1,n2(5)	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai số thực n1 và n2 thoả mãn n1 = n2. Kết quả được ghi lại vào bit đầu trong ngăn xếp.
57	OR>= n1,n2(5)	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai số thực n1 và n2 thoả mãn n1 ≥ n2. Kết quả ghi lại vào bit đầu trong ngăn xếp.
58	OW<= n1,n2(5)	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai từ n1 và n2 thoả mãn n1 ≤ n2. Kết quả được ghi lại vào bit đầu trong ngăn xếp.

TT	Tên lệnh		Mô tả
59	OW=	n1,n2(5)	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai từ n1 và n2 thoả mãn n1 = n2. Kết quả được ghi lại vào bit đầu trong ngăn xếp.
60	OW>=	n1,n2(5)	Thực hiện toán tử OR giữa bit đầu tiên của ngăn xếp với giá trị logic 1 nếu hai từ n1 và n2 thoả mãn n1 ≥ n2. 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.
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.

TT	Tên lệnh		Mô tả
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, OUI	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.
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).

TT	Tên lệnh		Mô tả
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	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 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 dộ 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.

TT	Tên lệnh		Mô tả
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.
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.

TT	Tên lệnh		Mô tả
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.
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 bộ thời gian Txx được 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.

TT	Tên lệnh	Mô tả
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-Tây đức)

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.
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.

TT	Tên lệnh	Mô tả
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.
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.

TT	Tên lệnh	Mô tả
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.

TT	Tên lệnh	Mô tả
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.
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).

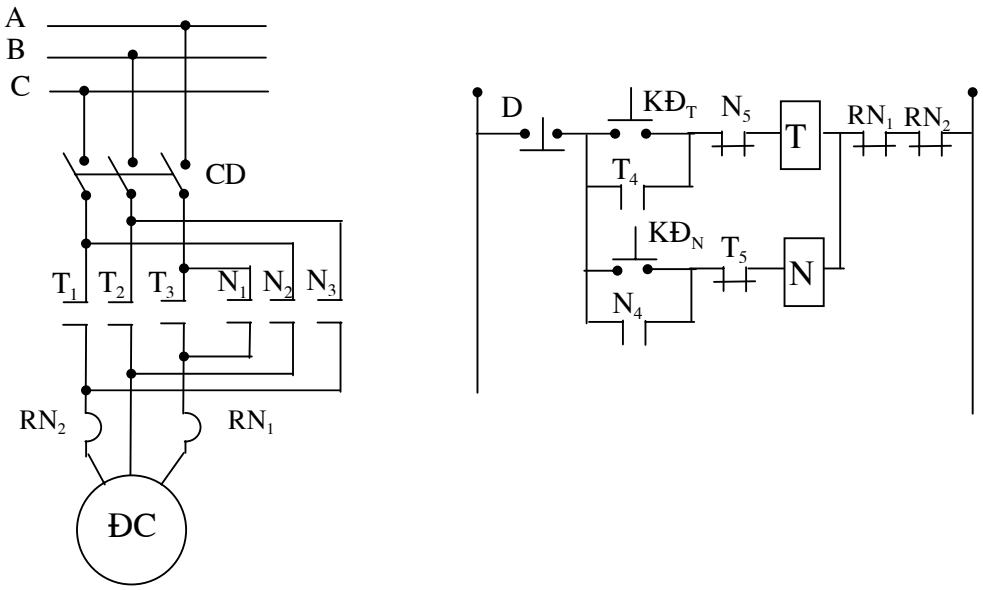
TT	Tên lệnh		Mô tả
83	JU	n	Nhảy sang làm việc ở nhãn n, không phụ thuộc RLO và RLO không bị ảnh hưởng.
84	JUO		Nhảy nếu (CC1 = 1, CC0 = 1).
85	JZ		Nhảy nếu kết quả là không (CC1 = 0, CC0 = 0).
86	L	n	Nội dung của đối tượng lệnh (đơn vị byte) được sao chép vào ACCU1 không phụ thuộc vào RLO, nội dung trước 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.
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.

TT	Tên lệnh	Mô tả
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).
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.

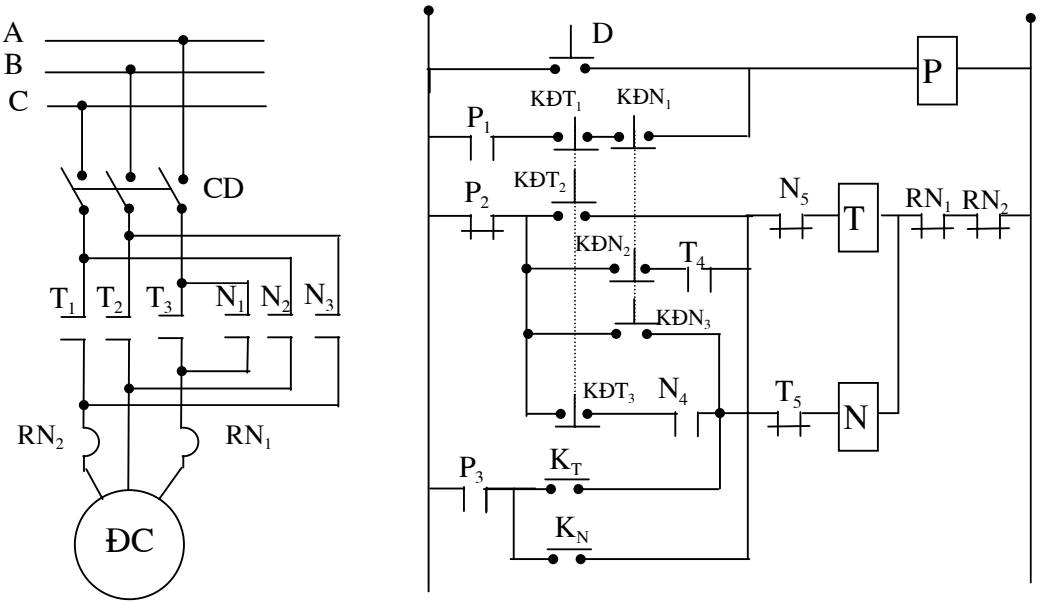
TT	Tên lệnh	Mô tả
128	RLDA	Quay tròn từ kép ở ACCU1 sang trái 1 bit qua CC 1.
129	RND	Đổi số thực 32 bit thành số nguyên 32 bit (bỏ phần thập phân).
130	RND+	Đổi số thực 32 bit thành số nguyên 32 bit, nếu là số dương thì làm tròn tăng, là số âm thì bỏ phần thập phân.
131	RND-	Đổi số thực 32 bit thành 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.

TT	Tên lệnh	Mô tả
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, các 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.
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.

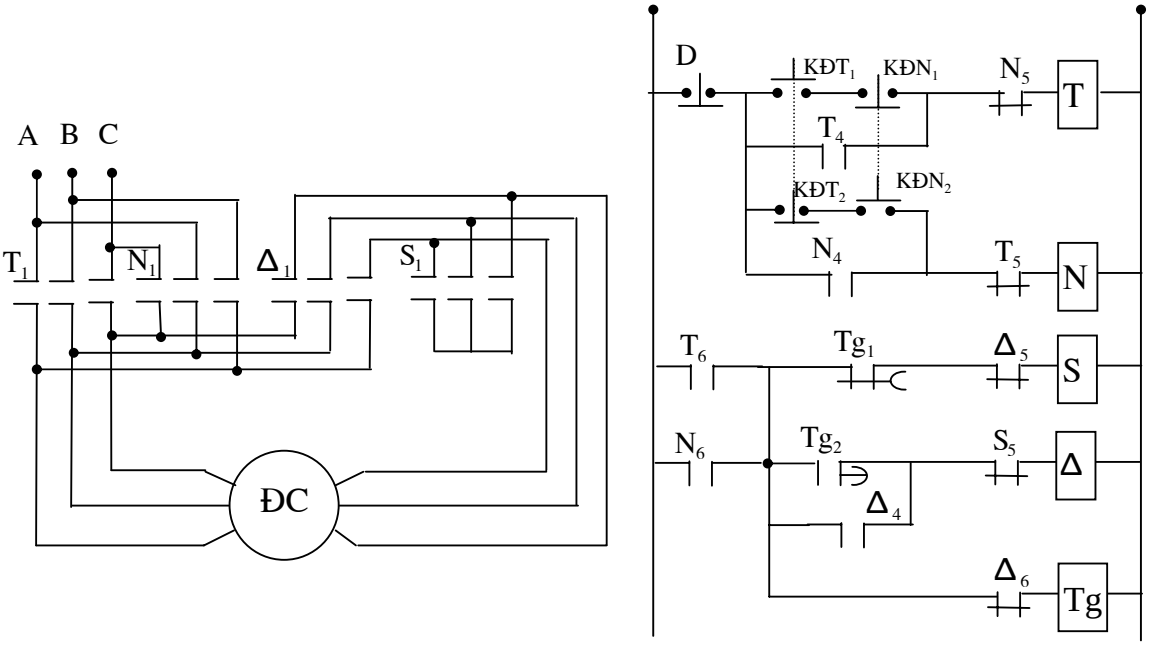
TT	Tên lệnh	Mô tả
162	XN(	Thực hiện lệnh OR (đặc biệt) giữa nội dung của RLO với giá trị nghịch đảo phép toán trong ngoặc (có đóng ngoặc), kết quả ghi vào RLO.
163	XOD	Thực hiện lệnh OR (đặc biệt) giữa các bit của hai từ kép.
164	XOW	Thực hiện lệnh OR (đặc biệt) giữa các bit của hai từ đơn.



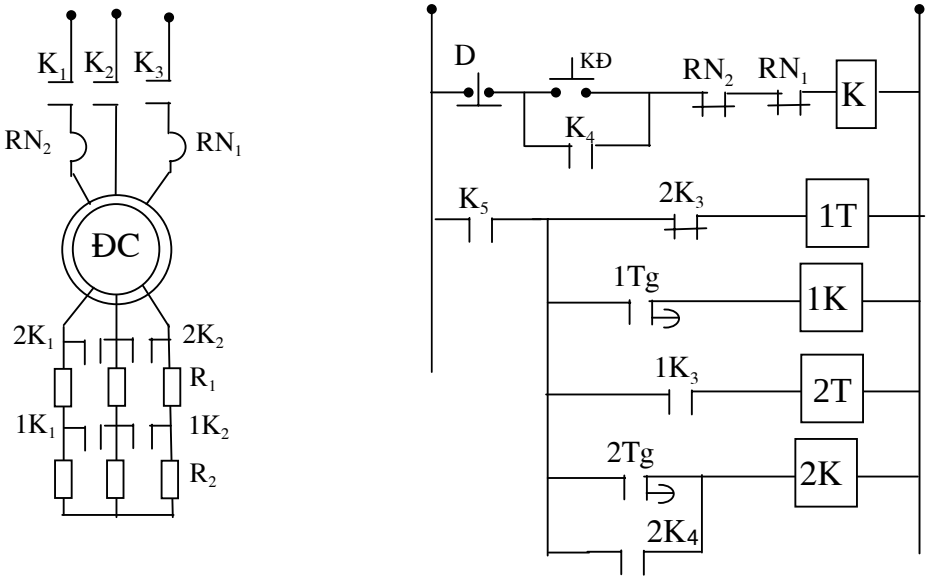
Hình 2.1



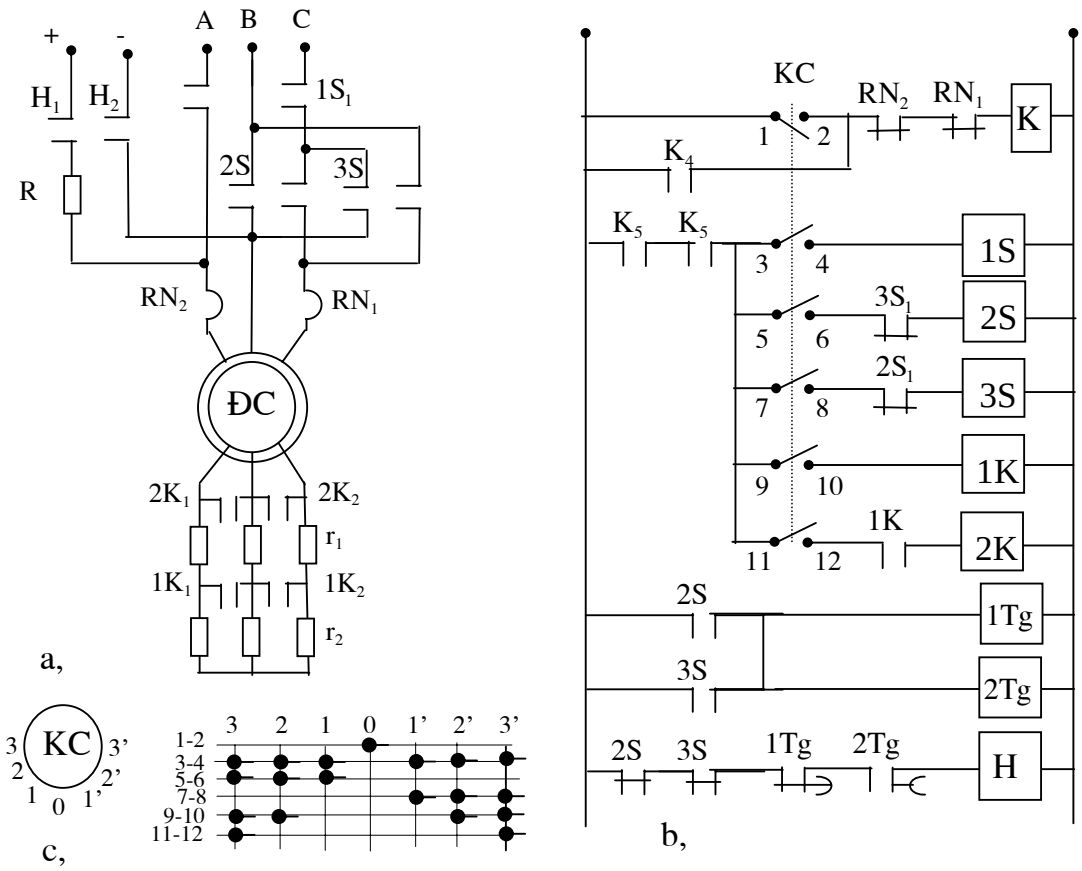
Hình 2.2



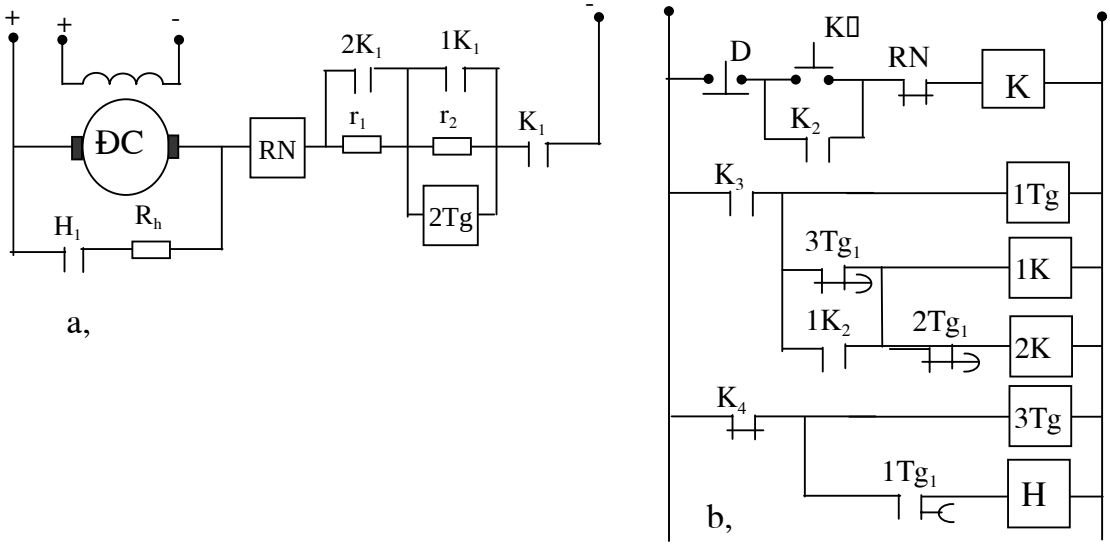
Hình 2.3



Hình 2.4



Hình 2.5



Hình 2.6

ỨNG DỤNG PLC

CHO HỆ THỐNG
KHỐNG CHẾ ĐIỀU KHIỂN THANG MÁY

Chương I

TÍN HIỆU HOÁ VÀ LÝ THUYẾT CHUNG VỀ TỐI ƯU LUẬT ĐIỀU KHIỂN THANG MÁY

1.1 TỐI ƯU HOÁ CHƯƠNG TRÌNH ĐIỀU KHIỂN THANG MÁY

1.1.1 Vấn đề tối ưu hoá trong điều khiển thang máy :

Như đã biết, trong các thang máy các nút ấn gọi thang được bố trí ở các tầng, tuỳ theo thiết kế mạch mà mỗi tầng sẽ có 1 hoặc 2 nút gọi thang. ở phương án này, tất cả các tầng (trừ tầng thượng chỉ có nút gọi xuống và tầng 1 chỉ có nút gọi lên) đều bố trí 2 nút ấn gọi thang, một nút gọi lên và một nút gọi xuống. Trong buồng thang cũng có một bàn phím gồm các nút ấn đến tầng, đóng mở cửa nhanh, dừng khẩn cấp, báo chuông khi cần thiết.

Các tín hiệu đó tác động vào hệ điều khiển thang máy không theo một quy luật nào cả. Do đó vấn đề đặt ra là : thang máy phải có một luật điều khiển sao cho vừa thoả mãn được các yêu cầu công nghệ, vừa đáp ứng được sự tối ưu về quãng đường mà buồng thang phải dịch chuyển, thời gian phục vụ cũng như năng lượng tiêu tốn, đồng thời mọi hành khách cảm thấy thoải mái khi sử dụng thang máy.

Như vậy, một vấn đề đặt ra là làm thế nào để có thể phục vụ được tất cả hành khách một cách tối ưu nhất, có thể nhớ được nhiều tín hiệu gọi Cabin và xử lý các tín hiệu nhớ này theo một luật tối ưu. Trong trường hợp này ta sử dụng lý thuyết hàng đợi.

1.1.2 Lý thuyết hàng đợi :

a. Khái niệm chung về hệ thống hàng đợi

Hệ thống hàng đợi (Queueing System) là hệ thống có các bộ phận phục vụ (Services) và các khách hàng đi đến hệ thống (Arriving Customers) để được phục vụ. Nếu khi khách hàng đến mà các bộ phận phục vụ đều bận thì các khách hàng

phải sắp hàng để đợi được phục vụ. Chính vì vậy mà hệ thống này có tên là hệ thống hàng đợi. Lý thuyết toán học để khảo sát các hệ thống hàng đợi được gọi là lý thuyết phục vụ đám đông (các khách hàng được coi là một đám đông được phục vụ).

b. Các đặc trưng cho hàng đợi

* *Chiều dài hàng đợi*

Là số khách hàng có trong hàng đợi (hạn chế hoặc không hạn chế).

* *Thời gian đợi*

Là khoảng thời gian từ khi khách hàng đến hệ thống cho đến khi bắt đầu được phục vụ. Thời gian đợi có thể hạn chế hoặc không hạn chế.

* *Luật sắp hàng*

Là phương thức chọn khách hàng trong hàng đợi. Thông thường có các luật sắp hàng như sau :

1. Đến trước phục vụ trước
2. Đến trước phục vụ sau
3. Ngẫu nhiên
4. Ưu tiên ...

c. Các thành phần chính của hệ thống hàng đợi

Hệ thống hàng đợi có ba bộ phận chính là :

* *Dòng khách hàng*

Là các phần tử, yêu cầu, sự kiện đi đến hệ thống để được phục vụ - được gọi chung là khách hàng. Đặc trưng cho dòng khách hàng là cường độ dòng khách hàng λ/đơn vị thời gian. Dòng khách hàng là một dòng sự kiện ngẫu nhiên, do đó khoảng cách thời gian giữa các khách hàng cũng là một đại lượng ngẫu nhiên.

**** Kênh phục vụ***

Là các cơ cấu để phục vụ khách hàng, thực hiện các yêu cầu của khách hàng. Thời gian phục vụ (Service time) và khoảng thời gian giữa các lần phục vụ là những đại lượng ngẫu nhiên. Tuỳ theo hệ thống có một hay nhiều điểm phục vụ mà người ta gọi là hệ thống có một hoặc nhiều kênh phục vụ. Đặc trưng cho kênh phục vụ là dòng phục vụ với cường độ là μ/đơn vị thời gian. Cường độ phục vụ là số khách hàng được phục vụ xong trên một đơn vị thời gian.

**** Hàng đợi (Queue)***

Là số khách hàng chờ đến lượt được phục vụ. Tuỳ theo số khách hàng đến nhiều hay ít (cường độ λ lớn hay bé), khả năng phục vụ (số kênh phục vụ, thời gian phục vụ) mà số khách hàng phải đợi trong hàng đợi nhiều hay ít. Vì vậy, độ dài hàng đợi cũng là một đại lượng ngẫu nhiên.

**** Luật sắp hàng***

Trong hệ thống hàng đợi có một kênh phục vụ thường có luật sắp hàng điều chỉnh sau đây:

- FIFO (First - In First - Out) : Khách hàng đến trước phục vụ trước . Luật FIFO thường được dùng ở những nơi như :

+ Sắp hàng trước quầy tính tiền của siêu thị

+ Sắp hàng vào cơ sở dịch vụ , phương tiện vận tải .

+ Các thiết bị sắp hàng trên băng tải chờ đến lượt được lắp ráp .v.v.

- LIFO (Last - In First - Out) : Khách hàng đến sau được phục vụ trước luật LIFO thường được dùng ở những nới như ;

+ Ra khỏi buồng thanh máy : người nào vào sau cùng sẽ được ra trước tiên .

+ Đọc giữ liệu trên băng từ : dữ liệu ghi sau sẽ được đọc trước .

+ Hàng hoá được xếp vào thùng chứa : hàng xếp sau cùng (phía trên của hàng chứa sẽ đươc lấy ra trước v.v...

- Ngẫu nhiên : các khách hành đều có chế độ ưu tiên như nhau và được phục vụ một cách ngẫu nhiên . Luật này thường được lấy ở các trường hợp sau như :

+ Phụ nữ trẻ em và người tàn tật được ưu tiên phục trước.

+ Luật FIFO cũng là trường hợp đặc biệt với đầu ưu tiên là đến trước .

+ Thời gian phục vụ ngắn được phục vụ trước (shortest job first). Ví dụ trên nút giao thông xe nhỏ gọn nhanh được ưu tiên đi trước so với xe to cồng kềnh di chuyển chậm v.v...

Chiều dài hàng đợi

Chiều dài hàng đợi là số khách hàng đứng đợi để đươc phục vụ. Nếu số vị trí để đứng đợi không hạn chế thì chiều dài hàng đợi có thể dài bất kỳ . Ngược lại nếu số vị trí đứng đợi là hạn chế thì thì chiều dài hàng đứng đợi không vượt quá số đã cho trước . Trong trường hợp này nếu khách hàng đến đúng vào lúc chiều dài hàng đợi đã đầy thì phải rời bỏ hệ thống và hệ thống sẽ bị mất khách hàng . Chiều dài hàng đợi là một đại lượng ngẫu nhiên phụ thuộc vào cường độ dòng khách hàng và dòng phục vụ.

