TẬP LỆNH PLC S7-300

- Cấu trúc và trạng thái kết quả lệnh.
- Nhóm lệnh logic.
- Nhóm lệnh tiếp điểm đặc biệt.
- Nhóm lệnh so sánh.
- Nhóm lệnh toán học.
- Nhóm lệnh chuyển đổi.
- Lệnh về Timer.
- Lệnh về Counter.
- Thư viện hàm S7-300.

CẤU TRÚC LỆNH

Lệnh STL của PLC S7-300 có dạng:

Tên lệnh + Toán hạng

Xét 2 lệnh trong ví dụ sau:

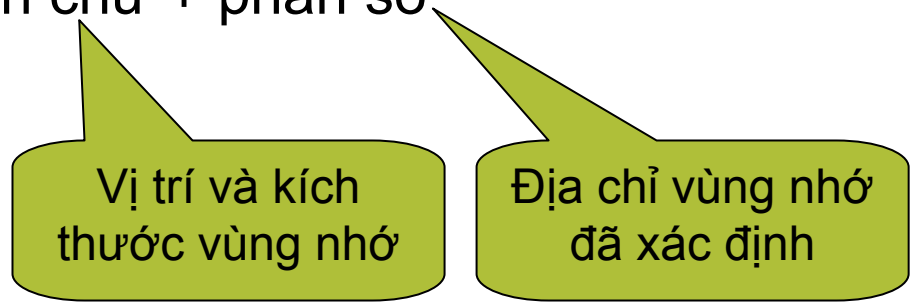
A I0.0

<> D

- A, <> là tên lệnh.
- I0.0, D là toán hạng.

CẤU TRÚC LỆNH

- Tên lệnh: xét cụ thể trong mục tập lệnh.
- Toán hạng: có 2 dạng
  - Toán hạng là địa chỉ: phần chữ + phần số
  - Toán hạng là dữ liệu:
    - Dữ liệu logic
    - Số nhị phân
    - Số thập lục phân
    - Số nguyên kiểu INT
    - Số thực kiểu REAL
    - Dữ liệu về thời gian
    - Dữ liệu của bộ đếm, định thời
    - Dữ liệu kiểu ký tự



THANH GHI TRẠNG THÁI

- Định nghĩa: là thanh ghi đặc biệt dài 16 bit, dùng để ghi lại trạng thái của các phép tính trung gian, kết quả tính toán khi thực hiện lệnh.
- Cấu trúc: chỉ sử dụng 9 bit thấp

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

THANH GHI TRẠNG THÁI

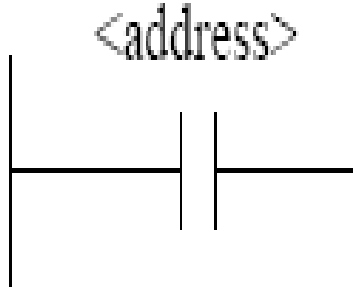
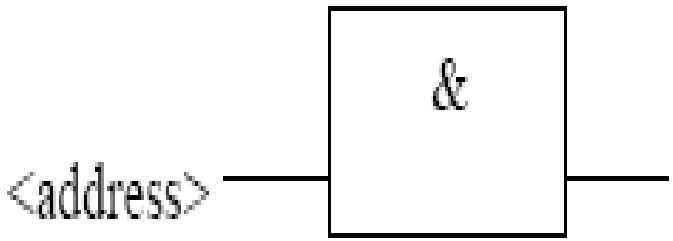
- FC – First Check: bit kiểm tra khi thực hiện các lệnh logic ^ (AND), v (OR), NOT.
  - Đang thực hiện lệnh: FC=1
  - Thực hiện xong lệnh: FC=0
- RLO – Result of Logic Operation: bit thể hiện kết quả tức thời của phép tính logic vừa thực hiện.
  - FC=0: ghi giá trị logic của tiếp điểm trong lệnh vào RLO
  - FC=1: thực hiện lệnh, ghi giá trị logic vào RLO.

THANH GHI TRẠNG THÁI

- STA – Status: bit trạng thái, luôn có giá trị logic của tiếp điểm được chỉ ra trong lệnh.
- OR: bit ghi lại giá trị logic của phép tính ^ cuối cùng để thực hiện phép v tiếp theo. (vì thực hiện ^ trước v)
- OS - Overflow Store: bit ghi kết quả phép tính bị tràn.
- OV – Overflow: bit báo kết quả phép tính bị tràn.
- CC0 và CC1 – Condition Code: bit báo trạng thái kết quả phép tính với số nguyên, thực, hoặc trong ACCU.
- BR – Binary Result: cho phép liên kết giữ STL và LAD

NHÓM LỆNH LOGIC

1. Lệnh And:

LAD	FBD
	

Với

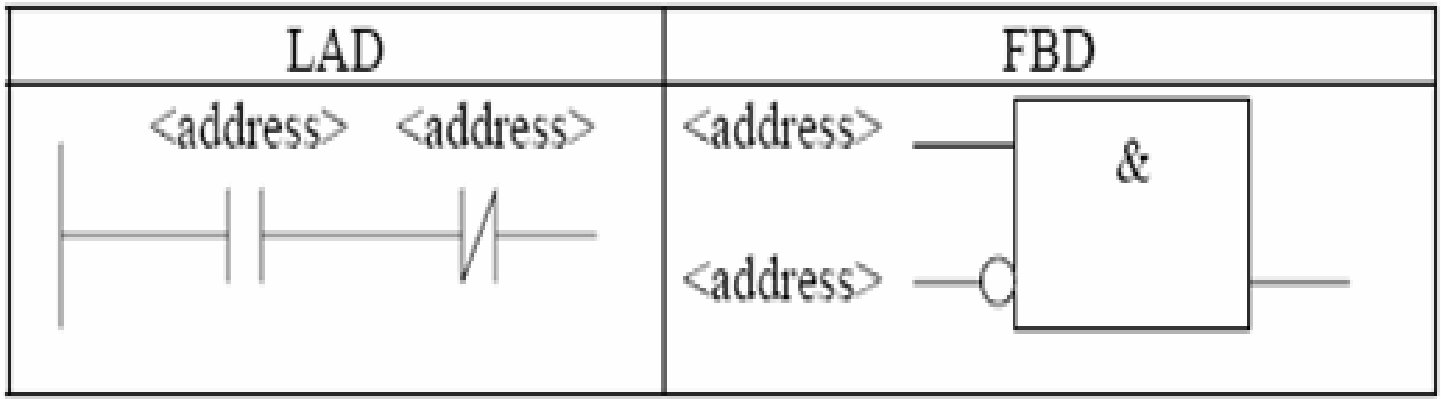
Thông số	Kiểu dữ liệu	Toán hạng	Mô tả
<address>	BOOL	I,Q,M,L,D,T,C	Kiểm tra bit

Ví dụ:

STL Program		Relay Logic
		Power rail
A	I 1.0	I 1.0 signal state 1 — NO contact
A	I 1.1	I 1.1 signal state 1 — NC contact
=	Q 4.0	Q 4.0 signal state 1 — Coil
Displays closed switch		

NHÓM LỆNH LOGIC

2. Lệnh And Not:



Với:

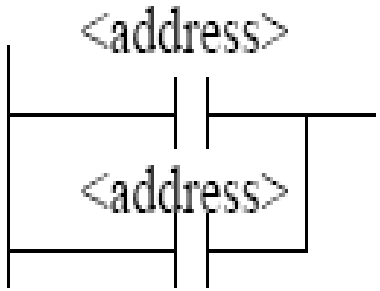
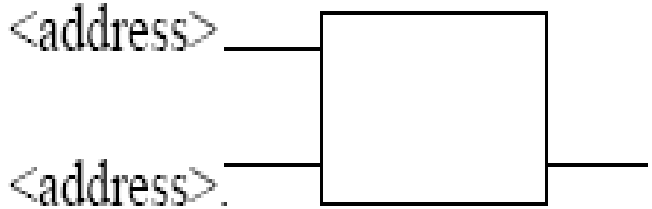
Thông số	Kiểu dữ liệu	Toán hạng
<address>	BOOL	I, Q, M, L, D, T, C

Ví dụ:

STL Program		Relay Logic
		Power rail
A	I 1.0	I 1.0 Signal state 0 — NO contact
AN	I 1.1	I 1.1 Signal state 1 — NC contact
=	Q 4.0	Q 4.0 Signal state 0 — Coil

NHÓM LỆNH LOGIC

3. Lệnh Or

LAD	FBD
	

Với :

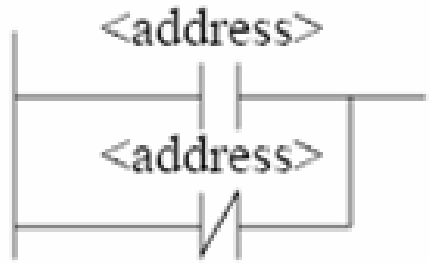
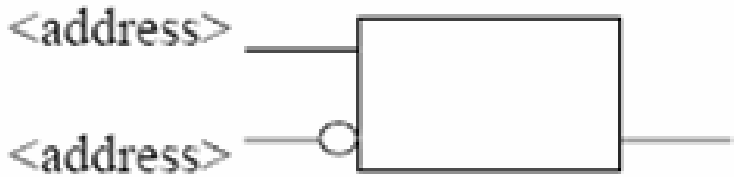
Thông số	Kiểu dữ liệu	Toán hạng
<address>	BOOL	I,Q,M,L,D,T,C

Ví dụ:

STL Program		Relay Logic
		Power rail
O	I 1.0	I 1.0 Signal state 1 No contact
O	I 1.1	I 1.1 Signal state 0 No contact
=	Q 4.0	Q 4.0 Signal state 1 Coil
		Displays closed switch

NHÓM LỆNH LOGIC

4. Lệnh Or Not:

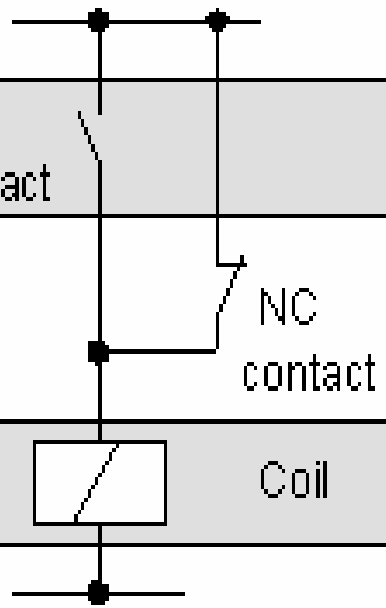
LAD	FBD
	

Với:

Thông số	Kiểu dữ liệu	Toán hạng
<address>	BOOL	I, Q, M, L, D, T, C

Ví dụ:

STL Program		Relay Logic
		Power rail
O	I 1.0	I 1.0 Signal state 0
ON	I 1.1	I 1.1 Signal state 1
=	Q 4.0	Q 4.0 Signal state 1



NHÓM LỆNH LOGIC

5. Lệnh Gán:

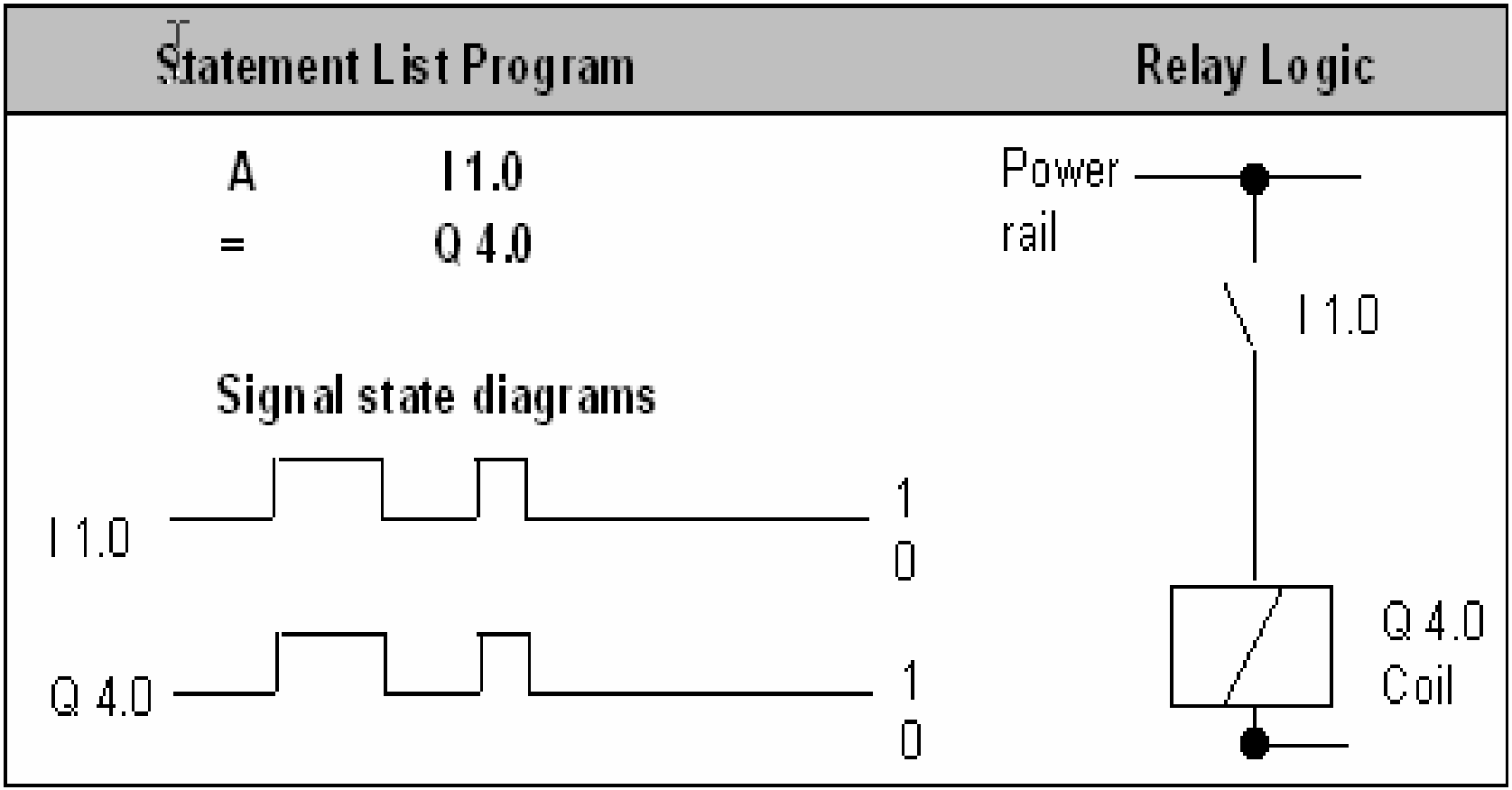
- gán giá trị của RLO đến ô nhớ được chỉ ra trong toán hạng

LAD	FBD
	

Với:

Thông số	Kiểu dữ liệu	Toán hạng	Mô tả
<address>	BOOL	I,Q,M,L,D	Địa chỉ bit được set

Ví dụ:


Statement List Program
Relay Logic
A I 1.0
= Q 4.0
Signal state diagrams
I 1.0
Q 4.0
1
0
1
0
Power rail
I 1.0
Q 4.0
Coil

NHÓM LỆNH LOGIC

6. Lệnh Gán 1:

- Gán giá trị 1 vào ô nhớ có địa chỉ xác định trong toán hạng

LAD	FBD
<address> —(S)—	<address> S

Với:

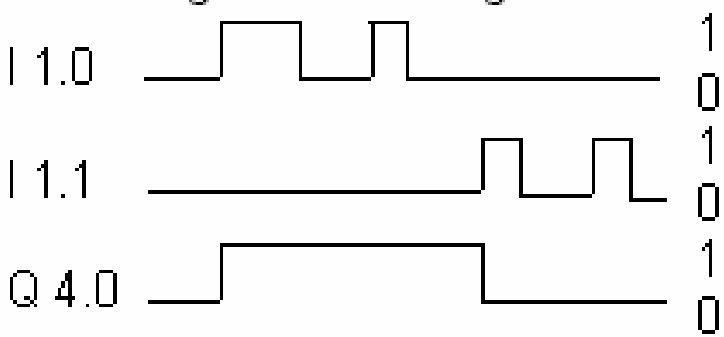
Thông số	Kiểu dữ liệu	Toán hạng	Mô tả
<address>	BOOL	I,Q,M,L,D	Địa chỉ bit được set

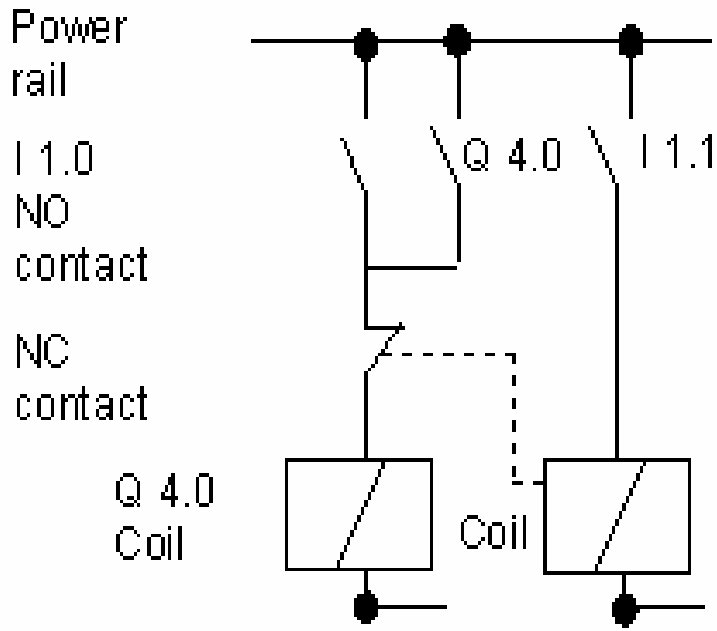
Ví dụ:

Statement List Program	Relay Logic

```
A    I 1.0
S    Q 4.0
A    I 1.1
R    Q4.0
```

Signal state diagrams





NHÓM LỆNH LOGIC

7. Lệnh Gán 0

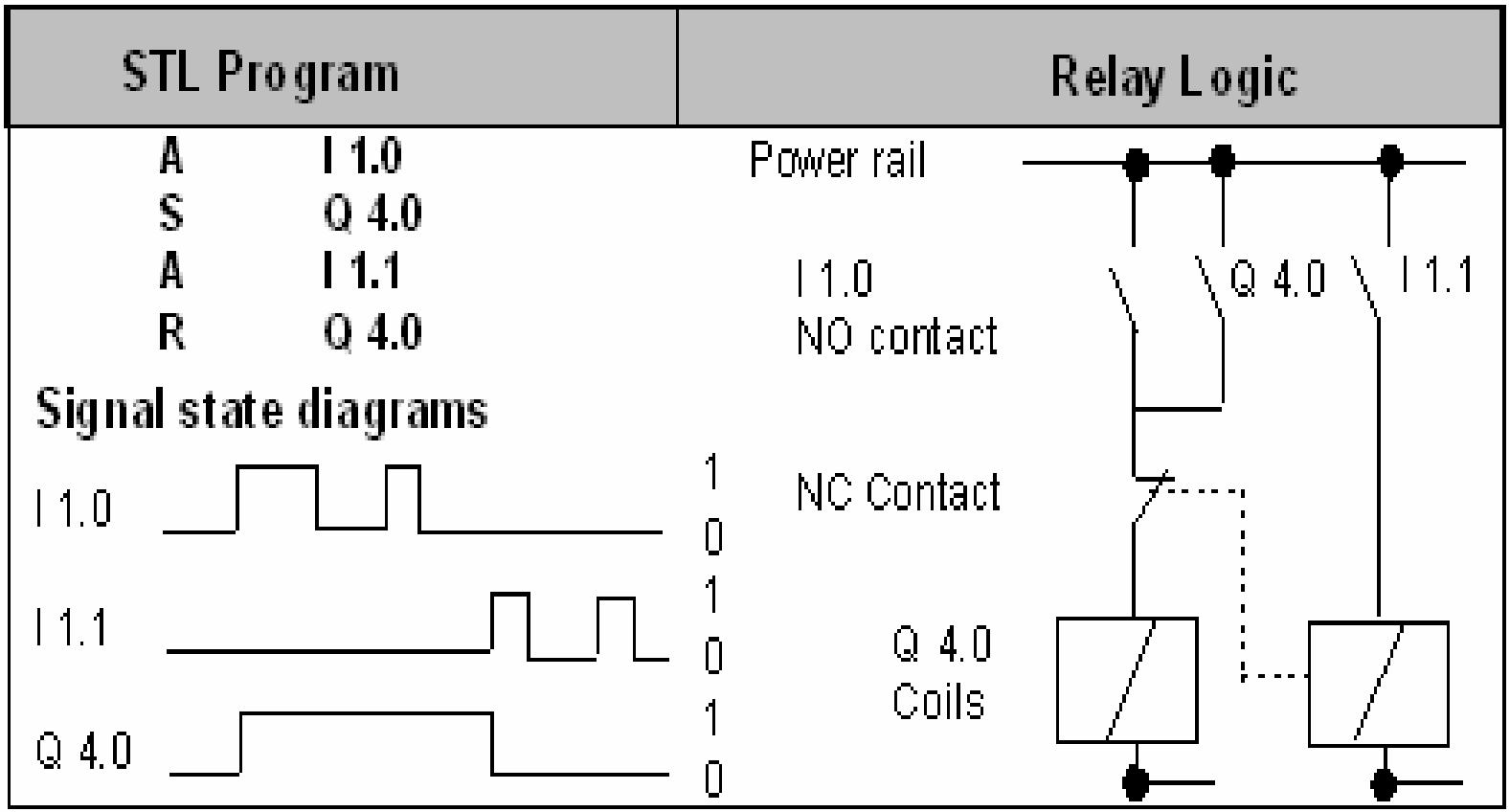
- Gán giá trị 0 vào bit có địa chỉ được xác định.

LAD	FBD
	

Với:

Thông số	Kiểu dữ liệu	Toán hạng	Mô tả
<address>	BOOL	I,Q,M,L,D	Địa chỉ bit được reset

Ví dụ:


STL Program
Relay Logic
A I 1.0
S Q 4.0
A I 1.1
R Q 4.0
Signal state diagrams
I 1.0
I 1.1
Q 4.0
1
0
1
0
1
0
Power rail
I 1.0
NO contact
Q 4.0
I 1.1
NC Contact
Q 4.0
Coils

NHÓM LỆNH LOGIC

8. Lệnh And tổ hợp và đóng tổ hợp:

- AND giữa bit RLO với giá trị logic của biểu thức trong dấu ngoặc sau nó và ghi lại kết quả vào RLO
- Cú pháp: A(
)
- Toán hạng: không có toán hạng.
- Thanh ghi trạng thái:

A(

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

)

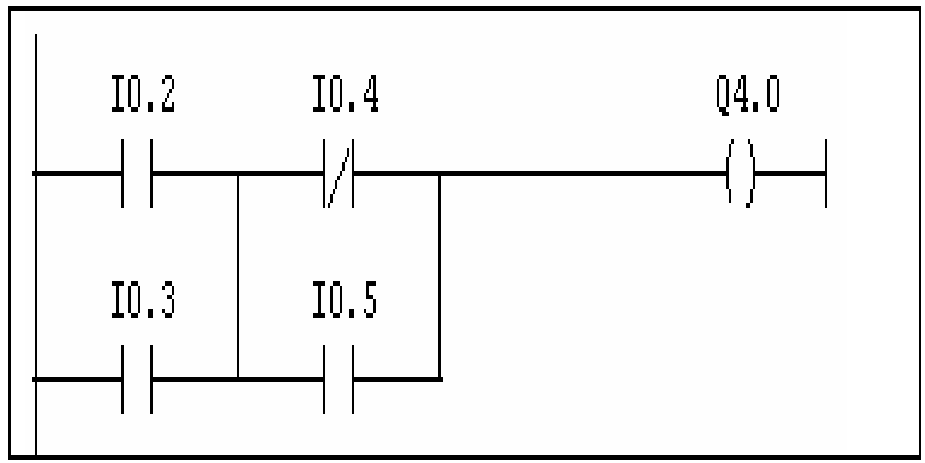
BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
-	-	-	-	-	x	1	x	1

Ví dụ:

Statement List Program		Relay Logic
		Power rail
A(O O)	I 0.0 M 10.0	I 0.0 M 10.0
A(O O)	I 0.2 M 10.3	I 0.2 M10.3
A	M 10.1	M 10.1
=	Q 4.0	Q 4.0 Coil

Thực hiện: Q4.0 = (I0.2 v I0.3) ^ (I0.4 v I0.5)

```
A(
O I0.2
O I0.3
)
A(
ON I0.4
O I0.5
)
= Q4.0
```



NHÓM LỆNH LOGIC

9. Lệnh Or tổ hợp và đóng tổ hợp:

- Thực hiện phép OR giữa bit RLO với giá trị của biểu thức trong dấu ngoặc sau nó và ghi kết quả vào RLO
- Cú pháp: O(
)
- Toán hạng: không có
- Thanh ghi trạng thái:

O(

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

)

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

Thực hiện Q4.0 = I0.2 v (I0.4 ^ I0.5)

A I0.2

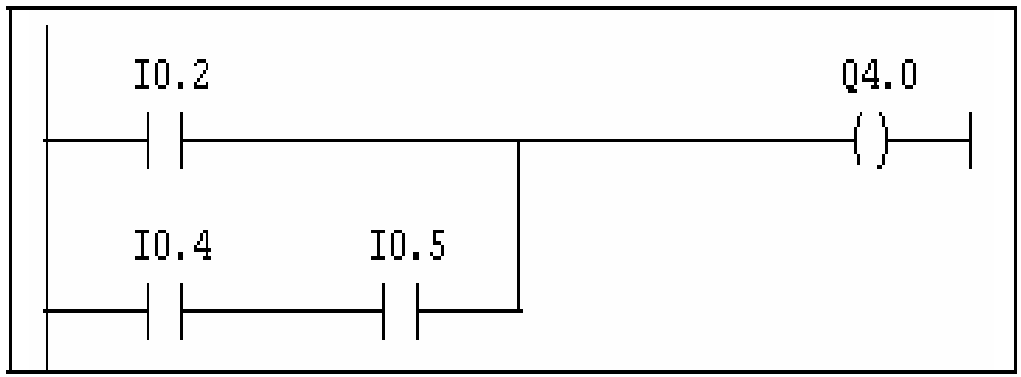
O(

A I0.4

A I0.5

)

= Q4.0



Ví dụ: chuyển sang LAD

Statement List Program		Relay Logic
		Power rail
A(O I 0.0 O M 10.0)		I 0.0 M 10.0
A(O I 0.2 O M 10.3)		I 0.2 M10.3
A	M 10.1	M 10.1
=	Q 4.0	Q 4.0 Coil

LỆNH VỀ TIẾP ĐIỂM ĐẶC BIỆT

- Lệnh Set:
  - Cú pháp: SET
  - Toán hạng: không có toán hạng.
  - Ý nghĩa: ghi giá trị 1 vào bit RLO.
  - Thanh ghi trạng thái:

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

  - LAD: không thực hiện.

LỆNH VỀ TIẾP ĐIỂM ĐẶC BIỆT

- Lệnh Clear:
  - Cú pháp: CLR
  - Toán hạng: không có toán hạng.
  - Ý nghĩa: ghi giá trị 0 vào bit RLO.
  - Thanh ghi trạng thái:

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

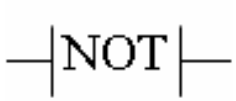
  - LAD: không thực hiện.

LỆNH VỀ TIẾP ĐIỂM ĐẶC BIỆT

- Lệnh Not:
  - Cú pháp: NOT
  - Toán hạng: không có toán hạng.
  - Ý nghĩa: đảo giá trị bit RLO.
  - Thanh ghi trạng thái:

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

  - LAD:

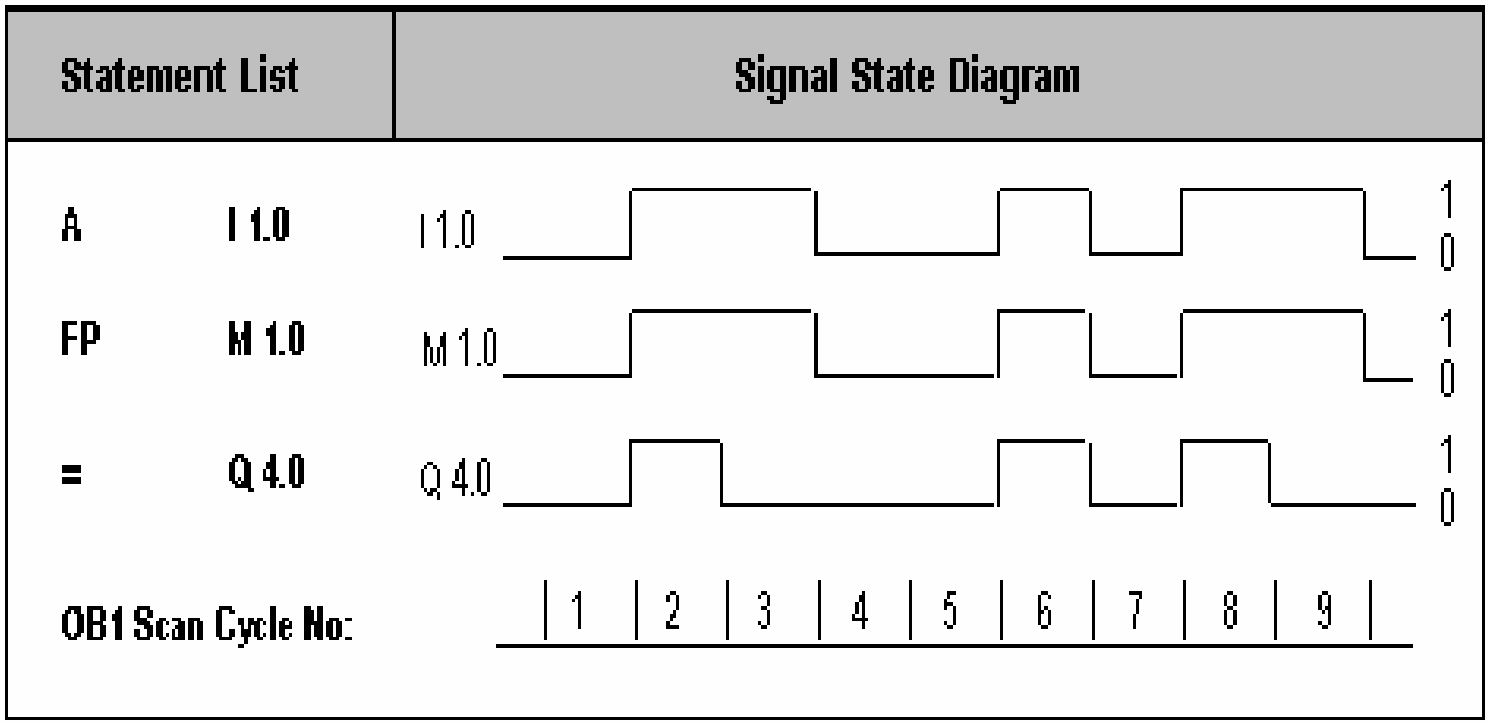
LAD	FBD
	

LỆNH VỀ TIẾP ĐIỂM ĐẶC BIỆT

- Lệnh phát hiện xung cạnh lên:
  - Cú pháp: FP <toán hạng>
  - Toán hạng: I, Q, M, L, D
  - Ý nghĩa: kiểm tra khi bit RLO chuyển từ 0 lên 1 thì cho RLO=1.
  - Thanh ghi trạng thái:

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

Ví dụ:

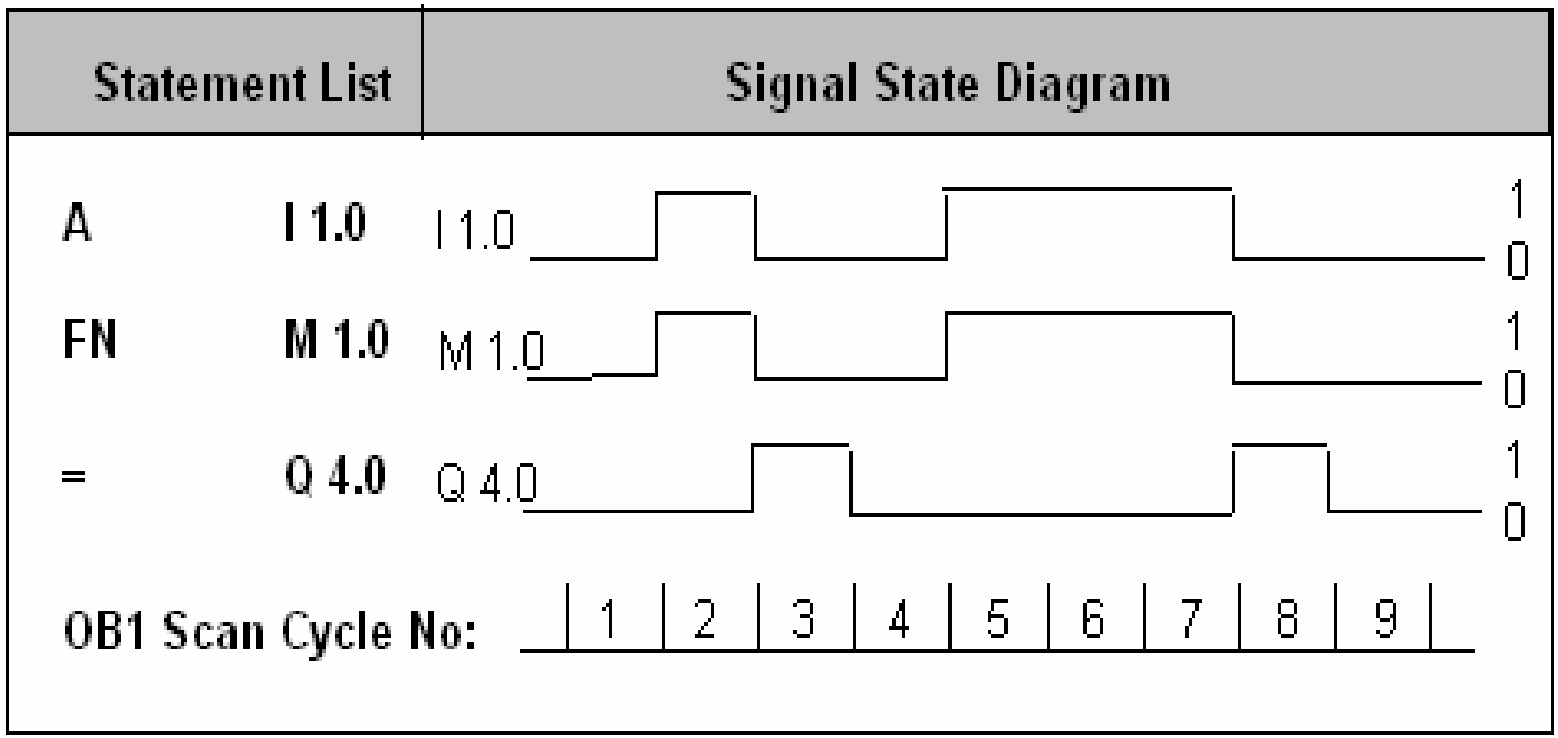

Statement List
Signal State Diagram
A I 1.0
FP M 1.0
= Q 4.0
I 1.0
M 1.0
Q 4.0
1
0
OB1 Scan Cycle No:
1 2 3 4 5 6 7 8 9

LỆNH VỀ TIẾP ĐIỂM ĐẶC BIỆT

- Lệnh phát hiện xung cạnh xuống:
  - Cú pháp: FN <toán hạng>
  - Toán hạng: I, Q, M, L, D
  - Ý nghĩa: kiểm tra khi bit RLO chuyển từ 1 xuống 0 thì cho RLO=1.
  - Thanh ghi trạng thái:

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

Ví dụ:


Statement List
Signal State Diagram
A I 1.0
FN M 1.0
= Q 4.0
I 1.0
M 1.0
Q 4.0
1
0
1
0
1
0
OB1 Scan Cycle No:
1 2 3 4 5 6 7 8 9

LỆNH NẠP VÀ TRUYỀN DỮ LIỆU

- Lệnh Load:
  - Cú pháp: **L <toán hạng>**
  - Toán hạng: địa chỉ của byte, word, Dword.
  - Ý nghĩa: nạp dữ liệu của byte, word, Dword có địa chỉ xác định trong toán hạng vào thanh ghi ACCU1 sau khi dữ liệu cũ của thanh ghi này được lưu vào thanh ghi ACCU2 và ACCU1 bị xóa về 0.
  - LAD: không thực hiện.

Ví dụ:

Contents of ACCU 1	ACCU1-H-H	ACCU1-H-L	ACCU1-L-H	ACCU1-L-L
before execution of load instruction	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
after execution of **L MB10** (L <Byte>)	00000000	00000000	00000000	<MB10>
after execution of **L MW10** (L <word>)	00000000	00000000	<MB10>	<MB11>
after execution of **L MD10** (L <double word>)	<MB10>	<MB11>	<MB12>	<MB13>

LỆNH NẠP VÀ TRUYỀN DỮ LIỆU

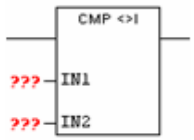
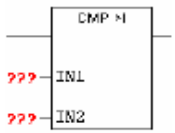
- Lệnh Transfer:
  - Cú pháp: **T <toán hạng>**
  - Toán hạng: địa chỉ của byte, word, Dword.
  - Ý nghĩa: truyền (copy) nội dung của thanh ghi ACCU1 đến địa chỉ của byte, word, Dword xác định trong toán hạng. Số lượng các byte của thanh ghi ACCU1 được truyền đi phụ thuộc vào toán hạng khai báo.
  - LAD: không thực hiện.

NHÓM LỆNH SO SÁNH

- Giới thiệu:
  - Thực hiện so sánh 2 thanh ghi ACCU1 và ACCU2
  - Gồm các phép so sánh: ==, <>, >, <, >=, <=.
  - Nếu phép so sánh đúng thì RLO=1.
- Có 3 lệnh so sánh.
  - So sánh số nguyên 16 bit.
  - So sánh số nguyên 32 bit.
  - So sánh số thực.