**** Thời gian sắp hàng***

Thời gian sắp hàng là quãng thời gian khách hàng đứng đợi trong hàng đợi chờ để chờ đến lượt phục vụ. Có loại khách hàng có thể đợi bao lâu cũng đươc, ngược lại có loại khách hàng chỉ có thể đợi trong một thời gian nhất định, hết thời gian đó khách hàng sẽ rời bỏ hệ thống mặc dầu vẫn còn chỗ để đứng đợi. Trong trường hợp này hệ thống sẽ mất khách hàng. Để giảm khả năng mất khách hàng hệ thống phải tăng cường độ dòng phục vụ hoặc tăng số kênh phục vụ.

1.2 THUẬT TOÁN TỐI ƯU ĐIỀU KHIỂN THANG MÁY :

Khi thiết kế thuật toán tối ưu điều khiển thang máy với hệ thống hàng đợi ta thấy có những đặc điểm cần lưu ý như sau :

- Nếu chiều dài hàng đợi lớn quá có thể xảy ra trường hợp hành khách không đợi được đã không đi thang máy. Trong khi đó, đến lượt được phục vụ thang máy vẫn chạy đến đúng vị trí gọi. Như vậy sẽ dẫn đến lãng phí thời gian và giảm hiệu suất hoạt động của thang. Do đó trong trường hợp này ta chọn chiều dài hàng đợi là 60.
- Khi xắp xếp hàng đợi, một vấn đề đặt ra là có thể ở một tầng có nhiều tín hiệu gọi thang của nhiều người. Vì vậy, mỗi khi có tín hiệu gọi thang cần phải duyệt toàn bộ hàng đợi xem tín hiệu này đã có mặt trong hàng đợi hay chưa, trước khi thêm vào hàng đợi.

- Khi hành khách đi vào thang máy và ấn nút gọi tầng, sau quá trình chuyển động, thang máy sẽ dừng lại ở vị trí tầng đã gọi. Tuy nhiên, có thể tầng này đã có mặt trong hàng đợi và như vậy, coi như tín hiệu gọi thang này đã được phục vụ. Vì vậy, cần phải loại tín hiệu này ra khỏi danh sách hàng đợi để quá trình phục vụ của thang máy không bị nhầm lẫn.

- Trong quá trình phục vụ có thể có những trường hợp thang máy không phục vụ kịp thời, dẫn đến tình trạng mất khách hàng do thang máy đã chuyển động đến tầng gọi nhưng không có người đi vào thang máy. Vì vậy cần phải có tín hiệu cảm biến sàn Cabin hoặc đặt thời gian trễ để sau khi cửa buồng thang đã khép lại nhưng không có người thì tín hiệu gọi thang tiếp theo trong hàng đợi sẽ được phục vụ.

Sơ đồ thuật toán điều khiển được mô tả như hình 3-6. Trong đó hàng đợi HĐ là một mảng 60 phần tử chứa tối đa 60 tín hiệu gọi sắp hàng. Ký hiệu HĐ[n] là tín hiệu gọi thứ n trong hàng đợi. Thuật toán này được giải thích trong phần sơ đồ thuật toán điều khiển hệ thống.

1.3 TÍN HIỆU HOÁ CHO HỆ THỐNG ĐIỀU KHIỂN LOGIC KHẢ TRÌNH

1.3.1 Thiết kế bộ tạo mã phím cho các công tắc và nút ấn :

a. Bàn phím gọi tầng

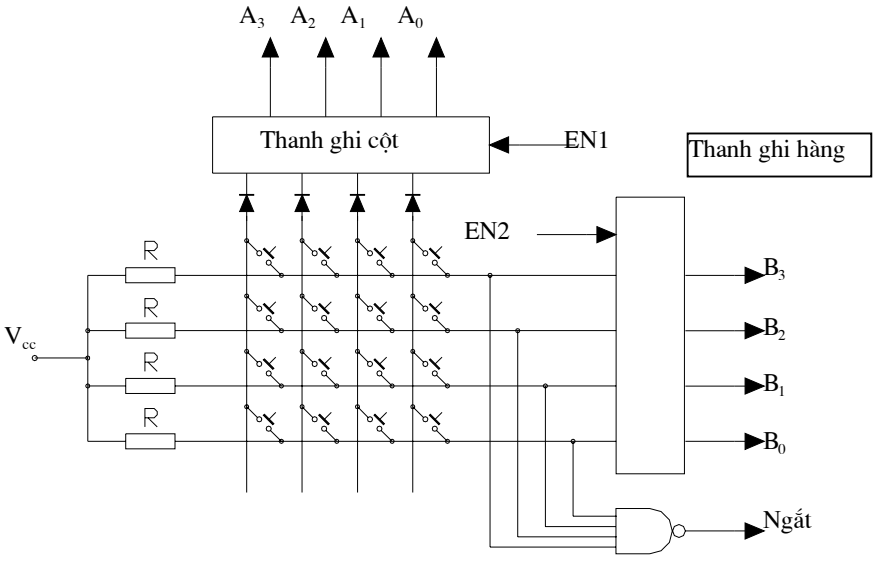
Khi số tầng ít, việc tạo mã phím cho các nút ấn gọi tầng, gọi thang và các tín hiệu cảm biến vị trí rất đơn giản. Các tín hiệu này sẽ tác động đến một bộ phát xung để phát ra các xung tương ứng với phím gọi. Các xung này sẽ được đưa đến một bộ đếm để có được mã phím. Tuy nhiên, khi số tầng nhiều, việc tạo mã như trên sẽ rất phức tạp và phải có rất nhiều dây dẫn tín hiệu. Vì vậy, chúng ta sẽ lựa chọn phương án thiết kế cho thang máy nhiều tầng với các bộ tạo mã theo ma trận phím dưới dạng các **mã quét (Scan code)**.

Trong thực tế có nhiều loại phím mà khi tiếp xúc sẽ gây ra những hiện tượng như :

- Thay đổi điện trở của phím.
- Thay đổi điện dung của phím.
- Thay đổi dòng điện chạy qua phím theo định luật Hall.

Để giảm số lượng dây dẫn phải biến đổi số thứ tự của các phím (mã hoá các phím) thành dạng nhị phân hoặc dạng số Hexa. Trong trường hợp này, người ta sử dụng một bộ đếm quét bàn phím. Khi có một phím được ấn bộ đếm sẽ được lệnh dừng lại và ở đầu ra của các bộ đếm sẽ thu được một mã nhị phân tương ứng với số thứ tự của phím. Mã này được gọi là mã quét bàn phím.

Nguyên tắc tạo mã quét cho bàn phím được minh hoạ như sơ đồ hình 2.1.



Hình 2-1: Sơ đồ tạo mã bàn phím

Người ta đưa ra các giá trị 1 lần lượt quét vào các cột, sau đó đọc vào các giá trị ứng với các cột khác nhau ở thanh ghi hàng từ đó có thể biết được mã của phím.

Sơ đồ cụ thể tạo mã quét của bảng 64 nút ấn như hình 2-1.

Vi mạch 4001 (4 cổng NOR) 2 lối vào) được mã thành mạch phát xung đồng hồ 50Hz có thể điều khiển chạy hoặc dừng được. Khi bộ phát xung chạy, hai tầng đếm nhị phân (dùng IC4520) sẽ đếm liên tục và thể hiện kết quả bằng xung điện áp ở các lối ra của chúng. Tầng đếm thứ hai đưa kết quả vào $A_2B_2C_2$ của vi mạch giải mã 4051 (demultiplexer 1-8) khống chế các cột của bàn phím.

Bảng các chế độ làm việc của 4051 như sau :

C	B	A	Z nối với
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Tầng đếm thứ nhất đưa kết quả vào $A_1B_1C_1$ của vi mạch 4051 khống chế hàng của bàn phím.

Khi ấn phím, hai đầu dây hàng và cột của phím đó được nối với nhau tạo nên điện áp +5V từ Z_2 đưa sang Z_1 để làm dừng bộ phát xung đồng hồ. Trên các lối ra 1..6 sẽ giữ nguyên trạng thái của hai tầng đếm lúc dừng và đó cũng chính là mã nhị phân tương ứng với ký tự ghi trên phím ấn.

Ví dụ khi ấn phím số "12" (hàng 3, cột 1), bộ phát xung đồng hồ tiếp tục chạy, các lối ra của bộ đếm cứ liên tục thay đổi và chỉ dừng khi $A_1B_1C_1$=110 (Z_1 nối với 3) và $A_2B_2C_2$ = 100 (Z_2 nối với 1). Như vậy ở đầu ra ta sẽ có mã nhị phân tương ứng với số 12 là 00001100.

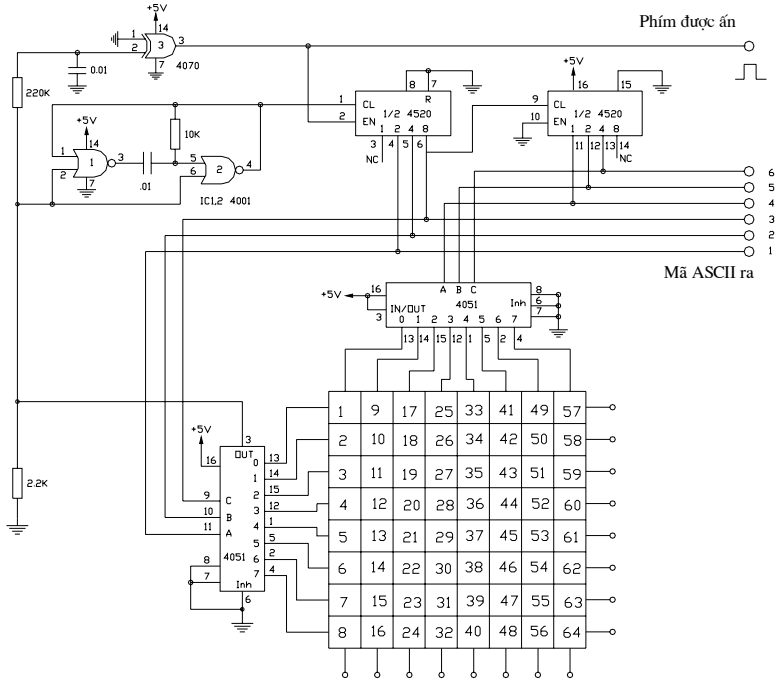
Khi nhả phím, hai bộ đếm lại tiếp tục biến đổi quay vòng chờ đến khi có một phím khác được ấn. Nếu có một phím thứ 2 được ấn trong khi phím thứ nhất chưa được nhả thì vẫn không có gì thay đổi cho đến khi phím thứ nhất được nhả. Sau đó quá trình biến đổi quay vòng lại được tiếp tục và chỉ dừng lại ứng với mã nhị phân của phím thứ hai.

Do thực tế đề tài thiết kế thang máy cho cao ốc 60 tầng, bàn phím chỉ đưa ra số nhị phân lớn nhất là 63, tức là chỉ sử dụng hết 6 đường truyền dữ liệu nên chỉ cần sử dụng 6 đầu vào (INPUT) của PLC.

b. Bàn phím gọi thang

Bàn phím gọi thang có cấu tạo và nguyên tắc hoạt động tương tự như bàn phím gọi tầng. Tuy nhiên do số lượng phím tăng lên gấp đôi nên kết cấu của bàn phím gọi thang có khác đôi chút, tức là phải tăng thêm số linh kiện để có thể đưa

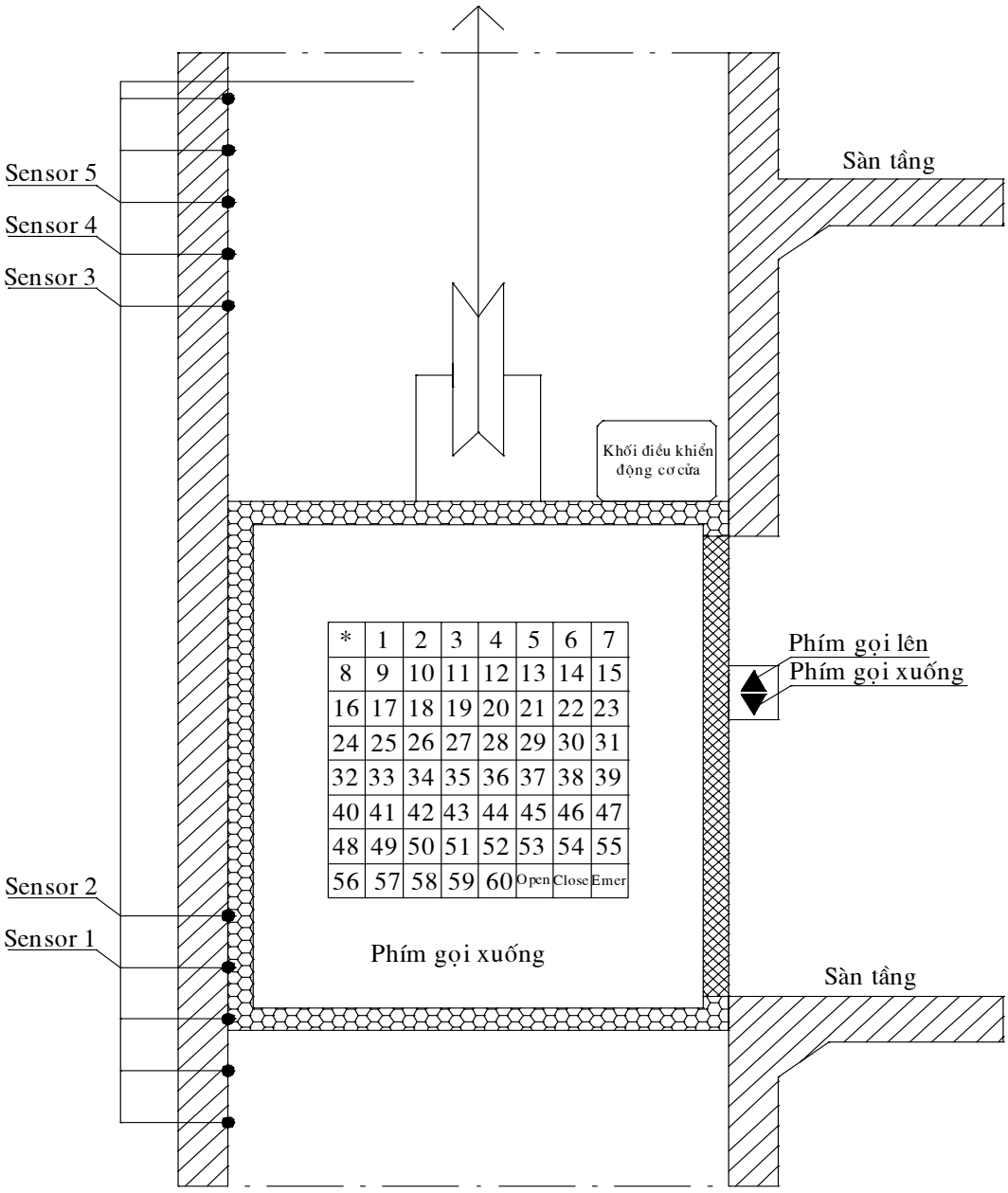
ra mã quét của các số từ 0 đến 127; trong đó các phím có mã từ 1 đến 59 dùng để gọi thang lên tương ứng với các tầng từ 1 đến 59, còn các phím có mã từ 62 đến 120 dùng để gọi thang xuống tương ứng với các tầng từ 2 đến 60. Các tín hiệu ra từ bàn phím gọi thang được đưa vào 7 đầu vào của PLC.



Hình 2-2: Bảng mã phím gọi tầng .

1.3.2 Thiết kế mạch cho các sensor

Như trên đã đề cập, để dừng chính xác buồng thang thì phải có tín hiệu báo giảm tốc trước khi phanh hãm đến sàn. Tại vùng dừng, người ta bố trí 5 sensor được bố trí như trên hình vẽ 2-3. Tất cả 5 sensor này được đấu song song và đưa vào một đầu vào ngắt số 0 của PLC. Chương trình ngắt sẽ phải giải mã để xác định thứ tự các tín hiệu để báo cho chương trình chính biết để có các phản ứng phù hợp.



*	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	Open	Close	Emer

Hình 2-3: Vùng dừng cho thang máy.

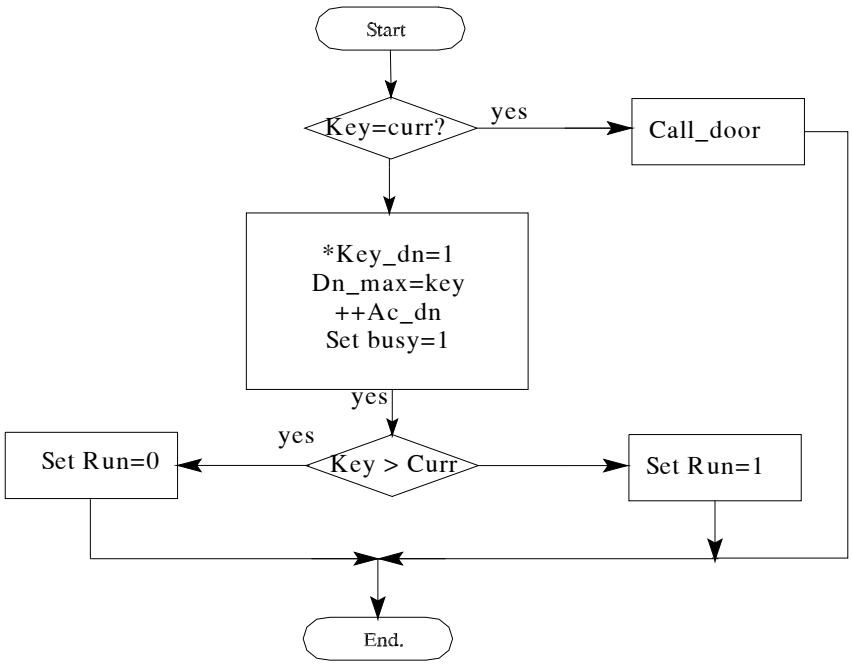
Chương II

CÁC SƠ ĐỒ THUẬT TOÁN SỬ DỤNG TRONG ĐIỀU KHIỂN THANG MÁY

Sau đây là phần thuyết minh các sơ đồ thuật toán đã được sử dụng trong chương trình:

2.1 CÁC CHƯƠNG TRÌNH XỬ LÝ PHÍM GỌI XUỐNG

2.1.1 Có phím gọi xuống khi thang máy đang dừng (SBN_1):

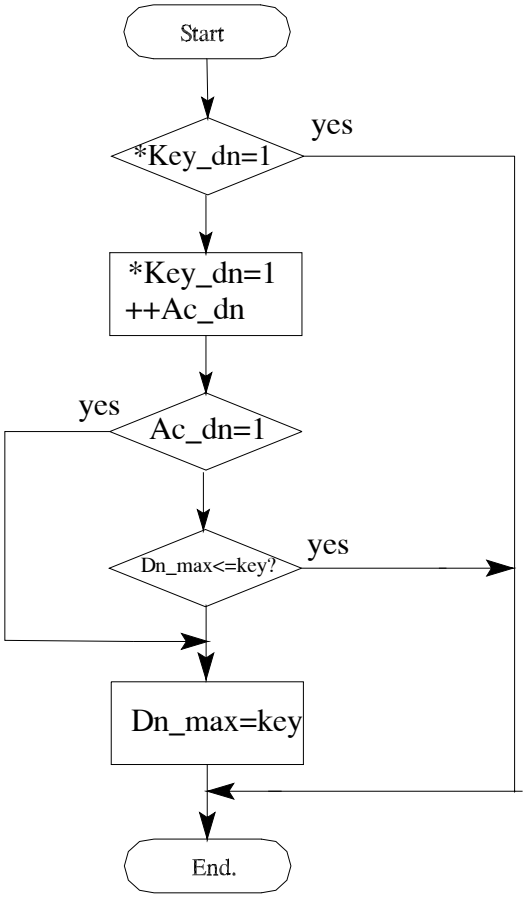


Hình 2 -1 : Sơ đồ thuật toán của chương trình bàn phím gọi xuống khi thang đang dừng (Busy = 0).

1. Kiểm tra trường hợp người gọi thang đứng ở đúng tầng mà Cabin thang máy đang dừng, nếu đúng thì sang bước 2, nếu sai thì sang bước 3.
2. Gọi chương trình mở - đóng cửa rồi sang bước 7.

3. Đưa vị trí tầng người đứng gọi thang (Key) vào hàng đợi xuống; đặt giá trị tầng lớn nhất trong hàng đợi bằng tầng được gọi; tăng số phần tử trong hàng đợi xuống lên một giá trị; thiết lập cờ busy (báo bận) = 1; sang bước 4.
4. So sánh vị trí tầng người đứng gọi thang với tầng hiện tại (Current), nếu lớn hơn thì sang bước 5, nếu sai sang bước 6.
5. Thiết lập cờ chạy lên (Run = 0), sang bước 7.
6. Thiết lập cờ chạy xuống (Run = 1), sang bước 7.
7. Kết thúc chương trình.

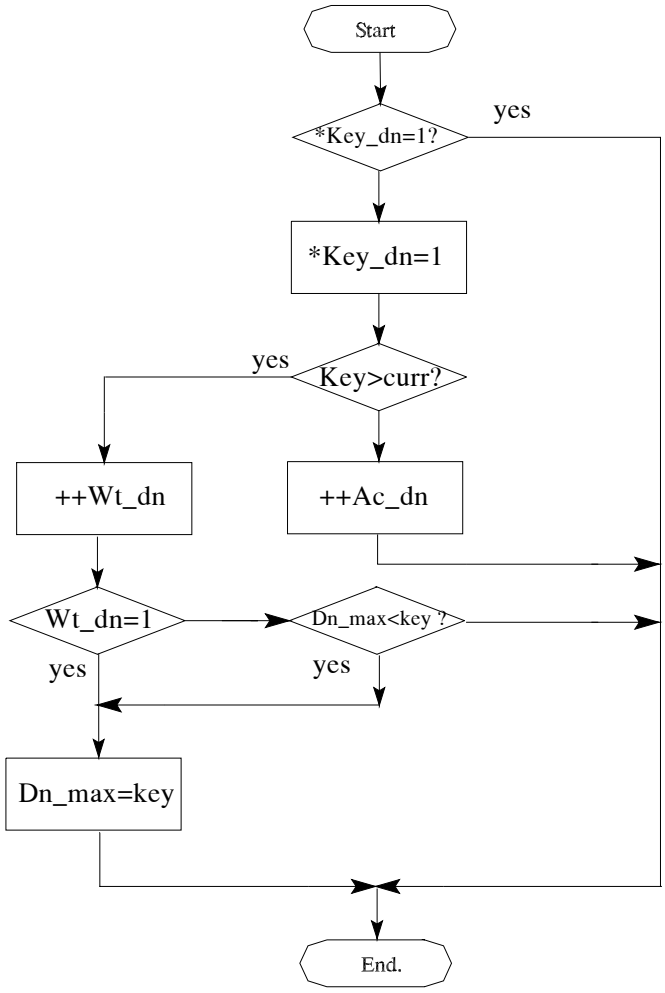
2.1.2 Có phím gọi xuống khi thang máy đang trong hành trình lên (SBN_2):



Hình 2 -2 : Sơ đồ thuật toán của chương trình bàn phím gọi xuống khi thang đang chạy lên (Run = 0)

1. Kiểm tra xem đã có số tầng trong hàng đợi xuống chưa, nếu đã có thì sang bước 6, nếu chưa thì sang bước 2.
2. Đưa vị trí tầng người đứng gọi thang (Key) vào hàng đợi xuống; tăng số phần tử trong hàng đợi xuống lên một giá trị; sang bước 3.
3. Kiểm tra xem có phải là người gọi đầu tiên không, nếu đúng thì sang bước 5, ngược lại thì sang bước 4.
4. Kiểm tra giá trị tầng lớn nhất (Dn_max) trong hàng đợi so với tầng được gọi, nếu Dn_max <= key thì sang bước 6, ngược lại thì sang bước 5.
5. Đặt Dn_max = key.
6. Kết thúc chương trình.

2.1.3 Có phím gọi xuống khi thang máy đang trong hành trình xuống(SBN_3):

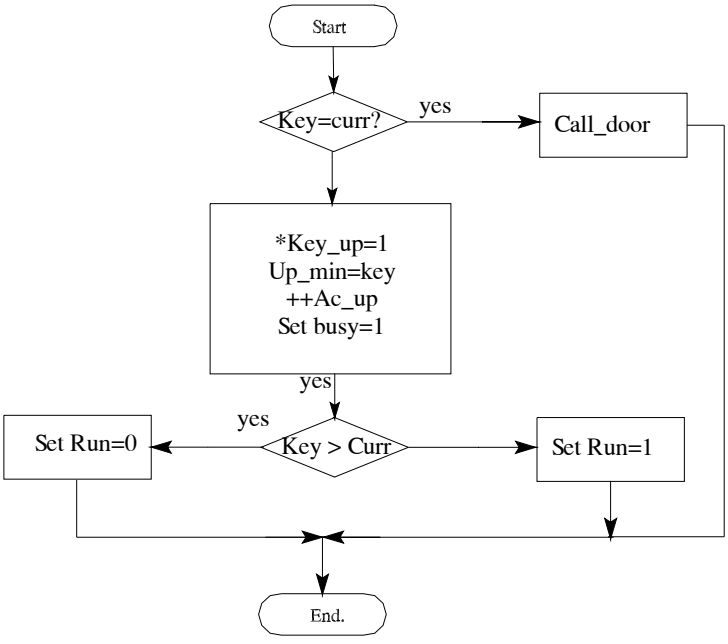


Hình 2-3: Sơ đồ thuật toán của chương trình bàn phím gọi xuống khi thang đang chạy xuống (Run = 1)

1. Kiểm tra xem đã có số tầng trong hàng đợi xuống chưa, nếu đã có thì sang bước 9, nếu chưa thì sang bước 2.
2. Đưa vị trí tầng người đứng gọi thang (Key) vào hàng đợi xuống; sang bước 3.
3. So sánh tầng gọi (key) với tầng hiện tại (current), nếu nhỏ hơn thì sang bước 4, ngược lại thì sang bước 5.
4. Tăng số phần tử đang được phục vụ (Wt_up) trong hàng đợi lên 1 giá trị, sang bước 9.
5. Tăng số phần tử chờ được phục vụ (Ac_up) trong hàng đợi lên 1 giá trị, sang bước 9.
6. Kiểm tra xem có phải là phần tử đầu tiên được đưa vào hàng đợi chờ phục vụ, nếu đúng thì sang bước 8, ngược lại sang bước 7.
7. So sánh phần tử nhỏ nhất trong hàng đợi lên (Up_min) với tầng gọi (key), nếu nhỏ hơn thì sang bước 8, ngược lại sang bước 9.
8. Đặt giá trị Up_min = key.
9. Kết thúc chương trình.

2.2 CÁC CHƯƠNG TRÌNH XỬ LÝ PHÍM GỌI LÊN

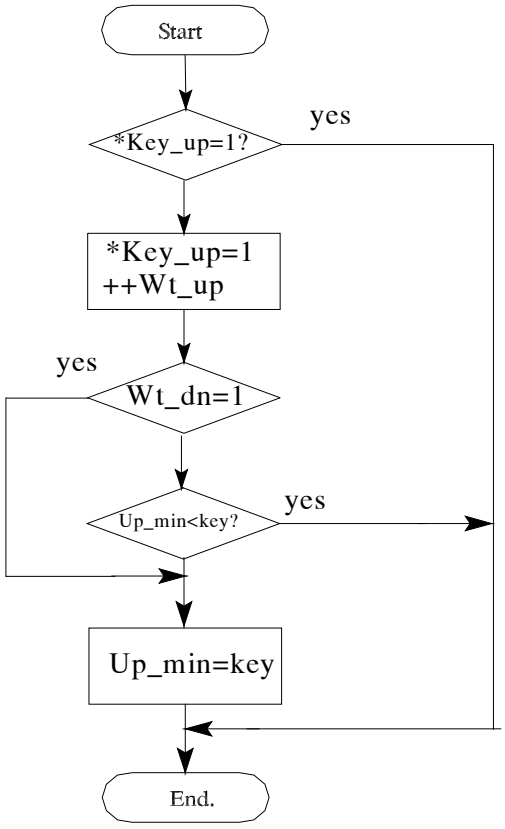
2.2.1 Có phím gọi lên khi thang đang dừng (SBN_4).



Hình 2-4: Sơ đồ thuật toán của chương trình bàn phím gọi lên khi thang đang dừng (Busy = 0)

1. Kiểm tra trường hợp người gọi thang đứng ở đúng tầng mà Cabin thang máy đang dừng, nếu đúng thì sang bước 2, nếu sai thì sang bước 3.
2. Gọi chương trình mở - đóng cửa (Call door) rồi sang bước 7.
3. Đưa vị trí tầng người đứng gọi thang (Key) vào hàng đợi lên; đặt giá trị tầng nhỏ nhất trong hàng đợi lên bằng tầng được gọi; tăng số phần tử trong hàng đợi lên một giá trị; thiết lập cờ busy (báo bận) = 1; sang bước 4.
4. So sánh vị trí tầng người đứng gọi thang với tầng hiện tại (Current), nếu lớn hơn thì sang bước 5, ngược lại thì sang bước 6.
5. Thiết lập cờ chạy lên (Run = 0), sang bước 7.
6. Thiết lập cờ chạy xuống (Run = 1), sang bước 7.
7. Kết thúc chương trình.

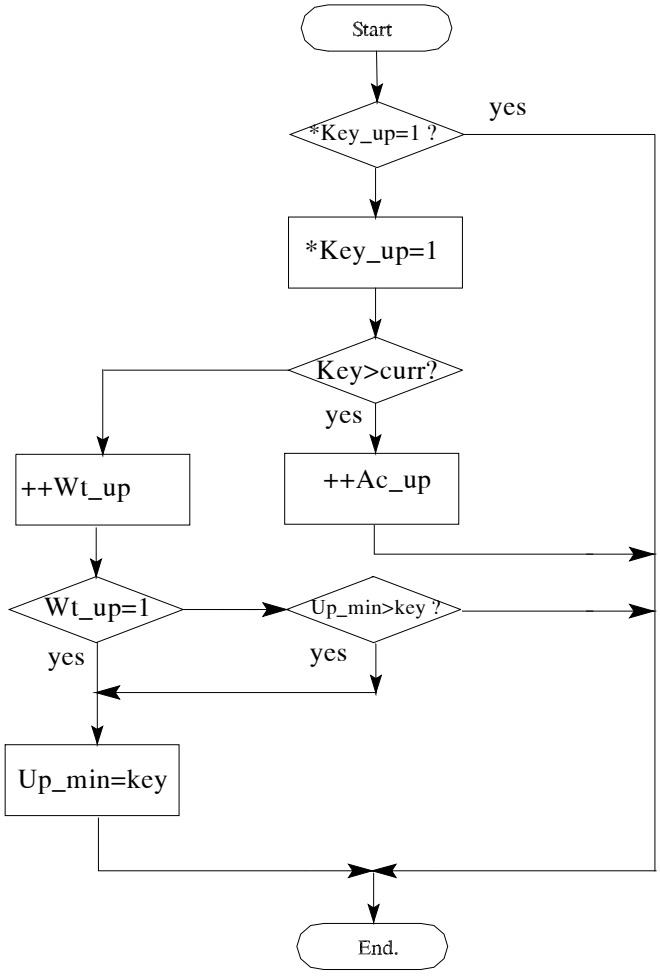
2.2.2 Có phím gọi lên khi thang máy đang trong hành trình xuống (SBN_5):



Hình 2-5 : Sơ đồ thuật toán của chương trình bàn phím gọi lên khi thang đang chạy xuống (Run = 1).

1. Kiểm tra xem đã có số tầng trong hàng đợi lên chưa, nếu đã có thì sang bước 6, nếu chưa thì sang bước 2.
2. Đưa vị trí tầng người đứng gọi thang (Key) vào hàng đợi lên; tăng số phần tử trong hàng đợi lên một giá trị; sang bước 3.
3. Kiểm tra xem có phải là người gọi đầu tiên không, nếu đúng thì sang bước 5, ngược lại thì sang bước 4.
4. Kiểm tra giá trị tầng nhỏ nhất (Up_min) trong hàng đợi so với tầng được gọi, nếu Up_min >= key thì sang bước 6, ngược lại thì sang bước 5.
5. Đặt Up_min = key.
6. Kết thúc chương trình.

2.2.3 Có phím gọi lên khi thang máy đang trong hành trình lên (SBN_6.

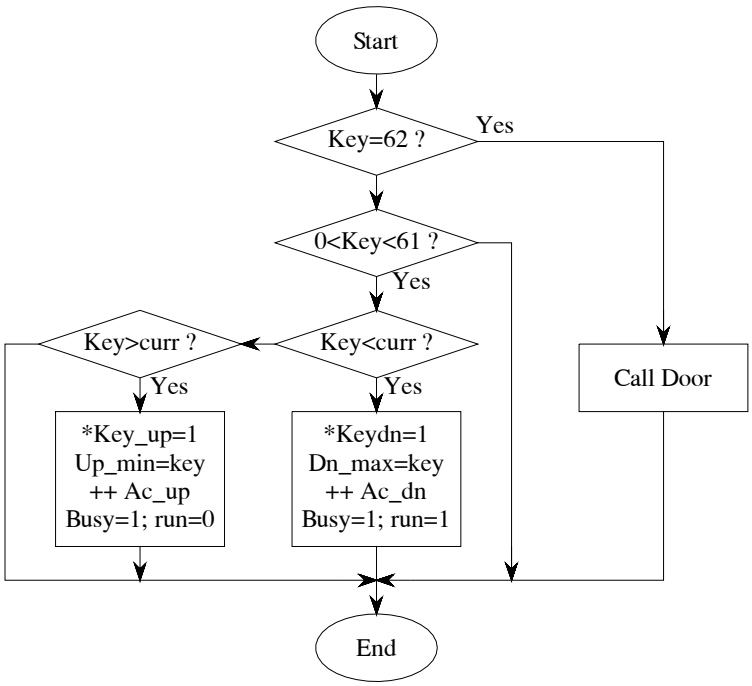


Hình 2-6 : Sơ đồ thuật toán của chương trình bàn phím gọi lên khi thang đang chạy lên (Run = 0).

1. Kiểm tra xem đã có số tầng trong hàng đợi lên chưa, nếu đã có thì sang bước 9, nếu chưa thì sang bước 2.
2. Đưa vị trí tầng người đứng gọi thang (Key) vào hàng đợi lên; sang bước 3.
3. So sánh tầng gọi (key) với tầng hiện tại (current), nếu nhỏ hơn thì sang bước 4, ngược lại thì sang bước 5.
4. Tăng số phần tử đang được phục vụ (Ac_dn) trong hàng đợi lên 1 giá trị, sang bước 9.
5. Tăng số phần tử chờ được phục vụ (Wt_dn) trong hàng đợi lên 1 giá trị, sang bước 9.
6. Kiểm tra xem có phải là phần tử đầu tiên được đưa vào hàng đợi chờ phục vụ, nếu đúng thì sang bước 8, ngược lại sang bước 7.
7. So sánh phần tử lớn nhất trong hàng đợi xuống (Dn_max)với tầng gọi (key), nếu nhỏ hơn thì sang bước 9, ngược lại sang bước 8.
8. Đặt giá trị Dn_max = key.
9. Kết thúc chương trình.

2.3 XỬ LÝ PHÍM GỌI TẦNG:

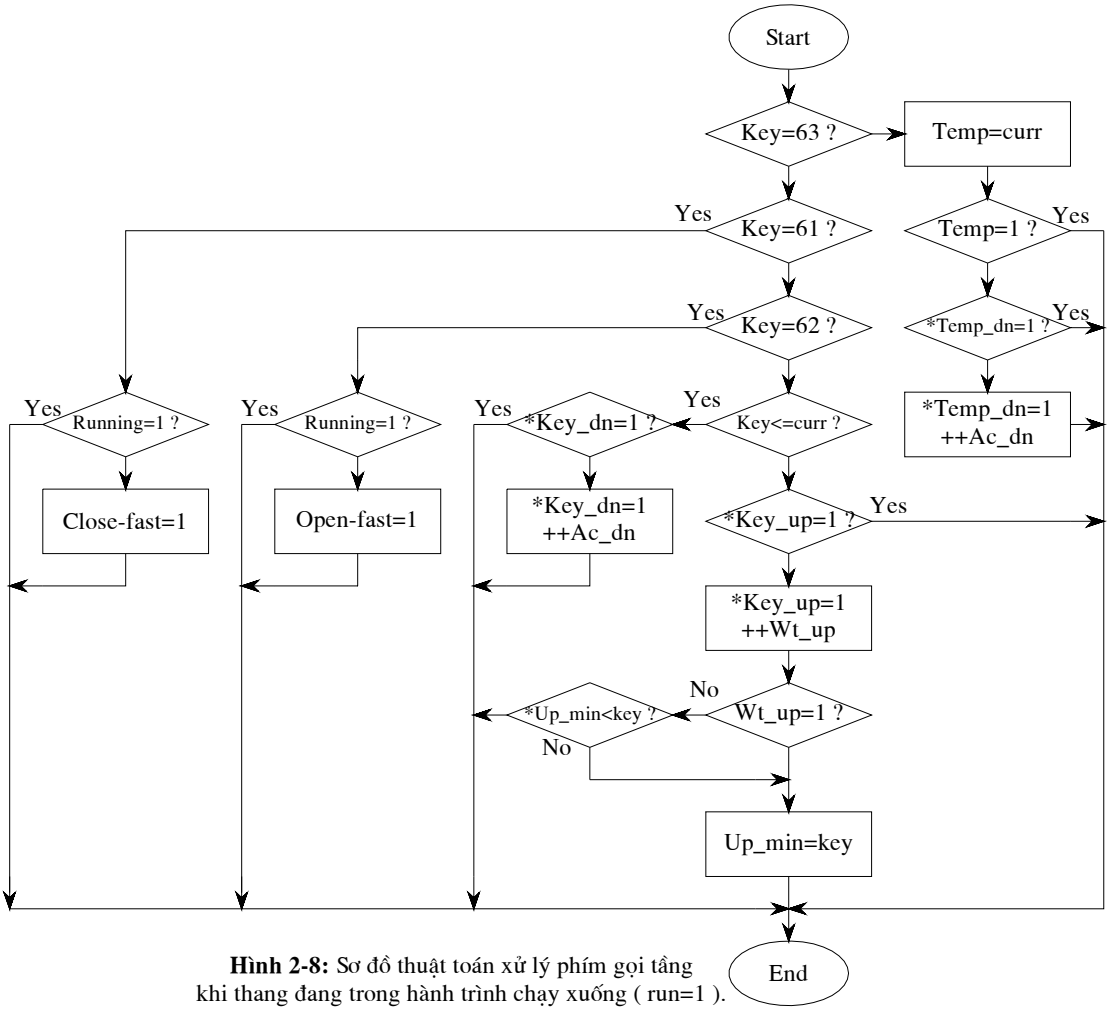
2.3.1 Có phím gọi tầng khi thang máy đang dừng (SBN_7).



Hình 2-7: Sơ đồ thuật toán chương trình xử lý phím gọi tầng khi thang đang dừng (run =0).

1. Kiểm tra phím mở cửa nhanh, nếu có thì chuyển sang bước 7, ngược lại thì chuyển sang bước 2.
2. Phát hiện xem phím bấm có nằm trong khoảng 0 < key < 61 không, nếu đúng thì sang bước 3, ngược lại sang bước 8. Trong trường hợp này, chỉ cho phép tín hiệu gọi tầng và tín hiệu mở cửa nhanh, cấm phím dừng khẩn cấp và phím đóng cửa nhanh.
3. So sánh tầng được gọi xem có nhỏ hơn tầng hiện tại không, nếu đúng thì sang bước 6, ngược lại thì sang bước 4.
4. Kiểm tra xem phím bấm có lơn hơn tầng hiện tại không, nếu đúng thì sang bước 5, ngược lại sang bước 8.
5. Đưa tầng được gọi vào hàng đợi lên; gán giá trị nhỏ nhất trong hàng đợi lên = tầng được gọi; tăng giá trị số phần tử trong hàng đợi lên một giá trị; thiết lập cờ busy = 1; bật cờ run = 0 (chạy lên), sang bước 8 .
6. Đưa tầng được gọi vào hàng đợi xuống; gán giá trị lớn nhất trong hàng đợi xuống = tầng được gọi; tăng giá trị số phần tử trong hàng đợi xuống lên một giá trị; thiết lập cờ busy = 1; bật cờ run = 1 (chạy xuống), sang bước 8 .
7. Gọi chương trình mở - đóng cửa (door), sang bước 8.
8. Kết thúc chương trình.

2.3.2 Có phím gọi tầng khi thang máy đang chạy xuống (SBN_8).

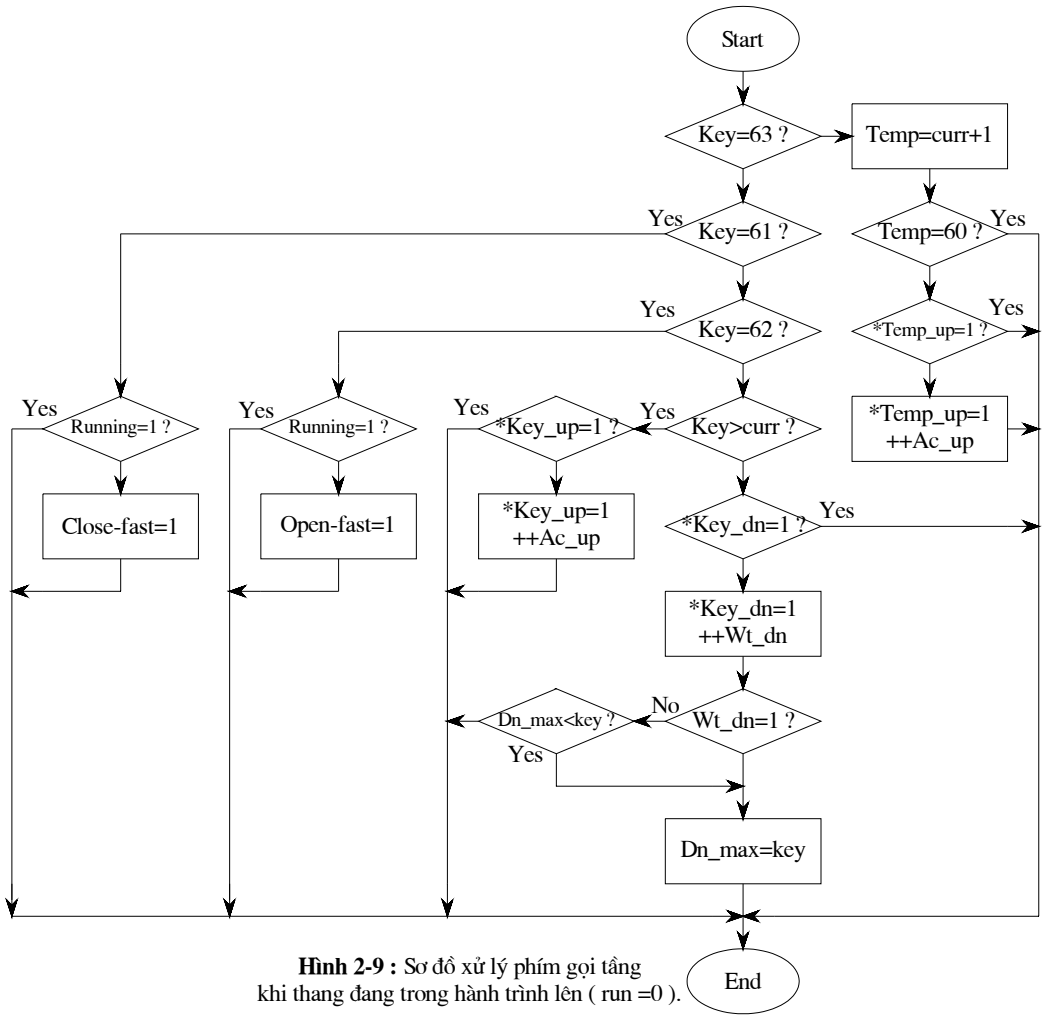


Hình 2-8: Sơ đồ thuật toán xử lý phím gọi tầng khi thang đang trong hành trình chạy xuống (run=1).

1. Kiểm tra phím dừng khẩn cấp có mã (Scan code) = 63 , nếu có thì chuyển sang bước 16, ngược lại chuyển sang bước 2.
2. Kiểm tra phím đóng cửa nhanh có mã (Scan code) = 61, nếu có thì chuyển sang bước 3, ngược lại thì chuyển sang bước 5.
3. Kiểm tra cờ đang chạy running, nếu được bật thì chuyển sang bước 20, không thì sang bước 4.
4. Thiết lập cờ đóng cửa nhanh, sang bước 20.
5. Kiểm tra phím mở cửa nhanh có mã (Scan code) = 62, nếu có thì chuyển sang bước 6, ngược lại thì chuyển sang bước 8.
6. Kiểm tra cờ đang chạy running, nếu được bật thì chuyển sang bước 20, không thì sang bước 7.
7. Thiết lập cờ mở cửa nhanh, sang bước 20.

8. So sánh tầng được gọi với giá trị tầng hiện tại, nếu lớn hơn thì chuyển sang bước 9, ngược lại sang bước 11.
9. Kiểm tra xem giá trị tầng được gọi đã có trong hàng đợi lên hay chưa, nếu có thì về bước 20, ngược lại sang bước 10.
10. Đưa giá trị tầng được gọi vào hàng đợi lên; tăng giá trị của phần tử có trong hàng đợi lên (Ac_up) một giá trị, sang bước 20.
11. Kiểm tra xem giá trị tầng được gọi đã có trong hàng đợi xuống hay chưa, nếu có thì về bước 20, ngược lại sang bước 12.
12. Đưa giá trị tầng được gọi vào hàng đợi xuống; tăng giá trị của phần tử có trong hàng đợi xuống (Wt_dn) lên một giá trị, sang bước 13.
13. Kiểm tra xem tầng được gọi có phải là phần tử đầu tiên trong hàng đợi xuống không, nếu đúng sang bước 15, ngược lại sang bước 14.
14. Kiểm tra xem tầng được gọi có lớn hơn giá trị lớn nhất trong hàng đợi xuống hay không, nếu đúng thì sang bước 15, ngược lại sang bước 20.
15. Gán giá trị Dn_max = giá trị tầng được gọi, sang bước 20.
16. Đưa giá trị tầng hiện tại cộng 1 vào ô nhớ tạm Temp, sang bước 17.
17. So sánh giá trị ô nhớ Temp với 60, nếu bằng về bước 20, ngược lại sang bước 18.
18. Kiểm tra giá trị ô nhớ Temp có trong hàng đợi hay chưa, có thì chuyển sang bước 20, chưa sang bước 19.
19. Gán giá trị ô nhớ Temp vào hàng đợi lên, tăng giá trị số phần tử có trong hàng đợi lên lên một giá trị, sang bước 20.
20. Kết thúc chương trình.