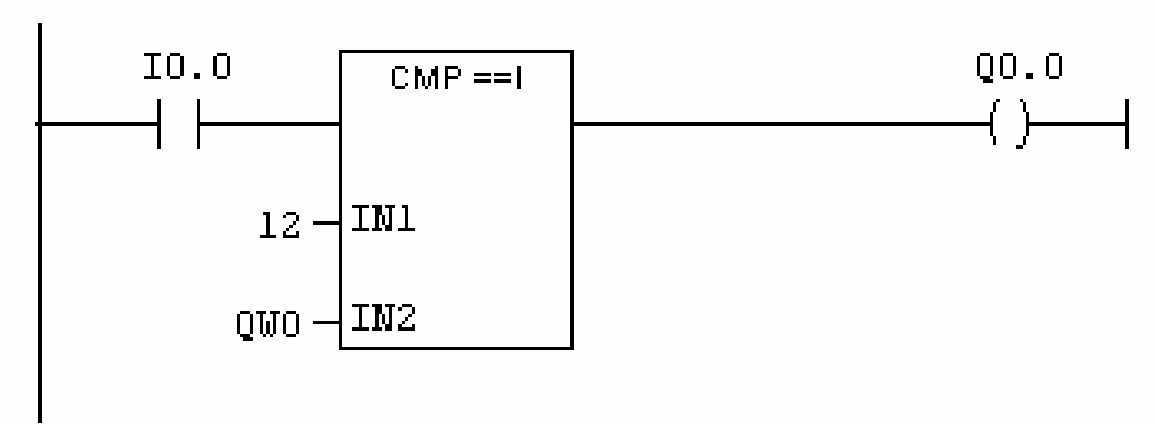
NHÓM LỆNH SO SÁNH

- Lệnh so sánh số nguyên 16 bit:

	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh so sánh bằng		= =I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1=IN2 -Ngõ vào lên mức 1.
Lệnh so sánh không bằng		<>I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<>IN2 -Ngõ vào lên mức 1.
Lệnh so sánh lớn hơn		>I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1>IN2 -Ngõ vào lên mức 1.
Lệnh so sánh nhỏ hơn		<I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<IN2 -Ngõ vào lên mức 1.
Lệnh so sánh lớn hơn hoặc bằng		>=I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1>=IN2 -Ngõ vào lên mức 1.
Lệnh so sánh nhỏ hơn hoặc bằng		<=I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<=IN2 -Ngõ vào lên mức 1.

```
STL         Explanation
L  MW10     //Load contents of MW10 (16-bit integer).
L  IW24     //Load contents of IW24 (16-bit integer).
>I          //Compare if ACCU 2-L (MW10) is greater (>) than ACCU 1-L
            (IW24).
=  M 2.0    //RLO = 1 if MW10 > IW24.
```

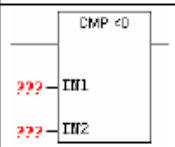
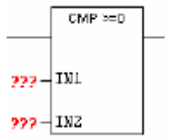
Ví dụ:




```
A       I       0.0
A(
L       12
L       QW      0
==I
)
=       Q       0.0
```

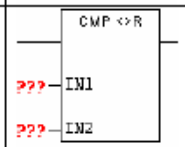
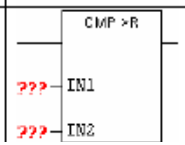
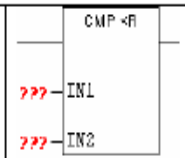
NHÓM LỆNH SO SÁNH

- Lệnh so sánh số nguyên 32 bit:

	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh so sánh bằng		==D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1=IN2 -Ngõ vào lên mức 1.
Lệnh so sánh không bằng		<>D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<>IN2 -Ngõ vào lên mức 1.
Lệnh so sánh lớn hơn		>D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1>IN2 -Ngõ vào lên mức 1.
Lệnh so sánh nhỏ hơn		<D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<IN2 -Ngõ vào lên mức 1.
Lệnh so sánh lớn hơn hoặc bằng		>=D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1>=IN2 -Ngõ vào lên mức 1.
Lệnh so sánh nhỏ hơn hoặc bằng		<=D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<=IN2 -Ngõ vào lên mức 1.

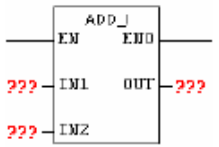
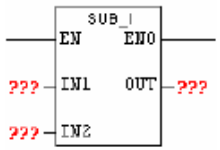
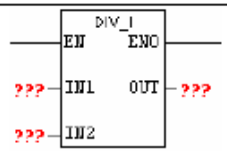
NHÓM LỆNH SO SÁNH

- Lệnh so sánh số thực:

	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh so sánh bằng		==R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1=IN2 -Ngõ vào lên mức 1.
Lệnh so sánh không bằng		<>R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<>IN2 -Ngõ vào lên mức 1.
Lệnh so sánh lớn hơn		>R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1>IN2 -Ngõ vào lên mức 1.
Lệnh so sánh nhỏ hơn		<R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<IN2 -Ngõ vào lên mức 1.
Lệnh so sánh lớn hơn hoặc bằng		>=R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1>=IN2 -Ngõ vào lên mức 1.
Lệnh so sánh nhỏ hơn hoặc bằng		<=R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<=IN2 -Ngõ vào lên mức 1.

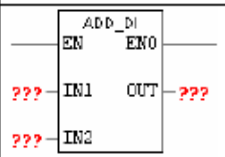
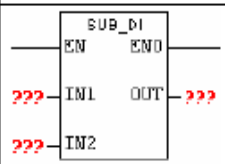
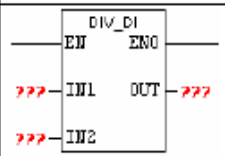
NHÓM LỆNH TOÁN HỌC

- Thực hiện với số nguyên 16 bit:

	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh cộng		+ I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const OUT I.Q.M.L.D, (INT) const	Lệnh cộng 2 số nguyên 16 bit trong IN 1 và IN 2. Kết quả cất vào OUT.
Lệnh trừ		- I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const OUT I.Q.M.L.D, (INT) const	Lệnh trừ 2 số nguyên 16 bit trong IN 1 và IN 2.Kết quả cất vào OUT.
Lệnh nhân		* I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const OUT I.Q.M.L.D, (INT) const	Lệnh nhân 2 số nguyên 16 bit trong IN 1 và IN 2. Kết quả cất vào OUT.
Lệnh chia		/ I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const OUT I.Q.M.L.D, (INT) const	Lệnh chia 2 số nguyên 16 bit trong IN 1 và IN 2. Kết quả cất vào OUT.

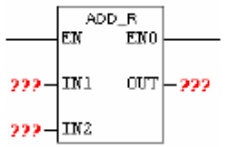
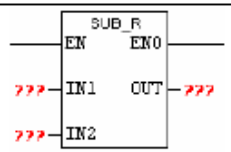
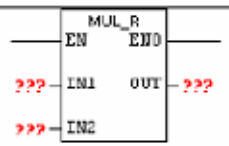
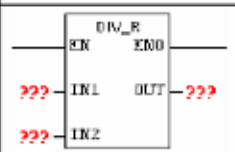
NHÓM LỆNH TOÁN HỌC

- Thực hiện với số nguyên 32 bit:

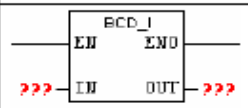
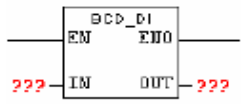
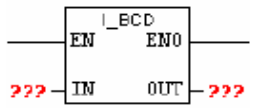
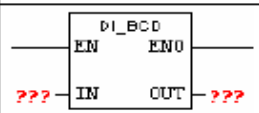
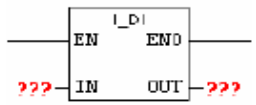
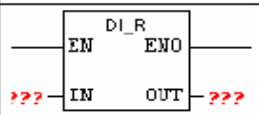
	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh cộng		+ D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const OUT I.Q.M.L.D, (DINT) const	Lệnh cộng 2 số nguyên 32 bit trong IN 1 và IN 2. Kết quả cất vào OUT.
Lệnh trừ		- D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const OUT I.Q.M.L.D, (DINT) const	Lệnh trừ 2 số nguyên 32 bit trong IN 1 và IN 2. Kết quả cất vào OUT.
Lệnh nhân		* D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const OUT I.Q.M.L.D, (DINT) const	Lệnh nhân 2 số nguyên 32 bit trong IN 1 và IN 2. Kết quả cất vào OUT.
Lệnh chia		/ D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const OUT I.Q.M.L.D, (DINT) const	Lệnh chia 2 số nguyên 32 bit trong IN 1 và IN 2. Kết quả cất vào OUT.

NHÓM LỆNH TOÁN HỌC

- Thực hiện với số thực:

	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh cộng		+ R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const OUT I.Q.M.L.D, (REAL) const	Lệnh cộng 2 số thực trong IN 1 và IN 2. Kết quả cất vào OUT.
Lệnh trừ		- R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const OUT I.Q.M.L.D, (REAL) const	Lệnh trừ 2 số thực trong IN 1 và IN 2. Kết quả cất vào OUT.
Lệnh nhân		* R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const OUT I.Q.M.L.D, (REAL) const	Lệnh nhân 2 số thực trong IN 1 và IN 2. Kết quả cất vào OUT.
Lệnh chia		/ R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const OUT I.Q.M.L.D, (REAL) const	Lệnh chia 2 số thực trong IN 1 và IN 2. Kết quả cất vào OUT.

LỆNH CHUYỂN ĐỔI BCD - INTEGER

	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh chuyển số BCD thành số nguyên 16 bit		BTI	IN I,Q,M,L,D (WORD) OUT I,Q,M,L,D (INT)	Lệnh chuyển số BCD trong IN thành số nguyên 16 bit cất trong OUT.
Lệnh chuyển đổi BCD thành số nguyên 32 bit		BTD	IN I,Q,M,L,D (DWORD) OUT I,Q,M,L,D (INT)	Lệnh chuyển số BCD trong IN thành số nguyên 32 bit cất trong OUT.
Lệnh chuyển đổi số nguyên 16 bit thành số BCD		ITB	IN I,Q,M,L,D (INT) OUT I,Q,M,L,D (WORD)	Lệnh chuyển số nguyên 16 bit trong IN thành số BCD cất trong OUT.
Lệnh chuyển đổi số nguyên 32 bit thành số BCD		DTB	IN I,Q,M,L,D (DINT) OUT I,Q,M,L,D (DWORD)	Lệnh chuyển số nguyên 32 bit trong IN thành số BCD cất trong OUT.
Lệnh chuyển đổi số nguyên 16 bit thành số nguyên 32 bit		ITD	IN I.Q,L.M.D (INT) OUT I,Q,M,L,D (DINT)	Lệnh chuyển số nguyên 16 bit trong IN thành số nguyên 32 bit trong OUT.
Lệnh chuyển số nguyên 32 bit thành số thực		DTR	IN I,QM,L,D (DINT) OUT I,Q,M,L,D (REAL)	Lệnh chuyển số nguyên 32 bit trongIN thành số thực cất trong OUT.

LỆNH VỀ TIMER

- Giới thiệu:
  - Timer là bộ tạo thời gian trễ giữa tín hiệu logic vào và ra, được đặt tên là Tx (0<x<255).
  - Timer có 2 thông số sử dụng: T-word và T-bit.
  - Khai báo thời gian trễ bằng word 16 bit.
    - Độ phân giải – R: 10ms, 100ms, 1s và 10s
    - Giá trị đặt – PV: số BCD từ 0 đến 999.
  - Thời gian trễ T = R*PV

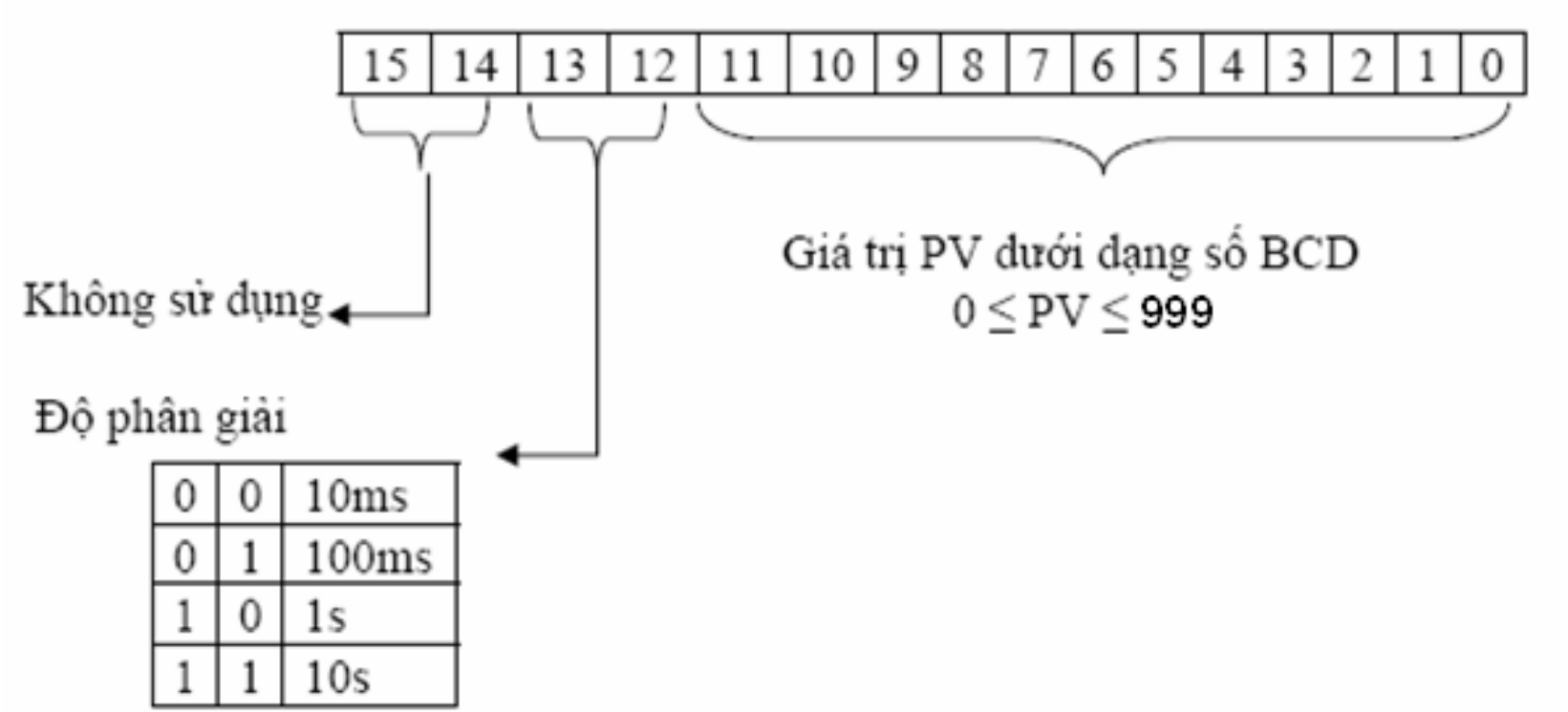
LỆNH VỀ TIMER

- Hoạt động:
  - Khi Timer được kích, giá trị PV sẽ được chuyển vào T-word của Timer.
  - T-word là thanh ghi chứa giá trị tức thời của Timer (gọi là giá trị CV)
  - Nội dung T-word sẽ giảm theo thời gian hoạt động của Timer.
  - Timer đạt được thời gian trễ đặt trước tương ứng với giá trị CV=0.
  - Báo hiệu thời gian trễ qua giá trị T-bit=1.

LỆNH VỀ TIMER

- Cấu trúc word khai báo thời gian trễ:

  Gồm 2 phần: giá trị đặt trước PV và độ phân giải.



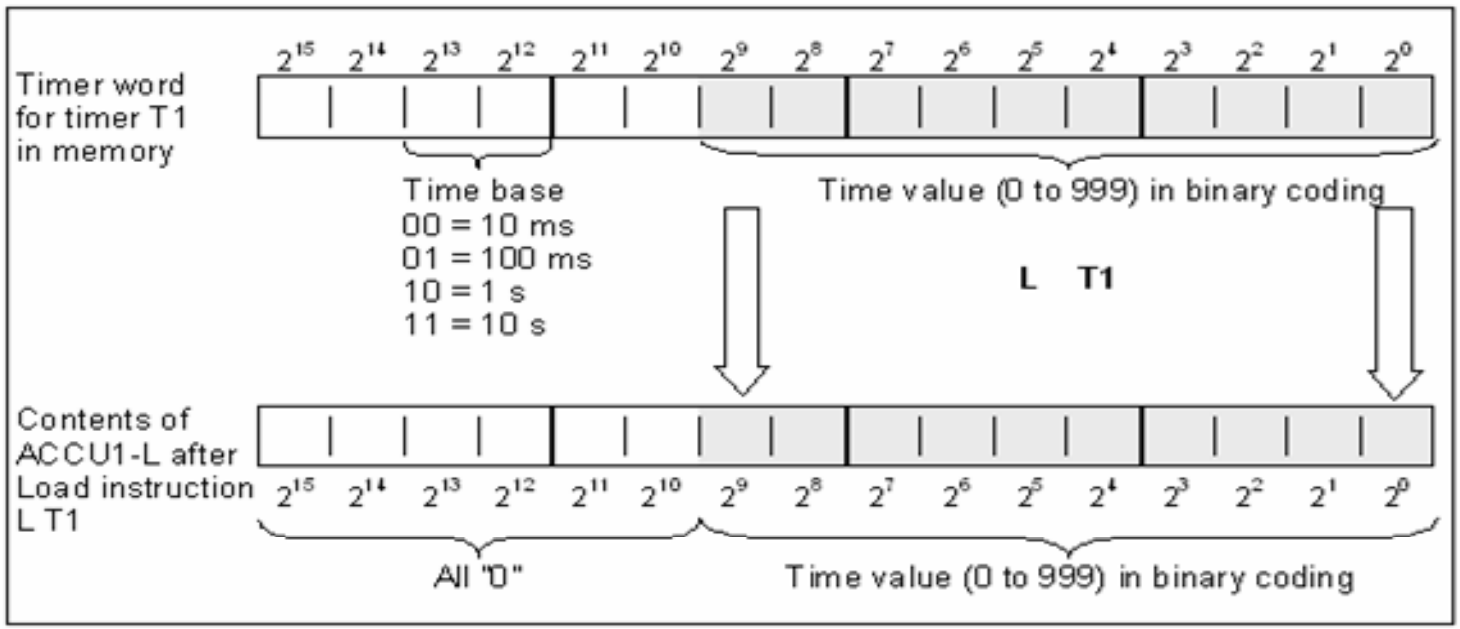
0	0	10ms
0	1	100ms
1	0	1s
1	1	10s

LỆNH VỀ TIMER

- Đọc nội dung thanh ghi T-word:

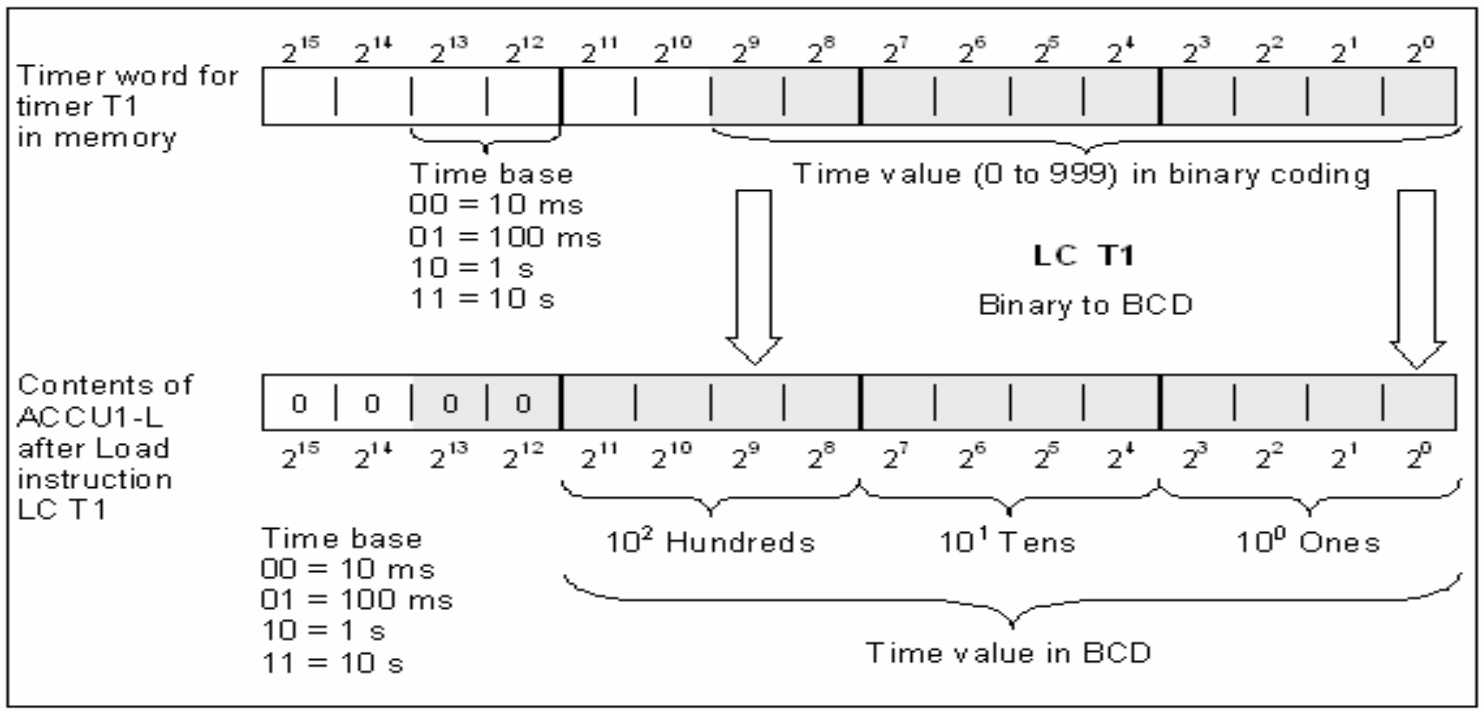
Nội dung T-word được đọc vào ACCU1: có 2 cách đọc

- Đọc số đếm tức thời:



LỆNH VỀ TIMER

- Đọc nội dung thanh ghi T-word:
  - Đọc thời gian trễ tức thời:



LỆNH VỀ TIMER

- Khai báo sử dụng Timer:
  - Khai báo tín hiệu enable (nếu muốn sử dụng tín hiệu chủ động kích).
  - Khai báo tín hiệu ngõ vào.
  - Khai báo tín hiệu trễ mong muốn.
  - Khai báo loại timer được sử dụng.
  - Khai báo tín hiệu xoá timer (tùy chọn).

Khai báo sử dụng Timer:

- Khai báo tín hiệu Enable:

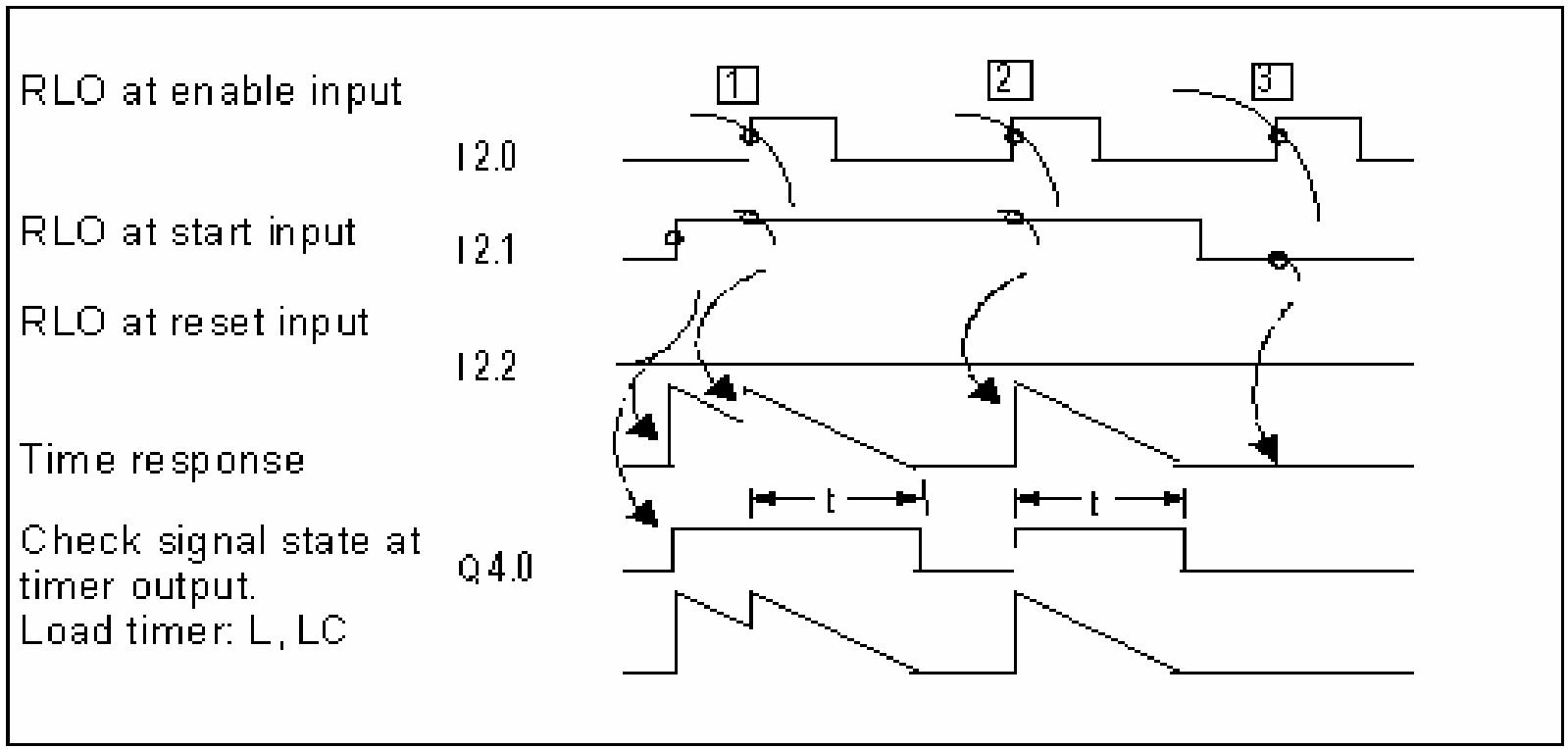
  Cú pháp: (**A <địa chỉ bit>)**

  FR <tên Timer>

  <địa chỉ bit>: xác định tín hiệu chủ động kích.

  <tên Timer>: loại Timer sử dụng, dạng Tx

Ví dụ:



t = programmed time interval

Khai báo sử dụng Timer:

- Khai báo tín hiệu ngõ vào

  Cú pháp **A <Địa chỉ bit>**

  <địa chỉ bit>: xác định tín hiệu đầu vào cho timer.

Khai báo sử dụng Timer:

- Khai báo word thời gian trễ:
  - Cú pháp: **L <constant>**

<constant> xác định thời gian trễ mong muốn,

Có 2 dạng constant:

  - Dạng dữ liệu thời gian trực tiếp:

    S5T#aHbMcSdMS

  - Dạng theo độ phân giải:

    L W#16#txyz

t: giá trị của 2 bit tính độ phân giải.

xyz: giá trị thời gian theo mã BCD.

Ví dụ:

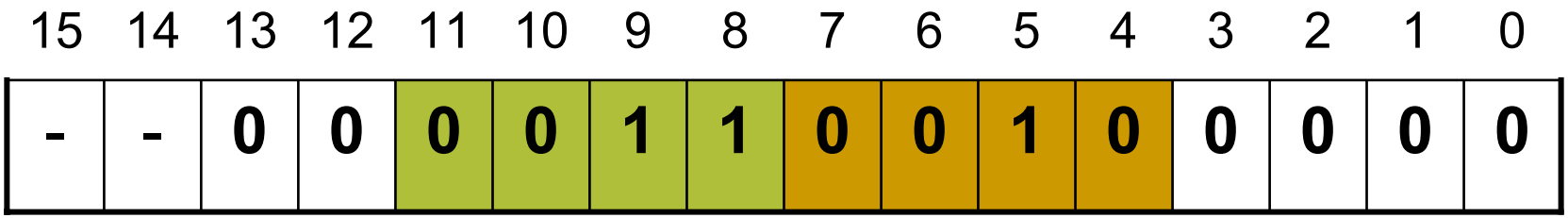
- Lệnh: L S5T#0h0m3s200ms

  khai báo thời gian trễ 3200ms.

- Lệnh: L W#16#0320

  khai báo thời gian trễ 3200ms.

  Nội dung word thời gian trễ:



Khai báo sử dụng Timer:

- Khai báo loại Timer: có 5 loại
  - Timer đóng chậm
  - Timer đóng chậm có nhớ
  - Timer xung
  - Timer giữ độ rộng xung
  - Timer mở chậm

Timer đóng chậm – On-delay Timer

- Cú pháp: SD <tên timer>
- Hoạt động:

  Khởi động timer khi RLO chuyển từ 0 lên 1.

  Thời gian trễ bắt đầu tính khi RLO=1.

  Trong khoảng thời gian này T-bit =0, khi thời gian trôi qua hết thời gian này thì T-bit=1.

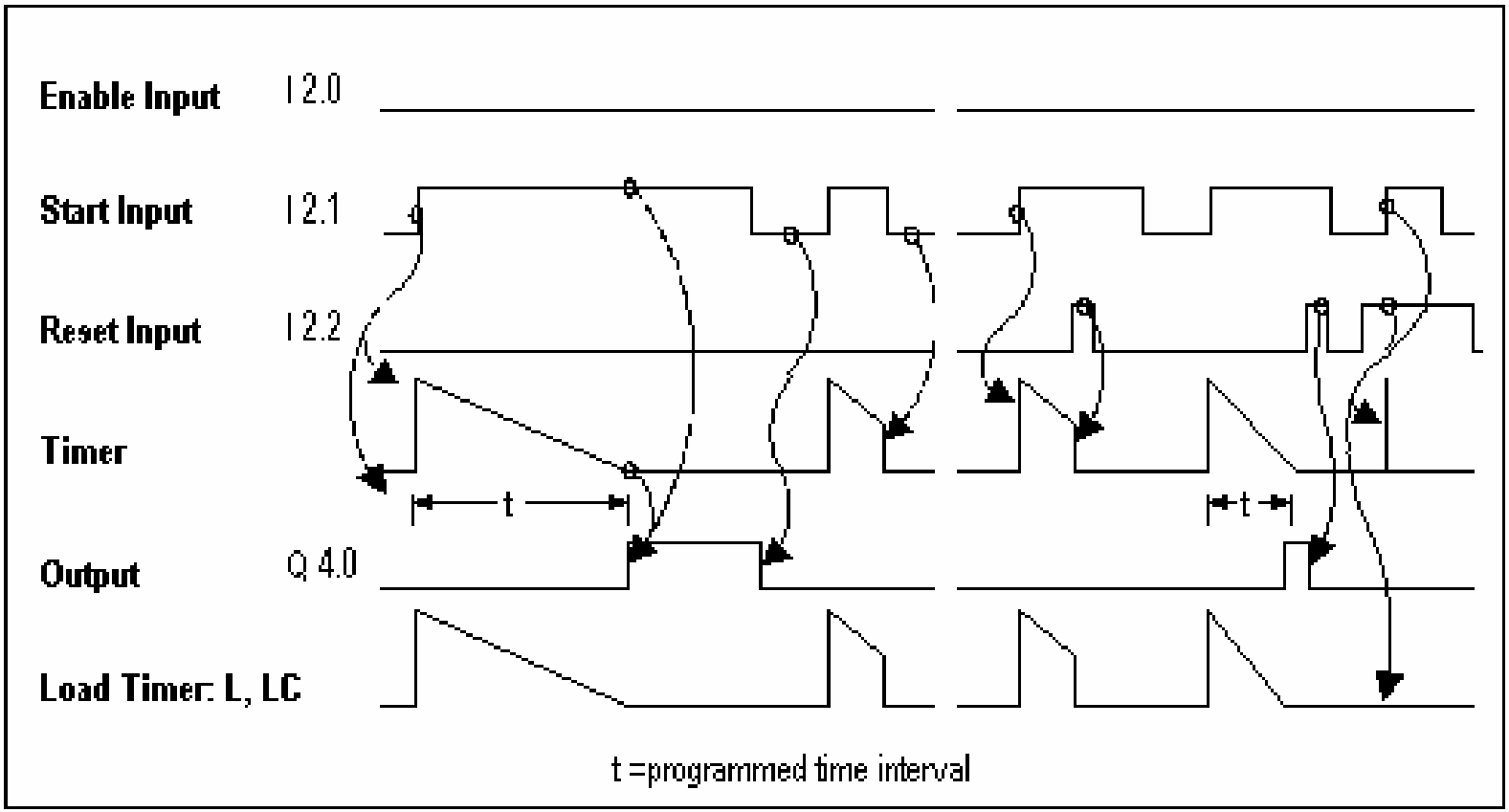
  Thời gian trễ là khoảng thời gian giữa RL0=1 và T-bit=1.

  Khi tín hiệu ngõ vào =0 thì T-bit và T-word =0

Ví dụ:

```
STL          Explanation
A   I 2.0
FR  T1       //Enable timer T1.
A   I 2.1
L   S5T#10s  //Preset 10 seconds into ACCU 1.
SD  T1       //Start timer T1 as an on-delay timer.
A   I 2.2
R   T1       //Reset timer T1.
A   T1       //Check signal state of timer T1.
=   Q 4.0
L   T1       //Load current timer value of timer T1 as binary.
T   MW10
LC  T1       //Load current timer value of timer T1 as BCD.
T   MW12
```

Ví dụ:


Enable Input
I 2.0
Start Input
I 2.1
Reset Input
I 2.2
Timer
t
t
Output
Q 4.0
Load Timer: L, LC
t =programmed time interval

Timer đóng chậm có nhớ – Retentive On-delay Timer

- Cú pháp: SS <tên timer>
- Hoạt động:

  Khởi động timer khi RLO chuyển từ 0 lên 1.

  Thời gian trễ bắt đầu tính khi RLO=1.

  Trong khoảng thời gian này T-bit =0, khi thời gian trôi qua hết thời gian này thì T-bit=1.

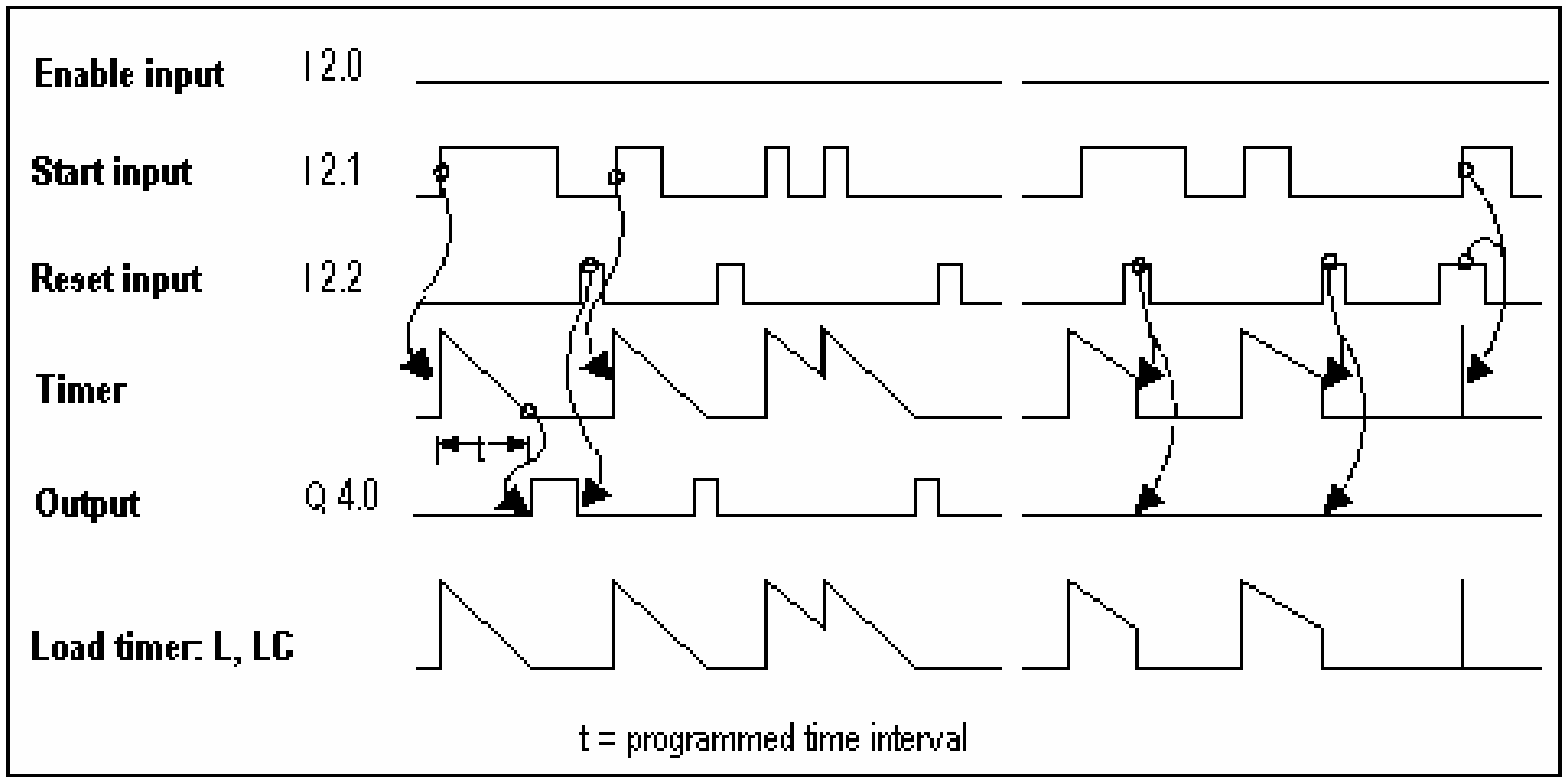
  Thời gian trễ là khoảng thời gian giữa RL0=1 và T-bit=1.

  Thời gian trễ vẫn được nhớ khi RLO chuyển về 0.

Ví dụ:

```
STL          Explanation
A   I 2.0
FR  T1       //Enable timer T1.
A   I 2.1
L   S5T#10s  //Preset 10 seconds into ACCU 1.
SS  T1       //Start timer T1 as a retentive on-delay timer.
A   I 2.2
R   T1       //Reset timer T1.
A   T1       //Check signal state of timer T1.
=   Q 4.0
L   T1       //Load current time value of timer T1 as binary.
T   MW10
LC  T1       //Load current time value of timer T1 as BCD.
T   MW12
```

Ví dụ:


Enable input
I 2.0
Start input
I 2.1
Reset input
I 2.2
Timer
t
Output
Q 4.0
Load timer: L, LC
t = programmed time interval

Timer mở chậm – Off-delay Timer

- Cú pháp: SF <tên timer>
- Hoạt động:

  Khởi động timer khi RLO chuyển từ 1 xuống 0.

  Thời gian trễ bắt đầu tính khi RLO=0.

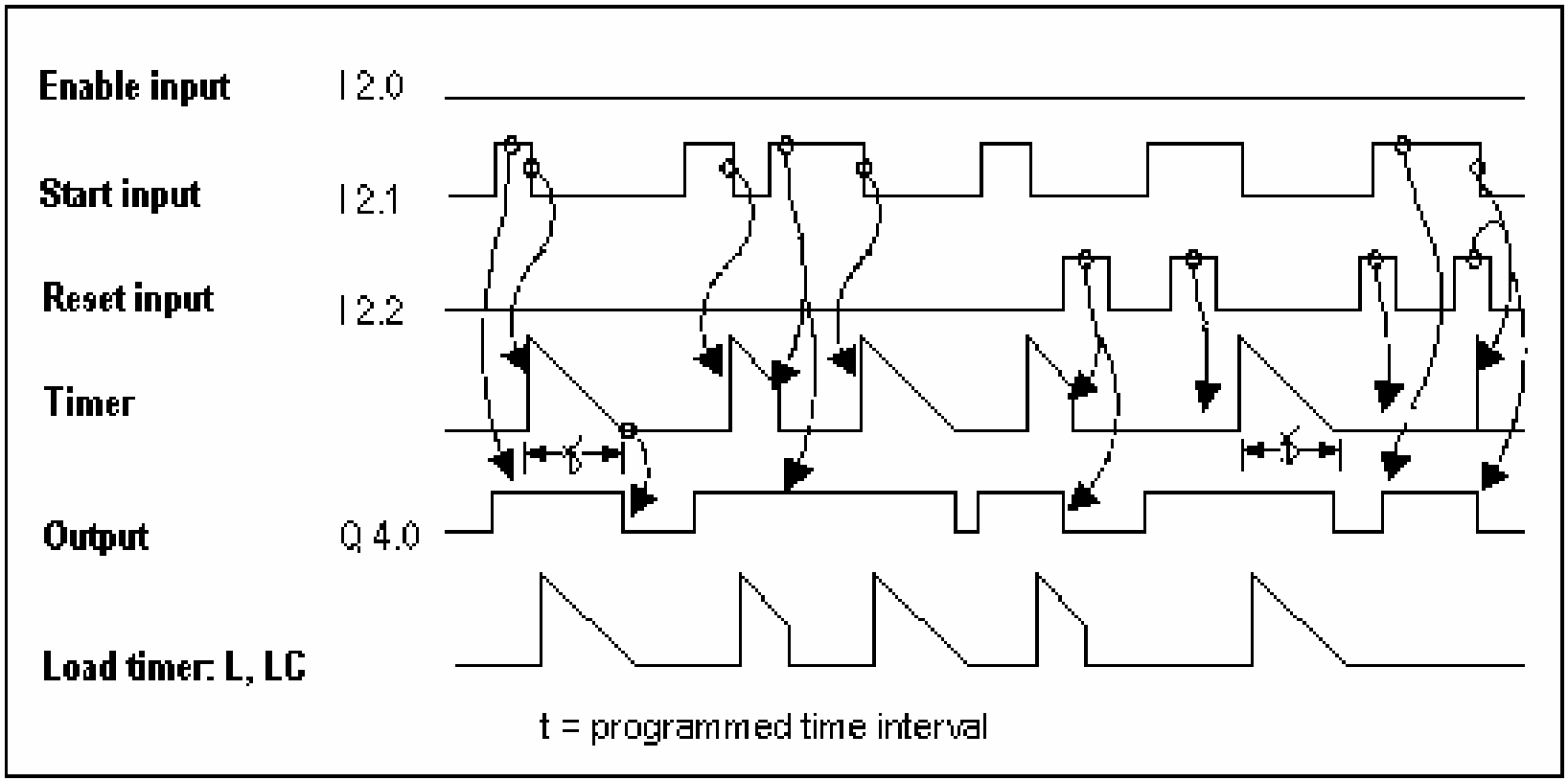
  Khi RLO=1 thì T-bit =1, cho đến khi hết thời gian trễ thì T-bit =0.

  Thời gian trễ được tính từ khi RL0 về 0 cho đến lúc T-bit về 0.

Ví dụ:

```
STL          Explanation
A   I 2.0
FR  T1       //Enable timer T1.
A   I 2.1
L   S5T#10s  //Preset 10 seconds into ACCU 1.
SF  T1       //Start timer T1 as an off-delay timer.
A   I 2.2
R   T1       //Reset timer T1.
A   T1       //Check signal state of timer T1.
=   Q 4.0
L   T1       //Load current timer value of timer T1 as binary.
T   MW10
LC  T1       //Load current timer value of timer T1 as BCD.
T   MW12
```

Ví dụ:


Enable input
I 2.0
Start input
I 2.1
Reset input
I 2.2
Timer
Output
Q 4.0
Load timer: L, LC
t = programmed time interval

Timer xung – Pulse-delay Timer

- Cú pháp: SP <tên timer>
- Hoạt động:

  Khởi động timer khi RLO chuyển từ 0 lên 1.

  Thời gian trễ bắt đầu tính khi RLO=1.

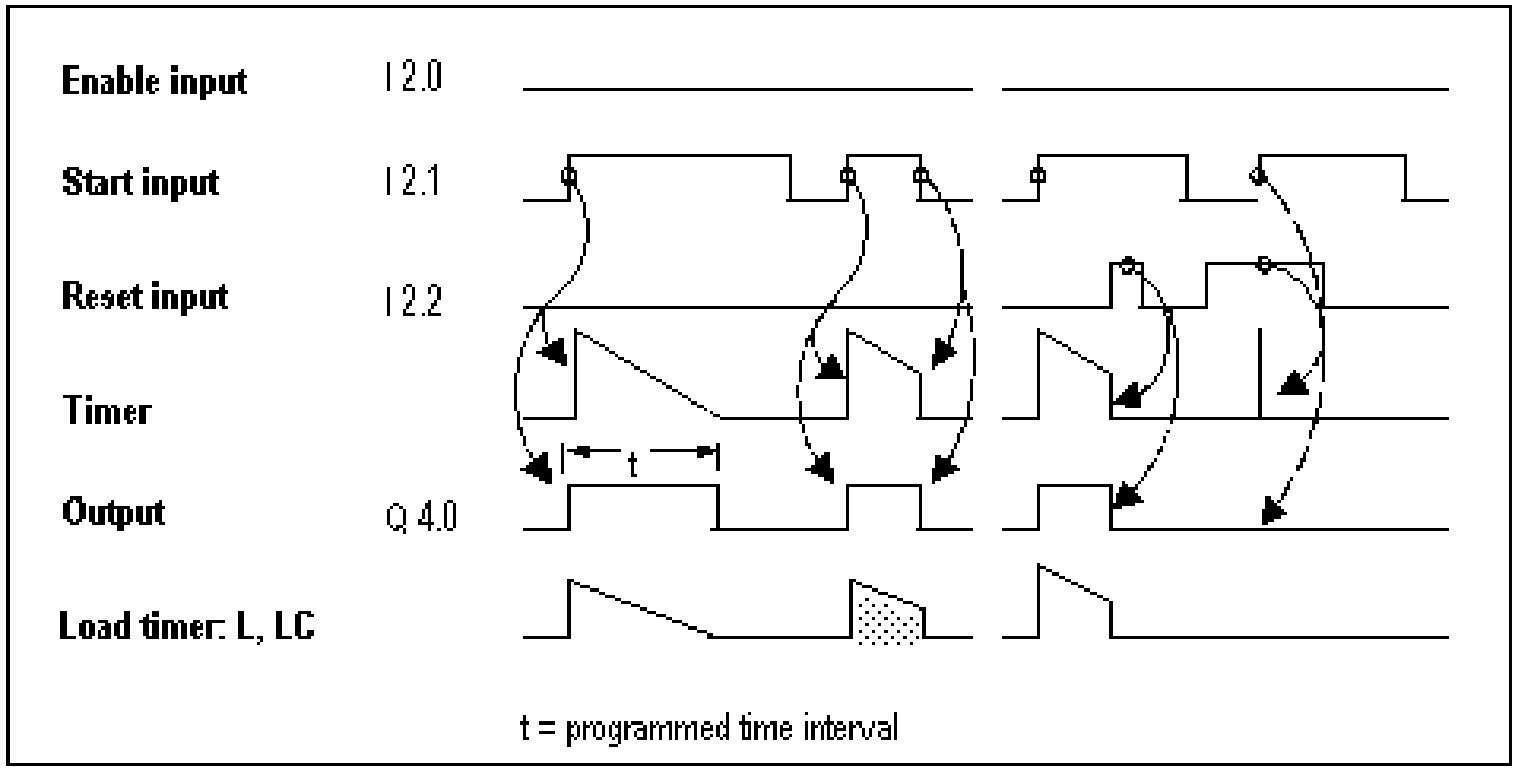
  Trong khoảng thời gian trễ thì T-bit =1, đến khi hết thời gian trễ thì T-bit =0.

  Thời gian trễ được tính từ khi RL0 lên 1 cho đến lúc T-bit về 0.

Ví dụ:

```
STL          Explanation
A   I 2.0
FR  T1       //Enable timer T1.
A   I 2.1
L   S5T#10s  //Preset 10 seconds into ACCU 1.
SP  T1       //Start timer T1 as a pulse timer.
A   I 2.2
R   T1       //Reset timer T1.
A   T1       //Check signal state of timer T1.
=   Q 4.0
L   T1       //Load current time value of timer T1 as binary.
T   MW10
LC  T1       //Load current time value of timer T1 as BCD.
T   MW12
```

Ví dụ:


Enable input
I 2.0
Start input
I 2.1
Reset input
I 2.2
Timer
t
Output
Q 4.0
Load timer: L, LC
t = programmed time interval

Khai báo Timer dạng LAD

LỆNH VỀ COUNTER

- Giới thiệu:
  - Counter là bộ đếm sườn xung của các tín hiệu ngõ vào, được đặt tên là Cx (0<x<255).
  - Counter có 2 thông số sử dụng: C-word và C-bit.

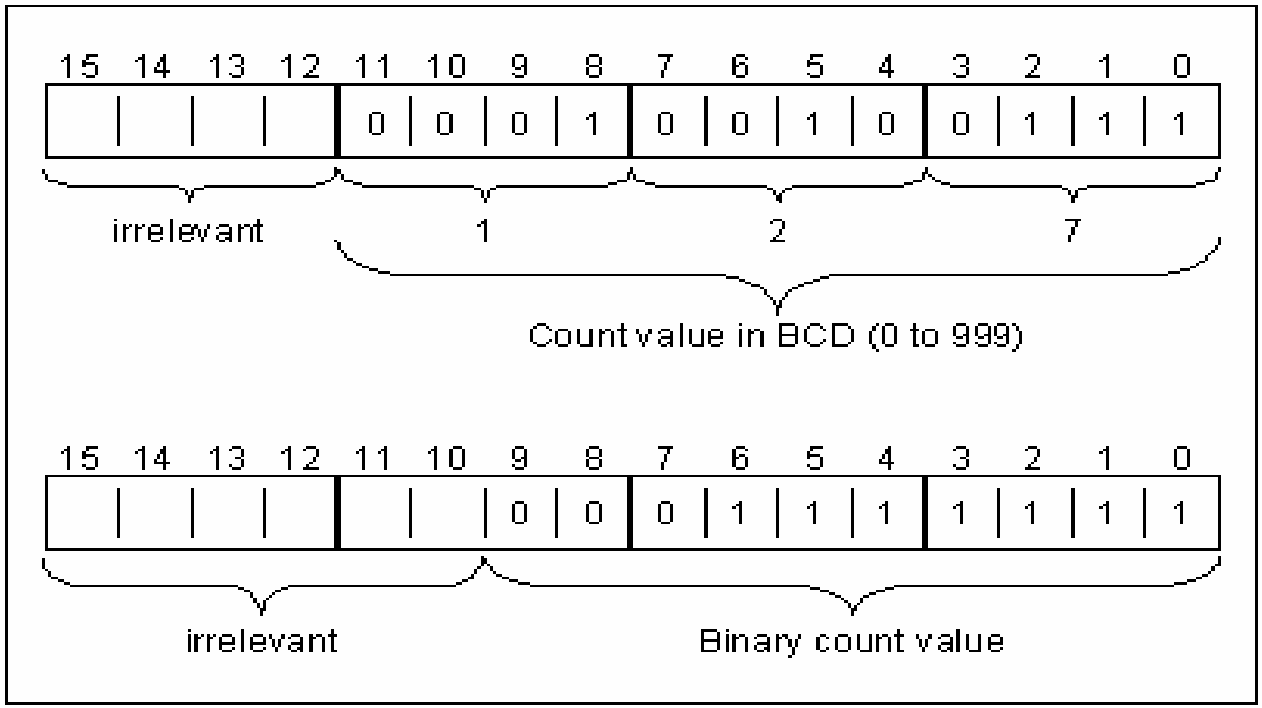
    C-word chứa giá trị đếm tức thời (CV≥0),

    C-bit báo trạng thái của C-word:

    CV<>0 thì C-bit=1; CV=0 thì C-bit=0.
  - Khai báo số đếm đặt trước PV bằng word 16 bit.
  - Giá trị PV được chuyển vào C-word khi có tín hiệu Set

LỆNH VỀ COUNTER

- Nội dung C-word sau khi nạp số đếm đặt trước:
  - 4 bit cao nhất: không sử dụng.
  - 12 bit thấp chứa mã BCD của 3 số từ 0 đến 999

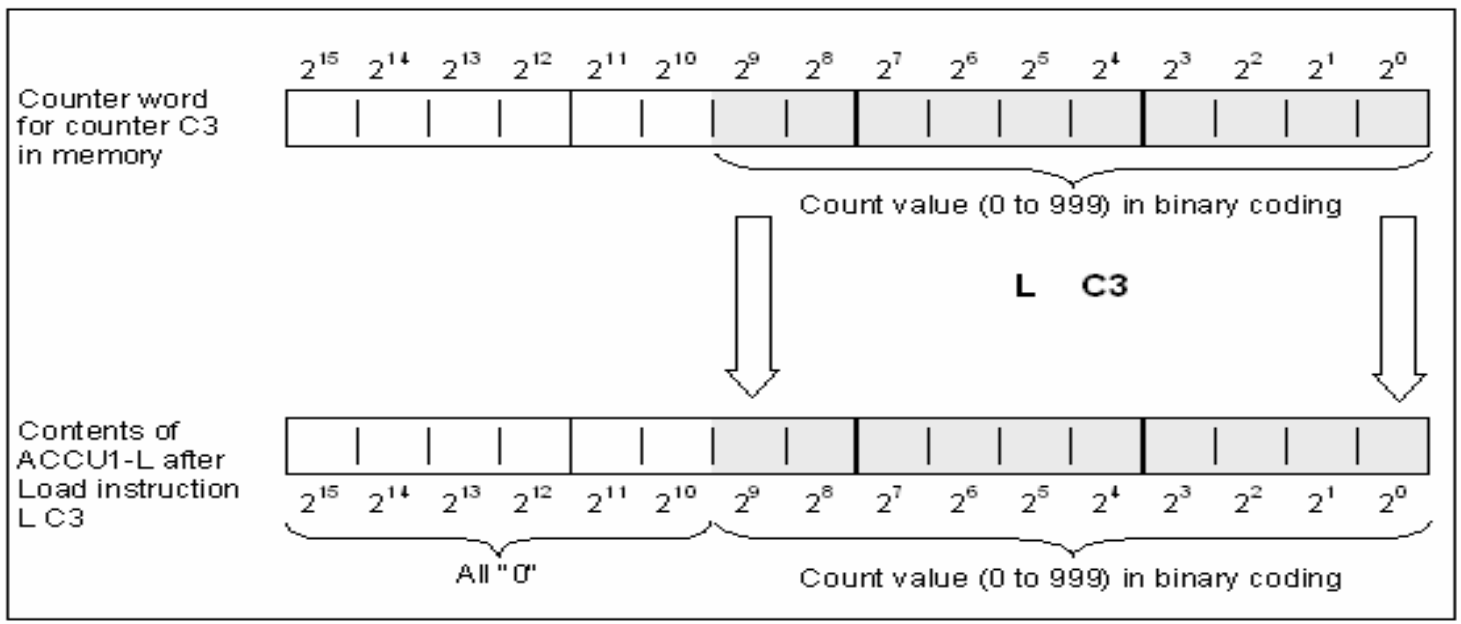


LỆNH VỀ COUNTER

- Đọc nội dung thanh ghi C-word:

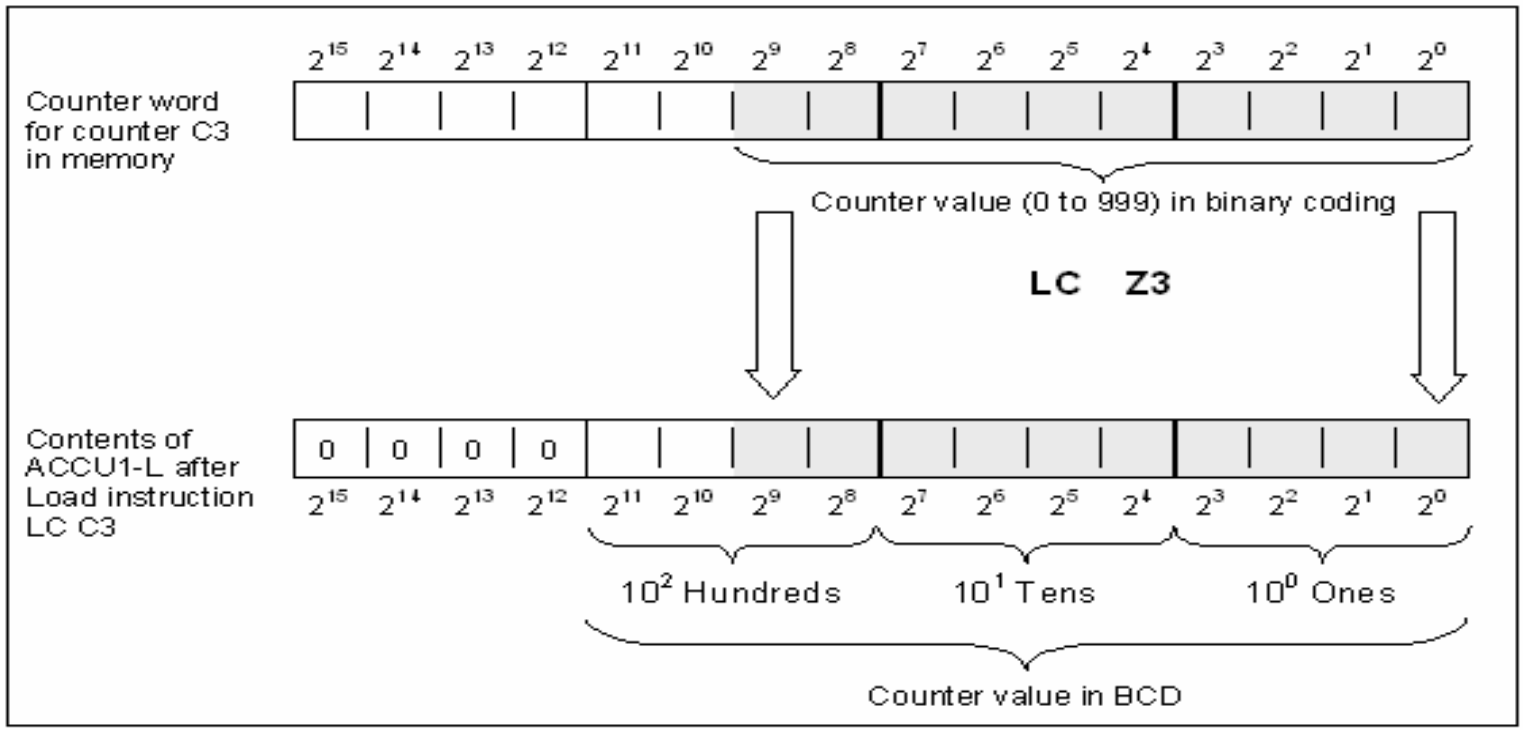
Nội dung C-word được đọc vào ACCU1: có 2 cách đọc

- Đọc số đếm tức thời dạng số nguyên nhị phân:



LỆNH VỀ COUNTER

- Đọc nội dung thanh ghi C-word:
  - Đọc số đếm tức thời dạng mã BCD:



LỆNH VỀ COUNTER

- Khai báo sử dụng Counter:
  - Khai báo tín hiệu enable nếu muốn sử dụng tín hiệu chủ động kích đếm.
  - Khai báo tín hiệu đầu vào CU đếm lên.
  - Khai báo tín hiệu đầu vào CD đếm xuống.
  - Khai báo tín hiệu đặt set và giá trị đặt trước PV.
  - Khai báo tín hiệu xóa reset.

Khai báo sử dụng Counter:

- Khai báo tín hiệu Enable:

Cú pháp: (**A <địa chỉ bit>)**

FR <counter>

<địa chỉ bit>: xác định tín hiệu chủ động kích.

<counter>: loại Counter sử dụng – dạng Cx

Khai báo sử dụng Counter:

- Khai báo tín hiệu vào CU:

Cú pháp: (**A <địa chỉ bit>)**

CU <counter>

<địa chỉ bit>: tín hiệu làm xung đếm.

<counter>: loại Counter đếm lên – tăng giá trị CV mỗi khi có xung đếm.

Khai báo sử dụng Counter:

- Khai báo tín hiệu vào CD:

Cú pháp: (**A <địa chỉ bit>)**

CD <counter>

<địa chỉ bit>: tín hiệu làm xung đếm.

<counter>: loại Counter đếm xuống – giảm giá trị CV mỗi khi có xung đếm.

Khai báo sử dụng Counter:

- Khai báo tín hiệu đặt SET:

Cú pháp: **(A <địa chỉ bit>**

L C#<hằng số>)

S <counter>

Ý nghĩa: nạp giá trị đếm từ thanh ghi ACCU1 vào <counter> khi RLO chuyển từ 0 lên 1.

```
STL      Explanation
A   I 2.3 //Check signal state at input I 2.3.
L   C#3   //Load count value 3 into ACCU 1-L.
S   C1    //Set counter C1 to count value if RLO transitions from 0
          to 1.
```

Khai báo sử dụng Counter:

- Khai báo tín hiệu xóa RESET:

Cú pháp: **(A <địa chỉ bit>)**

R <counter>

Ý nghĩa: xóa giá trị của <counter> về 0 khi RLO=1

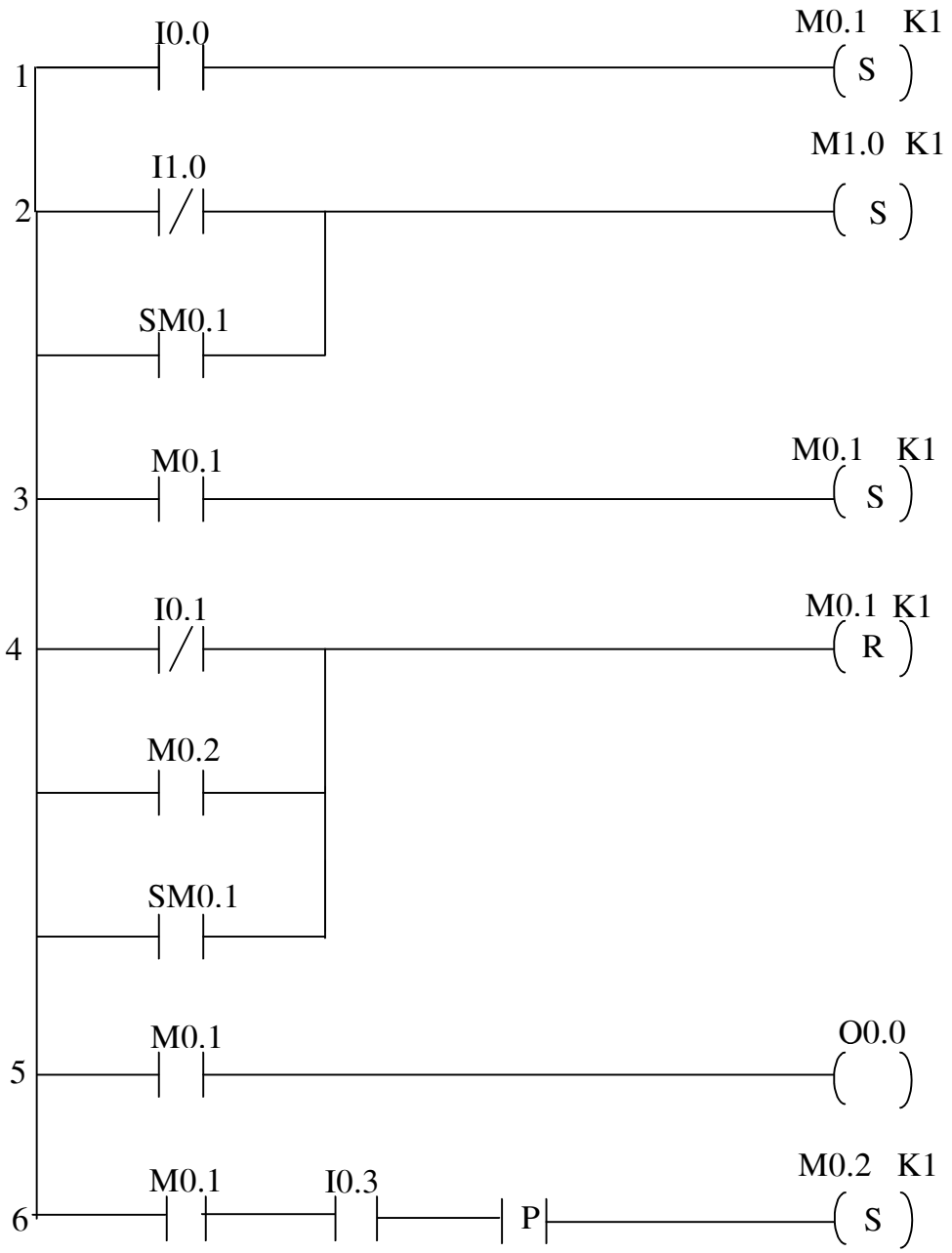
```
STL      Explanation
A   I 2.3 //Check signal state at input I 2.3.
R   C3    //Reset counter C3 to a value of 0 if RLO transitions from
          0 to 1.
```

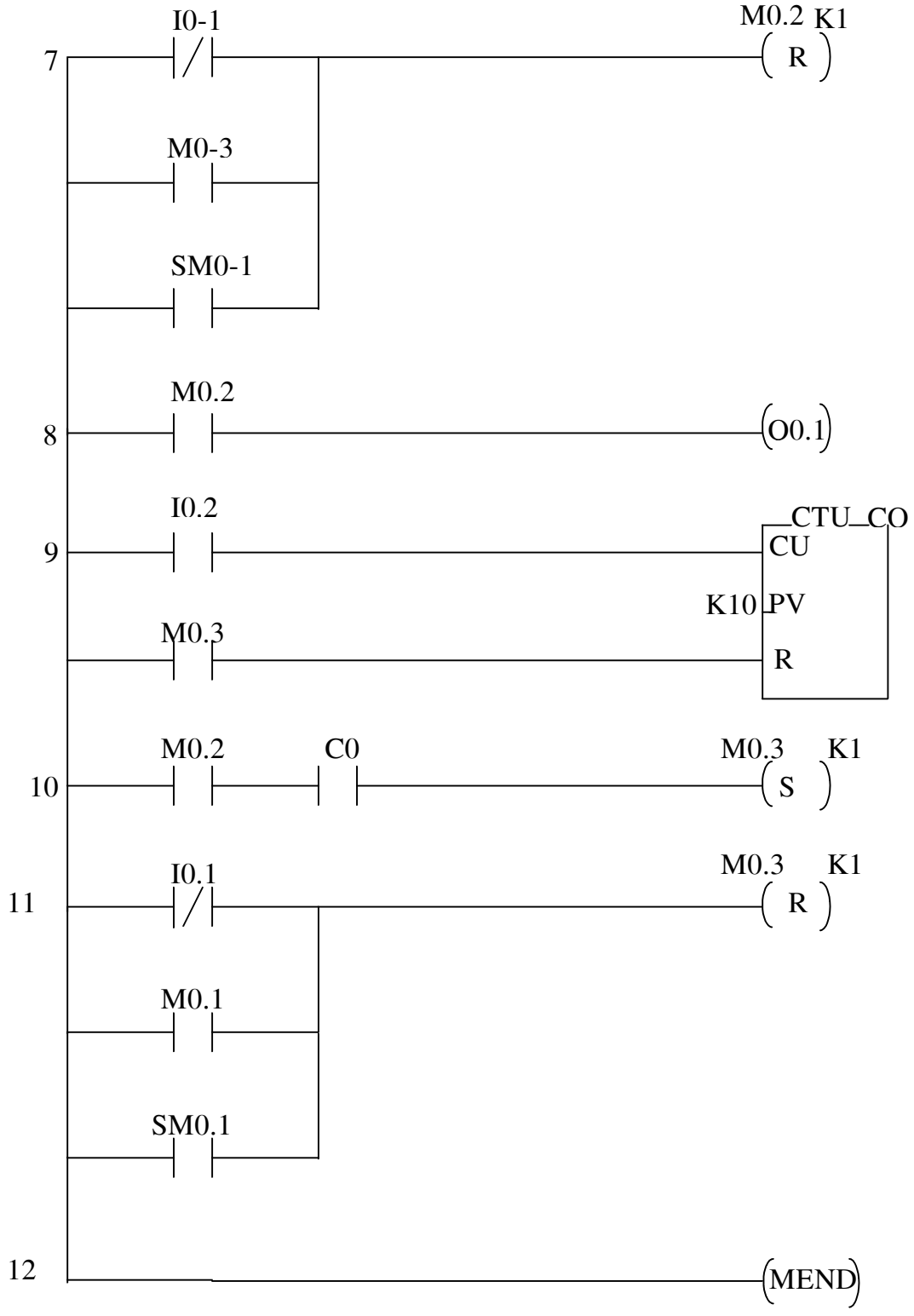
Khai báo Counter dạng LAD

CHƯƠNG 6

ỨNG DỤNG PLC VÀ CẢM BIẾN ĐỂ ĐIỀU KHIỂN DÂY CHUYỀN ĐÓNG HỘP

III.1. Sơ đồ công tắc :




7
I0-1
M0.2 K1
R
M0-3
SM0-1
8
M0.2
O0.1
9
I0.2
CTU_C0
CU
K10 PV
M0.3
R
10
M0.2
C0
M0.3 K1
S
11
I0.1
M0.3 K1
R
M0.1
SM0.1
12
MEND

III.2. Liệt kê lệnh :

```
NETWORK1
      0     LD      I0.0
      2     S       M1.0        K1
NETWORK2
      9     LDN     I0.1
      11    0       SM0.1
      13    R       M1.0        K1
NETWORK3
      20    LD      M1.0
      22    S       M0.1        K1
NETWORK4
      29    LDN     I0.1
      31    0       M0.2
      33    0       SM0.1
      35    R       M0.1        K1
NETWORK5
      42    LD      M0.1
      44    =       Q0.0
NETWORK6
      46    LD      M0.1
      48    A       I0.3
      50    EU
      51    S       M0.2        K1
NETWORK7
      58    LDN     I0.1
      60    0       M0.3
      62    0       SM0.1
      64    R       M0.2        K1
```

```
NETWORK8
      71    LD        I0.2
      73    =         Q0.1
NETWORK9
      75    LD        I0.2
      77    LD        M0.3
      79    CTU       C0              K10

NETWORK10
      85    LD        M0.2
      87    A         C0
      89    S         M0.3            K1
NETWORK11
      96    LDN       I0.1
98    0         M0.1
      100   0         SM0.1
      102   R         M0.3
NETWORK12
      109   MEND
```

❖ **Mô tả các toán hạng :**

I0.0 : Nút khởi động Start .
I0.0 : Nút dừng Stop .
I0.2 : Cảm biến số lượng .
I0.3 : Cảm biến thùng .
Q0.0: Động cơ của băng chuyền thùng .

Q0.1: Động cơ của băng chuyền táo .

III.3. Mô tả hoạt động:

Khi ấn nút khởi động (Start) thì băng chuyền thùng vận hành. Đế khi cảm biến nhận biết có thùng tới thì băng Chuyền dừng lại và băng chuyền táo bắt đầu hoạt động để cho táo vào thùng. Lúc này cảm biến đầu vào bộ đếm sẽ đếm, khi đếm đủ mười quả táo thì băng chuyền, và băng chuyền thùng tiếp tục vận hành. Bộ đếm được dặt lại và quá trình vận hành được lập lại cho đến khi ấn nút dừng (Stop).

THI CÔNG MÔ HÌNH

Sau khi viết chương trình và kiểm tra, em thấy chương trình đã chạy tốt nên em đã tiến hành thi công mô hình thí nghiệm.

Trình tự tiến hành như sau:

- Chuẩn bị vật tư thiết bị.
- Tiến hành kết nối đi dây để hoàn thành mô hình thí nghiệm.

KẾT LUẬN

Tuy thời gian có hạn hẹp, nhưng được sự hướng dẫn tận tình của thầy **Nguyễn Văn Mạnh** cùng với sự cố gắng của bản thân, em đã hoàn thành luận văn tốt nghiệp của mình đúng theo thời gian qui định.

Sau khi hoàn thành tập luận văn này, em cũng đã tìm hiểu và nắm vững hơn kiến thức về PLC, về cảm biến ánh sáng và ứng dụng thực tế của chúng.

Với thời gian có hạn, hơn nữa đề tài lại được làm độc lập bởi một sinh viên nên khó tránh khỏi những thiếu sót trong quá trình thi công mô hình và hoàn tất đề tài.

Thông qua đề tài này, ta thấy PLC được ứng dụng rất rộng rãi và đa dạng trong rất nhiều lĩnh vực sản xuất.

Cuối cùng, một lần nữa em xin gởi lời cảm ơn đến tất cả các Thầy, Cô của trường Đại Học Sư Phạm Kỹ Thuật đã dạy dỗ và cung cấp cho em nhiều kiến thức quí báu trong quá trình em theo học tại trường.

HƯỚNG DẪN SỬ DỤNG S7-300

Duy Minh Software

Chương trình học S7_200

A.Ôn tập kỹ thuật số:

I/ Các kiểu số:

1/ Số nhị phân(cơ số 2):

Là số mà hàng đơn vị chỉ có 2 giá trị là **0 (sai)** và **1 (đúng)** .

VD : theo chiều tăng dần ta có 0,1,10,11,100,101,110,111,1000,...

Số 8 biểu diễn trong hệ nhị phân là: **1000**

2/ Hệ cơ số 8:

Là số mà hàng đơn vị có 8 giá trị là 0,1,2,3,4,5,6,7.

VD : theo chiều tăng dần ta có 0,1,2,3,4,5,6,7,10,11,12,13,14,15,16,17,20,...

Số 15 biểu diễn trong hệ cơ số 8 là : **17**

3/ Hệ cơ số 10:

Là số mà hàng đơn vị có 10 giá trị là 0,1,2,3,...,9

VD :Theo chiều tăng dần ta có: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,...

Số 15 biểu diễn trong hệ cơ số 10 là **15**

4/ Số hex(cơ số 16):

Là số mà hàng đơn vị chỉ có 16 giá trị là 0,1,2,3,4,5,...,9,A,B,C,D,E,F.

VD :Theo chiều tăng dần ta có 0,1,2,3,...,A,B,C,D,E,F,10,11,...,19,1A,1B,1C,1E,1F,20.....

Số 20 biểu diễn trong hệ cơ số 16 là**14**

5/ Mã BCD số nguyên dương:

Mã BCD là dạng dùng biến hai trị (0 hoặc 1) để thể hiện những chữ số.

VD: Mã BCD số 259 là: 0010 0101 1001

2 5 9

6/ Cách qui đổi giá trị một số hệ cơ số n sang hệ thập phân:

Giả sử một số hệ cơ n có (m+1) chữ số tổng quát như sau:

$X_mX_{m-1}X_{m-2}.....X_2X_1X_0$

Trong đó, các giá trị m,(m-1), (m-2)... 2,1,0 được gọi là trọng số các chữ số.

Số đứng ở vị trí cao nhất (biên trái) gọi là số có trọng số cao nhất, số đứng ở vị trí thấp nhất (biên phải) gọi là số có trọng số thấp nhất .

Cách qui đổi giá trị thập phân như sau:

Giá trị thập phân = $X_m.n^m + X_{m-1}.n^{m-1} + X_{m-2}.n^{m-2} +...+ X_1.n^1 + X_0.n^0$

$n^1 = n$

$n^0 = 1$

VD: đổi số 24B3 trong hệ cơ 16 sang hệ thập phân

24B3

trọng số 3210

đổi sang hệ thập phân:

Giá trị = $2.16^3+4.16^2+11.16^1+3 =9395$ (B=11)

VD: đổi số 1011 trong hệ nhị phân sang hệ thập phân

1011

trọng số 3210

đổi sang hệ thập phân:

Giá trị $= 1.2^3+0.2^2+1.2^1+1 =11$

Cách qui đổi giá trị một số hệ nhị phân sang số HEX:

Để biểu diễn một số hàng đơn vị của số HEX bằng số nhị phân, ta cần một số nhị phân 4 bit, cụ thể như sau:

Hệ nhị phân	Số HEX
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Như vậy, để đổi một số nhị phân sang số HEX, ta có qui tắc như sau:

4 bit thấp nhất trong số nhị phân tương đương hàng đơn vị trong hệ HEX

4 bit kế tiếp tương đương hàng chục trong hệ HEX

...

VD: 1011 1000 0101 = B85

1000 0100 1111 1100 = 84FC

Qui tắt đổi ngược lại cũng tương tự.

Bài tập bài 1:

a/ Đổi số **HEX 12AB** sang số thập phân

b/ Đổi số nhị phân **1011010** sang hệ thập phân

c/ Đổi số HEX **A9C** sang hệ thập phân và nhị phân. Đổi số nhị phân trên sang số thập phân để kiểm tra kết quả.

Tính giá trị thập phân cao nhất cuả số nhị phân 4 bit. Rút ra qui luật tính giá trị cao nhất cuả số nhị phân n bit.

II/ Các khái niệm về số:

1/ Bit:

Chỉ có 2 giá trị: **1** (đúng) hoặc **0** (sai)

Ví dụ: Biến Motor bằng 1 thì Motor chạy

Ngược lại biến Motor bằng 0 thì Motor dừng.