2.3.3 Có phím gọi tầng khi thang máy đang chạy lên (SBN_9).

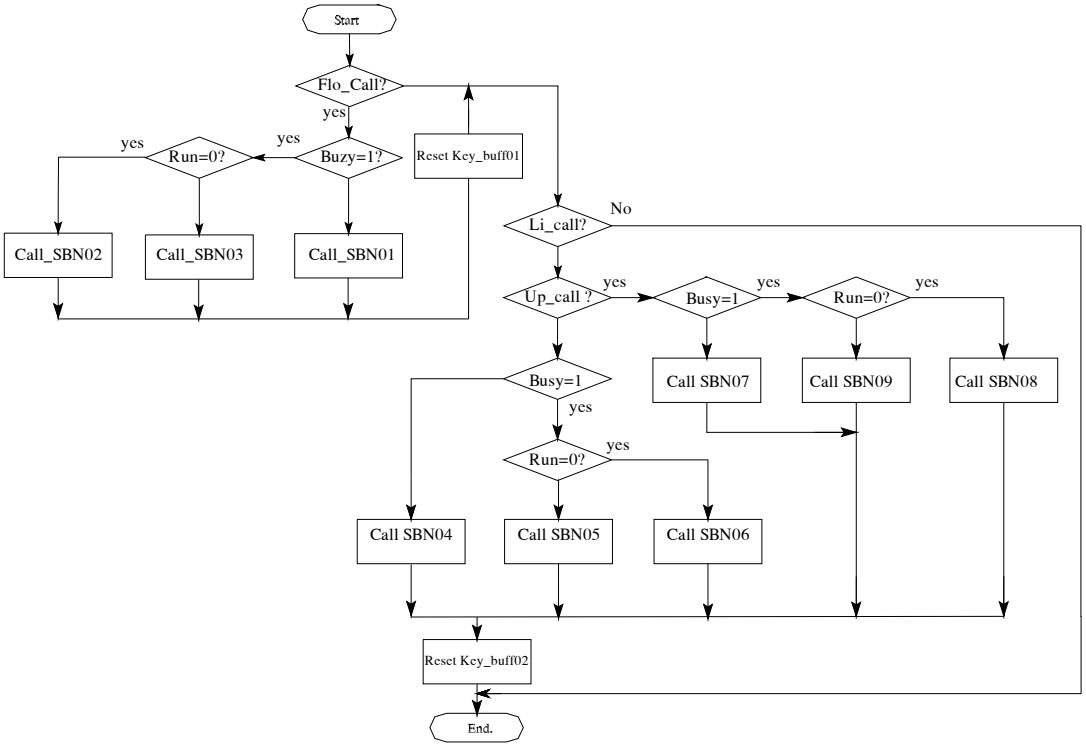


Hình 2-9 : Sơ đồ xử lý phím gọi tầng khi thang đang trong hành trình lên (run =0).

1. Kiểm tra phím dừng khẩn cấp có mã (Scan code) = 63 , nếu có thì chuyển sang bước 16, ngược lại chuyển sang bước 2.
2. Kiểm tra phím đóng cửa nhanh có mã (Scan code) = 61, nếu có thì chuyển sang bước 3, ngược lại thì chuyển sang bước 5.
3. Kiểm tra cờ đang chạy running, nếu được bật thì chuyển sang bước 20, không thì sang bước 4.
4. Thiết lập cờ đóng cửa nhanh, sang bước 20.
5. Kiểm tra phím mở cửa nhanh có mã (Scan code) = 62, nếu có thì chuyển sang bước 6, ngược lại thì chuyển sang bước 8.
6. Kiểm tra cờ đang chạy running, nếu được bật thì chuyển sang bước 20, không thì sang bước 7.
7. Thiết lập cờ mở cửa nhanh, sang bước 20.

8. So sánh tầng được gọi với giá trị tầng hiện tại, nếu nhỏ hơn hoặc bằng thì chuyển sang bước 9, ngược lại sang bước 11.
9. Kiểm tra xem giá trị tầng được gọi đã có trong hàng đợi xuống hay chưa, nếu có thì về bước 20, ngược lại sang bước 10.
10. Đưa giá trị tầng được gọi vào hàng đợi xuống; tăng giá trị của phần tử có trong hàng đợi xuống (Ac_dn) lên một giá trị, sang bước 20.
11. Kiểm tra xem giá trị tầng được gọi đã có trong hàng đợi lên hay chưa, nếu có thì về bước 20, ngược lại sang bước 12.
12. Đưa giá trị tầng được gọi vào hàng đợi lên; tăng giá trị của phần tử có trong hàng đợi xuống (Wt_up) lên một giá trị, sang bước 13.
13. Kiểm tra xem tầng được gọi có phải là phần tử đầu tiên trong hàng đợi lên không, nếu đúng sang bước 15, ngược lại sang bước 14.
14. Kiểm tra xem tầng được gọi có nhỏ hơn giá trị nhỏ nhất (Up_min) trong hàng đợi lên hay không, nếu đúng thì sang bước 15, ngược lại sang bước 20.
15. Gán giá trị Up_min = giá trị tầng được gọi, sang bước 20.
16. Đưa giá trị tầng hiện tại vào ô nhớ tạm Temp, sang bước 17.
17. So sánh giá trị ô nhớ Temp với 1, nếu bằng về bước 20, ngược lại sang bước 18.
18. Kiểm tra giá trị ô nhớ Temp có trong hàng đợi hay chưa, có thì chuyển sang bước 20, chưa sang bước 19.
19. Gán giá trị ô nhớ Temp vào hàng đợi xuống, tăng giá trị số phần tử có trong hàng đợi xuống lên một giá trị, sang bước 20.
20. Kết thúc chương trình.

2.4 CHƯƠNG TRÌNH XỬ LÝ PHÍM CHUNG KEYBOARD

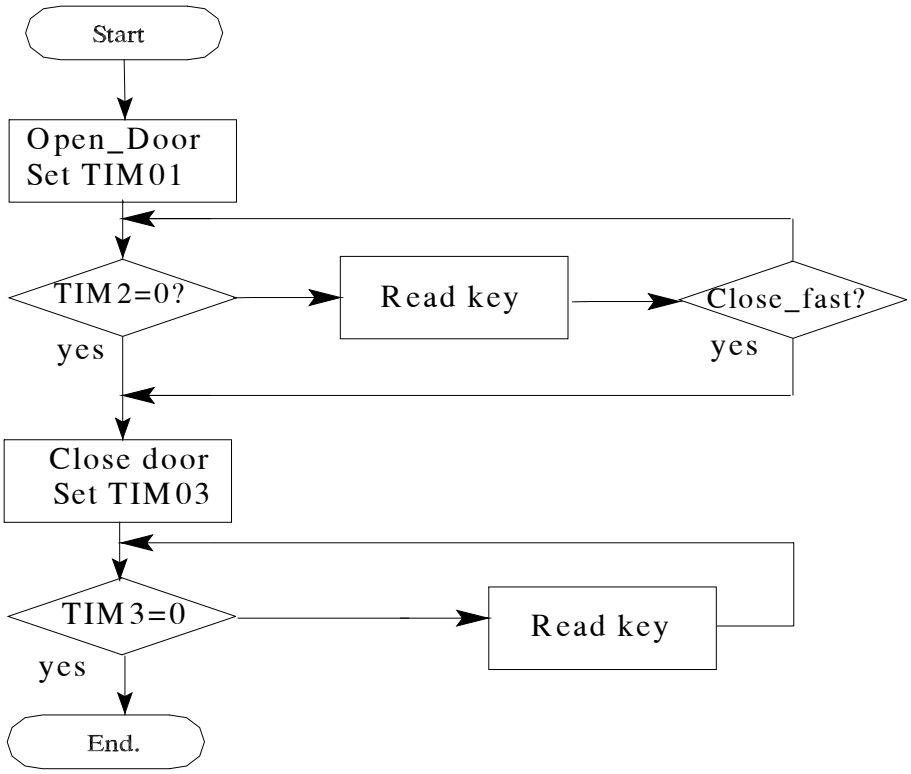


Hình 2-10 : Sơ đồ thuật toán của chương trình bàn phím

1. Kiểm tra xem có phím gọi tầng không, nếu có chuyển sang bước 2, ngược lại sang bước 8.
2. Kiểm tra cờ busy, nếu busy = 1 chuyển sang bước 3, ngược lại sang bước 6.
3. Kiểm tra cờ run, nếu run = 0 chuyển sang bước 4, nếu run = 1 chuyển sang bước 5.
4. Gọi chương trình con SBN 002, sang bước 8.
5. Gọi chương trình con SBN 003, sang bước 8.
6. Gọi chương trình con SBN 001, sang bước 8.
7. Reset cờ key_buff1 (có phím gọi tầng) = 0, sang bước 8.
8. Kiểm tra xem có phím gọi thang không, có sang bước 9, ngược lại sang bước 21.
9. Kiểm tra cờ run, run = 0 thì sang bước 10, run = 1 sang bước 12.
10. Kiểm tra cờ busy, busy = 1 thì sang bước 11, busy = 0 sang bước 13.
11. Kiểm tra cờ run, run = 0 thì sang bước 15, run = 1 sang bước 14.
12. Kiểm tra cờ busy, busy = 1 thì sang bước 17, busy = 0 sang bước 16.
13. Gọi chương trình con SBN 007, sang bước 20.
14. Gọi chương trình con SBN 009, sang bước 20.

15. Gọi chương trình con SBN 008, sang bước 20.
16. Gọi chương trình con SBN 004, sang bước 20.
17. Kiểm tra cờ run, run = 0 thì sang bước 19, run = 1 sang bước 18.
18. Gọi chương trình con SBN 005, sang bước 20.
19. Gọi chương trình con SBN 006, sang bước 20.
20. Reset cờ key_buff 02 (có phím gọi thang) = 0.
21. Kết thúc chương trình.

2.5 CHƯƠNG TRÌNH ĐÓNG - MỞ CỬA (DOOR).



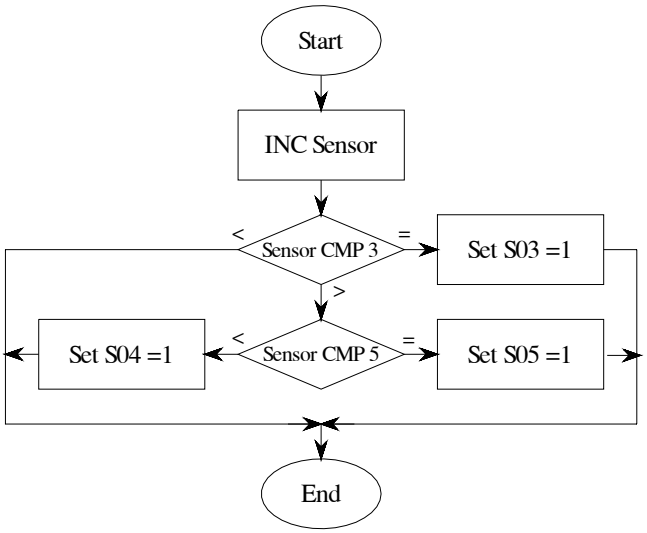
Hình 2-11: Sơ đồ thuật toán của chương trình cửa (Door).

1. Ra lệnh mở cửa; đặt bộ đếm TIM 002, chuyển sang bước 2.
2. Kiểm tra xem đã hết thời gian trễ chưa, nếu chưa thì sang bước 3, ngược lại sang bước 5.
3. Gọi chương trình đọc phím (Read_key), sang bước 4.
4. Kiểm tra xem có cờ đóng cửa nhanh không, nếu có thì sang bước 5, ngược lại về bước 2.

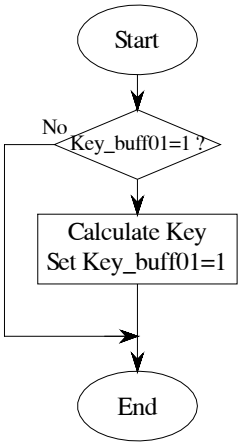
5. Ra lệnh đóng cửa, đặt bộ TIM 003, sang bước 6.
6. Kiểm tra xem đã hết thời gian trễ chưa, nếu chưa thì sang bước 7, ngược lại sang bước 8.
7. Gọi chương trình đọc phím (Read_key), sang bước 6.
8. Kết thúc chương trình.

2.6.1 CÁC CHƯƠNG TRÌNH NGẮT.

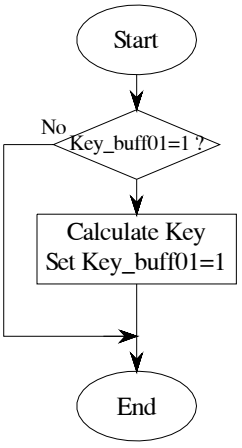
2.6.1 Chương trình ngắt sensor.



Sơ đồ thuật toán xử lý ngắt Sensor



Sơ đồ thuật toán xử lý ngắt bàn phím gọi tầng



Sơ đồ thuật toán xử lý ngắt bàn phím gọi thang

Hình 2-12 : Các sơ đồ thuật toán xử lý ngắt.

1. Tăng giá trị sensor lên một giá trị, chuyển sang bước 2.
2. So sánh giá trị của sensor với số 3, nếu bằng sang bước 3, lớn hơn sang bước 4, nếu nhỏ hơn sang bước 6.
3. Set cờ S03 = 1, sang bước 7.
4. So sánh giá trị của sensor với số 5, nếu bằng sang bước 5, nhỏ hơn sang bước 6.
5. Set cờ S05 = 1, Reset giá trị sensor = 0, chuyển sang bước 7.
6. Set cờ S04 = 1, chuyển sang bước 7.
7. Kết thúc chương trình.

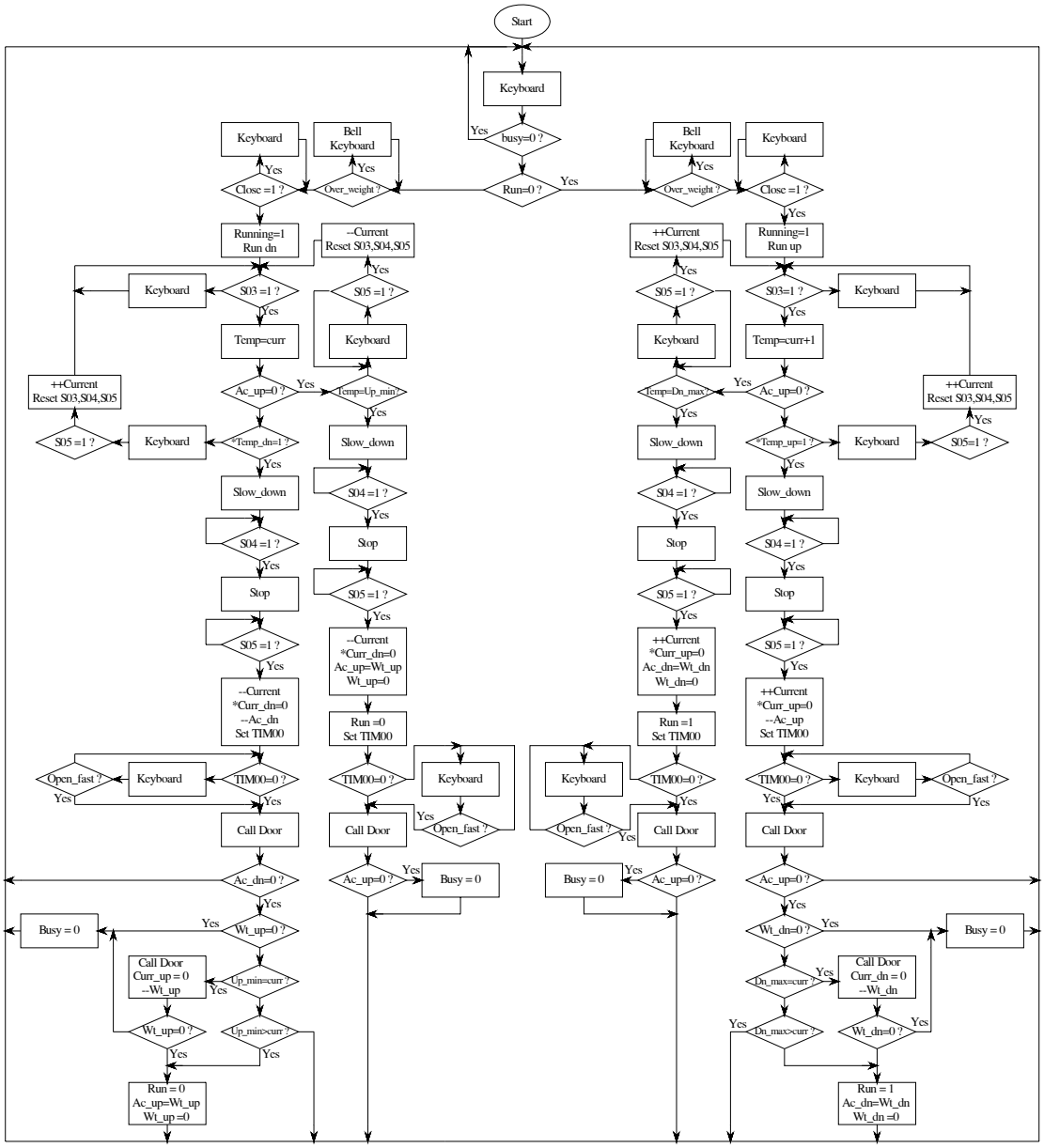
2.6.2 Chương trình ngắt đọc phím gọi tầng.

1. Kiểm tra xem phím bấm trước đó đã được xử lý chưa, nếu chưa (key_buff = 1) thì sang bước 3, rồi (key_buff1 = 0) thì sang bước 2.
2. Tính toán và đưa ra mã của phím được gọi (Scan code), set key_buff1=1.
3. Kết thúc chương trình.

2.6.3 Chương trình ngắt đọc phím gọi thang.

1. Kiểm tra xem phím bấm trước đó đã được xử lý chưa, nếu chưa (key_buff = 1) thì sang bước 3, rồi (key_buff2 = 0) thì sang bước 2.
2. Tính toán và đưa ra mã của phím được gọi (Scan code), set key_buff2=1.
3. Kết thúc chương trình.

2.7 CHƯƠNG TRÌNH CHÍNH



Hình 2-13: Lưu đồ thuật toán của chương trình chính điều khiển thang máy nhà cao tầng.

1. Gọi chương trình đọc bàn phím (Keyboard), chuyển sang bước 2.
2. Kiểm tra cờ busy, busy = 0 chuyển sang bước 1, busy=chuyển sang bước 3.
3. Kiểm tra cờ run, run = 0 chuyển sang bước 4, run = 1 chuyển sang bước 50.
4. Kiểm tra cờ Over_weight (quá tải), nếu Over_weight = 0 chuyển sang bước 6, Over_weight = 1 chuyển sang bước 5.
5. Báo chuông quá tải, gọi chương trình Keyboard, chuyển sang bước 4.
6. Kiểm tra cờ đóng cửa hoàn toàn (Close), close = 1 chuyển sang bước 8, ngược lại chuyển sang bước 7.
7. Gọi chương trình đọc bàn phím (Keyboard), chuyển sang bước 6.
8. Thiết lập cờ running = 1 (đang chạy), và ra lệnh chạy lên, chuyển sang bước 9.
9. Kiểm tra sensor S03, nếu có chuyển sang bước 11, ngược lại chuyển sang bước 10.
10. Gọi chương trình đọc bàn phím (Keyboard) , chuyển sang bước 9.
11. Gán Temp = current + 1 (current là tầng hiện tại), chuyển sang bước 12.
12. Kiểm tra Ac_up (số phần tử có trong hàng đợi lên cần được phục vụ trong hành trình hiện tại), Ac_up = 0 chuyển sang bước 37, ngược lại chuyển sang bước 13.
13. Kiểm tra xem tầng sắp đến có cần dừng không, nếu có chuyển sang bước 17, ngược lại chuyển sang bước 14.
14. Gọi chương trình đọc bàn phím (Keyboard) , chuyển sang bước 15.
15. Kiểm tra sensor S05, nếu có chuyển sang bước 16, ngược lại chuyển sang bước 14.
16. Tăng chỉ số tầng hiện tại, Reset các sensor S03, S04, S05, chuyển sang bước 9.
17. Ra lệnh giảm tốc, chuyển sang bước 18.
18. Kiểm tra Sensor S04, nếu có chuyển sang bước 19, ngược lại chuyển sang bước 18.
19. Ra lệnh dừng, chuyển sang bước 20.
20. Kiểm tra sensor S05, nếu có chuyển sang bước 21, ngược lại chuyển sang bước 20.
21. Tăng giá trị tầng hiện tại, xoá hàng đợi, trừ số phần tử trong hàng đợi lên đi một giá trị , set TIM0 (trễ thời gian chờ mở cửa), chuyển sang bước 22.
22. Kiểm tra TIM0, nếu hết thời gian trễ chuyển sang bước 25, ngược lại chuyển sang bước 23.
23. Gọi chương trình đọc bàn phím (Keyboard) , chuyển sang bước 24.

24. Kiểm tra cờ mở cửa nhanh (Open_fast), nếu có chuyển sang bước 25, ngược lại chuyển sang bước 22.
25. Gọi chương trình mở cửa (door) , chuyển sang bước 26.
26. Kiểm tra Ac_up (số phần tử có trong hàng đợi lên cần được phục vụ trong hành trình hiện tại), Ac_up = 0 chuyển sang bước 27, ngược lại chuyển sang bước 1.
27. Kiểm tra số phần tử trong hàng đợi xuống Wt_dn, Wn_dn = 0 thì chuyển sang bước 28, ngược lại chuyển sang bước 29.
28. Thiết lập cờ busy = 0, chuyển sang bước 1.
29. So sánh giá trị Dn_max (phần tử lớn nhất trong hàng đợi xuống) với tầng hiện tại , nếu Dn_max = current chuyển sang bước 30, ngược lại chuyển sang bước 31.
30. Gọi chương trình mở cửa, xoá phần tử trong hàng đợi xuống, giảm số phần tử trong hàng đợi xuống đi một giá trị, chuyển sang bước 32.
31. So sánh giá trị Dn_max (phần tử lớn nhất trong hàng đợi xuống) với tầng hiện tại , nếu Dn_max > current chuyển sang bước 1, ngược lại chuyển sang bước 33.
32. Kiểm tra xem có phần tử trong hàng đợi xuống không, nếu có chuyển sang bước 33, ngược lại chuyển sang bước 28.
33. Thiết lập cờ run =1, gán Ac_dn = Wt_dn, gán Wt_dn =0 (chuyển phần tử từ hàng đợi chờ được phục vụ sang hàng đợi cần được phục vụ), chuyển sang bước 1.
34. So sánh giá trị Temp với Dn_max, nếu Temp = Dn_max thì chuyển sang bước 38, ngược lại chuyển sang bước 35
35. Gọi chương trình đọc bàn phím (Keyboard), chuyển sang bước 37.
36. Kiểm tra sensor S05, nếu có chuyển sang bước 36, ngược lại chuyển sang bước 37.
37. Tăng giá trị tầng hiện tại, reset các sensor S03, S04 và S05, chuyển sang bước 9.
38. Ra lệnh giảm tốc, chuyển sang bước 39.
39. Kiểm tra Sensor S04, nếu có chuyển sang bước 40, ngược lại chuyển sang bước 39.
40. Ra lệnh dừng, chuyển sang bước 41.
41. Kiểm tra sensor S05, nếu có chuyển sang bước 42, ngược lại chuyển sang bước 41.