2/ Byte:

Là số có giá trị 8 bit,do vậy giá trị nhỏ nhất của Byte là 0 (00000000),và giá trị lớn nhất là 255 (11111111)

3/ Word:

Là số có giá trị 16 bit,do vậy giá trị nhỏ nhất của Word là 0 ,và giá trị lớn nhất là 2^{16}-1

4/Double Word:

Là số nguyên có giá trị 32 bit,do vậy giá trị nhỏ nhất của Double Word là 0 ,và giá trị lớn nhất là 2^{32}-1

5/ Số Int:

Là số có giá trị 16 bit,nhưng bit có trọng số lớn nhất là bit dấu,do vậy giá trị của số dạng này có giá trị từ $-(2^{15}-1)$ đến $(2^{15}-1)$.

6/ Số Double Int:

Là số nguyên có giá trị 32 bit,nhưng bit có trọng số lớn nhất là bit dấu,do vậy giá trị của số dạng này có giá trị từ $-(2^{31}-1)$ đến $(2^{31}-1)$.

6/ Số Real:

Là số thực có giá trị 32 bit,nhưng bit có trọng số lớn nhất là bit dấu,do vậy giá trị của số dạng này có giá trị từ $-(2^{31}-1)$ đến $(2^{31}-1)$.

III/ Các phép toán Logic:

1/ Phép AND

Bảng giá trị phép toán And:

X1	X2	X1 AND X2
0	0	0
0	1	0
1	0	0
1	1	1

2/ Phép OR:

Bảng giá trị phép toán OR:

X1	X2	X1 OR X2
0	0	0
0	1	1
1	0	1
1	1	1

3/ Phép XOR:

Bảng giá trị phép toán XOR:

X1	X2	X1 XOR X2
0	0	0
0	1	1
1	0	1
1	1	0

3/ Phép NOT:

Bảng giá trị phép toán NOT:

X1	NOT X1
0	1
1	0

Khi thực hiện phép toán AND,OR hay XOR cho 2 số có n bit thì các bit có trọng số bằng nhau sẽ được AND, OR hay XOR từng đôi một.

VD1: 1001
And
1101
Kết quả
1001

VD2: 1001
Xor
1101
Kết quả
0100

Bài tập bài 2 :

Thực hiện phép tính **And,Or,Xor,Not** 2 số sau:

1100 0110 0010 0011
1100 1010 1011 0001

4/ Các Tín hiệu kết nối với PLC:

a/Tín hiệu số : Là các tín hiệu thuộc dạng hàm Boolean, dạng tín hiệu chỉ có 2 trị 0 hoặc 1.

Đối với PLC Siemens :

Mức 0 : tương ứng với 0V hoặc hở mạch

Mức 1 : Tương ứng với 24V

Vd: Các tín hiệu từ nút nhấn ,từ các công tắc hành trình..... đều là những tín hiệu số

b/ Tín hiệu tương tự : Là tín hiệu liên tục, từ 0-10V hay từ 4-20mA....

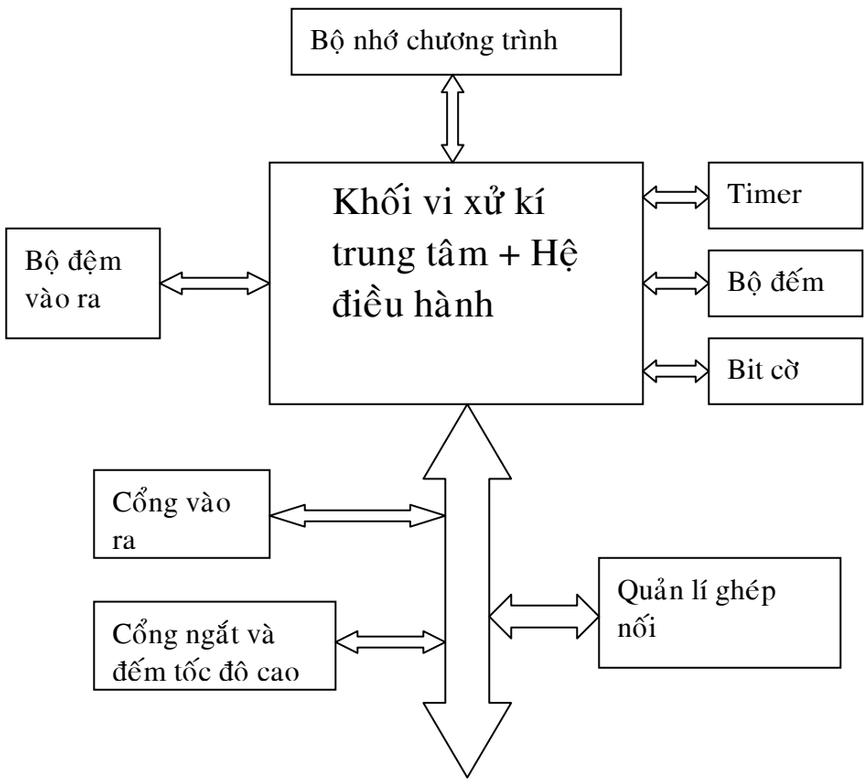
Vd: Tín hiệu đọc từ Loadcell,từ cảm biến lưu lượng...

c/ Tín hiệu khác : Bao gồm các tín hiệu giao tiếp với máy tính ,với các thiết bị ngoại vi khác bằng các giao thức khác nhau như giao thức RS232,RS485,Modbus....

B. Nhập Môn PLC:

I/ Thiết bị điều khiển Logic khả trình:

1/ Giới thiệu PLC:



Thiết bị điều khiển Logic khả trình PLC (Programmable Logic Control) là loại thiết bị cho phép thực hiện linh hoạt các thuật toán điều khiển số thông qua một ngôn ngữ lập trình ,thay cho việc phải thể hiện thuật toán đó bằng các mạch số .Như vậy với chương trình điều khiển trong mình .PLC trở thành bộ điều khiển số nhỏ gọn ,dễ dàng thay đổi thuật toán và đặc biệt dễ dàng trao đổi thông tin với môi trường xung quanh (Với các PLC khác hoặc với máy tính).

Toàn bộ chương trình được lưu nhớ trong bộ nhớ của PLC dưới dạng các khối chương trình con hoặc chương trình ngắt (Khối chính OB1). Trường hợp dung lượng nhớ của PLC không đủ cho việc lưu trữ chương trình thì ta có thể sử dụng thêm bộ nhớ ngoài hỗ trợ cho việc lưu chương trình và lưu dữ liệu (Catridge).

Để có thể thực hiện được một chương trình điều khiển ,tất nhiên PLC phải có tính năng như một máy tính ,nghĩa là phải có một bộ vi xử lí (CPU) ,một hệ điều hành ,một bộ nhớ để lưu chương trình điều khiển ,dữ liệu và tất nhiên là phải có các cổng vào ra để giao tiếp với các đối tượng điều khiển và để trao đổi thông tin với môi trường xung quanh .Bên cạnh đó nhằm phục vụ các bài toán điều khiển số ,PLC còn cần phải có thêm những khối chức năng đặc biệt khác như bộ đếm (Counter),bộ định thời gian (Timer) ....Và những khối hàm chuyên dụng.

2/ Bộ nhớ PLC: gồm 3 vùng chính.

a/Vùng chứa chương trình ứng dụng : Vùng chứa chương trình được chia thành 3 miền :

i/ OB1 (Organisation block) : miền chứa chương trình tổ chức,chứa chương trình chính,các lệnh trong khối này luôn được quét.

ii/ Subroutine (Chương trình con) : Miền chứa chương trình con ,được tổ chức thành hàm và có biến hình thức để trao đổi dữ liệu,chương trình con này sẽ được thực hiện khi nó được gọi trong chương trình chính.

iii/ Interrup (Chương trình ngắt) : Miền chứa chương trình ngắt ,được tổ chức thành hàmvà có khả năng trao đổi dữ liệu với bất cứ 1 khối chương trình nào khác .Chương trình này sẽ được thực hiên khi có sự kiện ngắt xảy ra. Có rất nhiều sự kiện ngắt như: Ngắt thời gian,ngắt xung tốc độ cao...

b/ Vùng chứa tham số của hệ điều hành: Chia thành 5 miền khác nhau

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

Q (Process Image Output): Miền bộ đệm các dữ liệu cổng ra số .Kết thúc giai đoạn thực hiện chương trình,PLC sẽ chuyển giá trị logic của bộ đệm Q tới các cổng ra số.Thông thường chương trình không trực tiếp gán giá trị tới tận cổng ra mà chỉ chuyển chúng tới bộ đệm Q.

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

T (Timer): Miền nhớ phục vụ bộ thời gian (Timer) bao gồm việc lưu trữ giá trị thời gian đặt trước (PV-Preset Value),giá trị đếm thời gian tức thời (CV –Current Value) cũng như giá trị Logic đầu ra của bộ thời gian.

C (Counter): Miền nhớ phục vụ bộ đếm bao gồm việc lưu trữ giá trị đặt trước (PV- Preset Value),giá trị đếm tức thời (CV _ Current Value)và giá trị logic đầu ra của bộ đệm.

c/ Vùng chứa các khối dữ liệu: được chia làm 2 loại:

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

L (Local data block) : Miền dữ liệu địa phương ,được các khối chương trình OB1,Chương trình con,Chương trình ngắt tổ chức và sử dụng cho các biến nháp tức thời và trao đổi dữ liệu của biến hình thức với những khối chương trình gọi nó .Nội dung của một khối dữ liệu trong miền nhớ này sẽ bị xoá khi kết thúc chương trình tương ứng trong OB1 ,Chương trình con,Chương trình ngắt.Miền này có thể được truy nhập từ chương trình theo bit (L),byte(LB) từ (LW) hoặc từ kép (LD).

3/ Vòng quét chương trình:

PLC thực hiện chương trình theo chu kì lặp .Mỗi vòng lặp được gọi là vòng quét (Scan) .Mỗi vòng quét được bắt đầu bằng giai đoạn chuyển dữ liệu từ các cổng vào số tới vùng bộ đệm ảo

I,tiếp theo là giai đoạn thực hiện chương trình .Trong từng vòng quét chương trình thực hiện từ lệnh đầu tiên đến lệnh kết thúc của khối OB (Block End).Sau giai đoạn thực hiện chương trình là giai đoạn chuyển các nội dung của bộ đệm ảo Qtới các cổng ra số .Vòng quét được kết thúc bằng giai đoạn truyền thông nội bộvà kiểm tra lỗi.

Chú ý rằng bộ đệm I và Q không liên quan tới các cổng vào ra tương tự nên các lệng truy nhập cổng tương tự được thực hiện trực tiếp với cổng vật lí chứ không thông qua bộ đệm.

Thời gian cần thiết để PLC thực hiện 1 vòng quét gọi là thời gian vòng quét (Scan Time).Thời gian vòng quét không cố định ,tức là không phải vòng quét nào cũng được thực hiện trong một khoảng thời gian như nhau .Có vòng quét được thực hiện lâu ,có vòng quét được thực hiện nhanh tuỳ thuộc vào số lệnh trong chương trình được thực hiện và khối dữ liệu truyền thông trong vòng quét đó.

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

Nếu sử dụng các khối chương trình đặc biệt có chế độ ngắt ,,ví dụ như khối OB40,OB80...., chương trình của các khối đó sẽ được thực hiện trong vòng quét khi xuất hiện tín hiệu báo ngắt cùng chủng loại.Các khối chương trình này có thể được thực hiện tại mọi điểm trong vòng quét chứ không bị gò ép là phải ở trong giai đoạn thực hiện chương trình.Chẳng hạn nếu 1 tín hiệu báo ngắt xuất hiện khi PLC đang ở giai đoạn truyền thông và kiểm tra nội bộ,PLC sẽ ngừng công việc truyền thông ,kiểm tra để thực hiện khối chương trình tương ứng với tín hiệu báo ngắt đó .Với hình thức xử lí tín hiệu ngắt như vậy,thời gian vòng quét sẽ càng lớn khi càng có nhiều tín hiệu ngắt xuất hiện trong vòng quét .Do đó để nâng cao tính thời gian thực cho chương trình điều khiển ,tuyệt đối không nên viết chương trình xử lí ngắt quá dài hoặc quá lạm dụng việc sử dụng chế độ ngắt trong chương trình điều khiển.

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

4 / Cấu trúc chương trình:

Chương trình trong S7_300 được lưu trong bộ nhớ của PLC ở vùng giành riêng cho chương trình và có thể được lập với 2 dạng cấu trúc khác nhau.

a/ Lập trình tuyến tính: toàn bộ chương trình nằm trong một khối trong bộ nhớ .Loại hình cấu trúc tuyến tính này phù hợp với những bài toán tự động nhỏ,không phức tạp .Khối được chọn phải là khối OB1 ,là khối mà PLC luôn quét và thực hiện các lệnh trong đó thường xuyên,từ lệnh đầu tiên đến lệnh cuối cùng và quay lại lệnh đầu tiên.

b/ Lập trình có cấu trúc: Chương trình được chia thành những phần nhỏ và mỗi phần thực thi những nhiệm vụ chuyên biệt riêng của no,từng phần này nằm trong những khối chương trình khác nhau .Loại hình cấu trúc này phù hợp với những bài toán điều khiển nhiều nhiệm vụ và phức tạp .PLC S7_200 có 3 loại khối cơ bản sau:

- Loại khối OB1 (Organization Block) : Khối tổ chức và quản lí chương trình điều khiển .Khối này luôn luôn được thực thi,và luôn được quét trong mỗi chu kì quét.
- Loại khối SBR (Khối chương trình con): Khối chương trình với những chức năng riêng giống như 1 chương trình con hoặc một hàm (chương trình con có biến hình thức).Một chương trình ứng dụng có thể có nhiều khối chương trình con và các khối chương trình con này được phân biệt với nhau bằng tên của chương trình con đó.

Loại khối INT (Khối chương trình ngắt) :Là loại khối chương trình đặc biệt có khả năng trao đổi 1 lượng dữ liệu lớn với các khối chương trình khác .Chương trình này sẽ được thực thi mỗi khi có sự kiện ngắt xảy ra.

5 / Các loại PLC S7_200 (Siemens):

Các loại PLC thông thường: CPU222, CPU224, CPU224XP (có 2 cổng giao tiếp), CPU226 (có 2 cổng giao tiếp), CPU226XM

Thông thường S7_200 được phân ra 2 loại chính:

a/ Loại cấp điện áp 220VAC :

Ngõ vào : tích cực mức 1 ở cấp điện áp +24VDC (15VDC – 30VDC)

Ngõ ra : Ngõ ra rơ le

Ưu điểm của loại này là ngõ ra rơ le,do đó có thể sử dụng ngõ ra ở nhiều cấp điện áp (có thể sử dụng ngõ ra 0V,24V,220V....

Tuy nhiên,nhược điểm của nó :do ngõ ra rơ le nên thời gian đáp ứng của rơle không được nhanh cho ứng dụng điều rộng xung,hoặc Output tốc độ cao...

a/ Loại cấp điện áp 24VDC :

Ngõ vào : tích cực mức 1 ở cấp điện áp +24VDC (15VDC – 30VDC)

Ngõ ra : Ngõ ra Transistor

Ưu điểm của loại này là ngõ ra Transistor,do đó có thể sử dụng ngõ ra này để điều rộng xung,hoặc Output tốc độ cao.....

Tuy nhiên,nhược điểm của nó :do ngõ ra Transistor nên ngõ ra chỉ có một cấp điện áp duy nhất là +24VDC,do vậy sẽ gặp rắc rối trong những ứng dụng có cấp điện áp ra là 0VDC,trong trường hợp này buộc ta phải thông qua 1 rơle 24Vdc đệm.

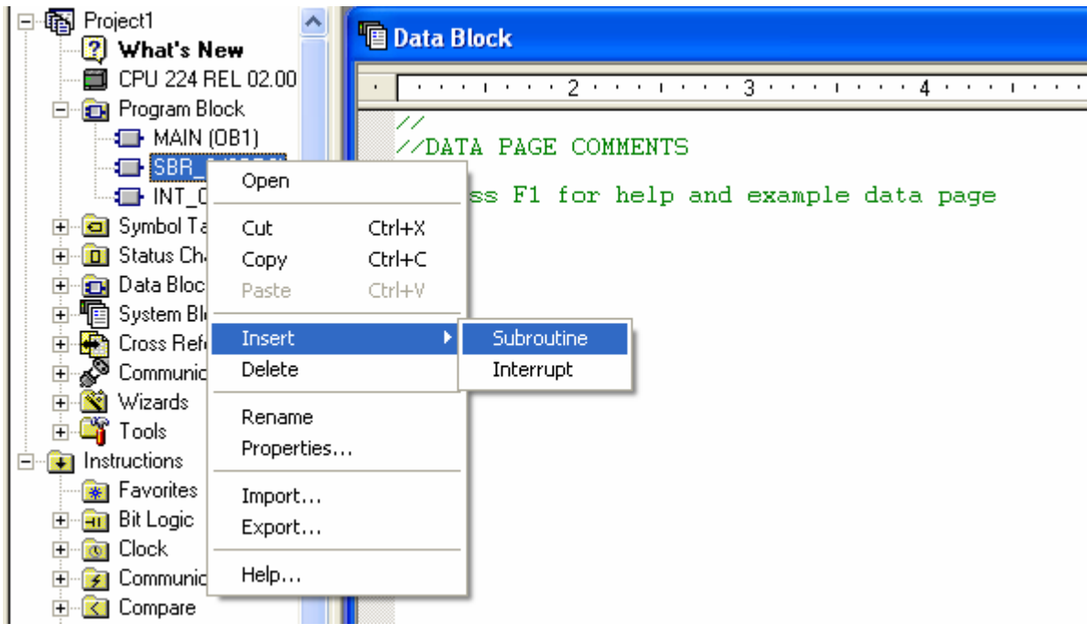
5 / Các khối trong S7_200 Siemens:

a/ Khối Program Block: Có 3 khối chính

i/ Khối OB1: Là khối chứa chương trình chính,và luôn được quét trong mỗi chu kì quét,là khối chính trong việc thiết kế chương trình.

ii/Khối chương trình con: Là khối chứa chương trình con,khối này sẽ được thực thi khi nó được gọi trong chương trình chính.

iii/Khối chương trình ngắt: Là khối chứa chương trình ngắt,khối này sẽ được thực thi khi có sự kiện ngắt xảy ra.



Trong một chương trình,luôn mặc định có một chương trình chính Main,chương trình con SBR_0,và chương trình ngắt INT_0,tuy nhiên ta có thể thêm một hoặc nhiều chương trình con hay chương trình ngắt cũng như có thể xoá nó khi không cần thiết bằng cách Click chuột phải ,rồi chọn Insert Subroutine hay Interrupt.

Tuy nhiên ta không thể thêm hoặc bớt một chương trình chính,do chương trình chính thì chỉ có 1.

b/ Khối Data Block:

Khối chứa dữ liệu của một chương trình,ta có thể định dạng trước dữ liệu cho khối này,và khi Download xuống PLC, thì toàn bộ dữ liệu này sẽ được lưu trong bộ nhớ.

c/ Khối System Block:

Có 10 khối chính:

i/Communication ports: Định dạng cho cổng giao tiếp bao gồm:

Địa chỉ PLC (PLC Address): Địa chỉ mặc định cho PLC là 2, ta có thể thay đổi địa chỉ cho PLC khác 2.Việc định địa chỉ cho PLC đóng vai trò quan trọng trong việc kết nối mạng.

Ngoài ra trong Port giao tiếp ta cũng cần chọn,tốc độ Baud cho việc truyền thông.Tốc độ Baud mặc định là 9600.

ii/Retentive Ranges:

Trong S7_200 cho phép ta chọn 5 phân vùng có thể lưu trữ dữ liệu khi mất điện,nếu ta chọn vùng dữ liệu nào trong Retentive thì giá trị của vùng đó sẽ vẫn không thay đổi khi mất điện,ngược lại giá trị đó sẽ bị reset về 0 khi mất điện.

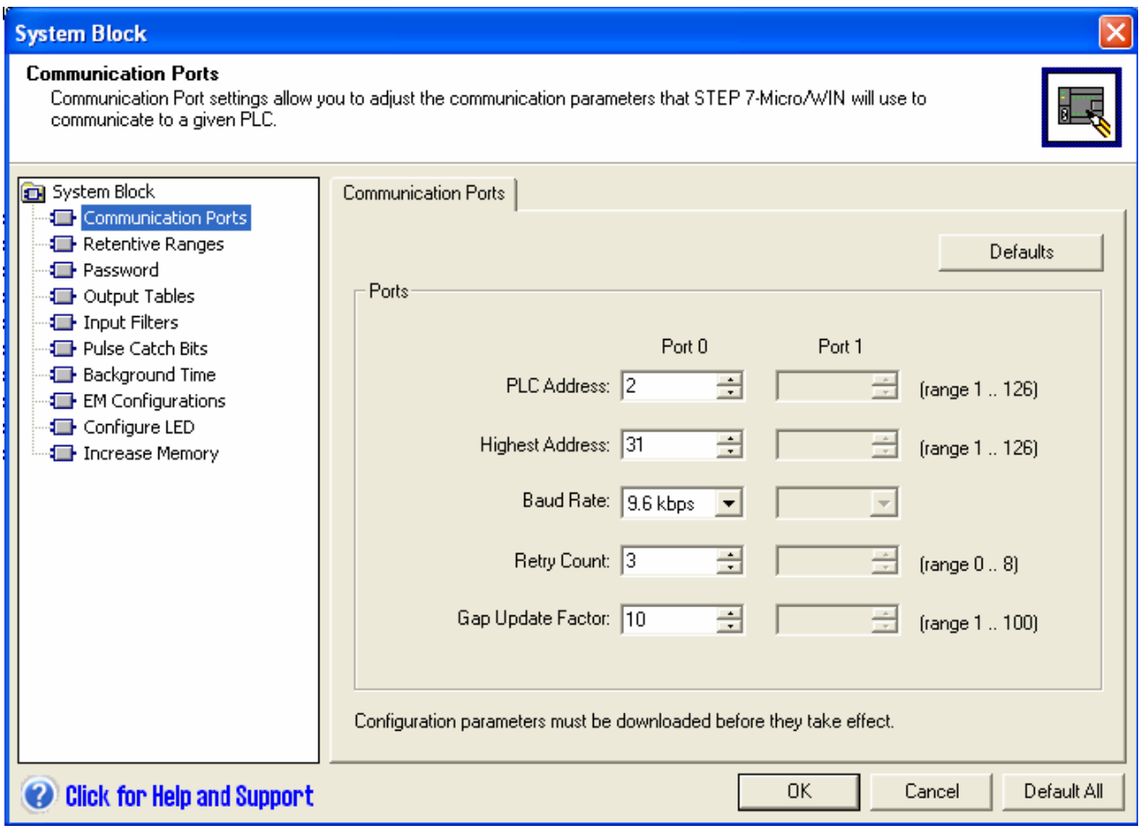
iii/Password:

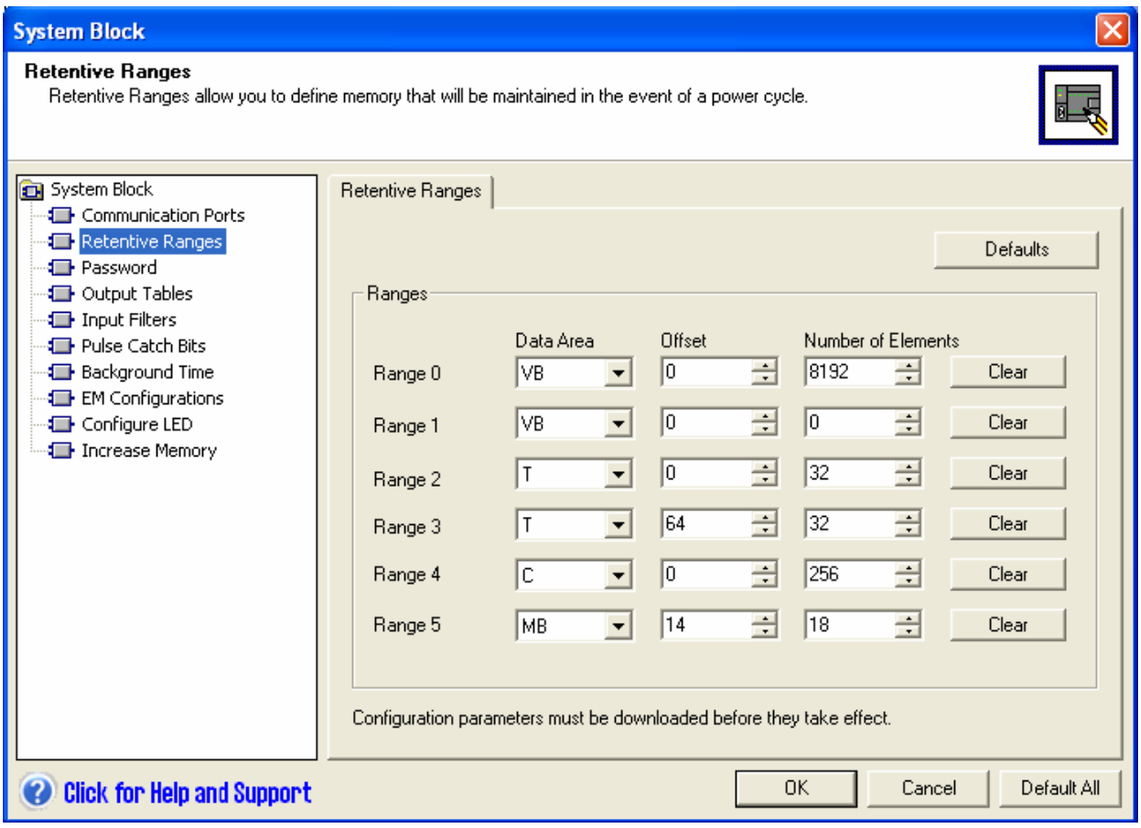
S7_200 có 3 mức (Level Password) : để bảo đảm bảo mật về bản quyền thông thường người sử nên chọn mức Password cao nhất. Số kí tự trong Password tối đa là 8 kí tự.

Trường hợp PLC đã cài Password thì người không có password,không thể upload chương trình từ PLC,nhưng ngược lại có thể Download chương trình mới xuống PLC bằng cách gõ **clearplc** khi phần mềm hỏi Password khi download,trường hợp khi ta gõ clearplc thì toàn bộ dữ liệu cũ sẽ hoàn toàn mất.

iv/Output table:

Ngõ ra của PLC cho phép ta chọn trạng thái **ON** hay **OFF** khi PLC chuyển từ trạng thái **Run** sang **Stop,** chế độ mặc định của phần mềm là tất cả trạng thái ngõ ra **OFF** khi chuyển trạng thái





v/ Input Filter:

S7_200 cho phép ta chọn thời gian lọc của các tín hiệu ngõ vào, thời gian lọc là thời gian mà ngõ vào phải không đổi trạng thái trong khoảng thời gian lọc đó thì PLC mới cho phép nhận trạng thái đó.

Thời gian lọc mặc định là: 6.4ms : Ngõ vào phải giữ **On** trong khoảng thời gian >= 6.4ms thì PLC mới hiểu ngõ vào đó lên 1.

vi/ Pulse catch Bits:

PLC cho phép người sử dụng chọn ngõ vào có thể bắt những tín hiệu nhanh khi chu kì quét chưa kịp quét. Tín hiệu đó sẽ được giữ cho tới khi chu kì quét được thực hiện.

vii/ Configure Led:

PLC cho phép ta định dạng trạng thái của Led System fault, hoặc led diagnostics,trạng thái Led này cho phép ta định dạng màu cam,đỏ,....khi chương trình gặp sự cố.

6/ Cách giao tiếp giữa máy tính và PLC:

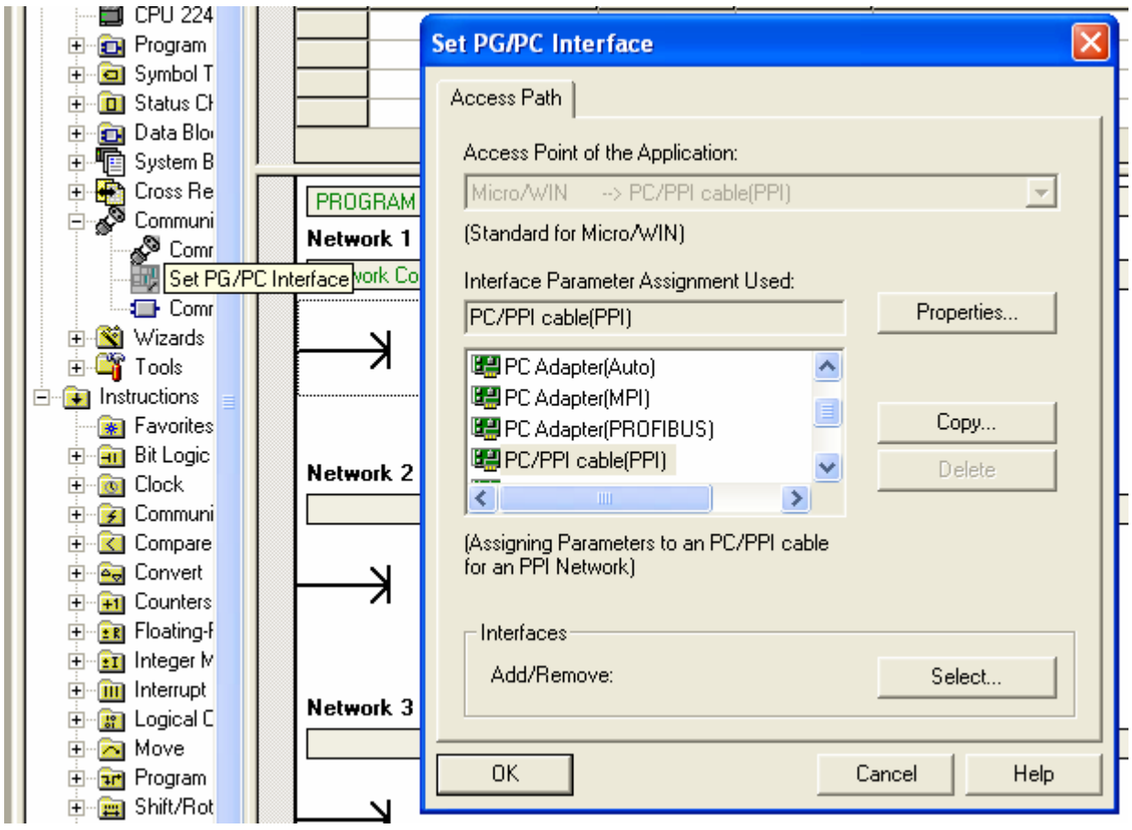
Để có thể giao tiếp giữa máy tính và PLC cho thực hiện việc Download hoặc Upload cho PLC, ta phải thực hiện các bước sau:

Chọn cổng giao tiếp:

Trường hợp cáp giao tiếp là cáp USB thì cổng giao tiếp phải chọn USB

Trường hợp cáp giao tiếp là cáp COM thì phải chọn đúng cổng giao tiếp của máy tính.

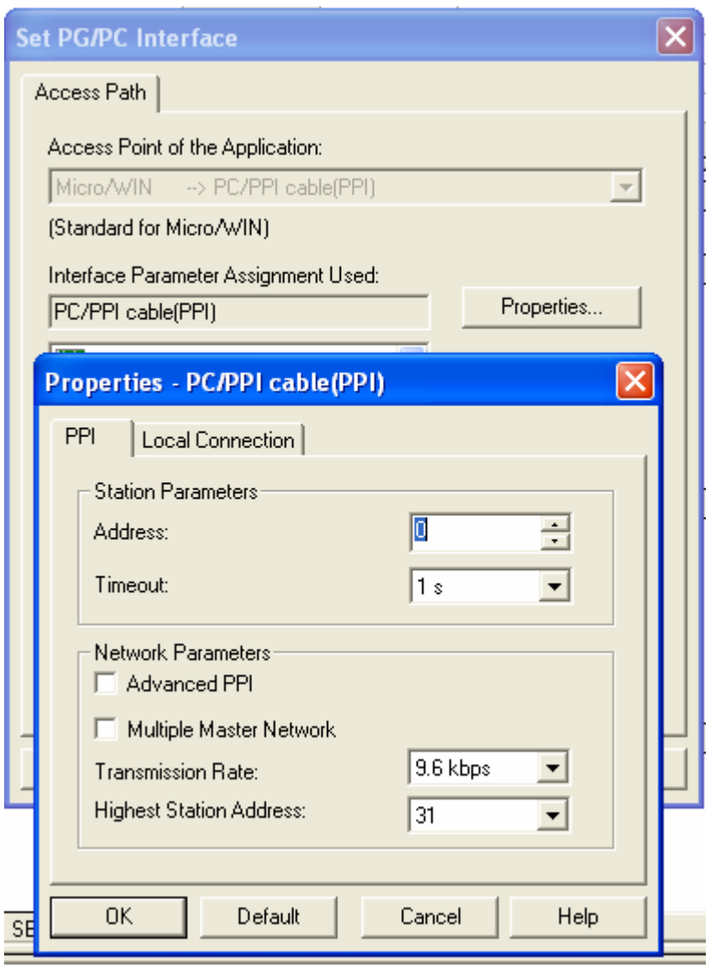
Để có thể chọn cổng giao tiếp,vào mục Communication,chọn Set PG/PC Interface



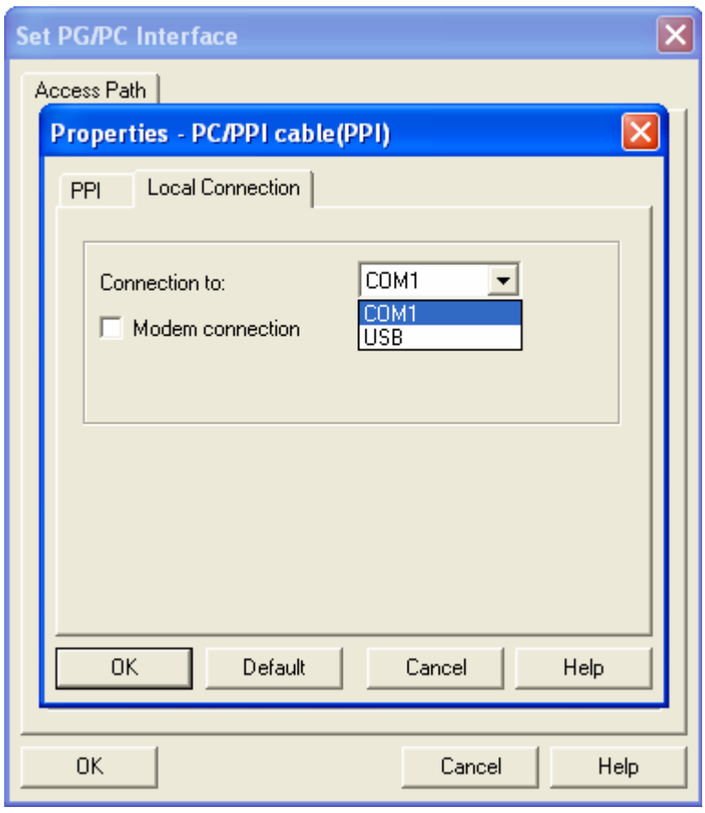
Sau đó chọn Properties của PC/PPI cable (PPI)

Trong Tab PPI: chọn đúng tốc độ Bauds ở phần Transmission Rate:

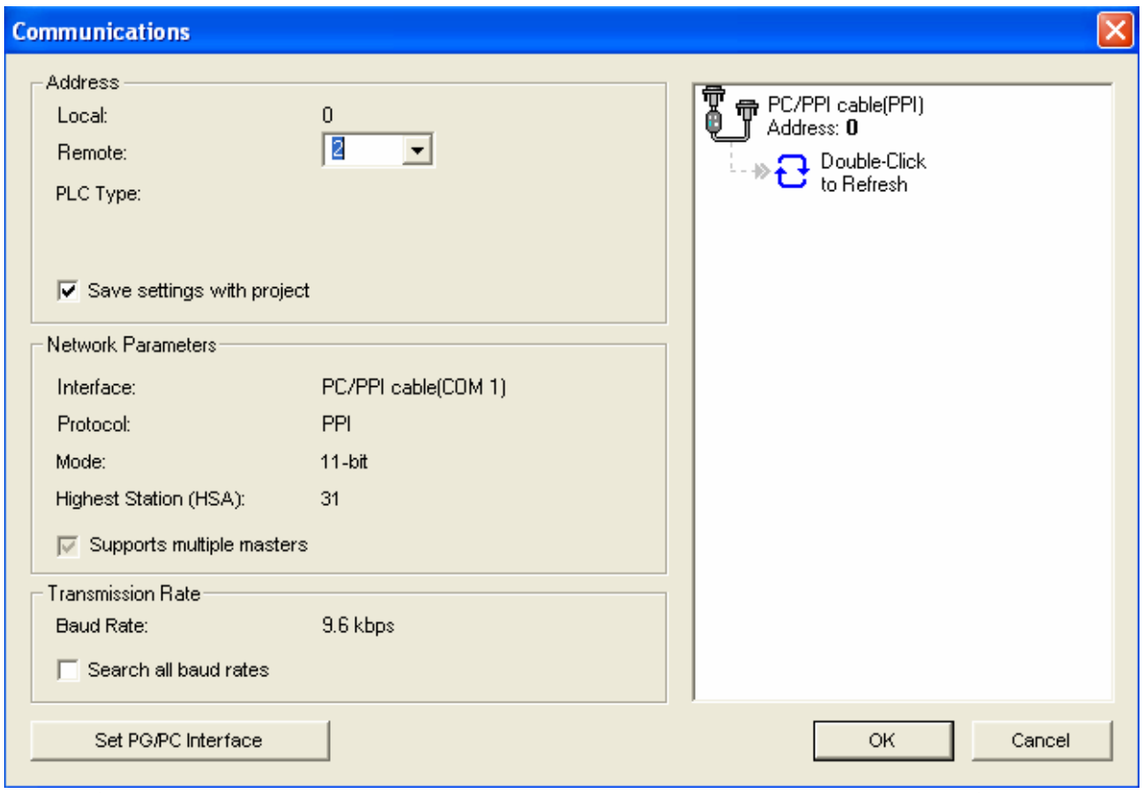
Tốc độ để mặc định là 9600, tốc độ Baud mặc định ở cáp cũng là 9600 (tốc độ Baud này chỉ áp dụng đối cáp cổng COM),trên cáp COM,cho phép ta chọn nhiều mức tốc độ Baud khác nhau.



Trong phần Local Connection: cho phép ta chọn cổng COM

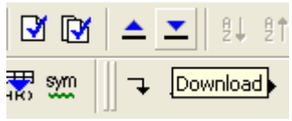


Sau khi chọn cổng COM,bước kế tiếp là phải chọn địa chỉ PLC,thông thường địa chỉ mặc định của PLC là 2,nếu địa chỉ PLC khác 2 thì ta phải chọn địa chỉ đúng trước khi thực hiện việc Communication.
Trường hợp nếu không biết địa chỉ PLC ta có thể thực hiện như sau:



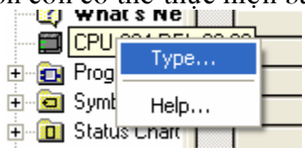
Vào phần Communication,chọn Search all baud rate sau đó double click vào phần " double click to refresh,khi đó chương trình sẽ tự nhận địa chỉ PLC .

Sau khi chọn xong cổng Com cũng như địa chỉ PLC, ta thực hiện việc Download cũng như Upload



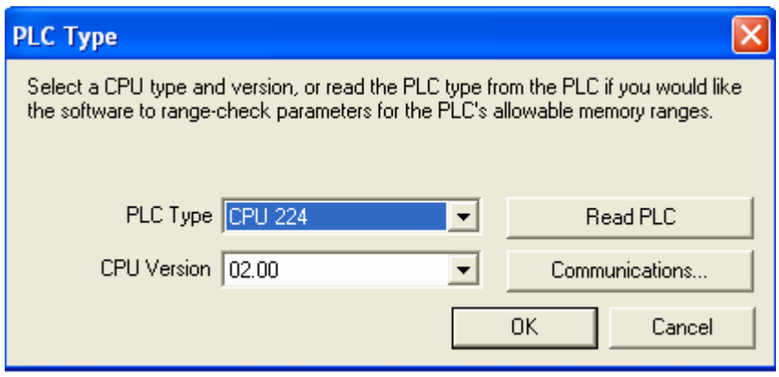
Chọn mũi tên xuống cho việc Download,mũi tên lên cho việc upload

Ngoài ra việc Communication còn có thể thực hiện bằng cách:



Vào CPU click chuột phải,chọn Type

Chọn Read PLC,nếu liên thông được thì chương trình có thể đọc được loại PLC,còn không thì nó sẽ báo,ta phải chọn lại cổng COM cũng như địa chỉ PLC trong phần Communications.


PLC Type
Select a CPU type and version, or read the PLC type from the PLC if you would like the software to range-check parameters for the PLC's allowable memory ranges.
PLC Type
CPU 224
Read PLC
CPU Version
02.00
Communications...
OK
Cancel

II/ Các vùng nhớ S7_200

1/ ***Trong S7_200 có các vùng nhớ sau:***

I: Input, các ngõ vào số.

Q: Output, các ngõ ra số.

M: Internal Memory, vùng nhớ nội.

V: Variable Memory, vùng nhớ biến

AIW: Analog Input, ngõ vào analog.

AQW: Analog Output, ngõ ra analog.

T: Timer.

C: Counter.

AC: con trỏ địa chỉ.

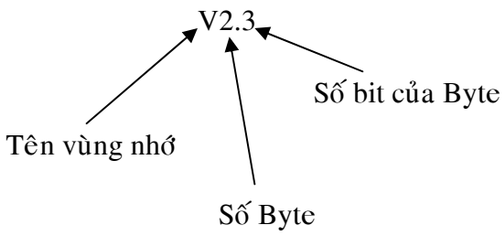
Giới hạn vùng nhớ trong S7_200:

CPU		221	222	224	226
Byte	VB	0-2047	0-2047	0-5119(V1.22) 0 – 8191(V 2.00) 0-10239(XP)	0-5119(V1.23) 0-10239(V2.00)
	IB	0-15	0-15	0-15	0-15
	QB	0-15	0-15	0-15	0-15
	MB	0-31	0-31	0-31	0-31
	SMB	0-179	0-299	0-549	0-549
	AC	0-3	0-3	0-3	0-3

2/Định dạng dữ liệu:

* **Kiểu Bool:**

VD: Q0.0, I0.0, V2.3, M1.7....



Một biến kiểu Bool chỉ có 2 giá trị là 0 hoặc 1 (**True** hoặc **False**).

Đối với ngõ IN :

Trạng thái mức 0 : Mức áp bé hơn 15VDC, hoặc ở trạng thái ngõ vào tổng trở cao

Trạng thái mức 1 :24V (15V – 30VDC) : so với 0VDC cấp cho chân **M** ở ngõ Input

Đối với ngõ OUT:

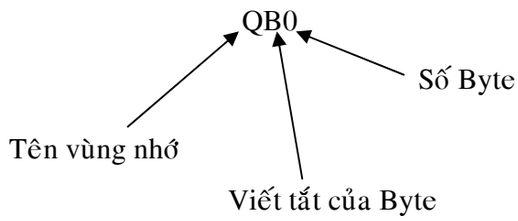
Trạng thái mức 0 : Hở tiếp điểm hoặc ngõ ra tổng trở cao (High Z)

Trạng thái mức 1: xuất 24V hoặc đóng tiếp điểm

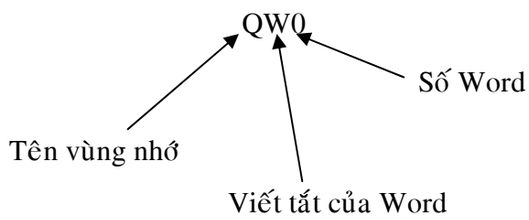
* **Kiểu Byte:**

1 Byte = 8 Bit. Suy ra, giá trị 1 Byte trong khỏang: 0 -(2^8-1) hay 0-255

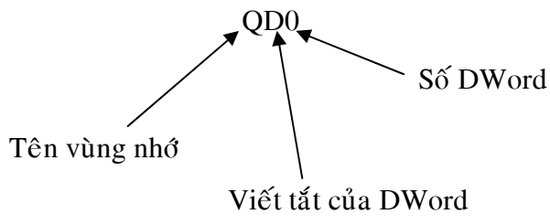
VD: QB0, MB3, VB10, SMB2...


QB0
Số Byte
Tên vùng nhớ
Viết tắt của Byte

* **Kiểu Word:**
1 Word = 2 Byte = 16 Bit. Suy ra, giá trị 1 Word trong khỏang: 0 -(2^{16}-1)
VD: IW0, QW0, MW3, VW10,...



QW0 = QB0+QB1, Trong đó, QB0 là byte cao, QB1 là Byte thấp.
* **Kiểu DWord:**
1 DWord = 2 Word = 4 Byte = 32 Bit. Suy ra, giá trị 1 Word trong khỏang: 0 -(2^{32}-1)
VD: ID0, QD0, MD3, VD10, ...

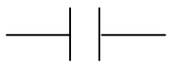


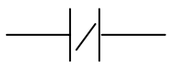
MD0=MW0+MW2=MB0+MB1+MB2+MB3, Trong đó, MB0 là byte cao nhất,MB3 là Byte thấp nhất.
* **Kiểu Int**: Số nguyên
Một biến kiểu Int tương đương một Word, nghĩa là dung lượng của 1 biến kiểu Int cũng gồm 16 bit. Tuy nhiên, biến kiểu Int và Word cũng có những điểm khác nhau như sau:
i/ Biến kiểu Word là biến ko dấu, biến kiểu Int có dấu(bit trọng số cao nhất là bit dấu).
ii/ Giá trị 1 Word: 0 -(2^{16}-1), giá trị một Int (-2^{15}) – (2^{15}-1) (do có 1 bit dấu)
iii/ Định dạng một biến kiểu Word phải có 16# đứng đầu, còn Int thì không.
VD: 16#1234, 16#ABCD: một Word
1,5,100,250...: một Int
* **Kiểu DInt**: Số nguyên
Một biến kiểu DInt tương đương một DWord, nghĩa là dung lượng của 1 biến kiểu Int cũng gồm 32 bit. Tuy nhiên, biến kiểu DInt và DWord cũng có những điểm khác nhau như sau:
i/ Biến kiểu DWord là biến ko dấu, biến kiểu DInt có dấu(bit trọng số cao nhất là bit dấu).
ii/ Giá trị 1 DWord: 0 -(2^{32}-1), giá trị một Int (-2^{31}) – (2^{31}-1) (do có 1 bit dấu)
iii/ Định dạng một biến kiểu DWord phải có 16# đứng đầu, còn DInt thì không.
VD: 16#12345678, 16#ABCDABCD: một Word
1,5,100,250...: một Dint
* **Kiểu Real**: Số thực.
Một biến kiểu Real 32 bit, nghĩa là vùng nhớ cũng là Dword.

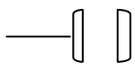
Định dạng: phải có dấu “.” Thập phân.
VD: 1.5, 2.3, 0.09, 1.0, 100.2 ...

III/ Tập Lệnh Trong S7_200:

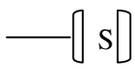
1/ Lệnh về bit:


: tiếp điểm thường hở.

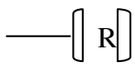

: tiếp điểm thường đóng.


: Cuộn coil, ngõ ra.

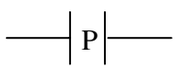
: đảo trạng thái bit.



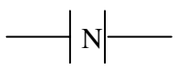
: Set bit



: Reset bit

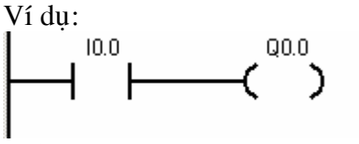


: Vi phân cạnh lên

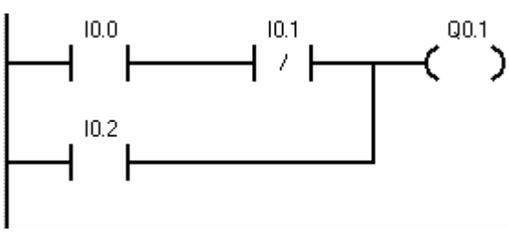


: Vi phân cạnh xuống.

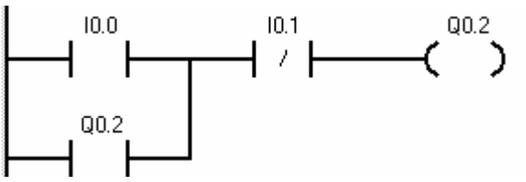
Ví dụ:



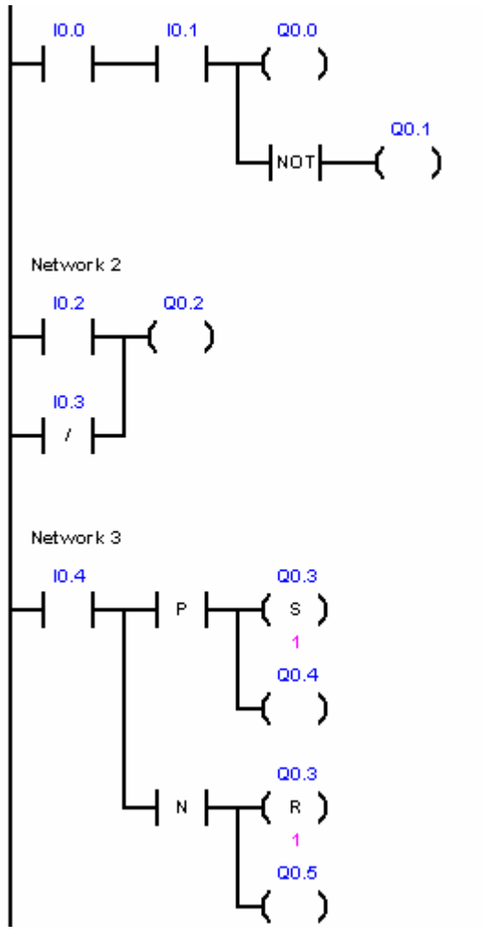
Nếu ngõ vào I0.0 =1 : Ngõ ra Q0.0 =1

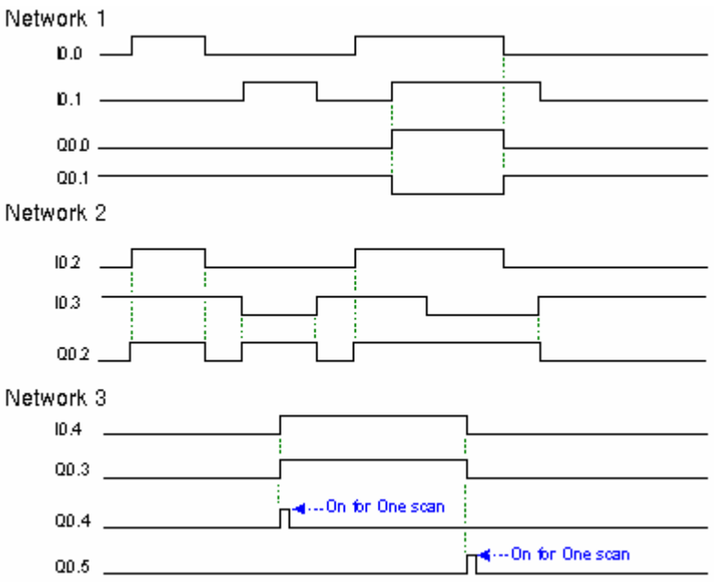


Ngõ ra Q0.1 =1 Nếu Ngõ vào I0.0=1 và I0.1=0 hoặc ngõ vào I0.2 =1



Mạch tự giữ: Ngõ vào I0.0=1 trong 1 chu kì Q0.2 =1 và tự giữ,Q0.2 =0 khi I0.1 =1 trong 1 chu kì (Ngõ vào I0.0 : Start ; I0.1 : Stop)



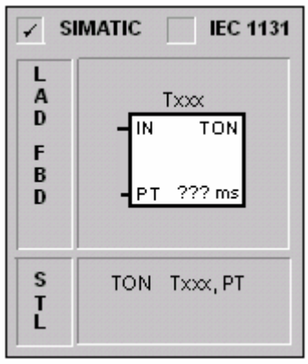


Ý nghĩa Các Network tương ứng.

2/ Timer: TON, TOF, TONR.

TON: Delay On.
TOF: Delay Off.
TONR: Delay On có nhớ

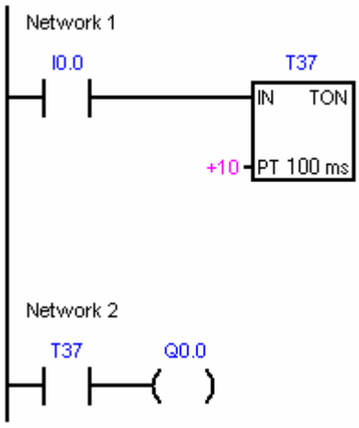
a/TON:



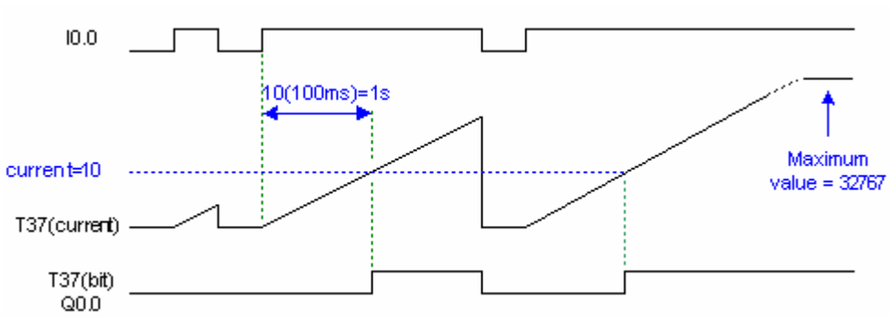
IN: BOOL: cho phép Timer.
PT: Int: giá trị đặt cho timer(VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *LD, *AC)
Txxx: số hiệu Timer.

Trong S7_200 có 256 Timer, ký hiệu từ T0-T255
Các số hiệu Timer trong S7_200 như sau:

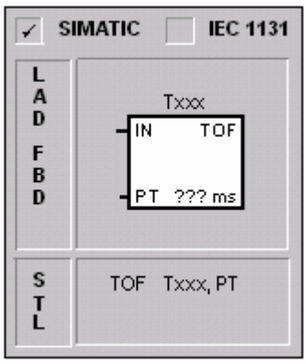
TONR	1 ms	32.767 s	T0, T64
	10 ms	327.67 s	T1-T4, T65-T68
	100 ms	3276.7 s	T5-T31, T69-T95
TON, TOF	1 ms	32.767 s	T32, T96
	10 ms	327.67 s	T33-T36, T97-T100
	100 ms	3276.7 s	T37-T63, T101-T255



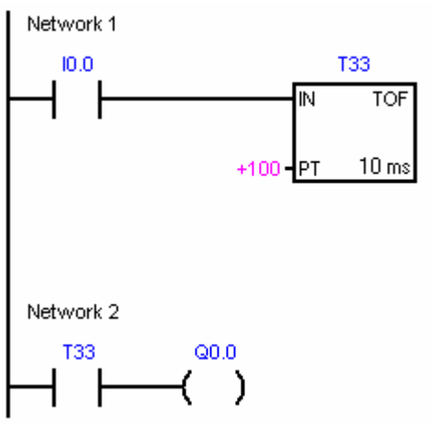
Khi ngõ vào I0.0 =1 Timer T37 được kích , Nếu sau 10x100ms =1s I0.0 vẫn giữ trạng thái thì Bit T37 sẽ lên 1 (Khi đó Q0.0 lên 1).
Nếu I0.0 =1 không đủ thời gian 1S thì bit T37 sẽ không lên 1.

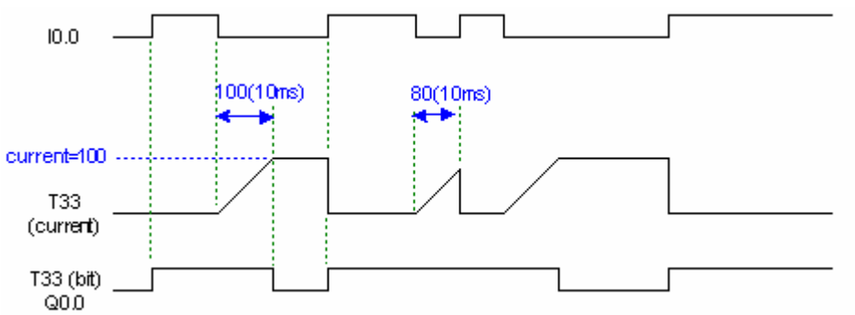


b/ TOF:



IN: BOOL: cho phép Timer.
PT: Int: giá trị đặt cho timer(VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *LD, *AC)
Txxx: số hiệu Timer.

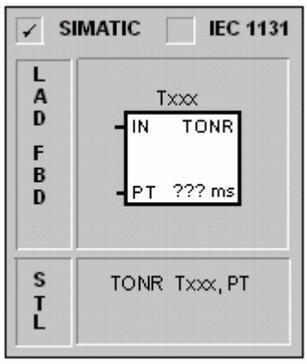




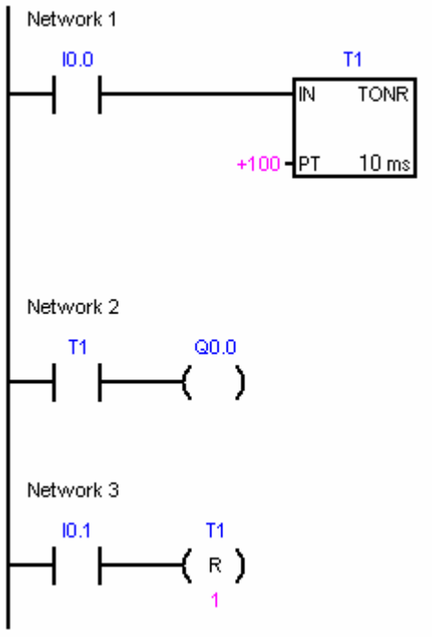
Khi Ngõ vào I0.0 = 1 thì bit T33 lên 1 (Ngõ ra Q0.0 lên 1)

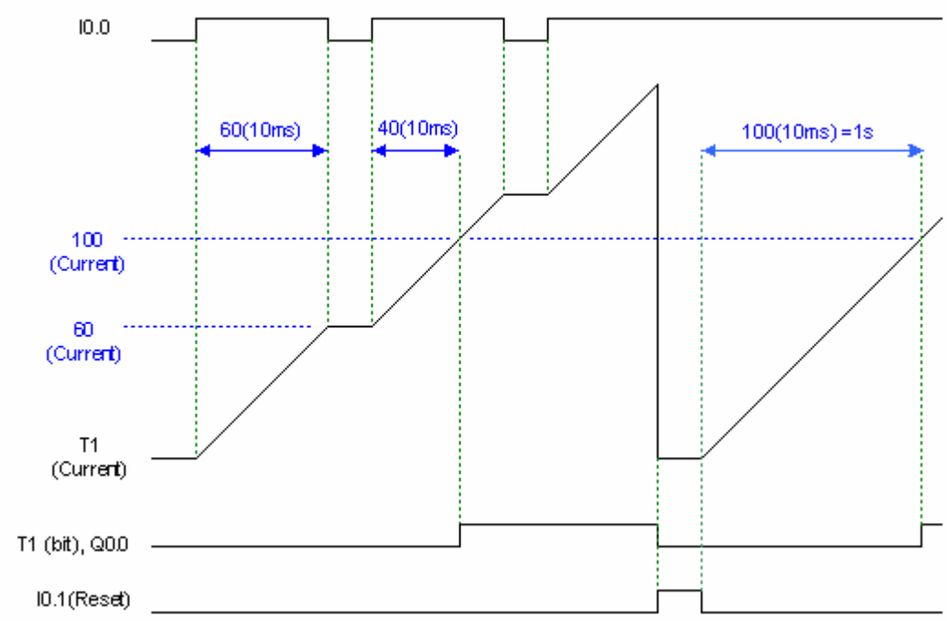
Khi I0.0 xuống 0,thời gian Timer bắt đầu tính ,đủ thời gian 1s = 100x10ms thì bit T33 sẽ tắt (Q0.0 tắt)
Nếu I0.0 xuống 0 trong khoảng thời gian chưa đủ 1s đã lên 1 lại thì bit T33 vẫn giữ nguyên trạng thái

c/**TONR:**



IN: BOOL: cho phép Timer.
PT: Int: giá trị đặt cho timer(VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *LD, *AC)
Txxx: số hiệu Timer.

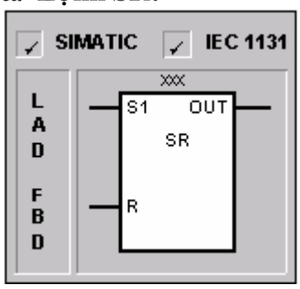




Ngõ vào I0.0 có tác dụng kích thời gian cho Timer,khi ngõ I0.0 =1 thời gian Timer được tính,khi I0.0=0 thời gian không bị Reset về 0.Khi đủ thời gian thì Bit T1 sẽ lên 1.
Thời gian Timer chỉ bị Reset khi có tín hiệu Reset Timer (tín hiệu từ ngõ I0.1)

4/ Lệnh RS và SR:

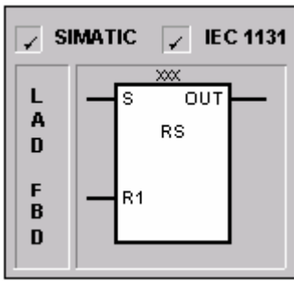
a/ Lệnh SR:



Ngõ vào: S1,R : Kiểu Bool (I,Q,M,T,C,V,S,SM,L)
Ngõ ra : OUT : Kiểu Bool (I,Q,M,T,C,V,S,SM,L)

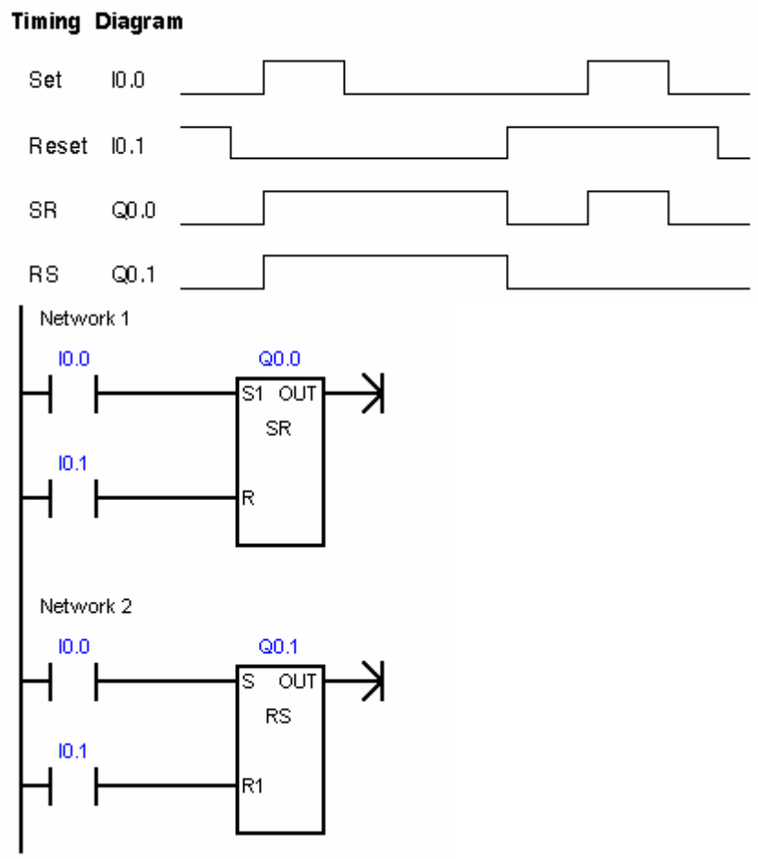
Cấu trúc:	**S1**	**R**	**OUT**
	0	**0**	**Giữ nguyên trạng thái**
	0	**1**	**0**
	1	**0**	**1**
	1	**1**	**1**

a/ Lệnh RS :



Ngõ vào S,R1 : Kiểu Bool (I,Q,M,T,C,V,S,SM,L)
Ngõ ra : OUT : Kiểu Bool (I,Q,M,T,C,V,S,SM,L)

Cấu trúc:	**S**	**R1**	**OUT**
	0	**0**	**Giữ nguyên trạng thái**
	0	**1**	**0**
	1	**0**	**1**
	1	**1**	**0**



BÀI TẬP:

1/ Sử dụng phương pháp mạch tự giữ để khởi động động cơ theo phương pháp sao /tam giác.

2/ Sử dụng các tập lệnh về Bit để thực hiện việc khởi động tuần tự 4 động cơ theo thứ tự sau:

Nhấn Start1 : động cơ 1 khởi động ,Stop1 động cơ 1 tắt

Khi động cơ 1 không đủ tải,nhấn Start2 động cơ 2 sẽ hoạt động,nhấn Stop2 động cơ 2 sẽ tắt (khi đã dư tải...)

Tương tự cho động cơ 3 và 4 (sẽ được khởi động khi tải tương ứng không đủ)

Trong quá trình hoạt động gặp sự cố ta có thể nhấn nút Emergency để dừng toàn bộ hệ thống.

3/ Phát hiện chiều di chuyển của vật: Để phát hiện chiều di chuyển của vật, ta phải sử dụng 2 Sensor kế tiếp nhau. Sensor1 và Sensor2 :

Trường hợp vật di chuyển theo chiều thuận : Sensor1 tác động rồi đến Sensor2.

Chiều ngược thì Sensor tác động theo thứ tự ngược lại .

Gợi ý : Bài tập có thể sử dụng lệnh P,N,Set,Reset Bit

Hay có thể sử dụng lệnh RS hay SR

4/ Điều khiển Đèn xanh đỏ tại ngã tư:

Xanh A : Trong 15s

Vàng A : Trong 3s

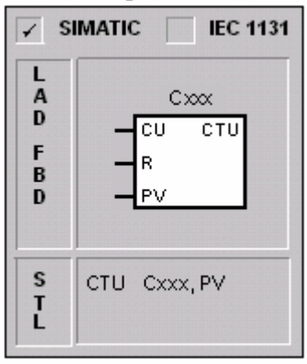
Xanh B : Trong 25s

Vàng B : Trong 4s

Ta có thể mở rộng bài toán cho điều khiển có thêm đường dành cho người đi bộ.

5/ Counter:

Counter Up(đếm lên):



Cxxx: số hiệu counter (0-255)

CU: kích đếm lên	Bool
R:reset	Bool
PV:giá trị đặt cho counter	INT

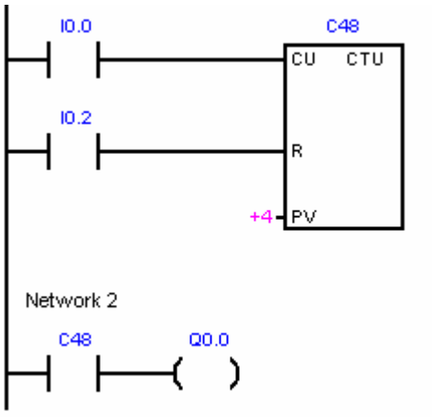
PV: VW, IW, QW, MW, SMW, LW, AIW, AC, T, C, Constant, *VD, *AC, *LD, SW

Mô tả:

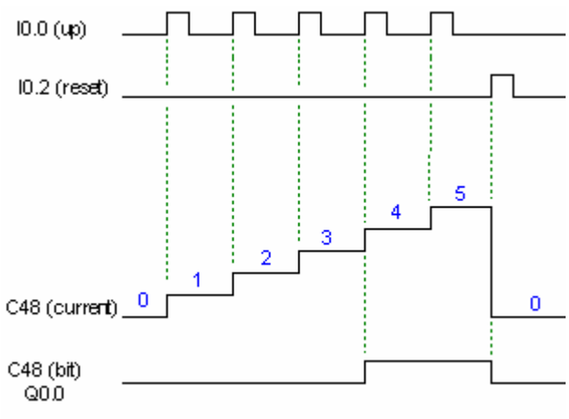
Mỗi lần có một sườn cạnh lên ở chân CU, giá trị bộ đếm (1 Word) được tăng lên 1.Khi giá trị hiện tại lớn hơn hoặc bằng giá trị đặt PV(Preset value), ngõ ra sẽ được bật lên ON.

Khi chân Reset được kích (sườn lên) giá trị hiện tại bộ đếm và ngõ ra được trả về 0.

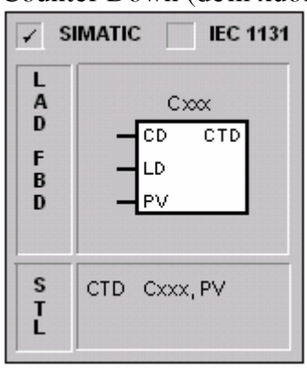
Bộ đếm ngưng đếm khi giá trị bộ đếm đạt giá trị tối đa là 32767 ($2^{16} - 1$).



giản đồ xung:


I0.0 (up)
I0.2 (reset)
C48 (current)
0
1
2
3
4
5
0
C48 (bit)
Q0.0

Counter Down (đếm xuống):



Cxxx: số hiệu counter (0-255)

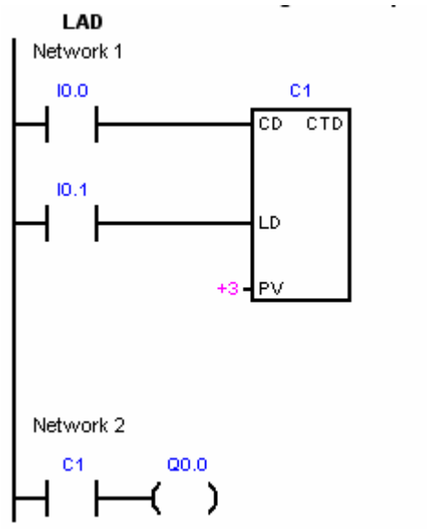
CD: kích đếm xuống	Bool
LD:Load	Bool
PV:giá trị đặt cho counter	INT

PV: VW, IW, QW, MW, SMW, LW, AIW, AC, T, C, Constant, *VD, *AC, *LD, SW

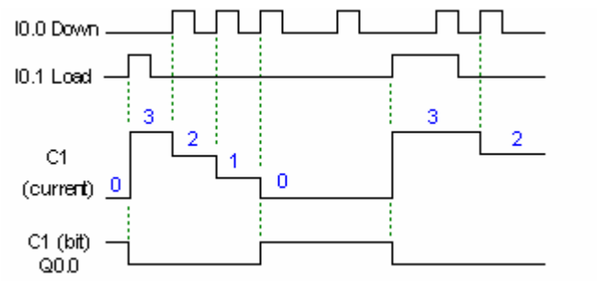
Mô tả:

Khi chân LD được kích (sườn lên) giá trị PV được nạp cho bộ đếm.

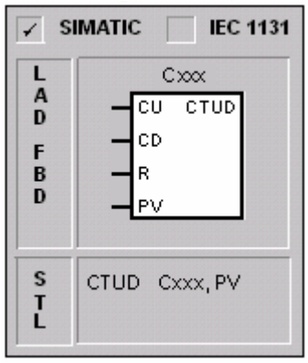
Mỗi lần có một sườn cạnh lên ở chân CD, giá trị bộ đếm (1 Word) được giảm xuống 1.Khi giá trị hiện tại của bộ đếm bằng 0, ngõ ra sẽ được bật lên ON và bộ đếm sẽ ngưng đếm.



Giản đồ xung:



Counter Up/Down (đếm lên/xuống):



Cxxx: số hiệu counter (0-255)

CU: kích đếm lên	Bool
CD: kích đếm xuống	Bool
R:reset	Bool
PV:giá trị đặt cho counter	INT

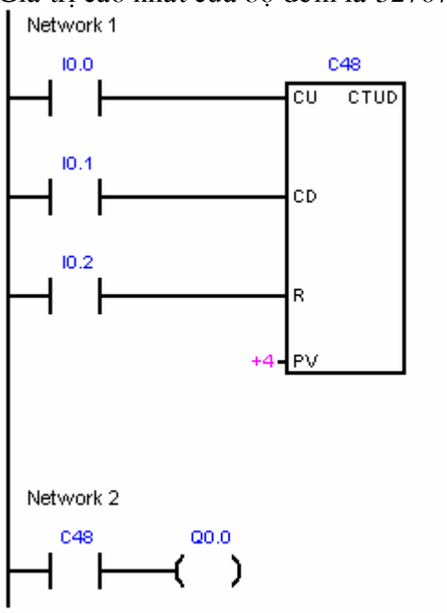
PV: VW, IW, QW, MW, SMW, LW, AIW, AC, T, C, Constant, *VD, *AC, *LD, SW

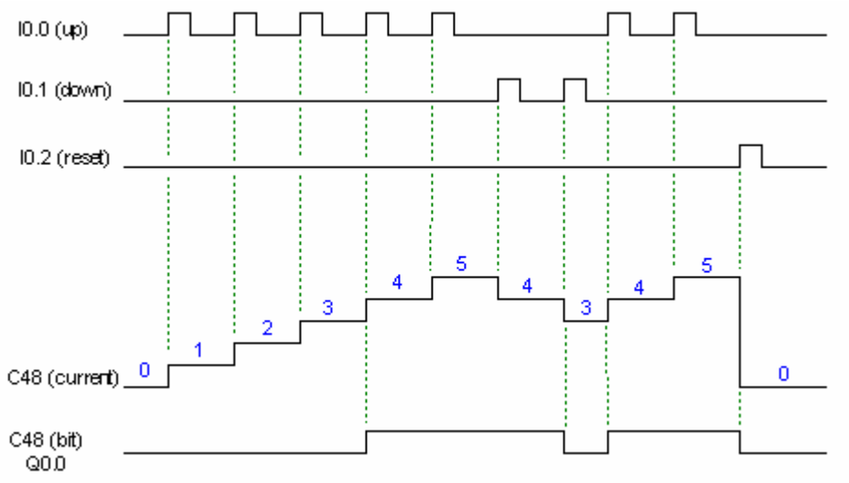
Mô tả:

Mỗi lần có một sườn cạnh lên ở chân CU, giá trị bộ đếm (1 Word) được tăng lên 1. Mỗi lần có một sườn cạnh lên ở chân CD, giá trị bộ đếm được giảm xuống 1.Khi giá trị hiện tại lớn hơn hoặc bằng giá trị đặt PV(Preset value), ngõ ra sẽ được bật lên ON.

Khi chân R được kích (sườn lên) giá trị bộ đếm và ngõ Out được trả về 0.