42. Tăng giá trị tầng hiện tại, xoá hàng đợi, trừ số phần tử trong hàng đợi xuống đi một giá trị , chuyển Ac_dn = Wt_dn, gán Wt_dn = 0, chuyển sang bước 44.
43. Thiết lập cờ run =1, set TIM0 (trễ thời gian chờ mở cửa),chuyển sang bước 44.
44. Kiểm tra TIM0, nếu hết thời gian trễ chuyển sang bước 47, ngược lại chuyển sang bước 45.
45. Gọi chương trình đọc bàn phím (Keyboard) , chuyển sang bước 46.
46. Kiểm tra cờ mở cửa nhanh (Open_fast), nếu có chuyển sang bước 47, ngược lại chuyển sang bước 44.
47. Gọi chương trình mở cửa (door) , chuyển sang bước 48.
48. Kiểm tra Ac_dn (số phần tử có trong hàng đợi xuống cần được phục vụ trong hành trình hiện tại), Ac_dn = 0 chuyển sang bước 49, ngược lại chuyển sang bước 1.
49. Thiết lập cờ busy = 0, chuyển sang bước 1.
50. Kiểm tra cờ Over_weight (quá tải), nếu Over_weight = 0 chuyển sang bước 51, Over_weight = 1 chuyển sang bước 50.
51. Báo chuông quá tải, gọi chương trình Keyboard, chuyển sang bước 50.
52. Kiểm tra cờ đóng cửa hoàn toàn (Close), close = 1 chuyển sang bước 53, ngược lại chuyển sang bước 54.
53. Gọi chương trình đọc bàn phím (Keyboard), chuyển sang bước 52.
54. Thiết lập cờ running = 1 (đang chạy), và ra lệnh chạy xuống, chuyển sang bước 55.
55. Kiểm tra sensor S03, nếu có chuyển sang bước 57, ngược lại chuyển sang bước 56.
56. Gọi chương trình đọc bàn phím (Keyboard) , chuyển sang bước 57.
57. Gán Temp = current (current là tầng hiện tại), chuyển sang bước 58.
58. Kiểm tra Ac_dn (số phần tử có trong hàng đợi lên cần được phục vụ trong hành trình hiện tại), Ac_dn = 0 chuyển sang bước 80, ngược lại chuyển sang bước 59.
59. Kiểm tra xem tầng sắp đến có cần dừng không, nếu có chuyển sang bước 63, ngược lại chuyển sang bước 60.
60. Gọi chương trình đọc bàn phím (Keyboard) , chuyển sang bước 61.
61. Kiểm tra sensor S05, nếu có chuyển sang bước 62, ngược lại chuyển sang bước 59.
62. Giảm chỉ số tầng hiện tại, Reset các sensor S03, S04, S05, chuyển sang bước 55.

63. Ra lệnh giảm tốc, chuyển sang bước 64.
64. Kiểm tra Sensor S04, nếu có chuyển sang bước 65, ngược lại chuyển sang bước 64.
65. Ra lệnh dừng, chuyển sang bước 66.
66. Kiểm tra sensor S05, nếu có chuyển sang bước 67, ngược lại chuyển sang bước 66.
67. Giảm giá trị tầng hiện tại, xoá hàng đợi, trừ số phần tử trong hàng đợi xuống đi một giá trị , set TIM0 (trễ thời gian chờ mở cửa), chuyển sang bước 68.
68. Kiểm tra TIM0, nếu hết thời gian trễ chuyển sang bước 69, ngược lại chuyển sang bước 71.
69. Gọi chương trình đọc bàn phím (Keyboard) , chuyển sang bước 70.
70. Kiểm tra cờ mở cửa nhanh (Open_fast), nếu có chuyển sang bước 71, ngược lại chuyển sang bước 68.
71. Gọi chương trình mở cửa (door) , chuyển sang bước 72.
72. Kiểm tra Ac_dn (số phần tử có trong hàng đợi lên cần được phục vụ trong hành trình hiện tại), Ac_dn = 0 chuyển sang bước 73, ngược lại chuyển sang bước 1.
73. Kiểm tra số phần tử trong hàng đợi xuống Wt_up, Wn_up = 0 thì chuyển sang bước 74, ngược lại chuyển sang bước 75.
74. Thiết lập cờ busy = 0, chuyển sang bước 1.
75. So sánh giá trị Up_min (phần tử nhỏ nhất trong hàng đợi lên) với tầng hiện tại , nếu Up_min = current chuyển sang bước 77, ngược lại chuyển sang bước 76.
76. Gọi chương trình mở cửa, xoá phần tử trong hàng đợi lên, giảm số phần tử trong hàng đợi lên đi một giá trị, chuyển sang bước 78.
77. So sánh giá trị Up_min (phần tử nhỏ nhất trong hàng đợi lên) với tầng hiện tại , nếu Up_min > current chuyển sang bước 1, ngược lại chuyển sang bước 79.
78. Kiểm tra xem có phần tử trong hàng đợi lên không, nếu có chuyển sang bước 79, ngược lại chuyển sang bước 74.
79. Thiết lập cờ run =1, gán Ac_up = Wt_up, gán Wt_up =0 (chuyển phần tử từ hàng đợi chờ được phục vụ sang hàng đợi cần được phục vụ), chuyển sang bước 1.
80. So sánh giá trị Temp với Up_min, nếu Temp = Up_min thì chuyển sang bước 84, ngược lại chuyển sang bước 81.
81. Gọi chương trình đọc bàn phím (Keyboard), chuyển sang bước 82.

82. Kiểm tra sensor S05, nếu có chuyển sang bước 83, ngược lại chuyển sang bước 80.
83. Giảm giá trị tầng hiện tại, reset các sensor S03, S04 và S05, chuyển sang bước 55.
84. Ra lệnh giảm tốc, chuyển sang bước 85.
85. Kiểm tra Sensor S04, nếu có chuyển sang bước 86, ngược lại chuyển sang bước 85.
86. Ra lệnh dừng, chuyển sang bước 87.
87. Kiểm tra sensor S05, nếu có chuyển sang bước 88, ngược lại chuyển sang bước 87.
88. Giảm giá trị tầng hiện tại, xoá hàng đợi, trừ số phần tử trong hàng đợi lên đi một giá trị , chuyển Ac_up = Wt_up, gán Wt_up = 0, chuyển sang bước 89.
89. Thiết lập cờ run =1, set TIM0 (trễ thời gian chờ mở cửa),chuyển sang bước 90.
90. Kiểm tra TIM0, nếu hết thời gian trễ chuyển sang bước 93, ngược lại chuyển sang bước 91.
91. Gọi chương trình đọc bàn phím (Keyboard) , chuyển sang bước 92.
92. Kiểm tra cờ mở cửa nhanh (Open_fast), nếu có chuyển sang bước 93, ngược lại chuyển sang bước 90.
93. Gọi chương trình mở cửa (door) , chuyển sang bước 94.
94. Kiểm tra Ac_up (số phần tử có trong hàng đợi xuống cần được phục vụ trong hành trình hiện tại), Ac_up = 0 chuyển sang bước 95, ngược lại chuyển sang bước 1.
95. Thiết lập cờ busy = 0, chuyển sang bước 1.

Trên cơ sở các thuật toán đã trình bày, chương trình điều khiển thang máy đã được viết theo sơ đồ Ladder trên phần mềm SYSMAC do hãng OMRON cung cấp, đã được soát lỗi và nạp vào PLC. Tuy nhiên do không có đủ điều kiện (thiếu các bàn phím đầu vào, các rơ le trung gian v.vv...) nên không thể chạy thử nghiệm chương trình. Do đó trong phần tiếp theo, em sẽ trình bày chương trình mô phỏng thang máy trên máy tính.

Chương III

THUYẾT MINH SƠ ĐỒ NGUYÊN LÝ

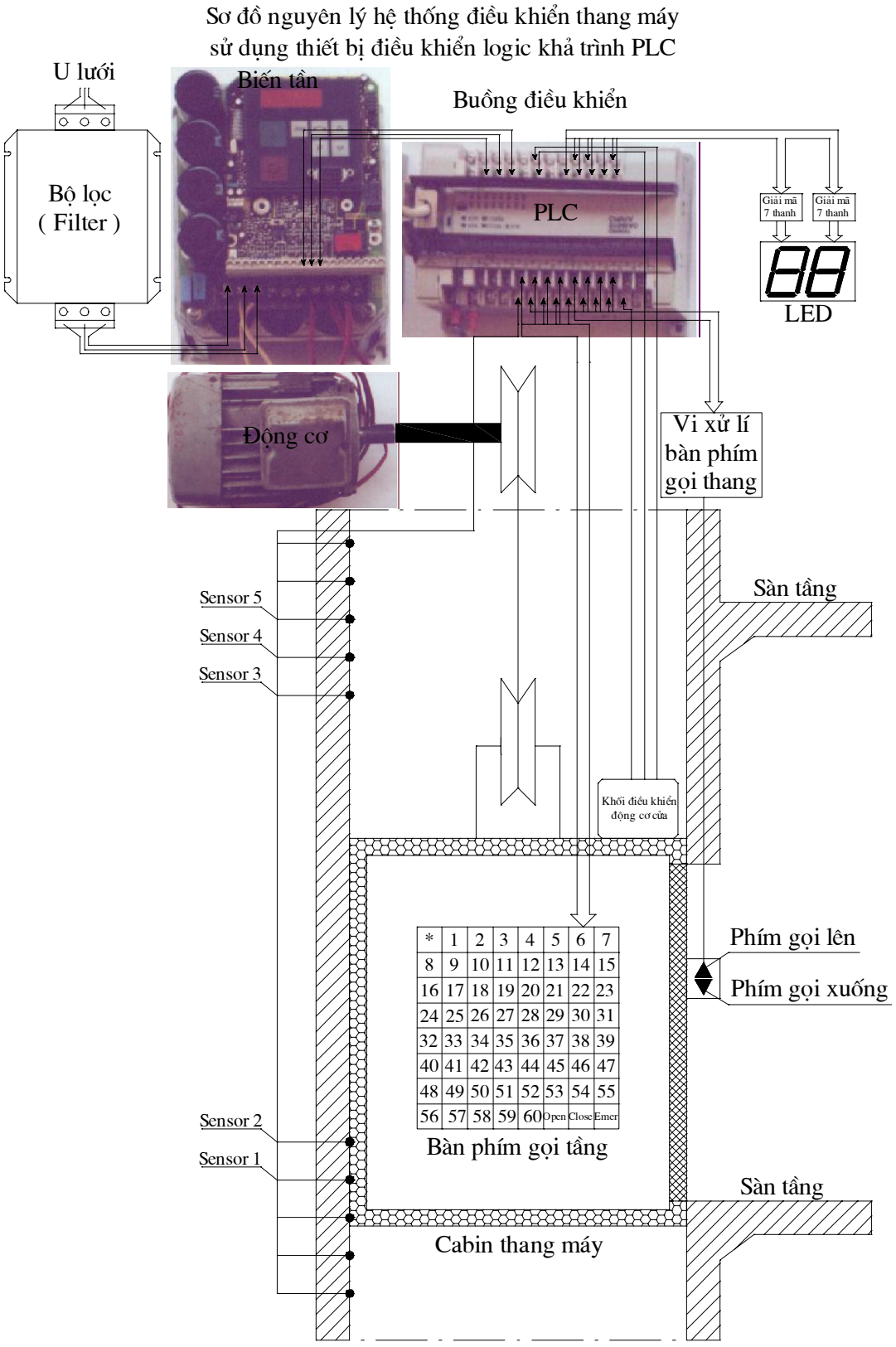
3.1 GIỚI THIỆU CHUNG SƠ ĐỒ NGUYÊN LÝ TOÀN HỆ THỐNG

Sơ đồ nguyên lý chung cho toàn hệ thống được mô tả trên hình 3-1. Trong đó đối tượng được điều khiển là cabin thang máy. Động cơ truyền động chính là động cơ không đồng bộ rotor lồng sóc. Động cơ được cung cấp nguồn bởi biến tần, là biến tần 3 pha loại MICRO MASTER của hãng SIEMENS (Đức) chế tạo. Trước đầu vào của biến tần có lắp bộ lọc để chống nhiễu ảnh hưởng đến lưới xoay chiều. Toàn bộ hệ thống được điều khiển bởi thiết bị điều khiển logic khả trình PLC , là loại PLC CPM1A-40CDR do hãng OMRON (Nhật bản) chế tạo gồm 24 đầu vào (Input), 16 đầu ra (Output).

Biến tần có 3 đầu vào số để tổ hợp thành 8 tần số đặt trước cho phép nó hoạt động khi có tín hiệu đầu vào tương ứng. Các đầu vào này được đánh số từ 10 đến 12 và được nối vào các đầu ra từ IR 010.00 đến IR 010.02 tương ứng trên PLC. PLC sẽ điều khiển biến tần hoạt động theo các tần số đã được đặt trước này phù hợp với giản đồ vận tốc tối ưu cho điều khiển thang máy.

Để cung cấp các tín hiệu cần thiết cho quá trình điều khiển, trong sơ đồ có sử dụng bàn phím gọi tầng được đặt trong cabin thang máy gồm 64 phím trong đó các phím từ 1 đến 60 được dùng cho việc gọi đến các tầng tương ứng, 3 phím khác là Open để gọi mở cửa nhanh, Close để gọi đóng cửa nhanh, Emer để gọi dừng thang máy khẩn cấp. Bàn phím gọi tầng có 7 đầu ra được nối vào đầu vào của PLC như sau :

- đầu báo có phím gọi (báo ngắt) được đưa vào đầu vào IR 000.04 để gọi chương trình ngắt SBN 001 cho xử lý phím gọi tầng. Chương trình này sẽ tổ hợp các đầu vào IR 000.06 đến IR 000.11 (6 đầu) tương ứng với các bit dữ liệu từ D0 đến D5 của bàn phím gọi tầng đưa đến theo mã nhị phân và xác định được vị trí tầng cần đến để đưa vào ô nhớ đệm và báo cờ keybuff01(có phím gọi tầng) cho chương trình chính xử lý.



Hình 3-1: Sơ đồ nguyên lý toàn hệ thống

Bàn phím gọi thang gồm 118 phím, trong đó tại mỗi tầng đặt 2 nút, một cho gọi thang máy đi lên, một cho gọi thang máy đi xuống, trừ trường hợp đặc biệt là tầng 1 chỉ có phím gọi lên và tầng thượng chỉ có nút gọi xuống. Bàn phím gọi thang có 8 đầu ra được nối vào đầu vào của PLC như sau :

- đầu báo có phím gọi (báo ngắt) được đưa vào đầu vào IR 000.05 để gọi chương trình ngắt SBN 002 cho xử lý phím gọi thang. Chương trình này sẽ tổ hợp các đầu vào IR 001.00 đến IR 001.06 (7 đầu) tương ứng với các bit dữ liệu từ D0 đến D6 của bàn phím gọi thang đưa đến theo mã nhị phân và xác định được vị trí tầng cần đến để đưa vào ô nhớ đệm và báo cờ keybuff02(có phím gọi thang) cho chương trình chính xử lý.

Để có thể phát hiện được vị trí thang máy khi cần điều chỉnh tốc độ cũng như hãm dừng, trong đồ án có sử dụng các sensor phi tiếp điểm theo nguyên lý quang học được đánh số từ Sensor 1 đến Sensor 5, tất cả các sensor này được đấu song song vào đầu vào ngắt 000.03 để gọi chương trình ngắt SBN 000 cho xử lý sensor.

Việc cung cấp thông tin về vị trí tầng hiện tại mà thang đang hoạt động được thực hiện nhờ các đèn LED. Các đèn LED này được nối vào các đầu ra IR 100.00 đến IR 100.03 cho chữ số hàng chục và IR 100.04 đến 100.07 cho chữ số hàng đơn vị thông qua các mạch giải mã 16 từ 4 sử dụng EPROM 2764.

Ngoài ra, hệ thống động cơ đóng mở cửa cũng được PLC điều khiển thông qua đầu vào IR 001.07 báo tín hiệu cửa đã đóng hoàn toàn để cho phép động cơ khởi động, trong trường hợp ngược lại thì động cơ sẽ không được phép khởi động; tín hiệu cho phép động cơ cửa quay theo chiều mở cửa ra được lấy trên đầu ra IR 010.06, tín hiệu cho phép động cơ cửa quay theo chiều đóng cửa vào được lấy trên đầu ra IR 010.07.

Để đảm bảo an toàn trong các trường hợp sự cố, các thiết bị an toàn hoạt động độc lập với phần điều khiển như phanh dù, lò xo thuỷ lực v.v... sẽ hoạt động. Ngoài ra trong buồng thang còn đặt một phím bấm chuông hoạt động nhờ nguồn một chiều cung cấp độc lập để báo tín hiệu khi có sự cố mất điện lưới.

3.2 NGUYÊN TẮC HOẠT ĐỘNG CỦA HỆ THỐNG

Khi chương trình đã được viết xong, được kiểm định và nạp vào PLC cùng với các điều kiện khác cho thang máy hoạt động được đảm bảo thì có thể khởi động hệ thống. Trước hết ta cấp nguồn cho PLC và nó chuyển sang trạng thái RUN (đèn RUN sáng). Sau đó đóng cầu dao cung cấp nguồn cho biến tần và thang máy sẵn sàng hoạt động. Tại thời điểm hoạt động lần đầu tiên, thang máy

được đặt tham số tầng hoạt động hiện tại là 1 và nó sẽ thay đổi trong suốt quá trình hoạt động sau này. Tham số này sẽ được lưu lại trong suốt quá trình hoạt động kể cả khi mất nguồn cung cấp và được các LED hiển thị khi thang máy hoạt động.

Để hệ thống hoạt động tốt thì phải tiến hành bảo dưỡng định kỳ theo các quy định của nhà sản xuất các thiết bị đã sử dụng trong hệ thống.

Chương IV

CHƯƠNG TRÌNH MÔ PHỎNG THANG MÁY

4.1 MỤC ĐÍCH

Để minh hoạ các thuật toán đã được sử dụng khi viết chương trình cho PLC, trong bản đồ án này em đã viết một chương trình mô phỏng thang máy trên ngôn ngữ C. Chương trình cũng đồng thời được sử dụng để kiểm tra đặc tính thực tế của hệ thống nhờ phần vẽ đồ thị tín hiệu phản hồi tốc độ được lấy qua bộ chuyển đổi A/D lắp trên card giao tiếp.

4.2 SỬ DỤNG CHƯƠNG TRÌNH

4.2.1 Màn hình của chương trình

Toàn bộ phần màn hình của chương trình được mô tả trên hình 4-1; trong đó gồm các phần :

1. Cabin thang máy.
2. Các phím gọi thang đặt tại cửa tầng.
3. LED hiển thị tầng hiện tại của thang máy.
4. Tầng hiên tại của toà nhà.
5. Phím gọi tầng gần nhất.
6. Phím gọi thang gần nhất.
7. Hàng đợi lên.
8. Hàng đợi xuống.
9. Đồ thị tốc độ thực của thang máy được vẽ theo số liệu đầu ra của biến tần.

4.2.2 Các quy định về sử dụng phím trong chương trình.

Do phải sử dụng bàn phím của máy tính nên trong chương trình, việc bấm phím được quy định như sau:

a. Phím gọi thang

Phím gọi thang lên: Người gọi phải bấm vào số tầng mà người đó đang đứng (từ 1 đến 999) nhờ sử dụng các phím số trên bàn phím và bấm phím mũi tên lên, ví dụ có người đang ở tầng 30 cần đi lên thì người đó phải ấn số 3, sau đó là số 0 rồi ấn phím mũi tên lên.

Phím gọi thang xuống: Người gọi phải bấm vào số tầng mà người đó đang đứng (từ 1 đến 999) và bấm phím mũi tên xuống.

b. Phím gọi tầng

Người gọi phải bấm vào số tầng mà người đó cần đến (từ 1 đến 999) và bấm phím Enter.

4.2.3. Khởi động chương trình.

Trước khi chạy chương trình, công việc cần thiết là phải kiểm tra các đầu nối điều khiển từ card giao tiếp đến biến tần, kiểm tra card giao tiếp, kiểm tra nguồn cung cấp cho biến tần để đảm bảo an toàn trong khi chạy.

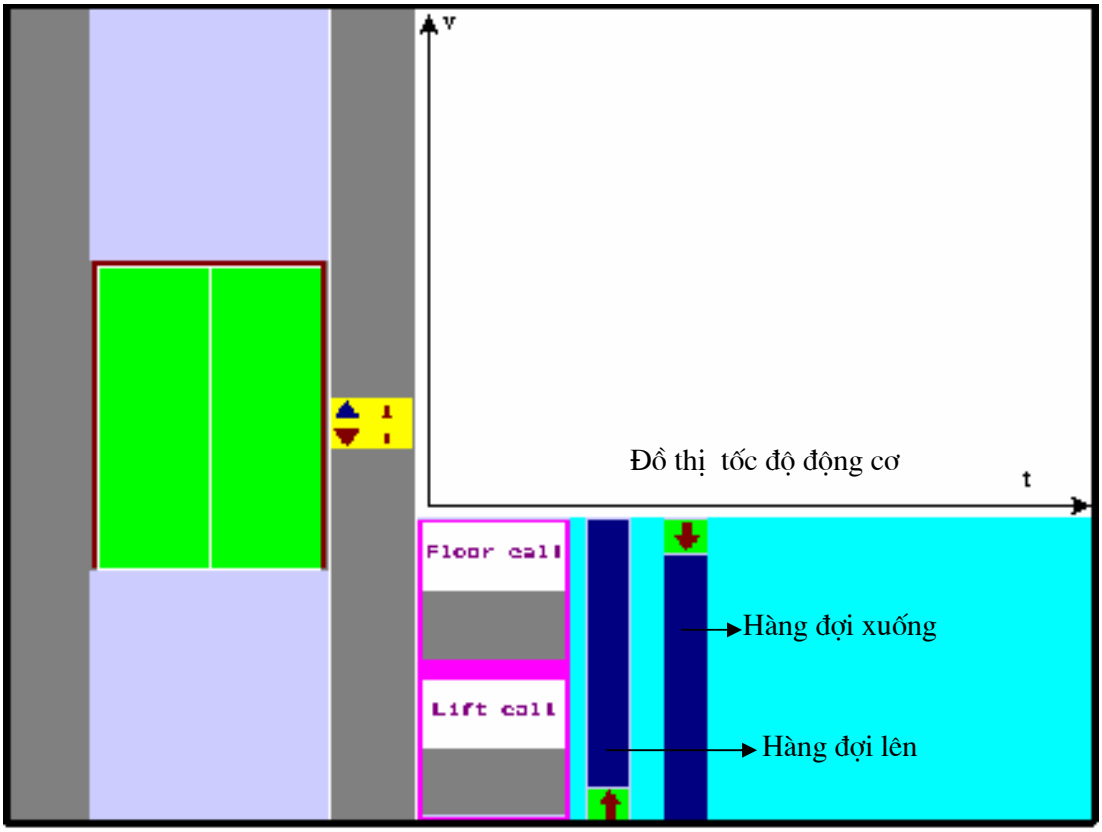
Chương trình mô phỏng thang máy nằm gọn trong một file có tên là Lift.exe; do chương trình sử dụng phần đồ hoạ nên nhất thiết bạn phải có các file đồ hoạ để trong cùng thư mục với chương trình nói trên.

Muốn khởi động chương trình, ta chỉ cần thực hiện việc chạy chương trình đuôi EXE thông thường trên DOS hoặc trên WINDOWS.

4.3. CÁC HOẠT ĐỘNG CỦA CHƯƠNG TRÌNH.

Khi khởi động xong, chương trình bắt đầu chạy thì thang máy được đặt tại tầng 1 và sẵn sàng chờ đọc các tín hiệu gọi thang cũng như gọi tầng. Nếu có tín hiệu gọi hợp lệ, chương trình sẽ quét và đưa vào hàng đợi.