Giá trị cao nhất của bộ đếm là 32767 và thấp nhất là –32768. Khi giá trị bộ đếm đạt ngưỡng





6/Lệnh Move:

Trong S7_200 có các hàm Move sau:

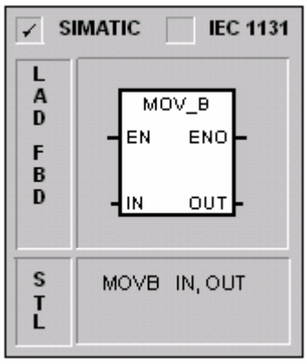
Move_B: Di chuyển các giá trị cho nhau trong giới hạn 1 Byte

Move_W: Di chuyển các giá trị nguyên cho nhau trong giới hạn 1 Word

Move_DW: Di chuyển các giá trị nguyên cho nhau trong giới hạn 1 DWord

Move_R: Di chuyển các giá trị thực cho nhau trong giới hạn 1 Dint

a/* Move_B*:



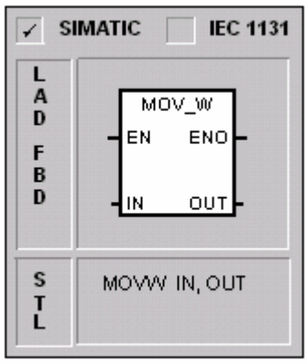
EN: ngõ vào cho phép

IN Ngõ vào: VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC

OUT: Ngõ ra VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

Khi có tín hiệu ở ngõ cho phép,lệnh sẽ chuyển nội dung của ô nhớ trong (IN) sang ô nhớ trong OUT

a/* Move_W*:



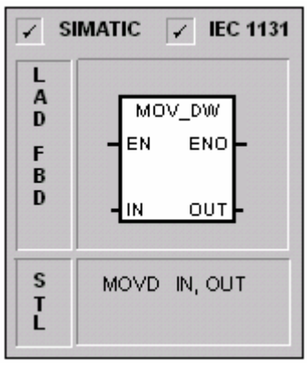
EN: ngõ vào cho phép

IN Ngõ vào: VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, Constant, AC, *VD, *AC, *LD

OUT Ngõ ra: VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD

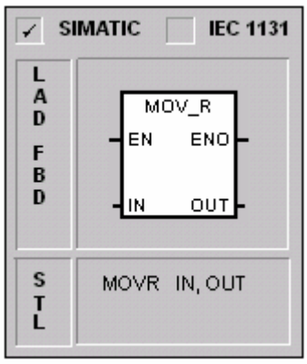
Khi có tín hiệu ở ngõ cho phép,lệnh sẽ chuyển nội dung của ô nhớ trong (IN) sang ô nhớ trong OUT

a/* Move_DW*:



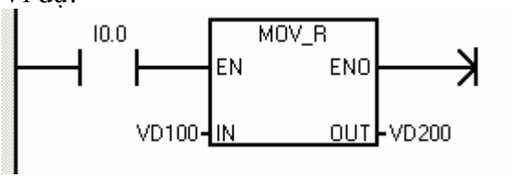
EN: ngõ vào cho phép
IN Ngõ vào: VD, ID, QD, MD, SD, SMD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, &SMB, &AIW, &AQW AC, Constant, *VD, *LD, *AC
OUT Ngõ ra: VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC
Khi có tín hiệu ở ngõ cho phép,lệnh sẽ chuyển nội dung của ô nhớ trong (IN) sang ô nhớ trong OUT

***a/* Move_R:**



EN: ngõ vào cho phép
IN Ngõ vào: VD, ID, QD, MD, SD, SMD, LD, AC, Constant, *VD, *LD, *AC
OUT Ngõ ra: VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC
Khi có tín hiệu ở ngõ cho phép,lệnh sẽ chuyển nội dung của ô nhớ trong (IN) sang ô nhớ trong OUT

Các tín hiệu ngõ vào cũng như ngõ ra của các lệnh Move phải được chọn đúng loại theo đã định dạng như vùng Dword đối với Move_R và Move_DW...
Nếu chọn sai định dạng thì chương trình biên dịch sẽ bị sai.
Ví dụ:



Khi I0.0 lên 1 thì chương trình sẽ chuyển nội dung ô nhớ trong VD100 sang ô nhớ VD200

7/Lệnh chuyển khối:

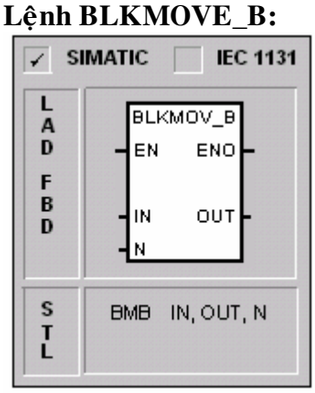
S7_200 có các lệnh chuyển khối sau:

BLKMOVE_B: chuyển khối Byte

BLKMOVE_W: chuyển khối Word

BLKMOVE_D: chuyển khối Double Word

Lệnh BLKMOVE_B:

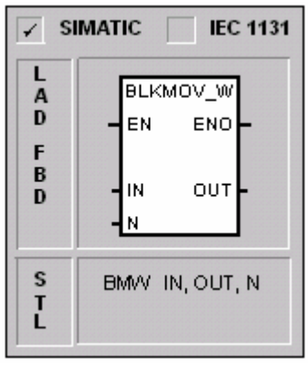


EN: ngõ vào cho phép
IN: vị trí Byte bắt đầu ngõ vào
N: số byte cần Move
OUT: vị trí Byte bắt đầu ngõ ra

IN	VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD	BYTE
N	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD	BYTE

Khi có tín hiệu ở ngõ vào (EN) : chương trình sẽ chuyển nội dung của N Byte (có vị trí Byte bắt đầu ở (IN) sang N Byte có vị trí bắt đầu ở OUT.

Lệnh BLKMOVE_W:

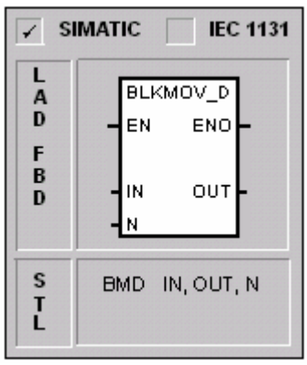


EN: ngõ vào cho phép
IN: vị trí Word bắt đầu ngõ vào
N: số byte cần Move
OUT: vị trí Word bắt đầu ngõ ra

IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, *VD, *LD, *AC	WORD
N	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC	BYTE
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, *VD, *LD, *AC	WORD

Khi có tín hiệu ở ngõ vào (EN) : chương trình sẽ chuyển nội dung của N Word (có vị trí Word bắt đầu ở (IN) sang N Word có vị trí bắt đầu ở OUT.

Lệnh BLKMOVE_W:

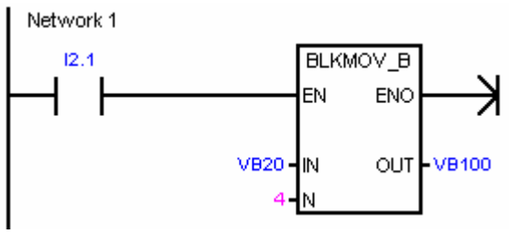


EN: ngõ vào cho phép
IN: vị trí DWord bắt đầu ngõ vào
N: số byte cần Move
OUT: vị trí DWord bắt đầu ngõ ra

IN, OUT	VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD	DWORD
N	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD	BYTE

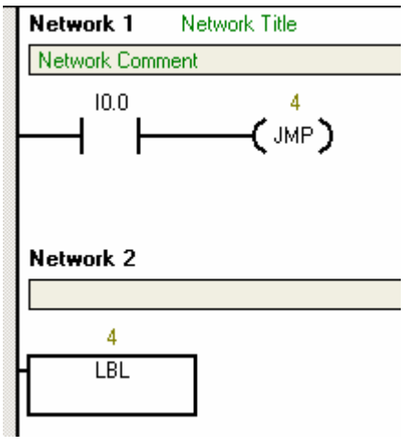
Khi có tín hiệu ở ngõ vào (EN) : chương trình sẽ chuyển nội dung của N DWord (có vị trí DWord bắt đầu ở (IN) sang N DWord có vị trí bắt đầu ở OUT

VD:



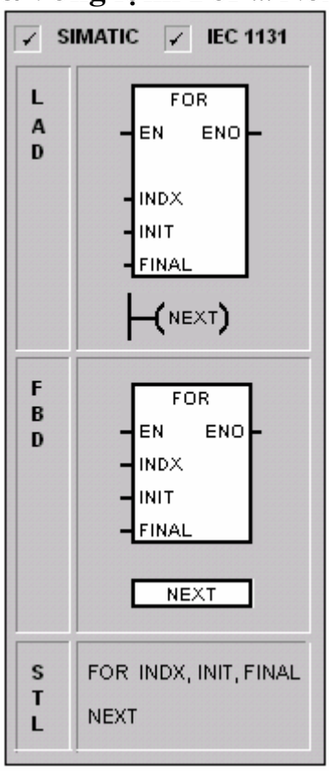
Khi I2.1 ON:
VB20 → VB100
VB21 → VB101
VB22 → VB102
VB23 → VB103

Lệnh Nhảy:



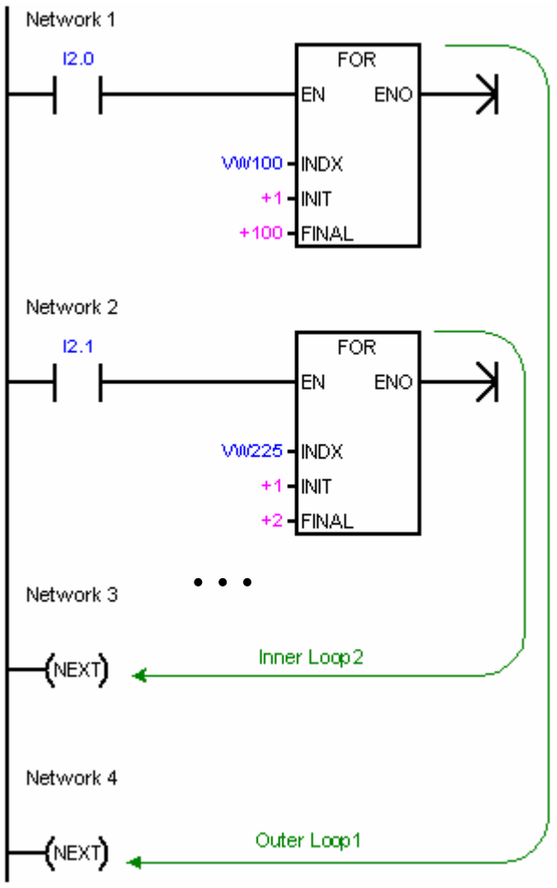
Khi I0.0 lên 1 chương trình sẽ thực hiện lệnh nhảy: Sẽ nhảy tới nhãn tương ứng,khi đó đoạn chương trình ở giữa lệnh nhảy và nhãn sẽ được bỏ qua ở chu kì đó.
Kí hiệu của nhãn nhảy phải là một số nguyên n.

8/Vòng lệnh For ... Next:



Vòng lệnh For ... Next thực thi đoạn chương trình giữa lệnh For và lệnh Next trong một số lần đặt trước.
INDEX: Lưu số vòng thực hiện.
INIT: giá trị bắt đầu.
FINAL: giá trị kết thúc.
Khi gặp lệnh FOR lần đầu tiên, giá trị INIT sẽ được đưa vào biến đếm INDX. Cứ mỗi lần thực hiện xong 1 chu trình For...Next, Biến đếm INDX sẽ tăng 1 đơn vị.đến khi INDX bằng giá trị cuối FINAL thì vòng lặp kết thúc. Chương trình kế tiếp vòng lệnh For ... Next được thực hiện.
VD: nếu INIT=1, FINAL=10, vòng lặp sẽ thực hiện 10 lần với các giá trị INDX 1,2,3 ... 10.

INDX	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	INT
INIT	VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, Constant, *VD, *LD, *AC	INT
FINAL	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, AIW, Constant, *VD, *LD, *AC	INT



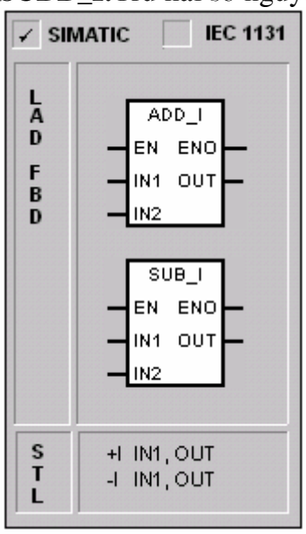
Khi I2.1 ON. Vòng lệnh Loop2 được thực hiện 2 lần.
Khi I2.0 ON. Vòng lệnh Loop1 được thực hiện 100 lần.
Nếu cả I2.0 và I2.1 ON thì chương trình trong vòng Loop2 sẽ thực thi 2x100=200 lần

9/Các hàm số học :

Lệnh cộng trừ:

ADD_I: Cộng hai số nguyên 16 Bit

SUBB_I: Trừ hai số nguyên 16 Bit



EN:Ngõ vào cho phép

IN1+IN2=OUT

IN1-IN2=OUT

IN1, IN2	VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, Constant, *VD, *LD, *AC	INT
OUT	VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *LD, *AC	INT

Khi ngõ vào cho phép lên 1 chương trình sẽ thực hiện việc cộng (hay trừ) 2 số nguyên 16 Bit ở IN1,IN2 tương ứng ,kết quả đưa vào OUT.

Tương tự, ta có:

ADD_DI: Cộng hai số nguyên 32 Bit

SUBB_DI: trừ hai số nguyên 32 Bit

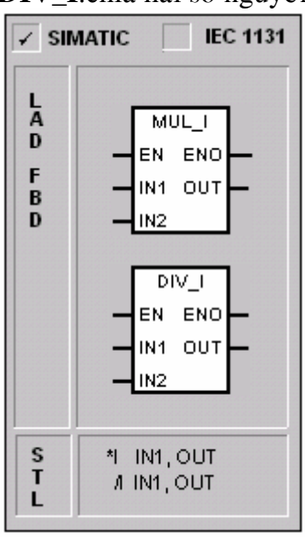
ADD_R: Cộng hai số thực

SUBB_R: trừ hai số thực

Lệnh nhân chia:

MUL_I: Nhân hai số nguyên 16 bit

DIV_I:chia hai số nguyên 16 bit



EN:Ngõ vào cho phép

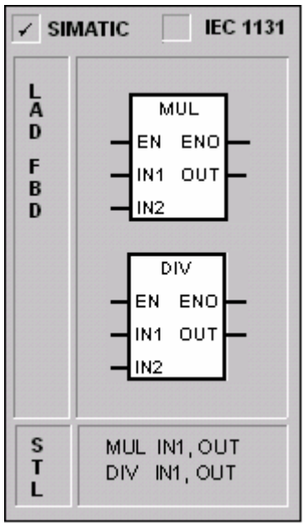
IN1*IN2=OUT

IN1/IN2=OUT

Nếu kết quả chia có dư thì phần dư sẽ được bỏ.

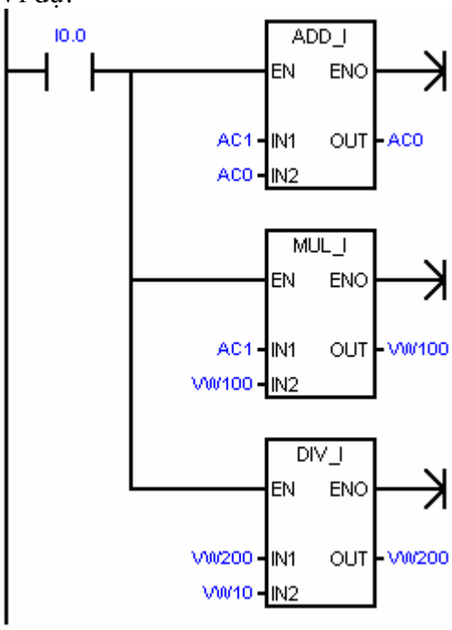
IN1, IN2	VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, Constant, *VD, *LD, *AC	INT
OUT	VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *LD, *AC	INT

Khi ngõ vào EN lên 1 ,chương trình sẽ thực hiện việc nhân (hay chia) 2 số nguyên 16 Bit,kết quả cất vào số nguyên 16 Bit
Trường hợp chia:do OUT là số nguyên 16 Bit,nên phần dư của phép chia sẽ bị bỏ.
Trường hợp nhân:nếu bị tràn bộ nhớ thì OUT sẽ chứa phần Byte thấp.
Tương tự, ta có:
MUL_DI: Nhân hai số nguyên 32 bit
DIV_DI:chia hai số nguyên 32 bit
MUL_R: Nhân hai số thực
DIV_R:chia hai số thực
Lệnh MUL,DIV : Tương tự lệnh nhân và chia,nhưng trong trường hợp này ngõ ra OUT là 32 Bit



Ta sẽ sử dụng lệnh MUL hay DIV khi không biết ngõ ra có bị tràn 16 Bit hay không.

Ví dụ:



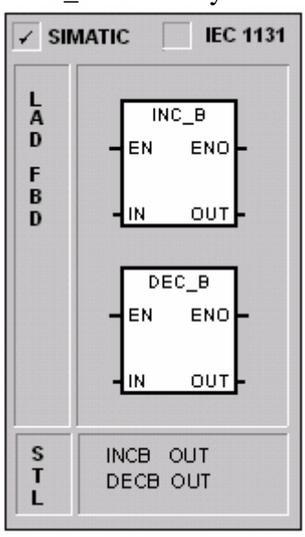
Khi I0.0 ON, chương trình thực thi:

IN1		IN2	OUT
40	+	60	100
AC1		AC0	AC0
40	*	20	800
AC1		VW102	VW100
4000	/	40	100
VW200		VW10	VW200

Lệnh tăng giảm:

INC_B: Tăng Byte

DEC_B: Giảm Byte



EN:Ngõ vào cho phép
IN1 + 1 = OUT

EN:Ngõ vào cho phép
IN1 - 1 = OUT

IN	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC	BYTE

Các hàm tương tự:

INC_W: Tăng Word

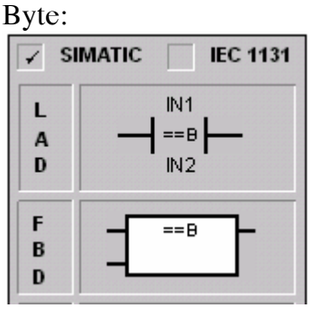
DEC_W: Giảm Word

INC_DW: Tăng DWord
DEC_DW: Giảm Dword
Ngoài ra còn một số hàm khác như: SQRT(khai căn), SIN.COS,TAN,LN, EXP...

10/Các lệnh so sánh:

a/So sánh bằng:

Byte:



Khi IN1=IN2 thì ngõ ra được tích cực.

Inputs	IB, QB, MB, SMB, VB, SB, LB, AC, Constant, *VD, *LD, *AC	BYTE
Output (FBD)	I, Q, M, SM, T, C, V, S, L, Power Flow	BOOL

Tương tự, ta có các hàm so sánh như sau:

<>: so sánh khác

>=: so sánh lớn hơn hoặc bằng

<=: so sánh nhỏ hơn hoặc bằng

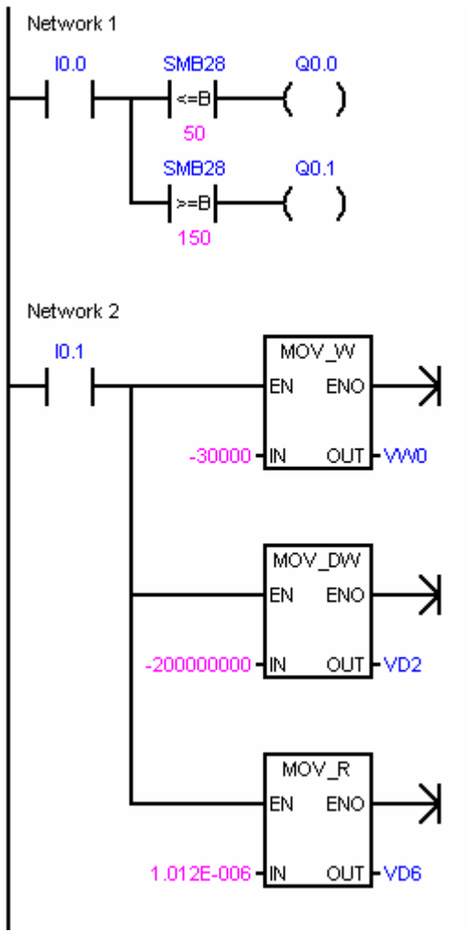
> : so sánh lớn

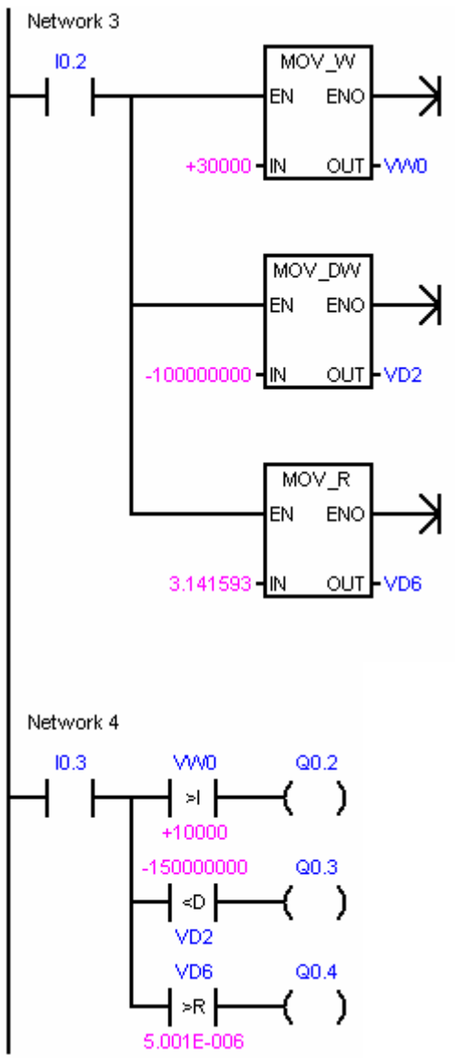
< : so sánh nhỏ

Tương tự các hàm so sánh cho Byte, ta cũng có các lệnh so sánh cho số Int, Dint, Real

Khi thực hiện các hàm so sánh thì IN1,IN2 phải được chọn đúng kiểu dữ liệu.

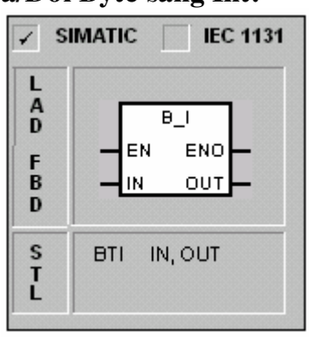
Ví dụ:


Network 1
I0.0
SMB28
<=B
50
Q0.0
SMB28
>=B
150
Q0.1
Network 2
I0.1
MOV_W
EN
ENO
-30000
IN
OUT
VW0
MOV_DW
EN
ENO
-200000000
IN
OUT
VD2
MOV_R
EN
ENO
1.012E-006
IN
OUT
VD6



11/Các hàm chuyển đổi:

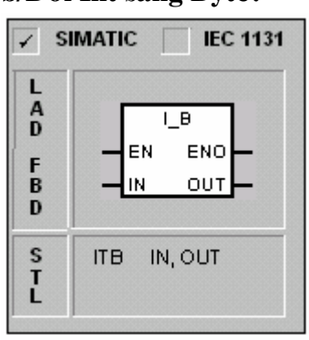
a/Đổi Byte sang Int:



EN: ngõ vào cho phép
Một số kiểu Byte ngõ vào được chuyển thành một số kiểu Int ở ngõ ra

IN	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *AC, *VD, *LD	BYTE
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	INT

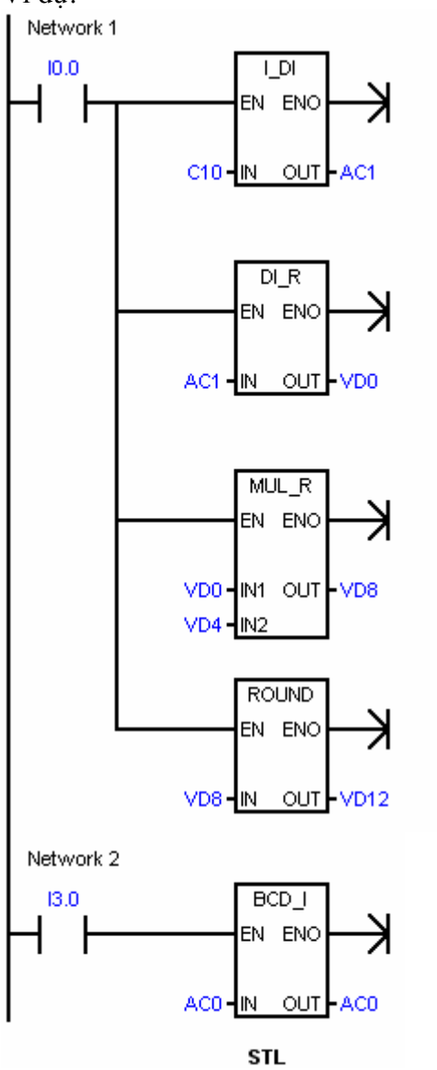
b/Đổi Int sang Byte:



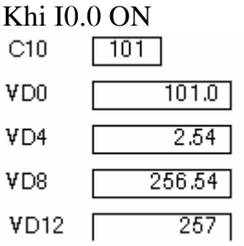
EN: ngõ vào cho phép
Một số kiểu Int ngõ vào (IN) được chuyển thành một số kiểu Byte ở ngõ ra (OUT)
Trong trường hợp ngõ vào nằm ngoài khoảng (0,255) thì ngõ ra không bị ảnh hưởng

IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, Constant, *VD, *LD, *AC	INT
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE

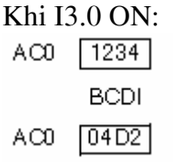
Tương tự, ta có các hàm chuyển đổi sau:
I_DI: đổi số nguyên 16 bit sang số nguyên 32 bit
DI_I: đổi số nguyên 32 bit sang số nguyên 16 bit
DI_R: đổi số nguyên 32 bit sang số thực
BCD_I: đổi số BCD 16 bit sang số nguyên 16 bit
I_BCD: đổi số nguyên 16 bit sang số BCD.
Trong trường hợp việc đổi từ số dung lượng nhỏ sang dung lương lớn hơn (như từ Byte sang Int,từ Int sang Dint..) thì chương trình luôn thực thi.
Còn trường hợp ngược lại: Nếu giá trị chuyển bị tràn ô nhớ thì chương trình sẽ không thực thi và Bit tràn SM1.1 sẽ bật lên 1.
Ví dụ: Khi chuyển số Int sang Byte,mà số Int lớn hơn 255 (8Bit),thì chương trình sẽ không thực thi và Bit SM1.1 bật lên 1.
Ví dụ:



Khi I0.0 ON



Khi I3.0 ON:



1234 dạng BCD là : 0001 0010 0011 0100

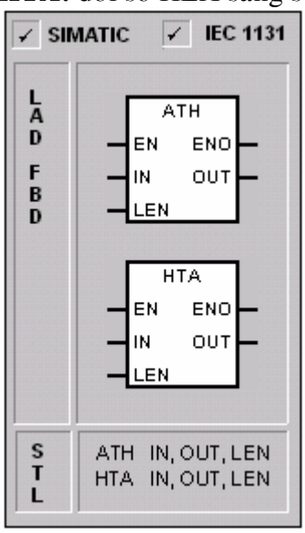
1234 dạng số Int: 0000 0100 1101 0010

Khi thực hiện việc chuyển đổi giữa số Int và số BCD hoặc ngược lại,thì giá trị lớn nhất của số BCD là 9999 trong khi giá trị lớn nhất số Int là 2^{16}-1.

Lệnh đổi số ASCII sang số HEX và ngược lại:

ATH: đổi số ASCII sang số HEX

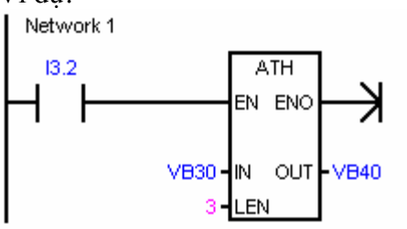
HTA: đổi số HEX sang số ASCII



EN: ngõ vào cho phép
IN: ngõ vào
LEN: chiều dài cần chuyển
OUT: ngõ ra

IN, OUT	VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD	BYTE
LEN	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *LD, *AC	BYTE

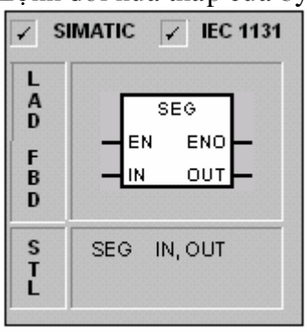
Ví dụ:



Khi I3.2 ON:

x: giá trị không bị thay đổi

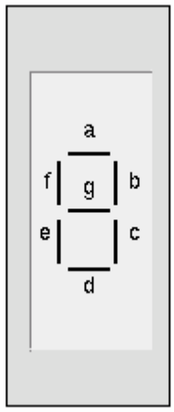
Lệnh đổi nửa thấp của byte sang mã LED 7 đoạn:



IN	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD	BYTE
OUT	VB, IB, QB, MB, SMB, LB, AC, *VD, *AC, SB, *LD	BYTE

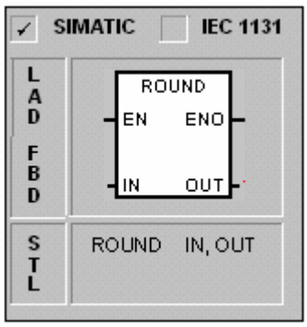
Hoạt động:

(IN) LSD	Segment Display	(OUT) - g f e d c b a
0	0	0 0 1 1 1 1 1 1
1	1	0 0 0 0 0 1 1 0
2	2	0 1 0 1 1 0 1 1
3	3	0 1 0 0 1 1 1 1
4	4	0 1 1 0 0 1 1 0
5	5	0 1 1 0 1 1 0 1
6	6	0 1 1 1 1 1 0 1
7	7	0 0 0 0 0 1 1 1



(IN) LSD	Segment Display	(OUT) - g f e d c b a
8	8	0 1 1 1 1 1 1 1
9	9	0 1 1 0 0 1 1 1
A	A	0 1 1 1 0 1 1 1
B	b	0 1 1 1 1 1 0 0
C	C	0 0 1 1 1 0 0 1
D	d	0 1 0 1 1 1 1 0
E	E	0 1 1 1 1 0 0 1
F	F	0 1 1 1 0 0 0 1

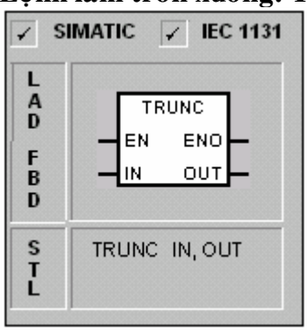
Lệnh làm tròn: ROUND



EN: ngõ vào cho phép
IN: ngõ vào
OUT: ngõ ra
Một giá trị số thực ở ngõ vào được làm tròn và chuyển thành số DInt ở ngõ ra. Nếu số lẻ >=0.5 thì giá trị số thực sẽ được làm tròn lên, ngược lại thì làm tròn xuống.

IN	VD, ID, QD, MD, SD, SMD, LD, AC, Constant, *VD, *LD, *AC	REAL
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	DINT

Lệnh làm tròn xuống: TRUNC



EN: ngõ vào cho phép
IN: ngõ vào
OUT: ngõ ra
Một giá trị số thực ở ngõ vào được làm tròn xuống và chuyển thành số DInt ở ngõ ra.
VD: 5.9 TRUNC 5

IN	VD, ID, QD, MD, SD, SMD, LD, AC, Constant, *VD, *LD, *AC	REAL
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	DINT

Bài Tập:

1/ Kiểm soát số lượng xe ra vào trong 1 trạm xe,điều khiển cửa đóng mở tự động ,kiểm soát xe đưa lên máy tính.
Để biết được chiều xe ra vào,ta sử dụng 2 Sensor
Để mở cửa tự động,ta sử dụng 2 Sensor trước sau,và các Sensor giới hạn trong,giới hạn ngoài.

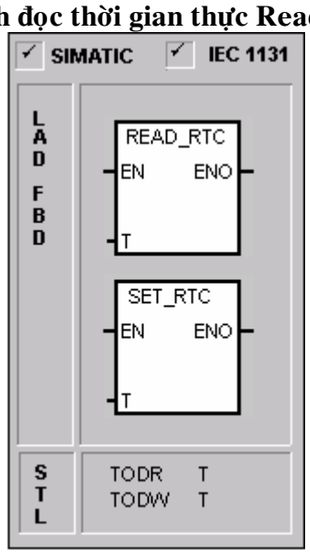
Mở rộng: Mỗi xe vào ra,có gắn 1 Tag (Mã vạch) Tại mỗi trạm đóng mở cửa,ta gắn 1 bộ Read,Write để có thể kiểm soát các thông số của xe vào ra,đồng thời có thể lập report.
Mở rộng cho bài toán tại trạm thu phí tự động,mỗi xe mua 1 SimCard,tại trạm thu phí đặt 1 bộ Read,Write kiểm soát từng xe ra vào.
2 / Ứng dụng trong nghành đá Granit:
Điều khiển trạm mài đá: Hệ thống mài đá có tất cả 10 đầu mài,khoảng cách giữa mỗi đầu mài là cố định,tốc độ băng tải đưa đá vào là cố định (hoặc được nhập vào bằng màn hình),do vậy thời gian giữa 2 đầu mài là biết trước.Đá đưa vào băng tải được phát hiện bằng 1 công tắc hành trình.
Gợi ý: Bài toán sử dụng Timer (định thời gian).
Cũng bài toán này,nếu ta sử dụng Encoder để có thể kiểm soát vị trí của băng tải ,từ vị trí đó ta có thể điều khiển đầu mài cho hợp lí.
Để có thể đọc Encoder,Encoder này được đưa về 1 bộ đếm trung gian để Scale lại thành số xung tốc độ thấp,đưa về PLC.

3/ Trạm ép gạch 1 trạm:

Hệ thống ép gạch bao gồm các công đoạn :
Công đoạn đổ nguyên liệu vào khuôn.(khi khuôn nằm ngoài)
Công đoạn rung khuôn (khi đạp rung khuôn)
Công đoạn khuôn đi vào trạm ép
Công đoạn ép gạch
Công đoạn khuôn lên
Công đoạn khuôn xuống
Công đoạn khuôn đi ra.
Kết thúc một chu trình ép gạch.

12/ Một số lệnh mở rộng:

a/Lệnh đọc thời gian thực Read_RTC:



Bit EN : Bit cho phép đọc thời gian thực
T (8byte): VB,IB,QB,MB,SB,LB,*AC,*VD,*LD
Được định dạng như sau:

T (byte)	Giá trị (định dạng BCD)
0 (năm)	0-99
1 (tháng)	0 -12
2 (ngày)	0 - 31
3 (giờ)	0 - 23
4 (phút)	0 - 59
5 (giây)	0 - 59
6 (00)	00
7 (ngày trong tuần)	1 – 7; 1: Sunday

b/ Lệnh Set thời gian thực Set_RTC:

Khi có tín hiệu EN thì thời gian thực sẽ được set lạ thông qua **T**
Cách định dạng Byte T hoàn toàn giống ở trên.

Bài tập:

Sử dụng lệnh đọc thời gian thực để ứng dụng trong điều khiển đèn giao thông tự động,tưới cây tự động.

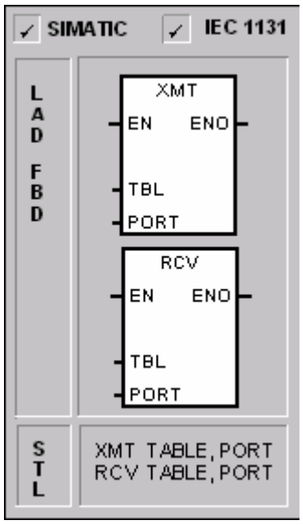
- Điều khiển đèn giao thông tự động:Thời gian từ 5g sáng đến 11g sáng: hoạt động bình thường

Từ 11g sáng đến 5g sáng ngày hôm sau: Đèn vàng chớp tắt xung 1s.

- Điều khiển tưới cây tự động: Để phục vụ cho việc tưới cây (trong phòng kiếng),Lan đòi hỏi nhu cầu tưới nước rất khắc nghiệt,đòi hỏi cách 1 khoảng thời gian nhất định cho việc tưới cây,và còn phụ thuộc theo từng tháng.Tháng mùa nóng nhu cầu tưới nước nhiều hơn mùa mưa.

c/ Lệnh Giao tiếp (Communication):

Lệnh truyền nhận: Lệnh truyền (**XMT**); Lệnh Nhận (**RCV**).



Bit EN : tín hiệu cho phép truyền dữ liệu qua cổng Com
TBL : VB,MB,IB,QB.SMB,*LD,*AC,*VD
Port : 0 cho CPU 221,222,224
0,1 cho CPU 224XP,CPU226

TBL : Byte chứa số lượng byte cũng như vị trí byte bắt đầu truyền qua cổng Port giao tiếp

Port : Chỉ Port thực hiện việc truyền nhận dữ liệu

Ví dụ : Muốn truyền chuỗi “TRI” qua cổng Port 0 ta thực hiện đoạn lệnh sau:

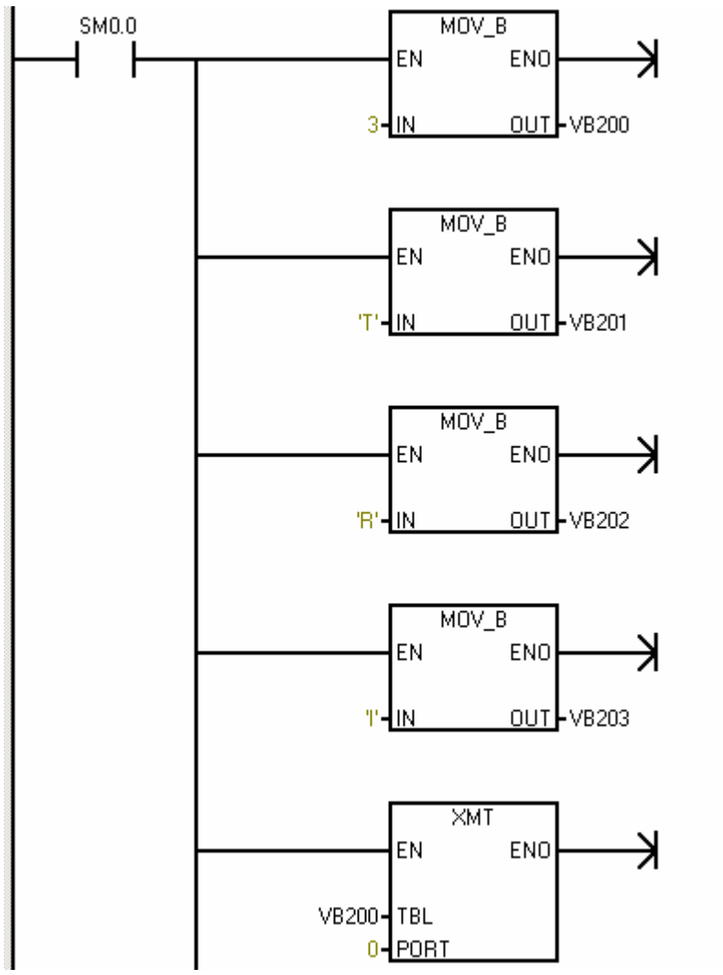
Trong đó : 3 là số Byte cần truyền ,được đưa vào VB200

‘T’ được đưa vào Byte VB201

‘R’ được đưa vào Byte VB202

‘I’ được đưa vào Byte VB203

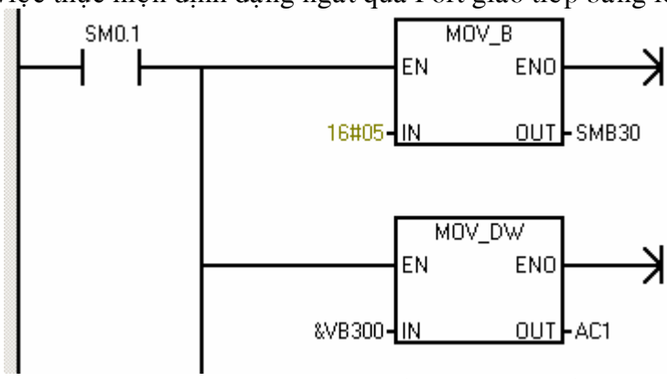
Lệnh truyền được thực hiện bằng lệnh XMT



Việc nhận dữ liệu qua Port giao tiếp được thực hiện bằng 2 cách:

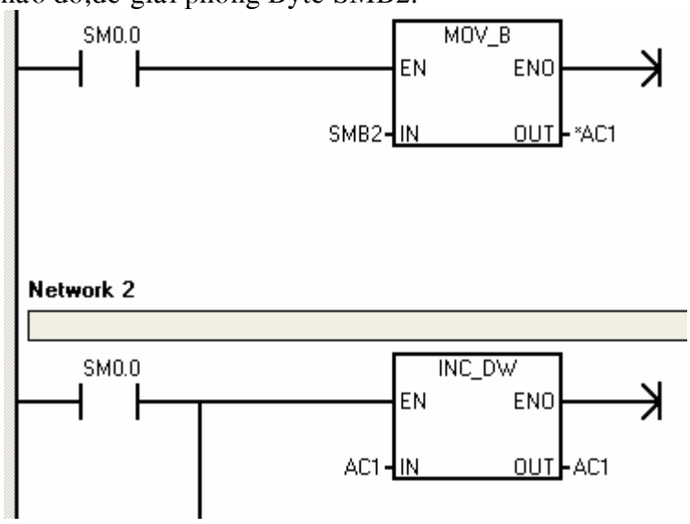
- Có thể thực hiện việc nhận dữ liệu bằng lệnh RCV (hoàn toàn tương tự việc truyền dữ liệu)
- Nhận dữ liệu bằng cách dùng ngắt thông qua Port giao tiếp,phương pháp này thường được dùng nhiều hơn,do phương pháp này có thể quản lí được số lượng Byte truyền nhận dễ dàng hơn.