Khi hàng đợi có người cần phục vụ, thang máy trong chương trình mô phỏng sẽ hoạt động theo đúng hành trình cần phục vụ. Đồng thời nhờ sử dụng mạch biến đổi trên cổng ra số nằm trên một card giao tiếp giữa máy tính với thiết bị ngoại vi nên chương trình có thể điều khiển trực tiếp một biến tần, mà được nối với một động cơ không đồng bộ rotor lồng sóc, với vận tốc tuân theo giản đồ tối ưu dành cho truyền động thang máy. Tốc độ động cơ cũng được vẽ mô phỏng theo thời gian thực nhờ sử dụng mạch chuyển đổi A/D trên card giao tiếp nói trên. Ngoài ra, vị trí tầng hiện tại được chương trình hiển thị ra trên hàng LED có trên card giao tiếp.



Hình 4-1: Màn hình chương trình mô phỏng hoạt động của thang máy.

TÀI LIỆU THAM KHẢO

[1] Vũ Quang Hồi, Nguyễn Văn Chất, Nguyễn Thị Liên Anh
Trang bị điện-điện tử máy công nghiệp dùng chung, NXB Giáo dục 1994

[2] **CPM1A- Programmable Controllers - OPERATION MANUAL**
OMRON 1996

[3] **Programmable Controllers - Biginner's Guide to PLC**, OMRON 1996

[4] **Programmable Controllers - PROGRAMMING MANUAL**, OMRON 1996

[5] Nguyễn Xuân Quỳnh
Lý thuyết mạch lôgic và kỹ thuật số , NXB Đại học và giáo dục chuyên nghiệp 1991

[6] Nguyễn Quốc Trung
Xử lý tín hiệu và lọc số , NXB Khoa học kỹ thuật 1998

[7] Ngô Diên Tập
Đo lường và điều khiển bằng máy tính , NXB Khoa học và kỹ thuật 1997

[8] Trần Bá Thái, Nguyễn Trí Công
Kỹ thuật vi xử lý , NXB Khoa học và kỹ thuật 1983

[9] Trần Bá Thái, Nguyễn Trí Công, Nguyễn Văn Tam, Vũ Duy Lợi, Phí Mạnh Lợi
Điều khiển và ghép nối các thiết bị ngoại vi , NXB Thống kê 1987

[10] Văn Thế Minh
Kỹ thuật vi xử lý , NXB Giáo dục 1997

[11] Nguyễn Mạnh Giang
Kỹ thuật ghép nối máy vi tính , NXB Giáo dục 1997

[12] Trần Quang Vinh
Cấu trúc máy vi tính , NXB Giáo dục 1997

[13] Lê Văn Doanh, Phạm Khắc Chương
Kỹ thuật vi điều khiển , NXB Khoa học kỹ thuật 1998

[14] Phạm Công Ngô
Lý thuyết điều khiển tự động , NXB Khoa học kỹ thuật 1994

[15] Nguyễn Phùng Quang
Điều khiển tự động truyền động điện xoay chiều ba pha, NXB Giáo dục 1996

[16] Bùi Quốc Khánh, Phạm Quốc Hải, Nguyễn Văn Liễn, Dương Văn Nghi
Điều chỉnh tự động truyền động điện, NXB Khoa học kỹ thuật 1996

[17] Perter Norton
Nhập môn Assembler , NXB giáo dục 1995

[18] **Microprocessor and IC families**
Intel Corporation 1993

[19] Võ Quang Lạp và Trần Xuân Minh
Kỹ thuật biến đổi, ĐH Kỹ thuật công nghiệp Thái nguyên.

MỤC LỤC

BUỔI THỰC HÀNH SỐ 1

TIẾP CẬN THIẾT BỊ, NGÔN NGỮ VÀ HOÀN CHỈNH BÀI THỰC HÀNH

1. TIẾP CẬN THIẾT BỊ

1.1. Giới thiệu thiết bị

- Module Standard CPU 200 & 300
- Digital Expanded Module
- Analog Expanded Module
- Communication Module

1.2. Kết nối thiết bị ngoại vi với CPU, Expanded Module

1.2.1. Các thiết bị vào ra số, tương tự được sử dụng trong khóa thực hành

- Các loại tác động cơ
- Các loại cảm biến số và tương tự
- Các thiết bị chấp hành: valve, motor, relay.

1.2.2. Xác định các đặc trưng của đầu nối

- Nguồn cung cấp, đặc tính
- Khái niệm Bit, Byte, Word, Double Word; Gán địa chỉ và tên gọi của biến vào ra

2. NGÔN NGỮ

2.1. Cú pháp, cấu trúc

- Nắm rõ chu kỳ quét của chương trình, bản chất các lệnh, tham số, toán hạng và tổ chức các lệnh theo quá trình hoạt động của hệ thống.

2.2. Công cụ lập trình

- Làm quen các chức năng phần mềm
- Sử dụng các câu lệnh
- Tải và đọc chương trình

2.3. Định nghĩa cấu hình

Nhà sản xuất thiết kế và chế tạo các loại CPU từ 300 trở lean với mục đích sử dụng cho các giải pháp mạng tích hợp hệ thống sản xuất tự động. Do đó để CPU làm việc và hiểu được các module tương tác với nó thì người dùng phải định nghĩa cấu hình cứng cho chúng.

Các bước định cấu hình phần cứng cho CPU S7-300

2.3.1. Sinh viên tự tạo cho bản thân một thư mục riêng biệt với MSSV với đường dẫn D:\TN_PLC\ .

2.3.2. Định nghĩa cấu hình cứng S7-300

⇨ File ⇨ New ⇨ sử dụng Browse để chọn thư mục đã tạo, gõ tên file vào hộp thoại Name, Enter và xuất hiện khung cửa sổ làm việc của môi trường S7-300 có chứa file có tên mà chúng ta vừa tạo.

⇨ Vào menu Insert⇨ Station ⇨ SIMATIC 300 Station.

⇨ Kích chuột vào ô dấu (+) của tên file, xuất hiện biểu tượng SIMATIC 300 và chọn chúng, biểu tượng Hardware bên cửa sổ phải của màn hình làm việc S7-300 xuất hiện.

⇨ Nhấn chuộc 2 lần vào Hardware thì xuất hiện cửa sổ định nghĩa cấu hình cứng cho S7 –300.

⇨ Kích vào dấu (+) của biểu tượng SIMATIC 300, sau đó kích vào dấu (+) của RACK 300 và chọn Rail.

⇨ Kích vào dấu (+) của PS-300 và chọn PS 307 2A.

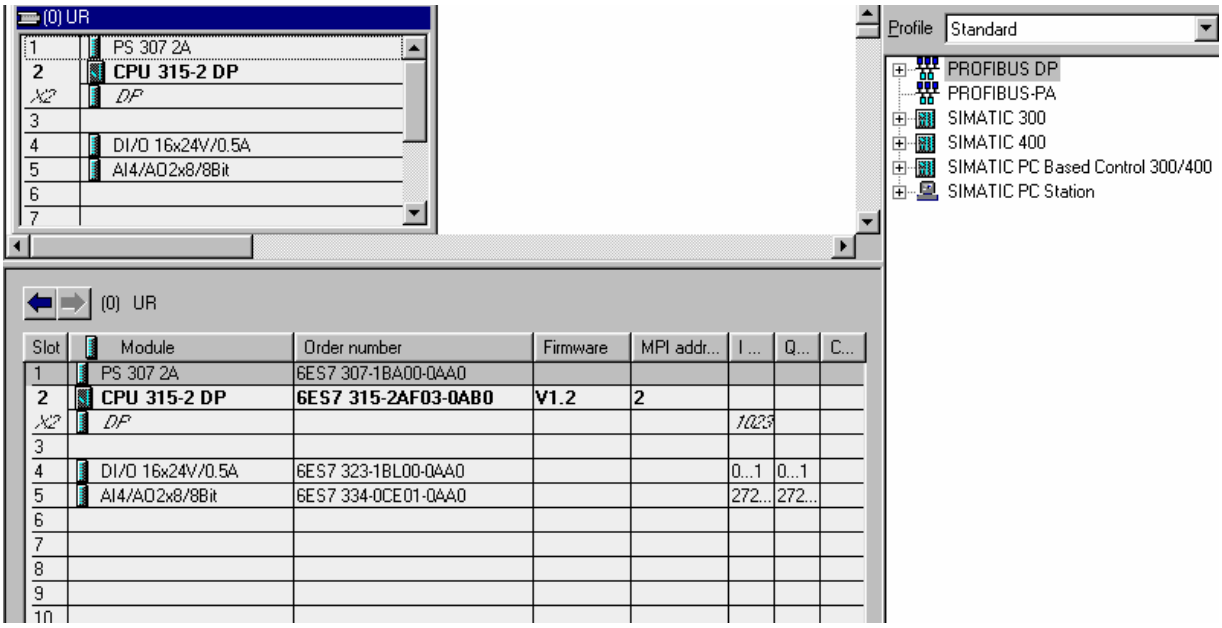
⇨ Kích dấu (+) CPU-300, chọn CPU 315-2 DP, sau đó chọn 6ES7 315-2AF03-0AB0.

⇨ Kích dấu (+) SM-300 rồi kích DI/DO –300 chọn SM 323 DI16/DO16x24V/0.5A

⇨ Kích dấu (+) SM-300 rồi kích AI/AO chọn SM334 AI4/AO2x8/8Bit.

⇨ Kích dấu (+) SIMATIC 300, sau đó chọn biểu tượng Blocks và xuất hiện cửa sổ viết chương trình cho S7-300. Có thể lập trình theo Ladder.

⇨ Chọn biểu tượng Download để tải chương trình xuống CPU S7-300.



2.3.3. Hiển thị trạng thái tín hiệu phụ thuộc chương trình : dùng để quan sát trạng thái hoạt động tín hiệu hiện hành của các toán hạng. Để thực hiện công việc này chúng ta vào menu Debug -> Monitor (ở các dạng phương pháp lập trình).

3. HOÀN CHỈNH BÀI THỰC HÀNH

Trình tự các bước thực hành được xây dựng logic trên cơ sở thiết kế và xây dựng một hệ thống điều khiển bằng PLC hoặc một hệ thống mạng PLC thực tiễn. Điều này giúp cho sinh viên đạt được kết quả tốt nhất khi tham gia vào quá trình công tác sau này.

Mô tả một ví dụ mẫu về việc hoàn chỉnh bài thực hành.

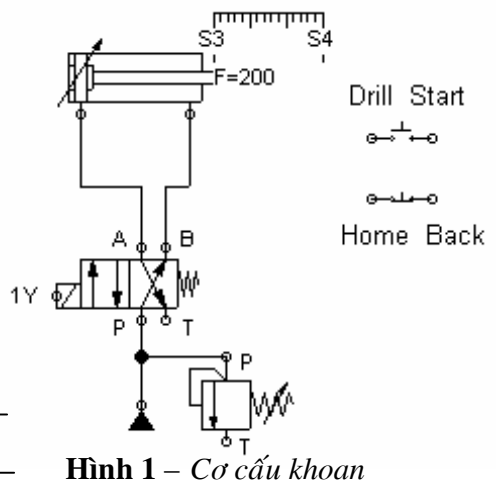
Phát biểu bài toán:

Viết chương trình cho PLC điều khiển thiết bị khoan thủy lực của một đầu khoan tự động mô tả hình 1, với yêu cầu kỹ thuật như sau: Đưa chi tiết vào vị trí cần khoan, rồi ấn nút Drill Start, mũi khoan xoay, đầu khoan tịnh tiến và khoan chi tiết. Đạt đủ chiều sâu khoan cần thiết (S4 tác động), đầu khoan tự động quay về và kết thúc một chu kỳ khoan

tại S3. Trong quá trình gia công nếu xảy ra sự cố ta ấn nút Home Back, đầu khoan tự động lui về.

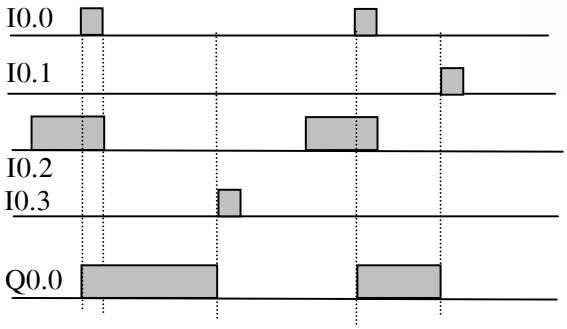
a) Bảng gán nhiệm vụ I/O

Input (ngỏ vào)		Output (ngỏ ra)	
Tên gọi	Địa chỉ	Tên gọi	Địa chỉ
Drill Start	I0.0	Sol 1Y	Q0.0
Home Back	I0.1		
S3	I0.2		
S4	I0.3		



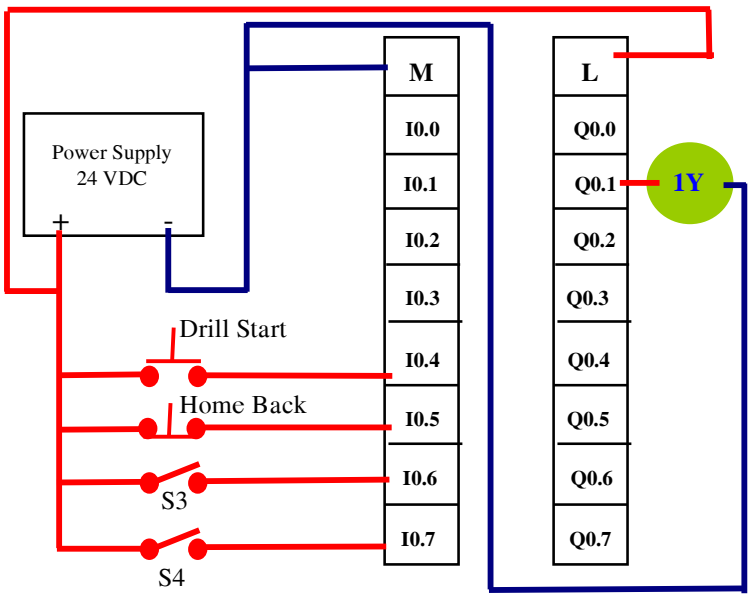
Hình 1 – *Cơ cấu khoan*

b) Biểu đồ trạng thái



Biểu đồ trạng thái theo thời gian

c) Kết nối thiết bị ngoại vi với PLC



d) Chương trình

Network1: *// Khi Drill Start là 1 thì 1Y được nhớ lên 1.*
A "Drill Start"
A "S3"
S "Solenoid 1Y"
Network2: *// S4 là 1 hoặc Home Back là 0 thì xóa 1Y*
O "S4"
ON "Home Back"
R "Solenoid 1Y"

BUỔI THỰC HÀNH SỐ 2
THỰC HÀNH CÁC LỆNH VỀ BIT LOGIC TRÊN S7-200

1. ***Mục đích.***

   Giúp SV sử dụng thông thạo được các lệnh về tiếp điểm qua các mô hình thực.

2. ***Yêu cầu:***
   - SV Chuẩn bị kiến thức trước cho buổi thực hành.
   - Tìm hiểu thiết bị ngoại vi đã lắp ráp trên mô hình.
   - Xác định các lỗ jack ứng với địa chỉ quy định để kết nối thiết bị với PLC.
3. ***Thời lượng thực hành***: 5 tiết
4. ***Nội dung các bước thực hiện*** (phần dành cho Sinh viên):
   a. ***Nội dung :***
      - Thực hành các lệnh tiếp điểm.
      - Thực hành các lệnh Set (S), Reset (R).
      - Thực hành các lệnh xét cạnh lên (EU), cạnh xuống (ED).

   b. ***Các bước thực hiện ở mỗi bài :***
      - Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).
      - Vẽ biểu đồ trạng thái của quá trình hoạt động hệ thống.
      - Lắp mô hình thí nghiệm (nếu có).
      - Trình bày chương trình ở dạng STL (Statement List).
5. ***Kết luận & đánh giá*** (phần dành cho Giảng Viên hướng dẫn):

Bài 1: *Đão chiều động cơ*

Viết chương trình điều khiển để đão chiều động cơ điện DC. Nhấn PB_CW để động cơ quay cùng chiều kim đồng hồ, nhấn PB_CCW để động cơ quay ngược chiều kim đồng hồ. Nhấn PB_STOP để dừng động cơ.

Ghi chú : có sử dụng mô hình (Động cơ DC).

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

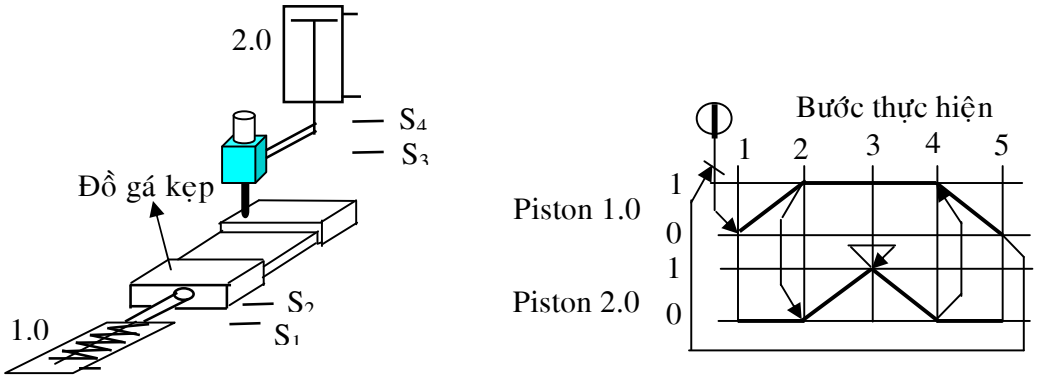
3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

Error!

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Bài 2: Hệ thống kẹp chặt và khoan chi tiết

Viết chương trình điều khiển hệ thống khoan chi tiết với phôi được cấp bằng tay



a. Sơ đồ nguyên lý *b. Biểu đồ trạng thái*

Hình 2 – *Nguyên lý làm việc của máy khoan*

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

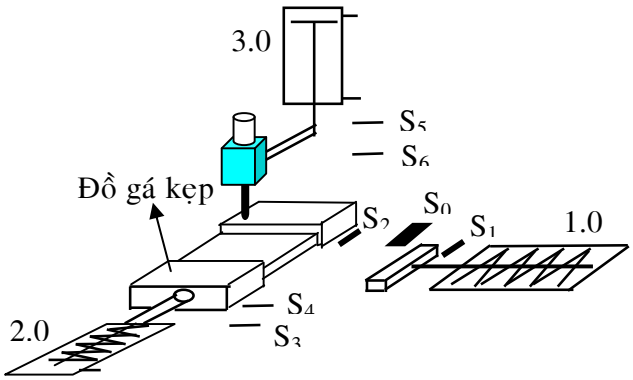
2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Bài 3:

Viết chương trình điều khiển hệ thống khoan chi tiết với phôi được cấp tự động, hệ thống được mô tả ở hình 3. Nguyên lý làm việc của hệ thống khoan làm việc như sau: Phôi được chuyển bằng băng tải, đến ngay vị trí gia công thì S_0 tác động làm piston 1.0 được tác động bởi van 5/2/1 side sẽ nay chi tiết vào vị trí kẹp phôi và S_2 được tác động, piston 1.0 trở về vị trí ban đầu. Khi S_1 tác động, piston 2.0 dịch chuyển má kẹp đến kẹp chặt phôi từ S_3 -> S_4 , khi S_4 tác động thì piston 3.0 sẽ mang đầu khoan đi xuống để thực hiện gia công lỗ và đạt đến chiều sâu lỗ, tức là S_6 tác động thì piston 3.0 giật về, khi S_5 tác động thì piston 2.0 giật má kẹp về vị trí ban đầu để tháo chi tiết ra.



Hình 3 – *Cơ cấu khoan với phôi cấp tự động*

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.

3. Vẽ sơ đồ kết nối thiết bị với PLC.

4. Trình bày chương trình ở dạng STL (Statement List) :

BUỔI THỰC HÀNH SỐ 3

THỰC HÀNH TIMER VÀ COUNTER TRÊN S7-200

1. ***Mục đích.***

   Giúp SV hiểu được bản chất của Timer và sử dụng thông thạo được các lệnh về Timer.

2. ***Yêu cầu:***
   - Chuẩn bị các thiết bị ngoại vi.
   - Xác định các lỗ jack ứng với địa chỉ quy định để kết nối thiết bị với PLC.
   - Nắm vững các kiến thức lý thuyết về PLC và kỹ thuật số.
3. ***Thời lượng thực hành***: 5 tiết
4. ***Nội dung các bước thực hiện*** (phần dành cho Sinh viên):
   a. ***Nội dung :***
      - Thực hành Timer loại TON, TOF.
      - Thực hành Timer loại có nhớ (TONR).
      - Thực hành các bô đếm: CU, CD, CUD

   b. ***Các bước thực hiện ở mỗi bài :***
      - Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).
      - Vẽ biểu đồ trạng thái của quá trình hoạt động hệ thống.
      - Lắp mô hình thí nghiệm (nếu có).
      - Trình bày chương trình ở dạng STL (Statement List).
5. ***Kết luận & đánh giá*** (phần dành cho Giảng Viên hướng dẫn):

Bài 4: ***Chuyển chế độ hoạt động của động cơ từ chế độ sao (Y) sang tam giác (Δ).***

Tác động tín hiệu khởi động (bằng nút nhấn PB Start) động cơ khởi động ở chế độ (Y), sau 5 giây thì động cơ chuyển sang hoạt động ở chế độ (Δ). Dừng động cơ tác động tín hiệu ngừng (bằng nút nhấn PB Stop).

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

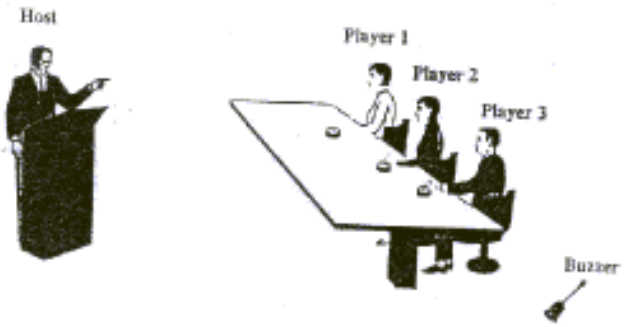
4. **Trình bày chương trình ở dạng STL (Statement List) :**

Bài 5: ***Thiết kế xác định định quyền ưu tiên.***

Yêu cầu của luật chơi:

1) Sau khi giám khảo hoàn tất câu hỏi.

2) Ba thí sinh sẽ nhấn công tắc gắn ở trên bàn nằm phía trước mặt họ để giành quyền đầu tiên là người trả lời câu hỏi.
3) Buzzer sẽ phát tiếng kêu trong 10 giây sau khi một trong các thí sinh chạm vào công tắc.
4) Đèn báo ở phía trước mỗi thí sinh sẽ sáng và chỉ sẽ được reset bằng công tắc của thầy giám khảo.



Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

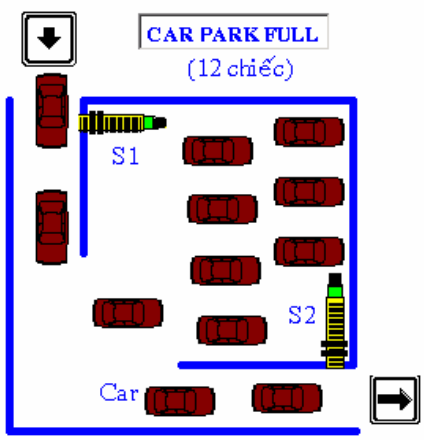
2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Bài 6: *Hệ thống điều khiển xe ra vào bãi xe*

Hệ thống điều khiển bãi đậu xe chứa tối đa là 12 chiếc mô tả **hình 4**. Mỗi lần xe vào, PLC tự động tăng thêm 1 bởi cảm biến phát hiện xe S1. Bất kỳ một chiếc xe nào đi ra khỏi bãi, PLC sẽ tự động giảm đi 1 bởi cảm biến phát hiện S2. Khi 12 chiếc xe được đăng ký, bảng hiệu đầy xe sẽ được sáng lên thông báo đến các xe không được vào nữa.