Để thực hiện việc truyền nhận dữ liệu thông qua phương pháp thứ 2 ,trước hết ta phải thực hiện việc thực hiện định dạng ngắt qua Port giao tiếp bằng lệnh:



Trong đó SMB30 là Byte định dạng cho ngắt Port giao tiếp.

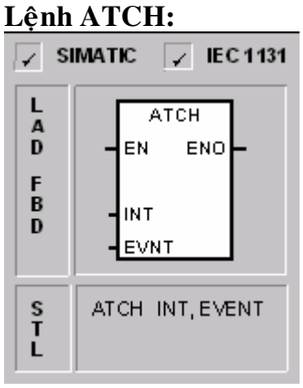
AC1 là con trỏ để sử dụng cho việc nhận dữ liệu từ Port giao tiếp.
Dữ liệu được nhận thông qua Port giao tiếp nằm trong Byte SMB2,do vậy sau mỗi lần nhận được dữ liệu thông qua chương trình ngắt,thì dữ liệu đó phải được cất vào 1 Byte tương ứng nào đó,để giải phóng Byte SMB2.



Dữ liệu nhận được đưa vào Byte VB300 sau đó tăng con trỏ lên 1,để trỏ tới Byte VB301,dữ liệu nhận tiếp theo sẽ được đưa vào Byte kế tiếp,tương tự như vậy khi số Byte nhận kết thúc,thì dữ liệu sẽ nằm trong 1 số Byte từ VB300,từ đó ta có thể sử lí Byte nhận được dễ dàng.

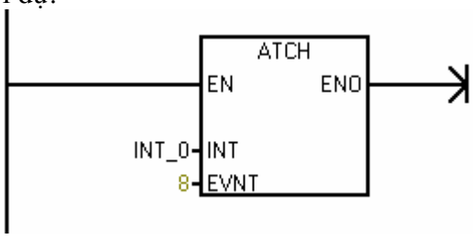
d/ Các lệnh về ngắt:

Lệnh ATCH:



Bit EN : tín hiệu cho phép thực hiện lệnh ATCH
INT : Chương trình ngắt được gọi khi có sự kiện ngắt xảy ra
EVNT : Số thứ tự sự kiện ngắt

Ví dụ:



Khi gặp sự kiện ngắt số 8 (Sự kiện ngắt Port nối tiếp) ,chương trình sẽ gọi chương trình ngắt INT_0.
Bảng sự kiện ngắt:

Event Number	Interrupt Description	Priority Group	Priority in Group	Supported by CPU 221	222	224	224 XP 226 226 XM
8	Port 0: Rcv character	Comm.	0	✓	✓	✓	✓
9	Port 0: Xmt complete	(Highest)	0	✓	✓	✓	✓
23	Port 0: Rcv msg complete		0	✓	✓	✓	✓
24	Port 1: Rcv msg complete		1				✓
25	Port 1: Rcv character		1				✓
26	Port 1: Xmt complete		1				✓
19	PTO 0 complete interrupt		0	✓	✓	✓	✓
20	PTO 1 complete interrupt		1	✓	✓	✓	✓
0	Rising edge, I0.0	Discrete	2	✓	✓	✓	✓
2	Rising edge, I0.1	(Middle)	3	✓	✓	✓	✓
4	Rising edge, I0.2		4	✓	✓	✓	✓
6	Rising edge, I0.3		5	✓	✓	✓	✓
1	Falling edge, I0.0		6	✓	✓	✓	✓
3	Falling edge, I0.1		7	✓	✓	✓	✓
5	Falling edge, I0.2		8	✓	✓	✓	✓
7	Falling edge, I0.3		9	✓	✓	✓	✓
12	HSC0 CV=PV		10	✓	✓	✓	✓
27	HSC0 direction changed		11	✓	✓	✓	✓
28	HSC0 external reset/Zphase		12	✓	✓	✓	✓
13	HSC1 CV=PV		13			✓	✓
14	HSC1 direction changed		14			✓	✓
15	HSC1 external reset		15			✓	✓
16	HSC2 CV=PV		16			✓	✓
17	HSC2 direction changed		17			✓	✓
18	HSC2 external reset		18			✓	✓
32	HSC3 CV=PV		19	✓	✓	✓	✓
29	HSC4 CV=PV		20	✓	✓	✓	✓
30	HSC4 direction changed		21	✓	✓	✓	✓
31	HSC4 external reset/Zphase		22	✓	✓	✓	✓
33	HSC5 CV=PV		23	✓	✓	✓	✓
10	Timed interrupt 0	Timed	0	✓	✓	✓	✓
11	Timed interrupt 1	(Lowest)	1	✓	✓	✓	✓
21	Timer T32 CT=PT interrupt		2	✓	✓	✓	✓
22	Timer T96 CT=PT interrupt		3	✓	✓	✓	✓

SMB30 và SMB130:

SMB30 là Byte điều khiển giao tiếp cho Cổng giao tiếp 0, SMB130 là Byte điều khiển giao tiếp cho Cổng giao tiếp 1,Ta có thể định dạng cho 2 Byte này,những byte này có chức năng định dạng cho từng Port giao tiếp như tốc độ Baud,dạng truyền thông....

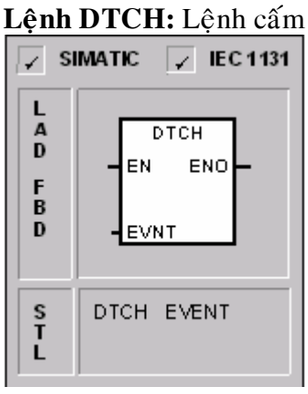
Bảng định dạng :

S7-200 Symbol Name	SM Address		Bit Format	Bit Format
	Port 0	Port 1	Bit Format	MSB 7 … 0 LSB: p p d b b b m m
P0_Config	SMB30			
P1_Config		SMB130		
	SM30.6 – SM30.7	SM130.6 – SM130.7	pp:	0 0 = No parity 0 1 = Even parity 1 0 = No parity 1 1 = Odd parity
	SM30.0 – SM30.1	SM130.0 – SM130.1	d:	0 = 8 data bits per character 1 = 7 data bits per character
	SM30.2 – SM30.4	SM130.2 – SM130.4	bbb:	0 0 0 = 38,400 bps 0 0 1 = 19,200 bps 0 1 0 = 9,600 bps 0 1 1 = 4,800 bps 1 0 0 = 2,400 bps 1 0 1 = 1,200 bps 1 1 0 = 115,200 bps* 1 1 1 = 57,600 bps* * Requires S7-200 CPU version 1.2 or later
P0_Config_0	SM30.0		mm:	0 0 = Point-to-Point Interface protocol (PPI/slave mode) 0 1 = Freeport protocol 1 0 = PPI/master mode 1 1 = Reserved (defaults to PPI/slave mode) **Note:** When you select code mm = 10 (PPI master), the CPU becomes a master on the network and allows the NETR and NETW instructions to be executed. Bits 2 through 7 are ignored in PPI modes.
P1_Config_0		SM130.0		
	SM30.1	SM130.1		

Ở ví dụ trên: SMB30=05 tương ứng:

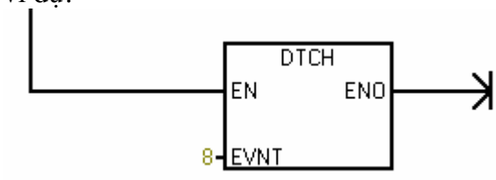
- Port giao tiếp : Port 0
- No parity (0 0)
- 8 data Bits (0)
- 19200 bps (0 0 1)
- Freeport protocol (0 1)

Lệnh DTCH: Lệnh cấm ngắt



Bit EN : tín hiệu cho phép thực hiện lệnh DTCH
EVNT : Số thứ tự sự kiện ngắt ,bị cấm

ví dụ:



Cấm sự kiện ngắt số 8,Sự kiện ngắt số 8 chỉ được cho phép trở lại bằng lệnh **ATCH**
Ngoài ra còn có các lệnh cho phép ngắt (**ENI**) và cấm ngắt (**DISI**) và lệnh trở về của chương tình ngắt (**RETI**).

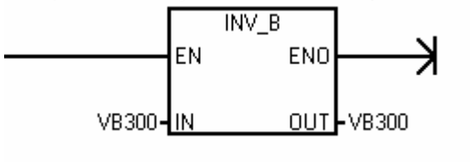
Bài Tập:

1/ Sử dụng lệnh giao tiếp và sử lí ngắt để giao tiếp giữa PLC và máy tính thông qua cổng 0 (Port 0).

2/ Sử dụng lệnh giao tiếp để giao tiếp PLC và đầu cân Redlion thông qua Port 0

13/ Các lệnh về sử lí logic (Logical Operation):

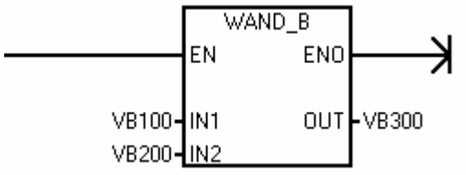
a/Lệnh đảo Byte INV_B: Thực hiện việc đảo các bit trong Byte



Ví dụ: VB300 : 0100 1001
Sau lệnh INV_B VB300 : 1011 0110

b/Lệnh đảo Word INV_W: Thực hiện việc đảo các bit trong Word

c/Lệnh đảo DWord INV_DW: Thực hiện việc đảo các bit trong Dword



VB300 = VB200 AND VB100
VB200 0010 1110
VB100 1111 1001
Kết quả: VB300 0010 1000

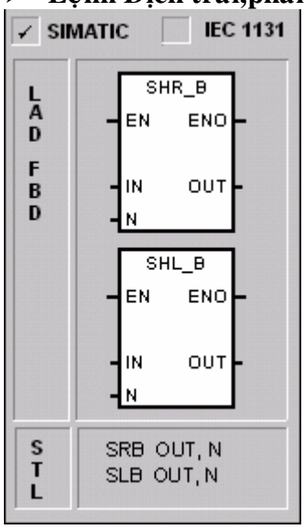
e/Lệnh WAND_W: Thực hiện việc AND 2 Word kết quả cất vào Word Out

e/Lệnh WAND_DW: Thực hiện việc AND 2 DWord kết quả cất vào DWord Out

Hoàn toàn tương tự ta có các lệnh **WOR_B, WOR_W, WOR_DW, WXOR_B , WXOR_W WXOR_DW.**

Các lệnh về dịch Bit:

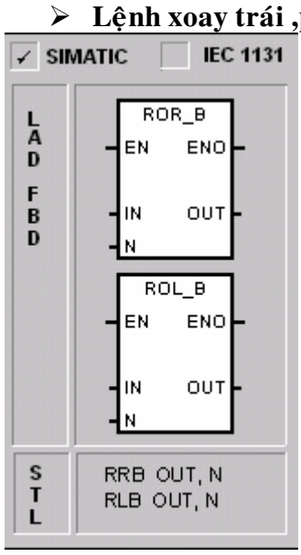
- **Lệnh Dịch trái,phải Byte:**



Bit EN : Bit cho phép thực hiện lệnh dịch trái,dịch phải
IN : Byte được dịch
OUT: Kết quả của Byte dịch
N : Số Byte dịch
Các Bit dịch ra ngoài,bị loại bỏ
Các số 0 được dịch vào Bit mới
Ví dụ:
Byte : 1101 1101 Sau lệnh dịch N=3 kết quả:
0001 1011 (3 Bit 000 mới được đẩy vào,3 Bit 101 bị đẩy ra)

- **Tương tự có lệnh dịch trái,phải Word,Dword:**

➢ **Lệnh xoay trái ,phải Byte:**

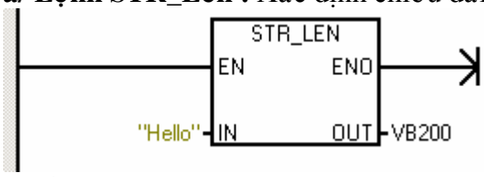


Bit EN : Bit cho phép thực hiện lệnh xoay trái,xoay phải
IN : Byte được xoay
OUT: Kết quả của Byte xoay
N : Số Byte xoay
Các Bit dịch ra ngoài được xoay trở lại Bit đầu
Ví dụ:
Byte : 1101 1101 Sau lệnh xoay N=2 kết quả:
0111 0111

Tương tự có lệnh xoay phải,trái Word,Dword.

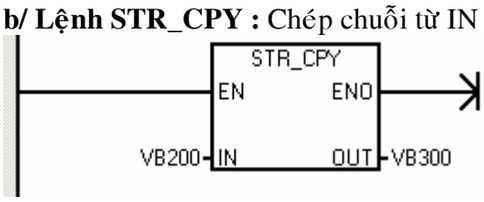
14/Các lệnh về xử lí chuỗi:

a/ Lệnh STR_Len : Xác định chiều dài của chuỗi(In) kết quả cất vào Byte Out



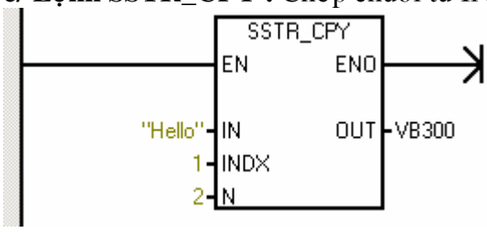
Chiều dài chuỗi Hello là 5,do đó VB200 = 5

b/ Lệnh STR_CPY : Chép chuỗi từ IN sang OUT



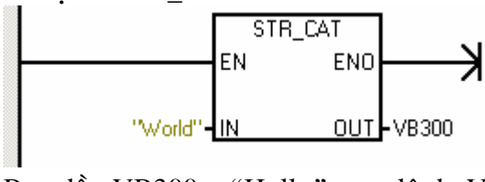
Chép chuỗi từ VB200 sang VB300

c/ Lệnh SSTR_CPY : Chép chuỗi từ IN từ vị trí INDX sang OUT (số kí tự Copy là N)



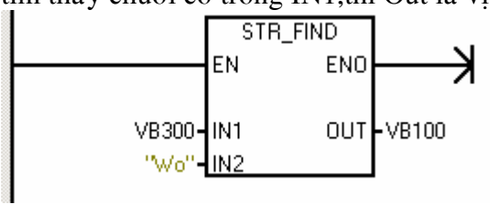
Copy chuỗi Hello từ vị trí thứ 1,số kí tự copy là 2,do đó VB300 = “He”

d/ Lệnh STR_CAT : Nối chuỗi từ IN thêm vào OUT



Ban đầu VB300 = “Hello” ; sau lệnh VB300 là “ Hello World “

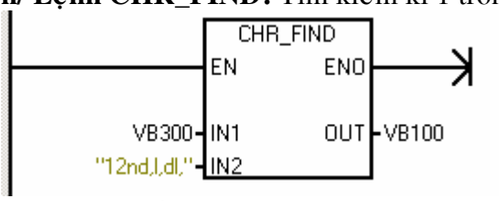
e/ Lệnh STR_FIND: Lệnh tìm kiếm chuỗi tồn tại trong **IN1,**chuỗi cần tìm trong **IN2** ,Nếu tìm thấy chuỗi có trong IN1,thì Out là vị trí tìm thấy trong chuỗi đó.



VB300 = “Hello World”

Sau lệnh trên VB100 = 7

h/ Lệnh CHR_FIND: Tìm kiếm kí 1 trong các kí tự trong **IN2** trong chuỗi **IN1**



IV/ Các Ứng dụng quan trọng trong S7_200:

1/ Xuất xung tốc độ cao:

CPU S7_200 có 2 ngõ ra xung tốc độ cao (**Q0.0 ,Q0.1**),dùng cho việc điều rộng xung tốc độ cao nhằm điều khiển các thiết bị bên ngoài.

Việc điều rộng xung được thực hiện thông qua việc định dạng Wizard

Có 2 cách điều rộng xung:điều rộng xung 50%,và điều rông xung theo tỉ lệ .

a/Điều rộng xung 50% (PTO):

Để thực hiện việc phát xung tốc độ cao (PTO) trước hết ta phải thực hiện các bước định dạng sau:

- Reset ngõ xung tốc độ cao ở chu kì đầu của chương trình
- Chọn loại ngõ ra phát xung tốc độ cao Q0.0 hay Q0.1
- Định dạng thời gian cơ sở (Time base) dựa trên bảng sau:

	Enable	Select Mode	PTO Segment Operation	PWM Update Method	Time Base	Pulse Count	Pulse Width	Cycle Time
16#81	Yes	PTO	Single		1 µs/cycle			Load
16#84	Yes	PTO	Single		1 µs/cycle	Load		
16#85	Yes	PTO	Single		1 µs/cycle	Load		Load
16#89	Yes	PTO	Single		1 ms/cycle			Load
16#8C	Yes	PTO	Single		1 ms/cycle	Load		
16#8D	Yes	PTO	Single		1 ms/cycle	Load		Load
16#A0	Yes	PTO	Multiple		1 µs/cycle			
16#A8	Yes	PTO	Multiple		1 ms/cycle			

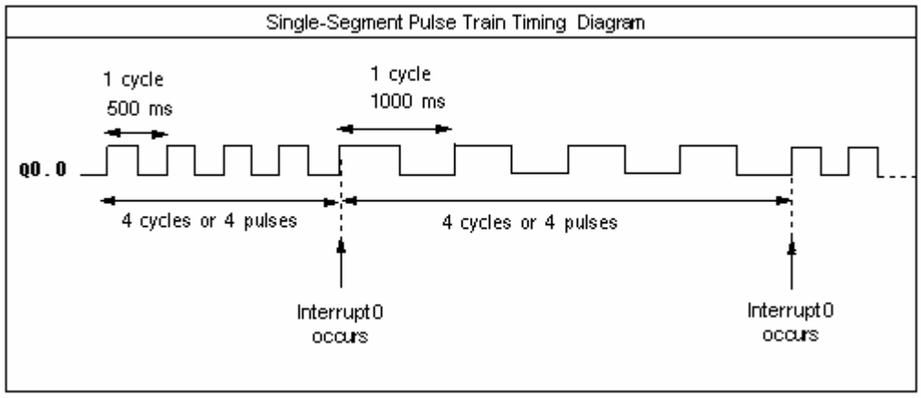
Các Byte cho việc định dạng **SMB67 (cho Q0.0)**

SMB77 (cho Q0.1)

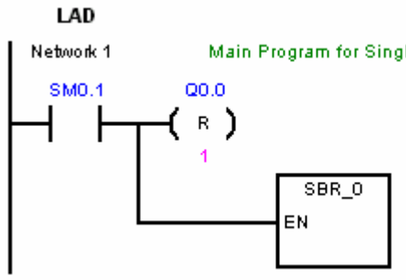
Ngoài ra:

Q0.0	**Q0.1**	
SMW68	**SMW78**	**:**Xác định chu kì thời gian
SMW70	**SMW80**	**:**Xác định chu kì phát xung
SMD72	**SMD82**	**:**Xác định số xung điều khiển

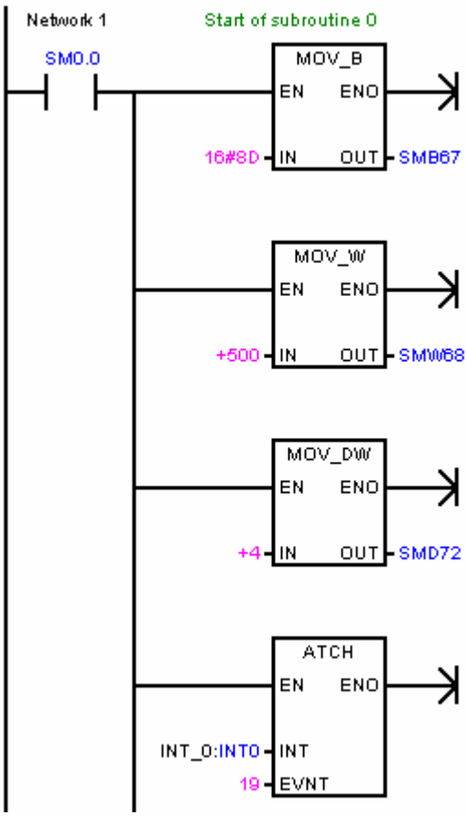
Ví dụ : Thực hiện việc điều rộng xung nhanh kiểu **PTO** theo giản đồ tại ngõ ra Q0.0:



Ta thực hiện chương trình như sau:



Reset Q0.0 ở chu kì quét đầu
Gọi chương trình con SBR_0

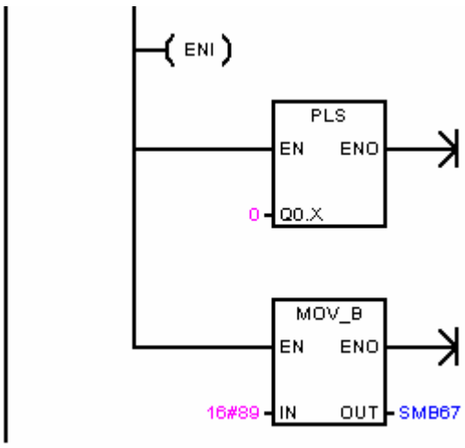


Định dạng SMB67 = 16#8D : Định dạng xung tốc độ cao ở ngõ ra Q0.0,
Thời gian cơ sở là 1ms/cycle,cho phép Load số xung và chu kì thời gian

Nạp chu kì thời gian là 500x1ms = 500ms

Nạp số xung là 4

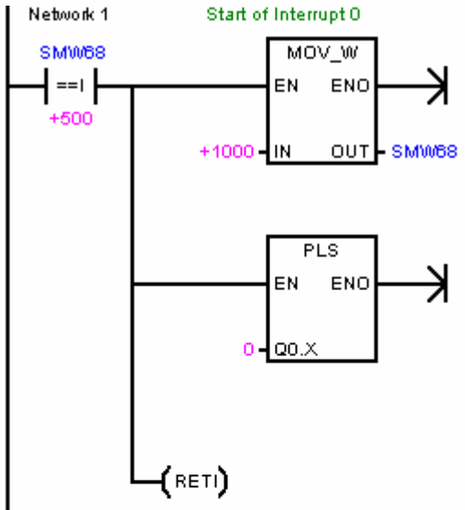
Liên kết với sự kiện ngắt số 18 (khi số xung phát ra bằng số xung đặt là 4)



Cho phép ngắt (ENI)

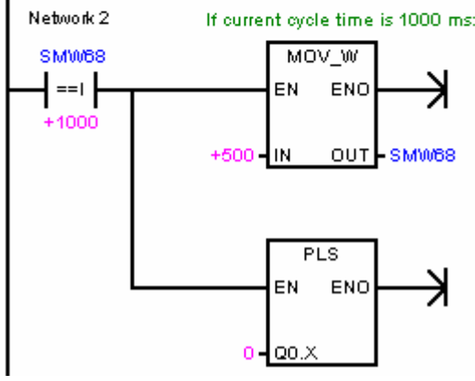
Lệnh xuất xung tốc độ cao ra Q0.0

SMB67 =16#89:thời gian cơ sở 1ms/cycle nhưng khi đó chỉ cho phép Load chu kì thời gian mà thôi.



Khi số xung phát ra bằng 4,chương trình ngắt INT_0 được thực thi
Nếu thời gian cơ sở =500ms,thì chuyển sang 1000ms rồi cho phát xung trở lại

Quay trở lại chương trình chính



Nếu thời gian cơ sở 1000ms ,thì chuyển sang 500ms

Cho phép xuất xung trở lại qua Q0.0

a/Điều rộng xung theo tỉ lệ (PWM):

Để thực hiện việc phát xung tốc độ cao (PWM) trước hết ta phải thực hiện các bước định dạng sau:

- Reset ngõ xung tốc độ cao ở chu kì đầu của chương trình
- Chọn loại ngõ ra phát xung tốc độ cao Q0.0 hay Q0.1
- Định dạng thời gian cơ sở (Time base) dựa trên bảng sau:

16#D1	Yes	PWM	Synchronous	1 µs/cycle		Load
16#D2	Yes	PWM	Synchronous	1 µs/cycle	Load	
16#D3	Yes	PWM	Synchronous	1 µs/cycle	Load	Load
16#D9	Yes	PWM	Synchronous	1 ms/cycle		Load
16#DA	Yes	PWM	Synchronous	1 ms/cycle	Load	
16#DB	Yes	PWM	Synchronous	1 ms/cycle	Load	Load

Các Byte cho việc định dạng **SMB67 (cho Q0.0)**

SMB77 (cho Q0.1)

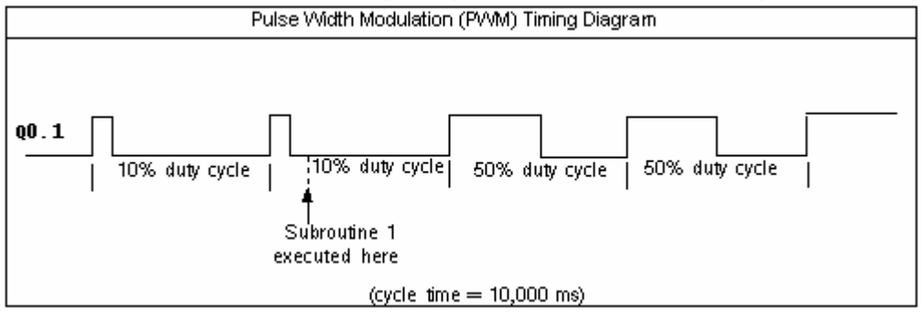
Ngoài ra: **Q0.0** **Q0.1**

SMW68 **SMW78** **:**Xác định chu kì thời gian

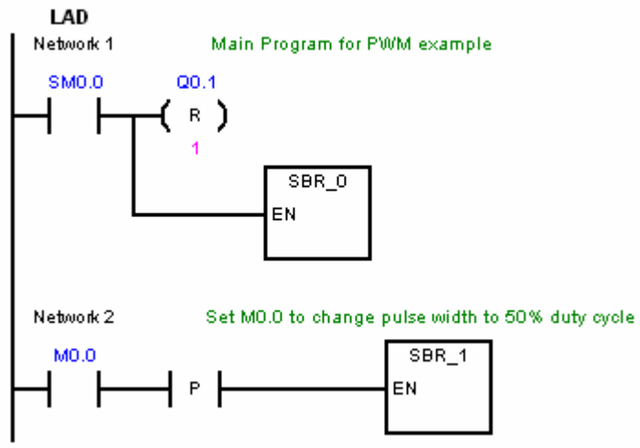
SMW70 **SMW80** **:**Xác định chu kì phát xung

SMD72 **SMD82** **:**Xác định số xung điều khiển

Ví dụ : Thực hiện việc điều rộng xung nhanh kiểu **PWM** theo giản đồ tại ngõ ra Q0.1:



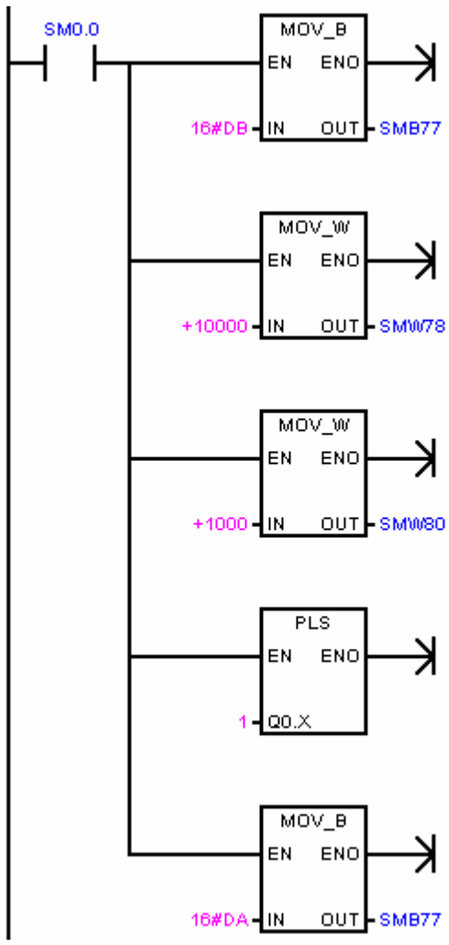
Chương trình được thực hiện như sau:



Reset Q0.1 ở đầu chương trình

Gọi chương trình con SBR_0

Khi có M0.0 gọi chương trình con SBR_1 để thay đổi độ rộng xung



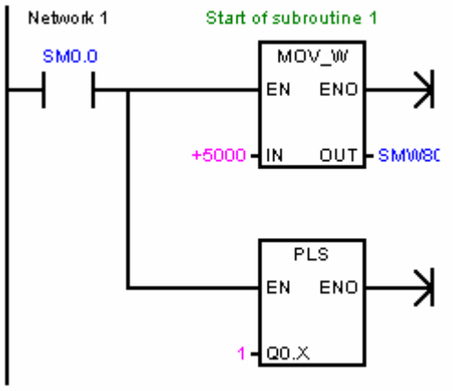
SMB77 = 16#DB : Định dạng ngõ ra xung Q0.1 ,thời gian cơ sở 1ms /cycle cho phép Load độ rộng xung cũng như chu kì thời gian.

SMW78 = 10000 : chu kì thời gian là 10000x1ms =10000ms

SMW80 = 1000 Độ rộng xung on là 1000ms

Phát xung ra tốc độ cao ở ngõ ra Q0.1

SMB77 =16#DA :thời gian cơ bản 1ms,chỉ cho phép Load độ rộng xung on

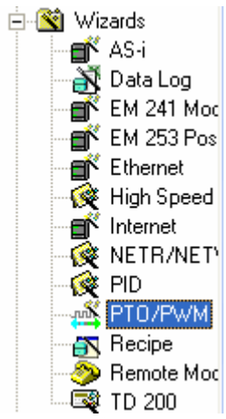


Load lại xung on là 5000ms,khi có tín hiệu M0.0

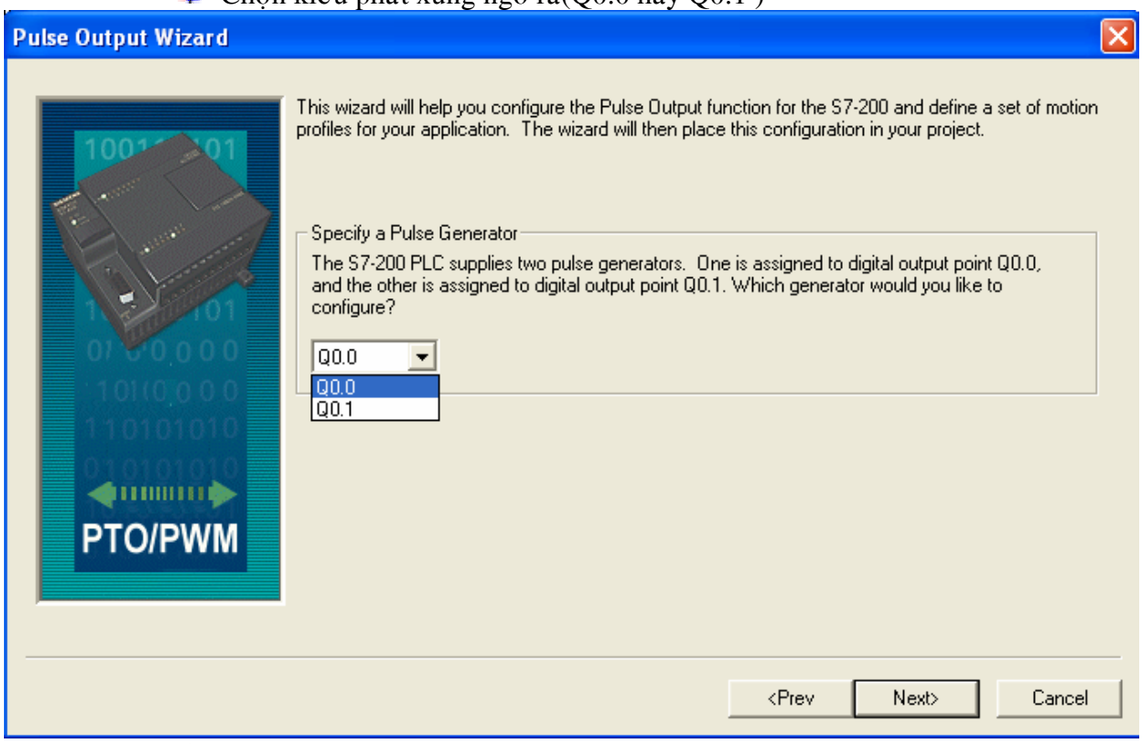
Cho phép xuất xung nhanh ra ngõ Q0.1

❖ **Ngoài ra ta có thể định dạng ngõ ra xung tốc độ cao thông qua việc định dạng Wizard theo các bước sau:**

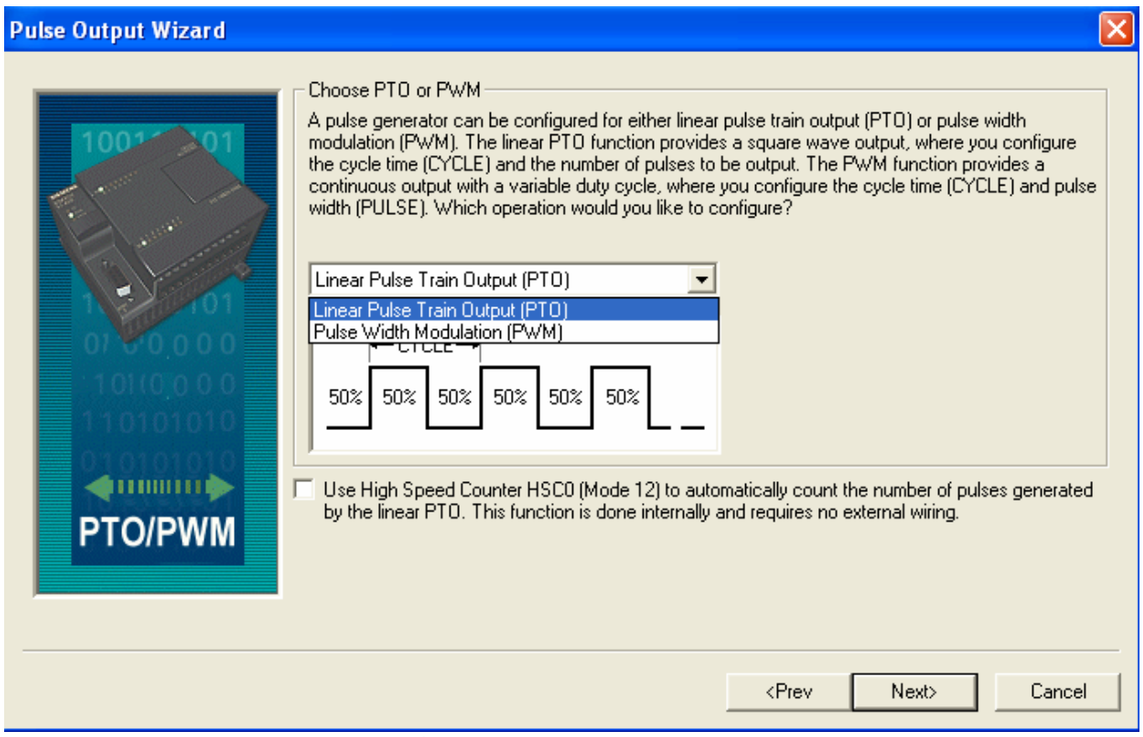
- Vào Wizard chọn PTO/PWM :



- Chọn kiểu phát xung ngõ ra(Q0.0 hay Q0.1)