Hình 4 – *Bãi đậu xe*

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

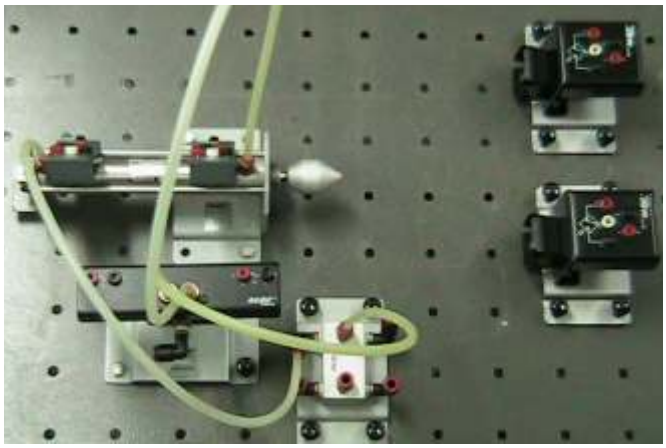
Buổi thực hành số 4:
THỰC HÀNH MỘT SỐ LỆNH BIT LOGIC TRÊN S7-300

1. ***Mục đích.***
   Giúp SV tiếp cận với bộ điều khiển PLC S7-300 và thực hiện một số lệnh cơ bản.
2. ***Yêu cầu:***
   - Chuẩn bị các thiết bị ngoại vi.
   - Xác định các lỗ jack ứng với địa chỉ quy định để kết nối thiết bị với PLC.
   - Nắm vững các kiến thức lý thuyết về PLC S7-300.
3. ***Thời lượng thực hành***: 5 tiết
4. ***Nội dung các bước thực hiện*** (phần dành cho Sinh viên):
   a. ***Nội dung :***
      - Định cấu hình phần cứng cho PLC S7-300.
      - Thực hành một số lệnh cơ bản về bit logic.

   b. ***Các bước thực hiện ở mỗi bài :***
      - Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).
      - Vẽ biểu đồ trạng thái của quá trình hoạt động hệ thống.
      - Lắp mô hình thí nghiệm (nếu có).
      - Trình bày chương trình ở dạng STL (Statement List).
5. ***Kết luận & đánh giá*** (phần dành cho Giảng Viên hướng dẫn):

Bài 7: *Hệ thống tự động đóng mở cửa cho xe ra vào kho hàng.*

Khi xe đang tiến về gần cửa kho, cảm biến SS1 nhận dạng được xe và cửa sẽ được mở ra đến gặp giới hạn hành trình trên LS2 thì cửa dừng lại rồi xe chạy vào. Khi cảm biến quang SS2 đặt phía trong cổng cửa nhận dạng được xe đã đi qua khỏi cửa thì cửa sẽ được đóng lại, chạm vào giới hạn hành trình dưới LS1 thì cửa dừng lại. Trường hợp xe đi chiều ngược lại cũng tương tự.



Hình 5 - mô hình đóng mở cửa tư động

Chú ý: mô hình hình 5 được thay thế cho hệ thống đóng cửa xe vào kho hàng. Đóng mở cửa bằng piston duỗi ra và thụt vào với valve 5/3/2 side.

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

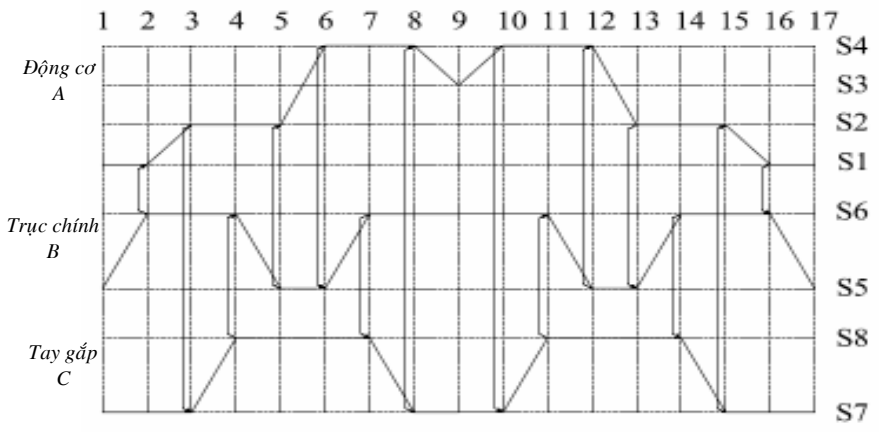
2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Bài 8:

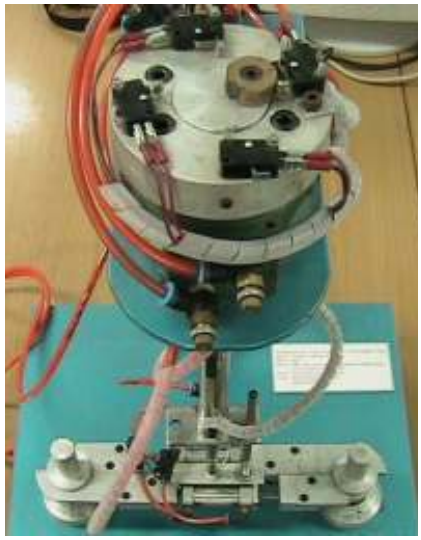
Viết chương trình điều khiển cơ cấu tháo và lắp dao cắt trong máy gia công CNC. Cơ cấu tháo lắp dao làm việc theo nguyên lý được mô tả ở biểu đồ trạng thái hình 6.



Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ



2. **Vẽ sơ đồ kết nối thiết bị với PLC.**

3. Trình bày chương trình ở dạng STL (Statement List) :

Buổi thực hành số 5:
THỰC HÀNH LỆNH TOÁN, CHUYỂN ĐỔI DỮ LIỆU, XỬ LÝ DỮ LIỆU TRÊN S7-300

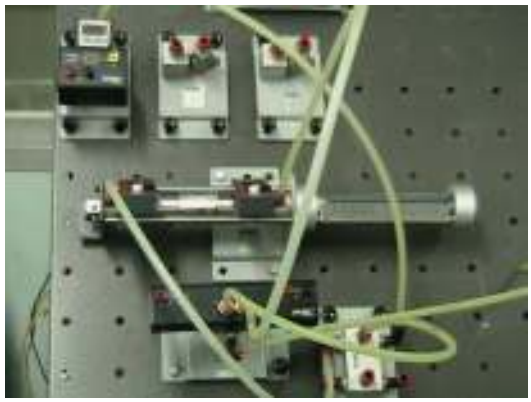
1. ***Mục đích.***
   Giúp SV name rõ và ứng dụng một số lệnh toán học, lệnh chuyển đổi dữ liệu, lệnh truyền và nhận dữ liệu trên thanh ghi và thực hiện việc xử lý dữ liệu PLC S7-300.
2. ***Yêu cầu:***
   - Chuẩn bị các thiết bị ngoại vi.
   - Xác định các lỗ jack ứng với địa chỉ quy định để kết nối thiết bị với PLC.
   - Nắm vững các kiến thức lý thuyết về PLC S7-300.
3. ***Thời lượng thực hành***: 5 tiết
4. ***Nội dung các bước thực hiện*** (phần dành cho Sinh viên):
   a. ***Nội dung :***
      - Thực hành một số lệnh truyền và nhận dữ liệu thanh ghi qua các ô nhớ PQW, PIW của module A/D và D/A.
      - Thực hành các lệnh toán học, so sánh, chuyển đổi.

   b. ***Các bước thực hiện ở mỗi bài :***
      - Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).
      - Vẽ biểu đồ trạng thái của quá trình hoạt động hệ thống.
      - Lắp mô hình thí nghiệm (nếu có).
      - Trình bày chương trình ở dạng STL (Statement List).
5. ***Kết luận & đánh giá*** (phần dành cho Giảng Viên hướng dẫn):

Bài 9:

Hệ thống kiểm tra lực ép của một piston khí nén được mô tả hình 7, trong đó tải ép được mô tả bằng một cơ cấu lò xo được gắn vào đầu thân xy lanh. Dùng cảm biến áp suất để đo áp suất khí tác động vào khoan piston. Ta thay đổi áp suất ép bằng cách điều chỉnh bằng van chỉnh áp.

Viết chương trình biểu diễn giá trị thực của lực ép lên tải ứng với áp suất khí là 350 kPa.



Hình 7 – mô hình kiểm tra lực ép

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Bài 10:

Hệ thống điều khiển vị trí của một quá trình ép thủy lực được mô tả hình 8, trong đó đại lượng vị trí được xác định nhờ một position transducer phản hồi về module A/D.

Viết chương trình biểu diễn giá trị vị trí thực của hành trình dịch chuyển ta có thể quan sát. Viết chương trình điều khiển piston dịch chuyển từ vị trí zêrô (0 mm) đến 90 mm rồi giật về vị trí 75mm rồ dừng hẳn.

Chú ý: tốc độ dịch chuyển chậm với vận tốc 180mm/phút bằng cách điều chỉnh van tiết lưu gắn ở ngõ vào trước van chỉnh hướng.

Hình 8 – Điều khiển vị trí

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ sơ đồ kết nối thiết bị với PLC.**

3. **Trình bày chương trình ở dạng STL (Statement List) :**

Bài 11:

Viết chương trình đọc và hiển thị tải trọng của một vật thể có khối lượng P=1750 kG được đặt trên một tấm phẳng có chiều dày, không bị biến dạng tại điểm O và tấm phẳng này được cố định trên 4 loadcell phân bố theo chiều ngang là 1,2 m, chiều dọc là 1,6 m.

Cho biết đặc tính loadcell m_{max} = 2000kG, điện áp ra 0 – 10 volt.

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Buổi thực hành 6
ĐIỀU KHIỂN TIMER

1. ***Mục đích.***

   Giúp SV tiếp cận và hiểu được bản chất của các Timer của S7-300.

2. ***Yêu cầu:***
   - Chuẩn bị các thiết bị ngoại vi.
   - Xác định các lỗ jack ứng với địa chỉ quy định để kết nối thiết bị với PLC.
   - Nắm vững các kiến thức lý thuyết về PLC S7-300.
3. ***Thời lượng thực hành***: 5 tiết
4. ***Nội dung các bước thực hiện*** (phần dành cho Sinh viên):
   a. ***Nội dung :***
      - Thực hành các chức năng và thành phần của Timer: SP, SE, SD, SF, SS.
      - Đặt các giá trị trì hoãn theo các kiểu dữ liệu khác nhau.

   b. ***Các bước thực hiện ở mỗi bài :***
      - Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).
      - Vẽ biểu đồ trạng thái của quá trình hoạt động hệ thống.
      - Lắp mô hình thí nghiệm (nếu có).
      - Trình bày chương trình ở dạng STL (Statement List).
5. ***Kết luận & đánh giá*** (phần dành cho Giảng Viên hướng dẫn):

Bài 12:

Mô tả việc điều khiển tín hiệu đèn giao thông ở các giao lộ với các thông số về thời gian như sau : Đèn Xanh 1 sáng 25 giây, đèn Đỏ 1 sáng 30 giây, đèn Vàng 1 sáng 5 giây. Khởi động hệ thống bằng cách nhấn PB_START, dừng bằng cách nhấn PB_STOP.
Chú ý : có sử dụng mô hình (Bộ tín hiệu đèn giao thông).

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

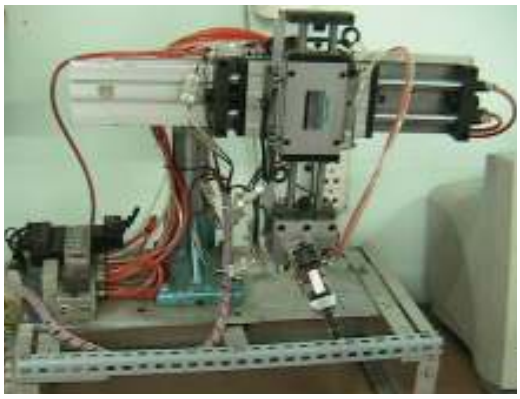
2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Bài 13:

Viết chương trình điều khiển tay máy khí nén toạ độ Castesian gắp sản phẩm và di chuyển đến một vị trí khác, trở đầu sản phẩm (xoay 180^0) rồ nhã ra. Tay máy được mô tả hình 10 bao gồm các chuyển động: OX, OY, XOY, kẹp trong mặt phẳng XOZ làm việc theo biểu đồ trạng thái dưới nay.



Hình 10 – Cơ cấu tay máy gắp sản phẩm

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Buổi thực hành số 7
ĐIỀU KHIỂN COUNTER

1. ***Mục đích.***

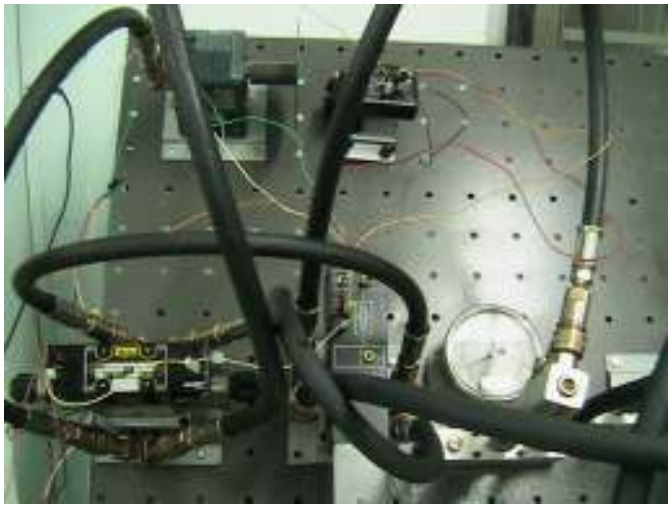
   Giúp SV sử dụng thông thạo được các lệnh về các bộ đếm của PLC S7-300.
2. ***Yêu cầu:***
   - Chuẩn bị các thiết bị ngoại vi.
   - Xác định các lỗ jack ứng với địa chỉ quy định để kết nối thiết bị với PLC.
   - Nắm vững các kiến thức về các bộ đếm,lý thuyết về PLC và kỹ thuật số .
3. ***Thời lượng thực hành***: 5 tiết
4. ***Nội dung các bước thực hiện*** (phần dành cho Sinh viên):
   a. ***Nội dung :***
      - Thực hành Counter loại đếm CU, CD, SC.
      - Gán các giá trị đếm đặt trước, đọc các giá trị đếm tức thời trong thanh ghi dưới dạng BCD, Binary.

   b. ***Các bước thực hiện ở mỗi bài :***
      - Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).
      - Vẽ biểu đồ trạng thái của quá trình hoạt động hệ thống.
      - Lắp mô hình thí nghiệm hoặc khảo sát các mô hình có sẵn.
      - Trình bày chương trình ở dạng STL (Statement List).
5. ***Kết luận & đánh giá*** (phần dành cho Giảng Viên hướng dẫn):

Bài 14:

Viết chương trình điều khiển tốc độ chuyển động của động cơ dầu chạy theo chiều kim đồng hồ. Tốc độ ban đầu của động cơ là 50 v/ph chạy trong 100 vòng tốc độ được tăng lên 100 v/ph, sau 300 vòng thì tốc độ tăng lean 150 v/ph và chạy trong 200 vòng thì dừng. Mô hình được mô tả hình 11. Cho biết:

Valve servo dầu để điều khiển thay đổi lưu lượng dầu vào xy lanh động cơ, dẫn đến tốc độ động cơ thay đổi. Điện áp điều khiển valve servo có tầm 0 – 10 volt tương vận tốc của động cơ 0 – 200v/ph.



Hình 11 – mô hình điều khiển tốc độ động cơ

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Bài 15: ***Thiết kế chương trình điều khiển cho cơ cấu đóng gói sản phẩm.***

Hệ thống đóng gói có nguyên tắc làm việc như sau: sản phẩm được vận chuyển trên một băng tải cấp sản phẩm, một cảm biến SS1 phát hiện sản phẩm đến vị trí chờ. Tiếp tục 3 sản phẩm tiếp theo và được nay đến vị trí chờ, lúc này vị trí chờ sẽ đủ 6 sản phẩm và được piston 3 đi xuống, piston 4di chuyển kẹp 6 chi tiết, rồi piston 3 rút lean và piston 2 di chuyển 6 sản phẩm đến và bỏ vào thùng đang name chờ sẵn trên băng tải vận chuyển sản phẩm đóng gói nhờ cảm biến phát hiện SS2.

Hình 12 – mô hình đóng gói sản

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.

3. Vẽ sơ đồ kết nối thiết bị với PLC.

4. Trình bày chương trình ở dạng STL (Statement List) :

Bài 16:

Viết chương trình phát 100 xung nhịp có tần số 2Hz. Khi nhấn nút phát thì bộ tạo xung mới làm việc. Nhấn nut Stop thì dừng phát.

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Buổi thực hành 8
CHƯƠNG TRÌNH CON TRÊN S7-300

1. ***Mục đích.***

   Giúp SV thực tập một số bài tập có độ phức tạp cao.
2. ***Yêu cầu:***
   - Chuẩn bị các thiết bị ngoại vi.
   - Xác định các lỗ jack ứng với địa chỉ quy định để kết nối thiết bị với PLC.
   - Nắm vững các kiến thức về cách thiết lập cấu hình cho PLC S7-300 .
3. ***Thời lượng thực hành***: 5 tiết
4. ***Nội dung các bước thực hiện*** (phần dành cho Sinh viên):
   a. ***Nội dung :***
      - Phân tích hệ thống làm việc, xác định cấu trúc hoạt động hệ thống, đưa ra kỹ thuật chương trình.
      - Thực hành bài tập về sử dụng chương trình con.

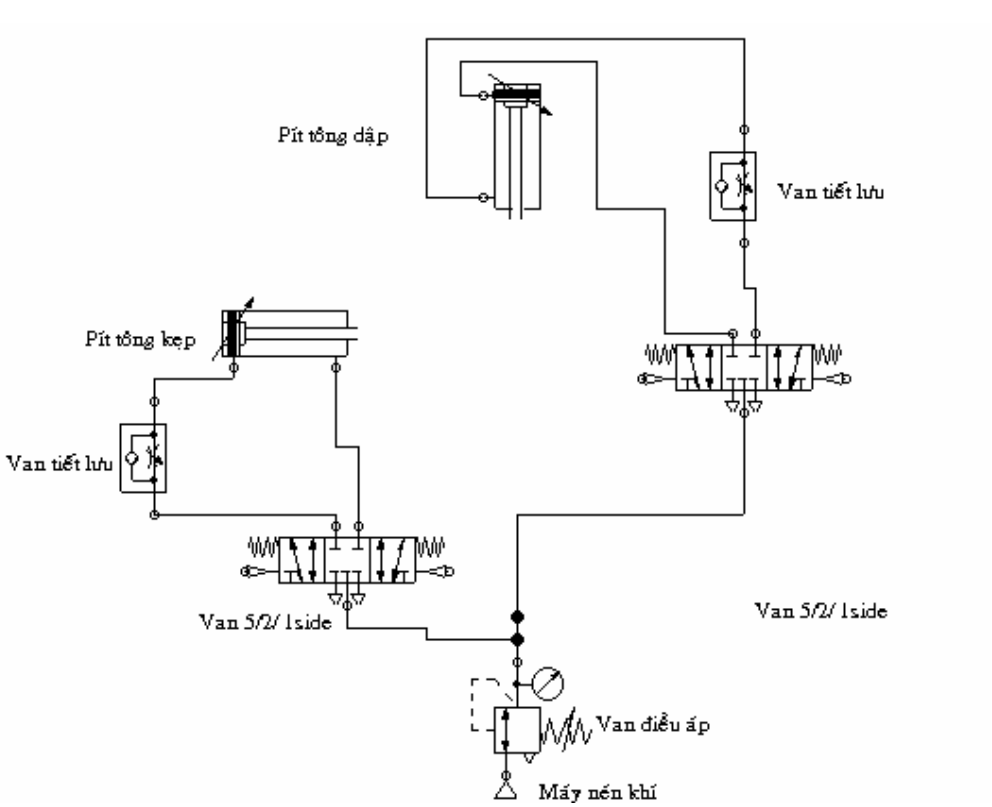
   b. ***Các bước thực hiện ở mỗi bài :***
      - Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).
      - Vẽ biểu đồ trạng thái của quá trình hoạt động hệ thống.
      - Lắp mô hình thí nghiệm (nếu có).
      - Trình bày chương trình ở dạng STL (Statement List).
5. ***Kết luận & đánh giá*** (phần dành cho Giảng Viên hướng dẫn):

Bài 17: Hệ thống điều khiển của một máy dập.

Thực tế trong một hệ thống, máy điều khiển có 2 chế độ vận hành: tự động và bằng tay. Chế độ tự động là máy chạy theo một chương trình đã tạo sẵn; chế độ bằng tay được dùng để thử từng động tác của một cơ cấu trong hệ để kiểm tra sản phẩm tạo ra trước khi làm việc tự động hoặc đôi khi ta sử dụng chế độ này để sản xuất thay cho tự động khi hư hỏng.

Viết chươngtrình sử dụng chương trình con cho hệ thống dập ở hai chế độ, nguyên tắc hoạt động như sau:

- Đầu tiên, chuyển qua chế độ tay đưa 2 pít tông về vị trí A và B. Do hầu hết pít tông nằm ở vị trí lưng chừng của xy lanh.
- Tác động tín hiệu khởi động (nút nhấn PB_START) pít tông kẹp chặt dịch chuyển từ vị trí A đến B thực hiện kẹp chặt phôi, lúc này LS2 được tác động và pít tông dập dịch chuyển từ vị trí C đến D để dập định hình phôi (theo hình dạng khuôn) lúc này LS4 tác động làm cho pít tông dập lùi về C và LS3 tác động. LS3 tác động làm cho pít tông kẹp dịch chuyển từ B về A và LS1 tác động thực hiện lần dập tiếp theo.
- *Chú ý:*
    - *Có sử dụng mô hình (Bộ thí nghiệp khí nén).*
    - *PLC chỉ nhận tín hiệu từ PB_START khi đồng thời LS1 và LS3 bị tác động.*
    - *Van điện từ là 5/3/2side.*



Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

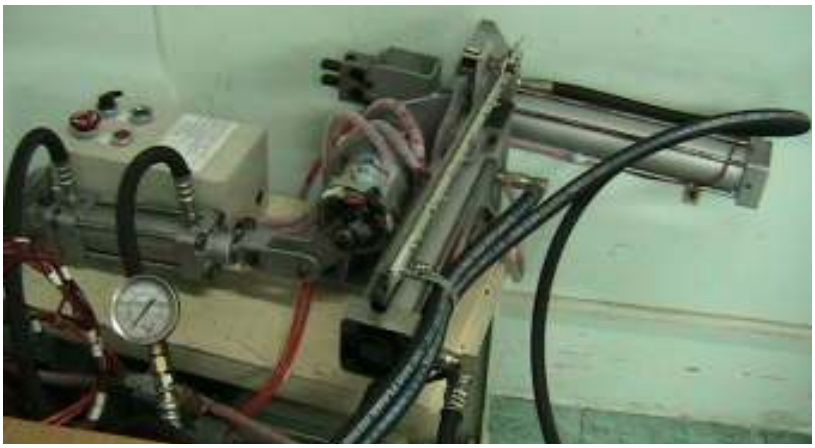
Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. **Trình bày chương trình ở dạng STL (Statement List) :**

Bài 18:



Hình 14 – tay máy gắp sản phẩm bằng thủy lực

Viết chương trình điều khiển tay máy thủy lực gắp sản phẩm nhựa trong các máy ép nhựa mô tả hình 14. Cho biết số sản phẩm có 3 loại khác nhau về kích thước. Kích thước sản phẩm khác nhau có nghĩa là khoảng cách từ vị trí gắp sản phẩm đến vị trí zêrô của tay máy sẽ khác nhau. Khoảng cách này được xác định bởi một cảm biến vị trí. Cho biết khoảng cách này là 240 mm, 275mm và 300mm.

Sinh viên phải thực hiện các phần sau:

1. **Thực hiện bảng gán nhiệm vụ I/0 (Input/Ouput).**

Input (ngỏ vào)		**Output (ngỏ ra)**	
Mô tả	Địa chỉ	Mô tả	Địa chỉ

2. **Vẽ biểu đồ trạng thái của quá trình hoat động hệ thống.**

3. **Vẽ sơ đồ kết nối thiết bị với PLC.**

4. Trình bày chương trình ở dạng STL (Statement List) :

Buổi thực hành số 9 + 10
Thực hành giao tiếp Máy tính và PLC

1. ***Mục đích.***

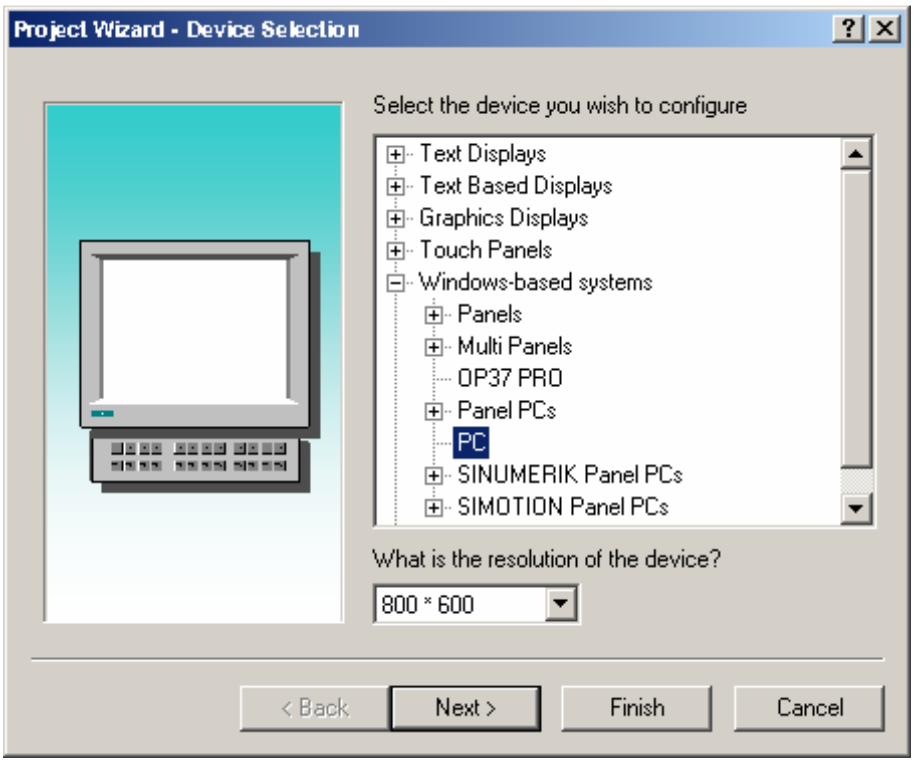
   Giúp SV tìm hiểu về một dạng điều khiển PLC mới, điều khiển và giám sát PLC thông qua máy tính nhờ công cụ Protool/Pro.

2. ***Yêu cầu:***
   - Chuẩn bị các thiết bị ngoại vi.
   - Xác định các lỗ jack ứng với địa chỉ quy định để kết nối thiết bị với PLC.
   - Nắm vững các kiến thức lý thuyết về PLC S7-300 và ProTool Pro.
3. ***Thời lượng thực hành***: 5 tiết
4. ***Nội dung các bước thực hiện*** (phần dành cho Sinh viên) :
   a. ***Nội dung :***
      - Giới thiệu về ProTool Pro V6.0.
      - Thực hành một số truyền nhận dữ liệu cơ bản trực tiếp giữa máy tính và PLC S7-300.

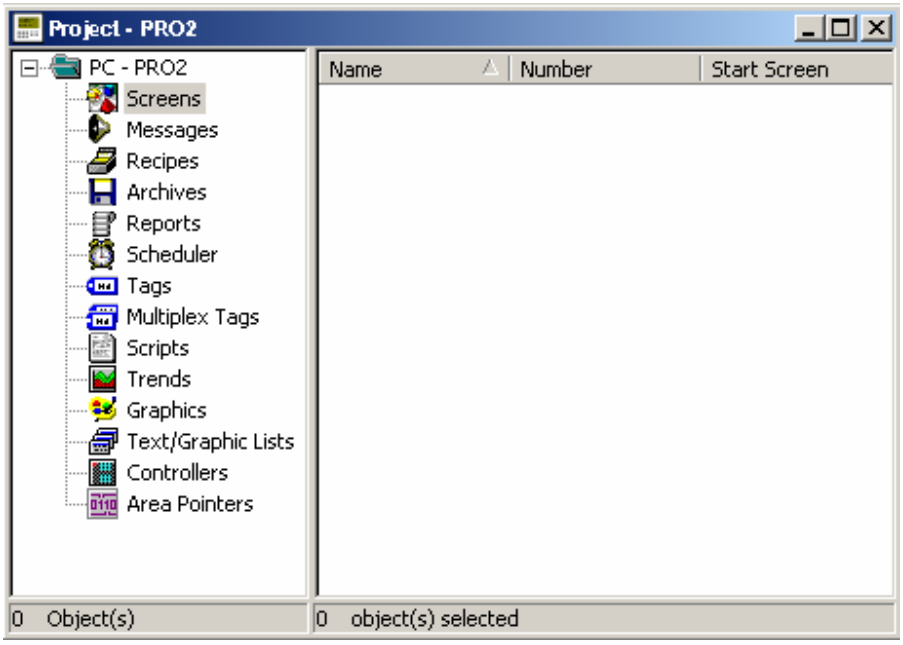
   b. ***Các bước thực hiện ở mỗi bài :***
      - Viết tiểu luận trình bày các bài tập trong phần ProTool.
      - Nhận xét về ProTool.
5. ***Kết luận & đánh giá*** (phần dành cho Giảng Viên hướng dẫn):

I . Giới thiệu về ProTool Pro CS :

- Chọn File ⇒ New để tạo một Project mới. Xuất hiện màn hình sau :



- Chọn PC (Giao tiếp qua máy tính) và chế độ màn hình 800 * 600. Bấm Finish để bắt đầu thực hiện chương trình. Màn hình Project sẽ xuất hiện :



- Nhấn Screens để bắt đầu soạn thảo chương trình.
- Thanh công cụ cho việc soạn thảo giao diện sẽ như sau :

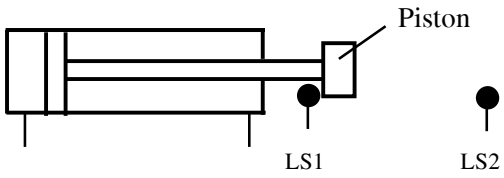
II . Một số bài tập về ProTool :

Bài 19 : Điều khiển hệ thống ép thủy lực.

Nhấn nut PB Start piston dịch chuyển theo chiều A thực hiện quá trình ép khi gặp LS2 piston sẽ giật lùi về gặp LS1 và lại tiếp tục quá trình ép. Ep được 10 lần thì ngưng ép tại vị trí LS1. Khi đang ép, nhấn nút PB Stop thì hệ thống sẽ dừng tại vị trí đó. Chú ý piston có thể name ở vị trí bất kỳ trước khi bắt đầu ép, do đó khi nút PB Start được nhấn thì piston sẽ lùi về LS1 rồi mới bắt đầu quá trình ép, hình 15.

Tạo giao diện mô tả quá trình:

- Tạo nút nhấn START, STOP.
- Thể hiện các giá trị I/O lên màn hình máy tính.
- Mô phỏng sự hoạt động của xylanh ép.
- Nhập vào một giá trị đặt couter, khi máy dập làm đủ số sản phẩm đúng bằng trị đặt trước thì ngừng máy.



Hình 15 – cơ cấu ép thủy lực

Bài 20: Xây dựng giao diện điều khiển và giám sát vị trí và tốc độ của piston ép thủy lực, mô tả hình 16 :

- Tạo nút nhấn START, STOP.
- Xác lập thông số vị trí và tốc độ cho quá trình qua màn hình máy tính.
- Thể hiện sự biến đổi của vị trí và tốc độ piston.

Hình 16 – Điều khiển vị trí và tốc độ piston

Buổi thực hành số 11 +12
Thực hành MẠNG PLC - PROFIBUS

1. ***Mục đích.***

Giúp SV hiểu rõ thêm về mạng PLC theo giải pháp Profibus, quá trình truyền thông dữ liệu trên mạng, những thành phần cơ bản về cứng của mạng, kết nối giữa các PLC chủ và tớ, và các vùng dữ liệu liên quan đến mạng.

2. ***Yêu cầu:***
   - Chuẩn bị các thiết bị ngoại vi.
   - Xác định các lỗ jack ứng với địa chỉ quy định để kết nối thiết bị với PLC.
   - Nắm vững các kiến thức lý thuyết về PLC S7-300 và ProTool Pro.
   - Xác định rõ cấu hình phần cứng mạng.
   - Hiểu rõ về HMI.
3. ***Thời lượng thực hành***: 5 tiết
4. ***Nội dung các bước thực hiện*** (phần dành cho Sinh viên) :
   a. ***Nội dung :***
      - Giới thiệu về mạng Profibus -DP.
      - Thực hành nối cáp mạng.
      - Định cấu hình mạng Profibus - DP.
      - Thu nhận dữ liệu từ các PLC.

   b. ***Các bước thực hiện ở mỗi bài :***
      - Viết tiểu luận trình bày các bài tập trong phần Profibus -DP.
      - Nhận xét về Profibus.

5. ***Kết luận & đánh giá*** (phần dành cho Giảng Viên hướng dẫn) :

I) Giới thiệu về hệ thống mạng Profibus :

1) Khái niệm về Profibus :

Giao thức Profibus – DP được thiết kế cho truyền thông tốc độ cao với các thiết bị xuất nhập từ xa. Có nhiều loại thiết bị Profibus sẵn có của nhiều hãng sản xuất khác nhau. Các thiết bị này cũng có nhiều loại, từ các module xuất nhập đơn giản đến các bộ điều khiển động cơ và các bộ điều khiển lập trình. Trên các mạng Profibus thường có một thiết bị chủ (Master) và các thiết bị tớ (Slave), thiết bị chủ khởi động mạng và cho phép các thiết bị tớ trên mạng hiểu được cấu hình. Thiết bị chủ liên tục ghi dữ liệu ngõ ra đến các thiết bị tớ và đọc các dữ liệu ngõ vào từ các thiết bị tớ. Khi một chủ DP đặt xong c ấu hình cho một tớ thì nó sẽ quản lý thiết bị tớ đó. Nếu có một chủ thứ 2 trên mạng thì nó sẽ bị hạn chế truy xuấtđến thiết bị tớ mà thiết bị chủ thứ nhất quản lý.

2) Định cấu hình phần cứng cho mạng :

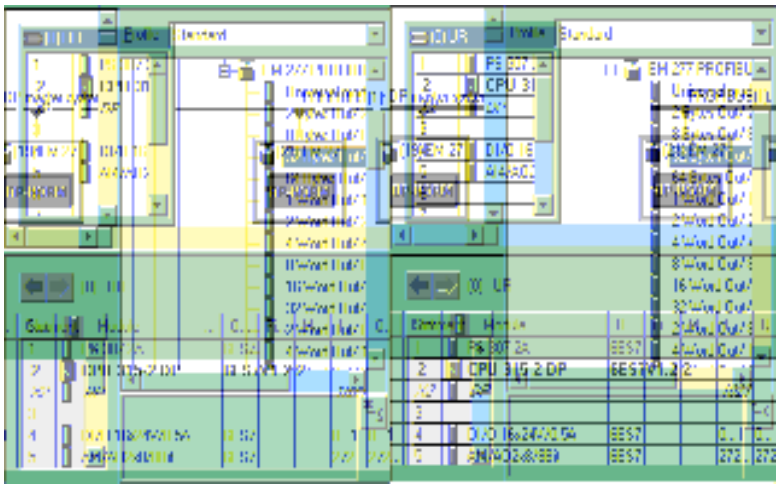
i) Thực hiện nối cáp mạng.

ii) Định cấu hình cho PLC S7-300 (PLC Master) : tương tự cách địng cấu hình cho PLC trước đây.

iii) Chèn cấu hình cho các PLC Slave :

Khi tiến hành đặt cấu hình mạng cho các PLC thì trước tiên ta phải chọn module kết nối, vì module kết nối của PLC S7-300 có đuôi theo sau là DP nên khi tiến hành đặt cấu hình mạng thì ta dựa trên cấu hình phần cứng cho trạm ta đặt.

- Nhấp đúp vào đuôi DP thì nó sẽ hiện ra nhánh Profibus.
- Chọn địa chỉ cho Master bằng cách chúng ta nhấp đúp vào nhánh Profibus thì sẽ hiện ra địa chỉ, địa chỉ mặc định của nó là1.
- Chọn tốc độ truyền thông cho mạng.
- Chèn module EM277 vào trong cấu hình mạng : module EM277 là thiết bị truyền thông Profibus – DP với tốc độ truyên lên tới 12MB. Địa chỉ của điều khiển tớ được định nghĩa trên EM277.



- Tải cấu hình mạng xuống cho PLC, và bắt đầu thực hiện việc định địa chỉ các vùng biến viết chương trình cho PLC S7-300.

II) Bài tập thực hành :

Bài 21: Thực hành kết nối và truyền nhận dữ liệu giữa các PLC, máy tính.

- Kết nối các trạm mạng, gồm 1 PLC S7-300 làm master và 1 PLC S7-200 làm slaver.
- Định cấu hình mạng.
- Thực hiện truyền nhận dữ liệu giữa các PLC thể hiện bằng các bit I/O thông qua màn hình ProTool.

Bài 22: Thực hành kết nối và truyền nhận dữ liệu giữa các PLC, máy tính. Mô tả một quá trình tự động gồm nhiều khâu sản xuất :

- Kết nối các trạm mạng, gồm 1 PLC S7-300 (Master) và 2 PLC S7-200 (Slave).
- Định cấu hình mạng.
- Thực hiện việc truyền nhận và giám sát dữ liệu ở các khâu sản xuất. Giả sử công việc ở các khâu như sau :
    - PLC S7-300 thực hiện công việc điều khiển tốc độ đông cơ thủy lực.
    - 1 PLC S7-200 thực hiện công việc cho xylanh dập một hướng làm việc.
    - 1 PLC S7-200 thực hiện công việc đọc tín hiệu từ cảm biến áp suất.

Chú ý : có sử dụng mô hình (mô hình khí nén, mô hình thủy lực, động cơ DC).

PHỤ LỤC

Một số bài giải tham khảo

Bài 1:

Network 1 NETWORK TITLE (single line)

```
LD    PB_CW
O     Clockwise_Coil
AN    Counterclockwise_Coil
AN    PB_Stop
=     Clockwise_Coil
```

Symbol	Address	Comment
Clockwise_Coil	Q0.0	Motor run clockwise
Counterclockwise_Coil	Q0.1	Motor run Counterclockwise
PB_CW	I0.0	Starting Clockwise motor
PB_Stop	I0.2	Stoping Motor

Network 2

```
LD    PB_CCW
O     Counterclockwise_Coil
AN    Clockwise_Coil
AN    PB_Stop
=     Counterclockwise_Coil
```

Symbol	Address	Comment
Clockwise_Coil	Q0.0	Motor run clockwise
Counterclockwise_Coil	Q0.1	Motor run Counterclockwise
PB_CCW	I0.1	Starting counterclockwise motor
PB_Stop	I0.2	Stoping Motor

Bài 4:

Network 1 NETWORK TITLE (single line)

```
LD    Start_PB
O     MC1
AN    Stop_PB
AN    T110
=     MC1
=     MC3
```

Symbol	Address	Comment
MC1	Q0.0	Link R,S,T nodes to one point
MC3	Q0.2	Supply three phases for motor
Start_PB	I0.0	Srating Star motor
Stop_PB	I0.1	Starting Triangle motor

Network 2

```
LD    MC3
TON   T110, +50
```

Symbol	Address	Comment
MC3	Q0.2	Supply three phases for motor

Network 3

```
LD    T110
O     MC2
AN    Stop_PB
=     MC3
=     MC2
```

Symbol	Address	Comment
MC2	Q0.1	Link nodes U and S, V and T, W and R
MC3	Q0.2	Supply three phases for motor
Stop_PB	I0.1	Starting Triangle motor

Bài 5:

Network 1 Interlocked

```
LD    PB1
AN    Player_3
AN    Player_2
LD    PB2
AN    Player_1
AN    Player_3
OLD
LD    PB3
AN    Player_1
AN    Player_2
OLD
O     M0.0
AN    M0.1
AN    T110
=     M0.0
```

Symbol	Address	Comment
PB1	I0.0	push button for first player
PB2	I0.1	push button for second player
PB3	I0.2	push button for third player
Player_1	Q0.0	Light of first player
Player_2	Q0.1	Light of second player
Player_3	Q0.2	Light of third player

Network 2 ON Buzzer when any switch is pressed and timer will cut buzzer after specified time

```
LD    M0.0
=     Buzzer
TON   T110, +100
```

Symbol	Address	Comment
Buzzer	Q0.3	Buzzer operates

Network 3 Player 1 network

```
LD    PB1
O     Player_1
AN    Player_2
AN    Player_3
AN    RST
=     Player_1
```

Symbol	Address	Comment
PB1	I0.0	push button for first player
Player_1	Q0.0	Light of first player
Player_2	Q0.1	Light of second player
Player_3	Q0.2	Light of third player
RST	I0.3	Reset push butoon of Host

Network 4 Player 2 network

```
LD    PB2
O     Player_2
AN    Player_1
AN    Player_3
AN    RST
=     Player_2
```

Symbol	Address	Comment
PB2	I0.1	push button for second player
Player_1	Q0.0	Light of first player
Player_2	Q0.1	Light of second player
Player_3	Q0.2	Light of third player
RST	I0.3	Reset push butoon of Host

Network 5 Player 3 network

```
LD    PB3
O     Player_3
AN    Player_1
AN    Player_2
AN    RST
=     Player_3
```

Symbol	Address	Comment
PB3	I0.2	push button for third player
Player_1	Q0.0	Light of first player
Player_2	Q0.1	Light of second player
Player_3	Q0.2	Light of third player
RST	I0.3	Reset push butoon of Host

Bài 6:

Network 1 NETWORK TITLE (single line)

```
LDW<  CTUD , +12
AN   Full_Car
=    Go_In
```

Symbol	Address	Comment
CTUD	C0	Counter Up and Down
Full_Car	Q0.0	
Go_In	Q0.1	

Network 2

```
LDW>=  CTUD , +12
=    Full_Car
```

Symbol	Address	Comment
CTUD	C0	Counter Up and Down
Full_Car	Q0.0	

Network 3

```
LD   SS1
AN   Full_Car
LD   SS2
AN   M0.0
LD   I0.2
CTUD   CTUD , +20
```

Symbol	Address	Comment
CTUD	C0	Counter Up and Down
Full_Car	Q0.0	
SS1	I0.0	sensor sensin entrans car
SS2	I0.1	Sensor detect out car

Network 4

```
LDW=   CTUD , +0
=    M0.0
```

Symbol	Address	Comment
CTUD	C0	Counter Up and Down

Bài 10:

```
Network: 1

      A(
      O     "Start_PB"
      O     M      0.1
      )
      S     "Sol_1A"
      R     "Sol_1B"

Network: 2

      A     "Sol_1A"
      FP    M      0.2
      R     M      0.1

Network: 3

      A     "Sol_1A"
      A(
      L     "Transduce position"
      L     9261                  //35 mm
      >=I
      )
      =     M      0.0

Network: 4

      A     M      0.0
      R     "Sol_1A"
      S     "Sol_1B"
```

```
Network: 5

      A     "Sol_1B"
      A(
      L     "Transduce position"
      L     5292                    //20 mm
      <=I
      )
      R     "Sol_1B"
```

Bài 12:

```
Network: 1

      A     I      0.0
      AN    T      1
      L     W#16#1900
      SD    T      1

Network: 2

      L     W#16#1600
      LC    T      1
      <=I
      JC    xng
      L     W#16#1550
      >I
      JC    vng
      S     Q      4.0
      S     Q      5.2
      R     Q      4.1
      R     Q      4.2
      R     Q      5.0
      LC    T      1
      L     W#16#1100
      <=I
      JC    vng1
      BEU

Network: 3

xng:  S     Q      4.2
      S     Q      5.0
      R     Q      4.1
      R     Q      4.0
      R     Q      5.1
      R     Q      5.2
      BEU

Network: 4

vng:  S     Q      4.1
      S     Q      5.0
      R     Q      4.2
      R     Q      4.0
      BEU

Network: 5

vng1: S     Q      5.1
      R     Q      5.2
      BEU
```

Bài 14:

```
Network: 1

      A     "Start_PB"
      S     "Sol_1A"
      R     "Sol_1B"

Network: 2

      A     "Sol_1A"
      FP    M      0.0
      JNB   _000
      L     6615
      T     "Flow adjusted valve"
_000: NOP   0

Network: 3

      A     "Sol_1A"
      L     C#0
      S     "Counter"

Network: 4

      A     "Speed count"
      CU    "Counter"

Network: 5

      L     "Counter"
      T     MW     2

Network: 6

      L     MW     2
      L     10
      ==I
      S     M      0.1

Network: 7

      A     M      0.1
      FP    M      1.0
      JNB   _001
      L     13230
      T     "Flow adjusted valve"
_001: NOP   0

Network: 8

      L     MW     2
      L     40
      ==I
      S     M      0.2

Network: 9

      A     M      0.2
      FP    M      1.1
      JNB   _002
      L     19845
      T     "Flow adjusted valve"
_002: NOP   0


Network: 10

      L     MW     2
      L     60
      ==I
      =     M      0.3
```

```
Network: 11

      A     M      0.3
      FP    M      0.4
      R     "Sol_1A"
      JNB   _003
      L     0
      T     "Flow adjusted valve"
_003: NOP   0

Network: 12

      A     "Reset counter"
      R     "Counter"


Network: 13

      A     "Stop_PB"
      R     "Sol_1A"
```

Tài liệu tham khảo

[1]. "Automation with Micro PLC SIMATIC S7-200"
Siemens, Germany.

[2]. " A beginner's guide to PLC"
OMRON, Japan.

[3]. Statement List for S7-300 and S7-400 Programming"
Siemens, Germany.

[4]. Lê Văn Tiến Dũng. "Điều khiển lập trình PLC và mạng"
Đại học Kỹ thuật Công nghệ TP.HCM, năm 2003.

[5]. Lê Văn Tiến Dũng. "Nghiên cứu khoa học – Xây dựng hệ thống giám sát và điều khiển thiết bị ngoại vi và PLC bằng máy tính"
Đại học Kỹ thuật Công nghệ TP.HCM, năm 2003.

[6]. Software: Protool/Pro, Simatic manager S7-300, Step7-200
Siemens, Germany.