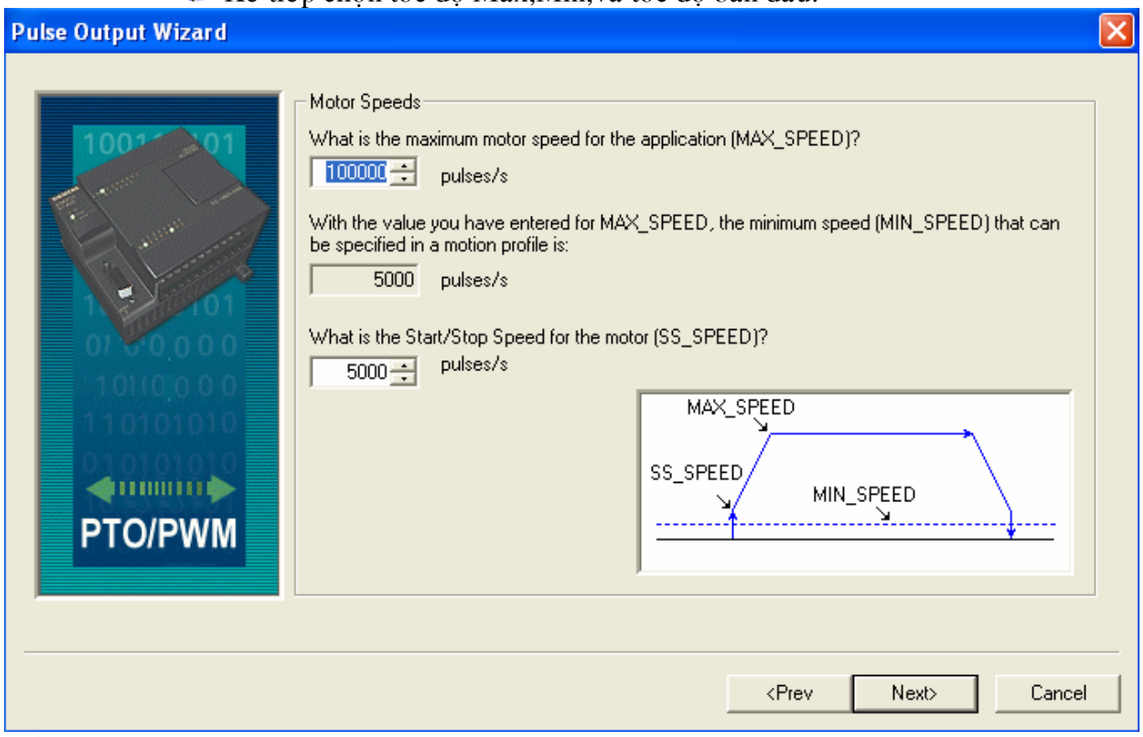


- Sau khi chọn loại ngõ ra,chọn next



Sau đó chọn loại xung cho việc điều rộng: xung PTO hay xung PWM

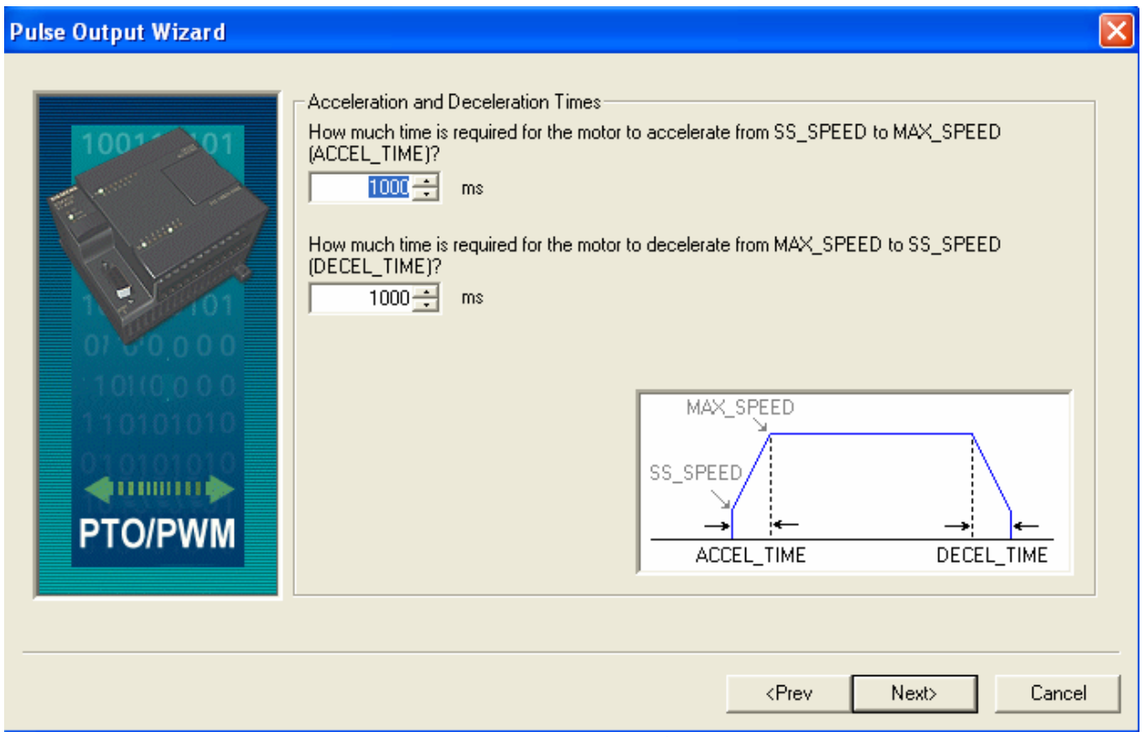
- Kế tiếp chọn tốc độ Max,Min,và tốc độ ban đầu:



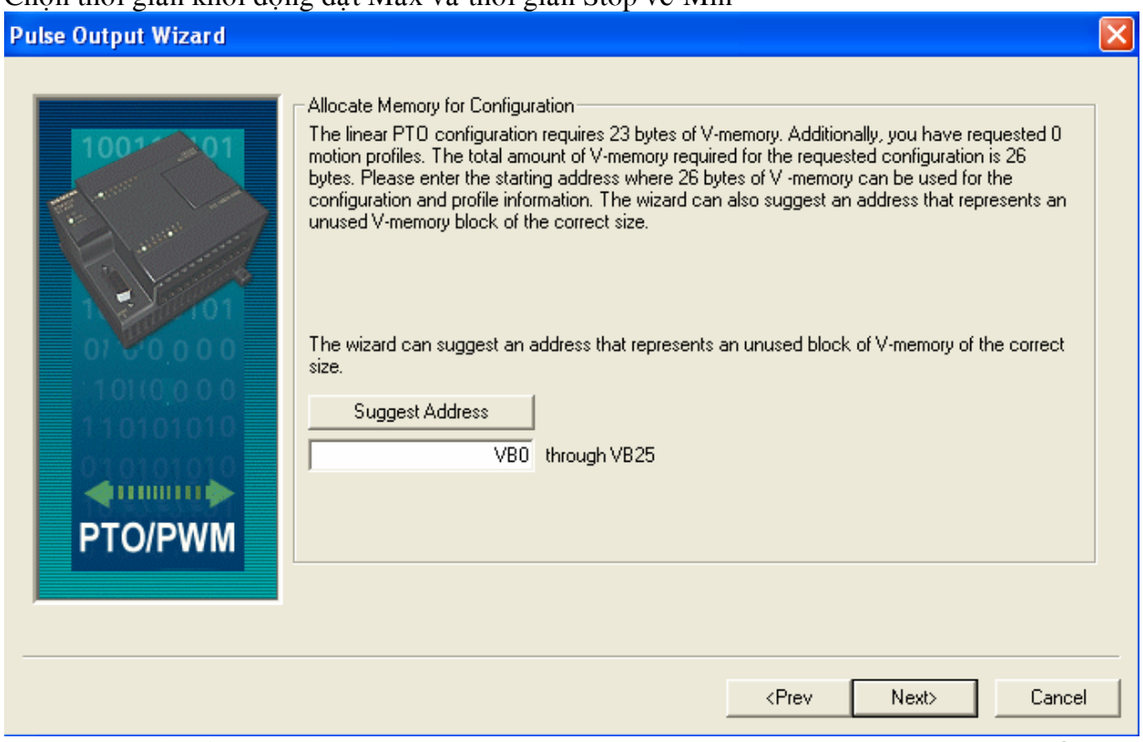
Tốc độ Max: 100000pulse/s

Tốc độ Min : 5000pulse/s

Tốc độ Start: 5000pulse/s



Chọn thời gian khởi động đạt Max và thời gian Stop về Min



Chọn Byte bắt đầu của số Byte cho việc định Wizard,cuối cùng chọn Next và Finish để kết thúc,

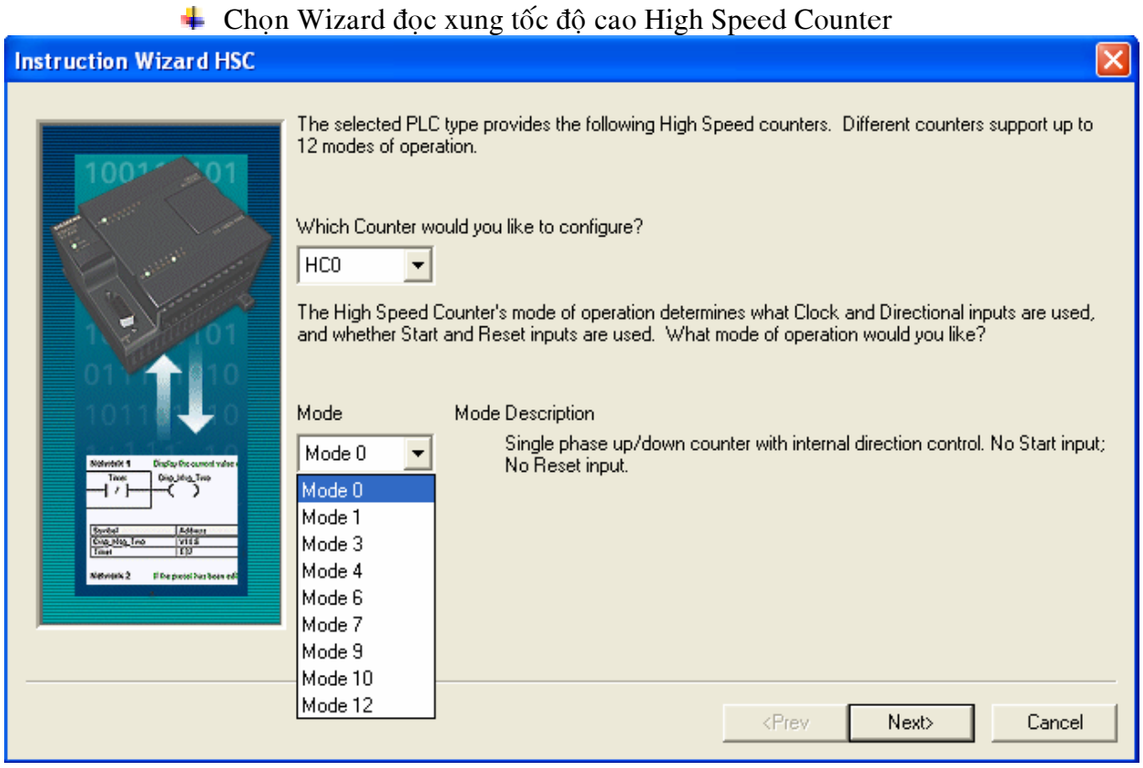
Sau khi kết thúc việc định Wizard,chương trình sẽ tạo ra 2 chương trình con cho việc định dạng phát xung tốc độ cao,2 chương trình con đó là : **X_CTRL, và X_MAN**

Sau đó ta chỉ sử dụng 2 chương trình con này cho việc định dạng.

2/ Đọc xung tốc độ cao:

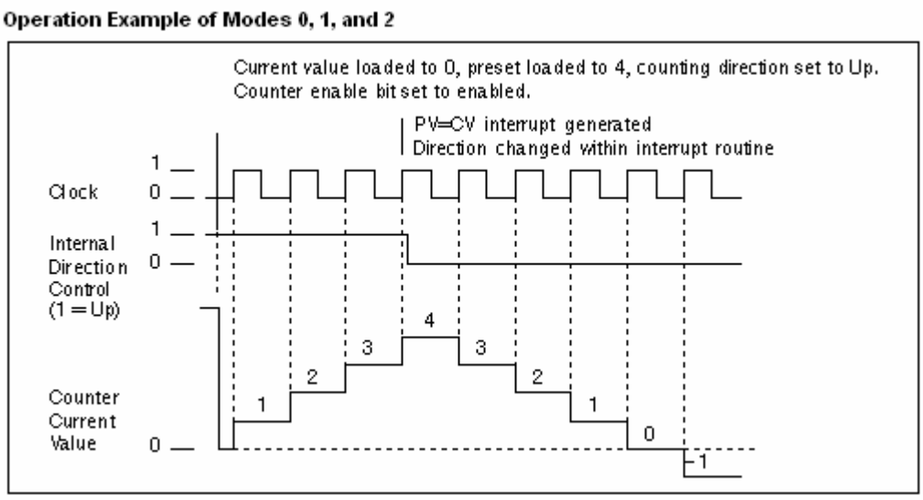
Để đọc xung tốc độ cao ,ta thực hiện các bước sau cho việc định dạng Wizard:

- Chọn Wizard đọc xung tốc độ cao High Speed Counter



Chọn Mode đọc xung tốc độ cao và loại Counter nào (HC0,HC1...)

Tuỳ từng loại ứng dụng mà ta có thể chọn nhiều Mode đọc xung tốc độ cao khác nhau,có tất cả 12 Mode đọc xung tốc độ cao như sau:

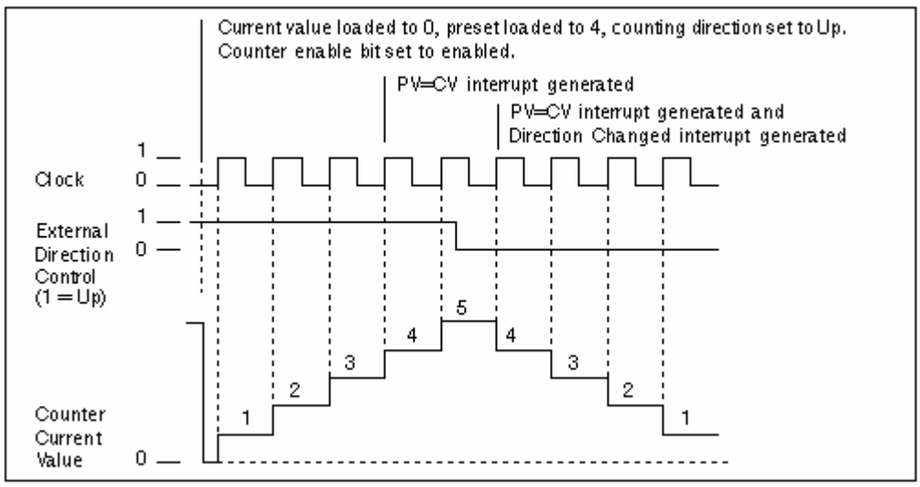


Mode 0,1,2 : Dùng đếm 1 pha với hướng đếm được xác định bởi Bit nội

Mode 0: Chỉ đếm tăng hoặc giảm,không có Bit Start cũng như bit Reset

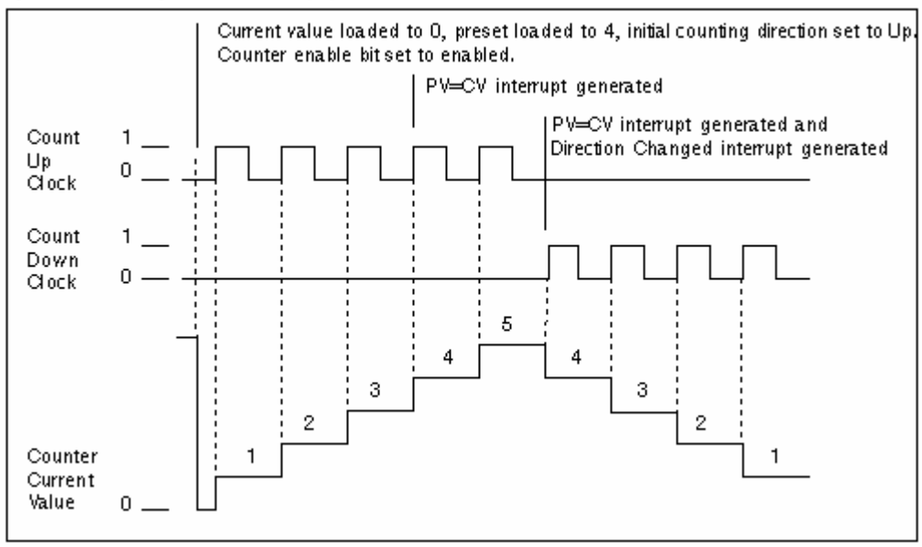
Mode 1: Đếm tăng hoặc giảm,có bit Reset nhưng không có bit Start
Mode 2: Đếm tăng hoặc giảm,Có Bit Start cũng như bit Reset để cho phép chọn bắt đầu đếm cũng như chọn thời điểm bắt đầu Reset.Các Bit Start cũng như Reset là các ngõ Input chọn từ bên ngoài.

Operation Example of Modes 3, 4, and 5



Mode 3,4,5: Dùng đếm 1 pha với hướng đếm được xác định bởi Bit ngoại,tức là có thể chọn từ ngõ vào input.
Mode 3: Chỉ đếm tăng hoặc giảm,không có Bit Start cũng như bit Reset
Mode 4: Đếm tăng hoặc giảm,có bit Reset nhưng không có bit Start
Mode 5: Đếm tăng hoặc giảm,Có Bit Start cũng như bit Reset để cho phép chọn bắt đầu đếm cũng như chọn thời điểm bắt đầu Reset.Các Bit Start cũng như Reset là các ngõ Input chọn từ bên ngoài.

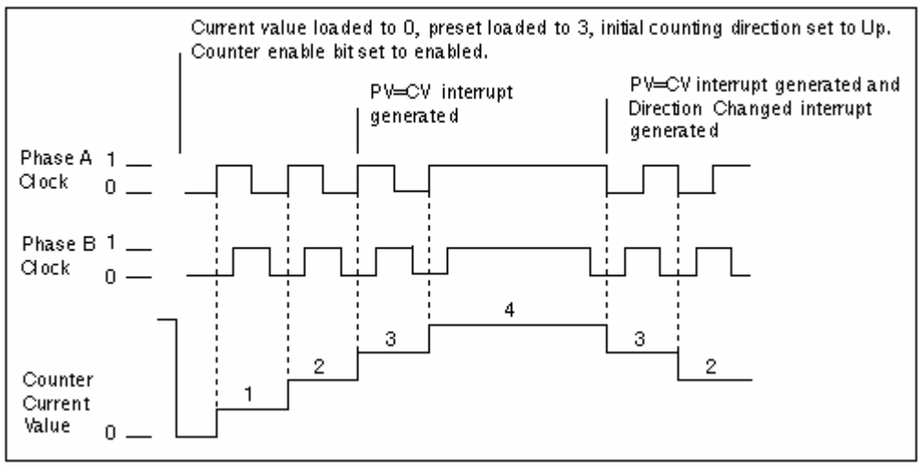
Operation Example of Modes 6, 7, and 8



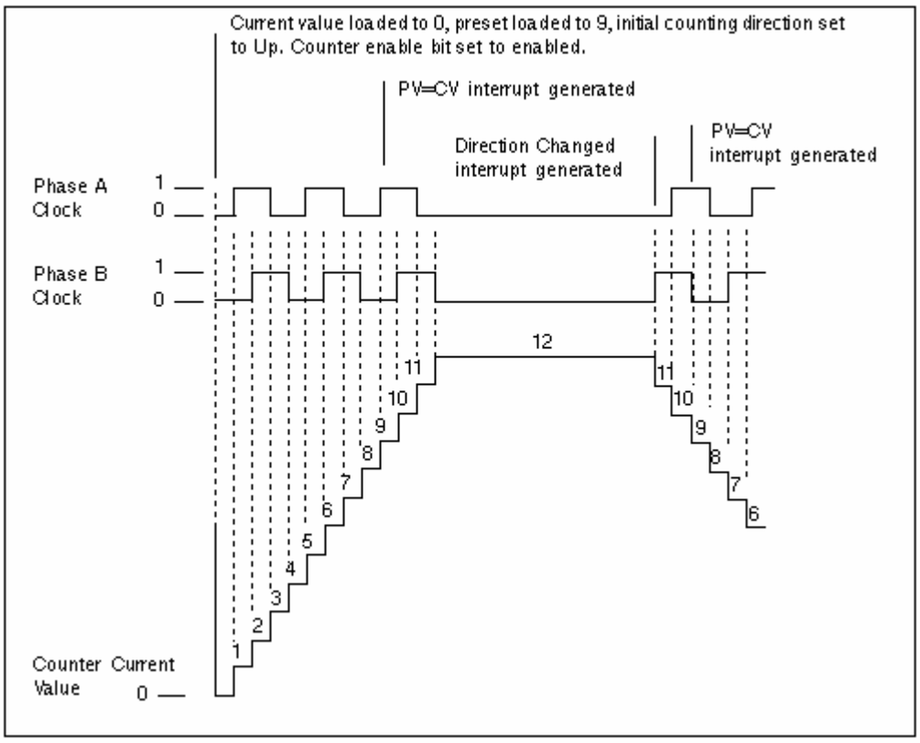
Mode 6,7,8: Dùng đếm 2 pha với 2 xung vào,1 xung dùng để đếm tăng và một xung đếm giảm
Mode 6: Chỉ đếm tăng giảm,không có Bit Start cũng như bit Reset

Mode 7: Đếm tăng giảm,có bit Reset nhưng không có bit Start
Mode 8: Đếm tăng giảm,Có Bit Start cũng như bit Reset để cho phép chọn bắt đầu đếm cũng như chọn thời điểm bắt đầu Reset.Các Bit Start cũng như Reset là các ngõ Input chọn từ bên ngoài.

Operation Example of Modes 9,10, and 11 (Quadrature 1x Mode)



Operation Example of Modes 9,10, and 11 (Quadrature 4x Mode)



Mode 9,10,11 : Dùng để đếm xung A/B của Encoder,có 2 dạng:
Dạng 1 (Quadrature 1x mode): Đếm tăng 1 khi có xung A/B quay theo chiều thuận,và giảm 1 khi có xung A/B quay theo chiều ngược.
Dạng 2 (Quadrature 4x mode): Đếm tăng 4 khi có xung A/B quay theo chiều thuận,và giảm 4 khi có xung A/B quay theo chiều ngược.
Mode 9: Chỉ đếm tăng giảm,không có Bit Start cũng như bit Reset

Mode 10: Đếm tăng giảm,có bit Reset nhưng không có bit Start

Mode 11: Đếm tăng giảm,Có Bit Start cũng như bit Reset để cho phép chọn bắt đầu đếm cũng như chọn thời điểm bắt đầu Reset.Các Bit Start cũng như Reset là các ngõ Input chọn từ bên ngoài.

Mode 12: Chỉ áp dụng với HSC0 và HSC3,HSC0 dùng để đếm số xung phát ra từ Q0.0,và HSC3 đếm số xung từ Q0.1 (Được phát ra ở chế độ phát xung nhanh) mà không cần đấu phần cứng,nghĩa là PLC tự kiểm tra từ bên trong.

HSC Mode	Description	Inputs			
	HSC0	I0.0	I0.1	I0.2	
	HSC1	I0.6	I0.7	I0.2	I1.1
	HSC2	I1.2	I1.3	I1.1	I1.2
	HSC3	I0.1			
	HSC4	I0.3	I0.4	I0.5	
	HSC5	I0.4			
0	Single-phase counter with **internal** direction control	Clock			
1		Clock		Reset	
2		Clock		Reset	Start
3	Single-phase counter with **external** direction control	Clock	Direction		
4		Clock	Direction	Reset	
5		Clock	Direction	Reset	Start
6	Two-phase counter with 2 clock inputs	Clock Up	Clock Down		
7		Clock Up	Clock Down	Reset	
8		Clock Up	Clock Down	Reset	Start
9	A / B phase quadrature counter	Clock A	Clock B		
10		Clock A	Clock B	Reset	
11		Clock A	Clock B	Reset	Start
12	Only HSC0 and HSC3 support mode 12. HSC0 counts the number of pulses going out of Q0.0. HSC3 counts the number of pulses going out of Q0.1.				

Bảng Mô tả chế độ đếm cũng như loại HSC ,quy định địa chỉ vào.

Căn cứ vào bảng trên để có thể chọn loại HSC cho từng ứng dụng phù hợp.

VD: Không thể sử dụng HSC0 cho Mode 5,Mode 8 cũng như Mode 11,vì các Mode này cần 4 chân Input trong khi đó HSC0 chỉ có 3 chân Input.

1 Số Bit được sử dụng để điều khiển các chế độ của HSC:

				HDEF Control Bits (used only when HDEF is executed)
HSC0	**HSC1**	**HSC2**	**HSC4**	**Description**
SM37.0	SM47.0	SM57.0	SM147.0	Active level control bit for Reset**: 0 = Reset active high 1 = Reset active low
	SM47.1	SM57.1		Active level control bit for Start**: 0 = Start active high 1 = Start active low
SM37.2	SM47.2	SM57.2	SM147.2	Counting rate selection for Quadrature counters: 0 = 4x counting rate 1 = 1x counting rate

- Bit chọn : Reset mức cao hay Reset mức thấp.
- Bit chọn : Start mức cao hay mức thấp.

- Bit chọn : Chế độ đếm 1x hay 4x

SM Control Bits for HSC Parameters

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Description
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	Counting direction control bit: 0 = count down 1 = count up
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	Write the counting direction to the HSC: 0 = no update 1 = update direction
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	Write the new preset value to the HSC: 0 = no update 1 = update preset
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	Write the new current value to the HSC: 0 = no update 1 = update current
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	Enable the HSC: 0 = disable the HSC 1 = enable the HSC

- Bit chọn : Chọn hướng đếm tăng hoặc hướng đếm giảm
- Bit chọn : Chọn cho phép Update hướng hay không Update
- Bit chọn : Chọn cho phép Update giá trị Preset hay không cho phép
- Bit chọn : Chọn cho phép Update giá trị hiện tại hay không cho phép
- Bit chọn : Cho phép HSC hoạt động hay ngưng hoạt động.

Value to be Loaded	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
New current	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158
New preset	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162

- Nạp giá trị hiện tại cho việc Update
- Nạp giá trị đặt cho việc Update

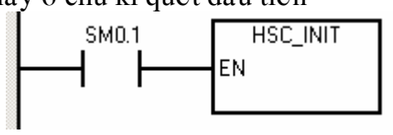
Status Bits for HSC0, HSC1, HSC2, HSC3, HSC4, and HSC5

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Description
SM36.0	SM46.0	SM56.0	SM136.0	SM146.0	SM156.0	Not used
SM36.1	SM46.1	SM56.1	SM136.1	SM146.1	SM156.1	Not used
SM36.2	SM46.2	SM56.2	SM136.2	SM146.2	SM156.2	Not used
SM36.3	SM46.3	SM56.3	SM136.3	SM146.3	SM156.3	Not used
SM36.4	SM46.4	SM56.4	SM136.4	SM146.4	SM156.4	Not used
SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	Current counting direction status bit: 0 = counting down; 1 = counting up
SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	Current value equals preset value status bit: 0 = not equal; 1 = equal
SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	Current value greater than preset value status bit: 0 = less than or equal; 1 = greater than

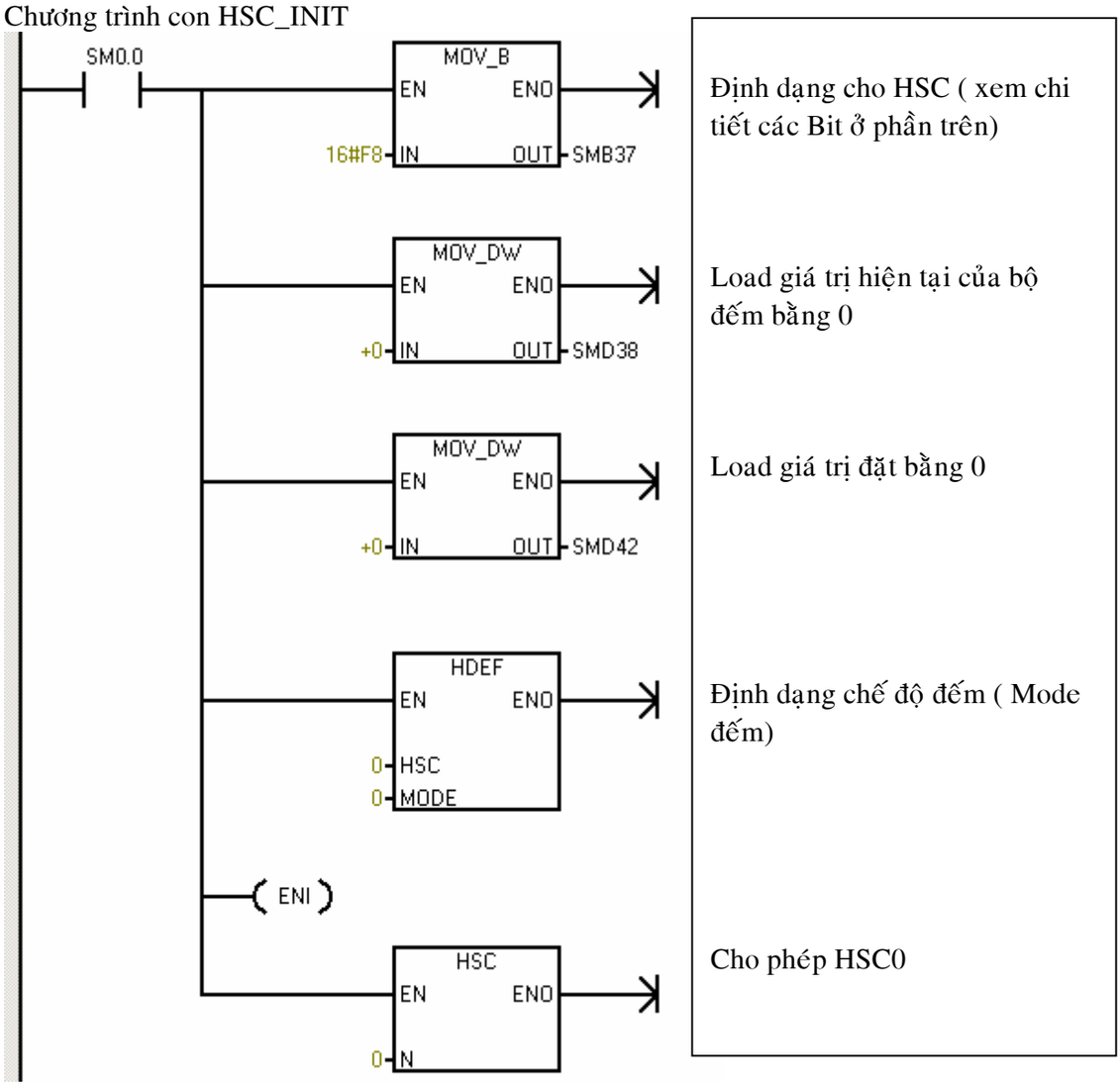
Các bit trạng thái mô tả:

- Hướng đếm
- Khi giá trị hiện tại bằng giá trị đặt
- Khi giá trị hiện tại lớn hơn hay bé hơn hoặc bằng giá trị đặt.

Ví dụ: Muốn sử dụng HSC0 cho việc đếm xung tốc độ cao,trước hết ta định dạng Wizard,sau khi định dạng Wizard,chương trình sẽ tạo ra 1 chương trình con,HSC_INIT,ta phải gọi chương trình này ở chu kì quét đầu tiên



Chương trình con HSC_INIT



Giá trị hiện tại của HSC0 sẽ nằm trong biến HC0

Ngoài ra ta còn có thể định dạng cho HSC với những chế độ ngắt khác nhau như:

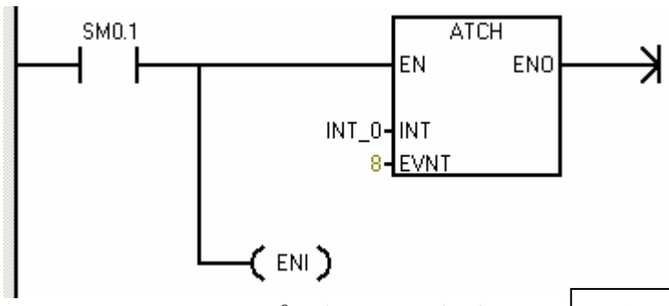
- Chương trình ngắt sẽ được thực thi khi giá trị HSC bằng với giá trị đặt
- Chương trình ngắt sẽ được thực thi khi hướng đếm thay đổi (thay đổi từ chiều đếm thuận sang đếm ngược,đếm tăng ,đếm giảm)
- Chương trình ngắt được thực thi khi Bit Reset được thực thi.

3/ Sử dụng chương trình ngắt:

Việc sử dụng chương trình ngắt là hết sức cần thiết trong việc lập trình S7_200,trong S7_200 có một số loại ngắt như sau:

Event Number	Interrupt Description	Priority Group	Priority in Group	Supported by CPU 221	222	224	224 XP 226 226 XM
8	Port 0: Rcv character	Comm.	0	✓	✓	✓	✓
9	Port 0: Xmt complete	(Highest)	0	✓	✓	✓	✓
23	Port 0: Rcv msg complete		0	✓	✓	✓	✓
24	Port 1: Rcv msg complete		1				✓
25	Port 1: Rcv character		1				✓
26	Port 1: Xmt complete		1				✓
19	PTO 0 complete interrupt		0	✓	✓	✓	✓
20	PTO 1 complete interrupt		1	✓	✓	✓	✓
0	Rising edge, I0.0	Discrete	2	✓	✓	✓	✓
2	Rising edge, I0.1	(Middle)	3	✓	✓	✓	✓
4	Rising edge, I0.2		4	✓	✓	✓	✓
6	Rising edge, I0.3		5	✓	✓	✓	✓
1	Falling edge, I0.0		6	✓	✓	✓	✓
3	Falling edge, I0.1		7	✓	✓	✓	✓
5	Falling edge, I0.2		8	✓	✓	✓	✓
7	Falling edge, I0.3		9	✓	✓	✓	✓
12	HSC0 CV=PV		10	✓	✓	✓	✓
27	HSC0 direction changed		11	✓	✓	✓	✓
28	HSC0 external reset/Zphase		12	✓	✓	✓	✓
13	HSC1 CV=PV		13			✓	✓
14	HSC1 direction changed		14			✓	✓
15	HSC1 external reset		15			✓	✓
16	HSC2 CV=PV		16			✓	✓
17	HSC2 direction changed		17			✓	✓
18	HSC2 external reset		18			✓	✓
32	HSC3 CV=PV		19	✓	✓	✓	✓
29	HSC4 CV=PV		20	✓	✓	✓	✓
30	HSC4 direction changed		21	✓	✓	✓	✓
31	HSC4 external reset/Zphase		22	✓	✓	✓	✓
33	HSC5 CV=PV		23	✓	✓	✓	✓
10	Timed interrupt 0	Timed	0	✓	✓	✓	✓
11	Timed interrupt 1	(Lowest)	1	✓	✓	✓	✓
21	Timer T32 CT=PT interrupt		2	✓	✓	✓	✓
22	Timer T96 CT=PT interrupt		3	✓	✓	✓	✓

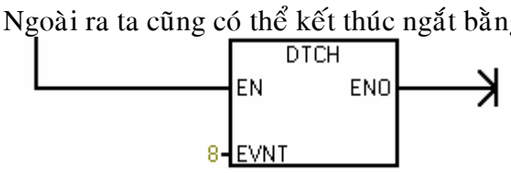
Mỗi loại ngắt trong S7_200 tương ứng với một sự kiện ngắt tương ứng.
Ví dụ: Sự kiện ngắt số 8 tương ứng với sự kiện khi việc nhận dữ liệu ở Port 0 xảy ra (Khi có dữ liệu truyền đến Port 0 thì chương trình ngắt mà liên kết với sự kiện ngắt số 8 được thực thi).



Liên kết sự kiện ngắt số 8 với chương trình ngắt INT_0 (Khi xảy ra sự kiện số 8 thì chương trình INT_0 được thực thi)

Cho ph p ngắt (**ENI**)

Ngoài ra ta cũng có thể kết thúc ngắt bằng lện



Kết thúc sự kiện ngắt số 8,sự kiện ngắt số 8 sẽ được cho phép lại khi có lệnh **ENI**

Ngắt thời gian Timer_0,ngắt thời gian Timer_1:
Để định dạng cho việc ngắt thời gian Timer 0 cũng như Timer 1,thì ngoài việc thiết lập sự kiện ngắt cho việc định dạng ngắt Timer_0 hay Timer_1,ta còn phải chọn thời gian ngắt
Thời gian ngắt sẽ được chọn như sau:

S7-200 Symbol Name	SM Addr.	Timed Interrupt Interval in milliseconds
Time_0_Intrvl	SMB34	Timed Interrupt 0: Time interval value (in 1 ms increments from 1 ms to 255 ms**).
Time_1_Intrvl	SMB35	Timed Interrupt 1: Time interval value (in 1 ms increments from 1 ms to 255 ms**).

Ô nhớ SMB34 dùng cho việc định dạng thời gian ngắt cho Timer_0
Ô nhớ SMB35 dùng cho việc định dạng thời gian ngắt cho Timer_1
Ví Dụ: Định dạng cho SMB34=10 :Cứ 10ms thì chương trình ngắt sẽ được thực thi 1 lần,và nó chỉ chấm dứt khi tín hiệu không cho phép ngắt được thực thi.

4/ Đọc tín hiệu Analog:

Tín hiệu Analog là các tín hiệu tương tự (0 – 10VDC,hoặc 4-20mA.....),Hầu hết các ứng dụng của chương trình PLC Siemens nói riêng hay các ứng dụng khác đều cần phải đọc các tín hiệu analog.Tín hiệu analog có thể là tín hiệu từ các cảm biến đo khoảng cách,cảm biến áp suất,cảm biến đo trọng lượng.....
Các bước đọc tín hiệu Analog:

a/Đọc tín hiệu analog từ Modul EM231:

Các tín hiệu có thể đọc được từ Modul EM231(tuỳ thuộc việc chọn các Switch trên modul):

- Tín hiệu đơn cực (Tín hiệu điện áp): 0-10VDC, 0-5VDC
- Tín hiệu lưỡng cực (tín hiệu điện áp): -5VDC – 5VDC, -2.5VDC – 2.5VDC
- Tín hiệu dòng điện :0 – 20mA (có thể đọc được 4-20mA)

Tín hiệu Analog sẽ được đọc vào AIW0,AIW2 tương ứng,tuỳ thuộc vào vị trí của tín hiệu đưa vào modul
Modul EM231 có 4 ngõ vào Analog,do vậy vị trí các ngõ vào tương ứng là:
AIW0,AIW2,AIW4,AIW6
Tín hiệu analog là tín hiệu điện áp ,tuy nhiên giá trị mà AIW đọc vào không phải là giá trị điện áp ,mà là giá trị đã được quy đổi tương ứng 16bit.
Trường hợp đơn cực : Giá trị từ 0 – 64000 tương ứng với (0-10V,0-5V hay 0-20mA)

Trường hợp lưỡng cực : Giá trị từ -32000 – 32000 tương ứng với (-5VDC – 5VDC hay -2.5VDC – 2.5VDC).

Vídụ :

- Trường hợp đơn cực: giá trị đọc vào của AIW0 = 32000,khi đó giá trị điện áp tương ứng là : (32000x10VDC/ 64000) = 5VDC (Tầm chọn 0 – 10VDC)
- Trường hợp lưỡng cực : Giá trị đọc vào của AIW0 = 16000,khi đó giá trị điện áp tương ứng là : (16000x5VDC/32000) =2.5VDC (Tầm đo -2.5VDC – 2.5VDC)

Do vậy căn cứ vào giá trị đọc vào của AIW ta có thể dùng quy tắc "tam suất",từ đó có thể tính được giá trị điện áp tương ứng.Từ giá trị điện áp ta có thể suy ra giá trị mong muốn.

- Thông thường các tín hiệu Analog đọc vào bao giờ người sử dụng cũng mong muốn đọc được chính giá trị mong muốn (Ví dụ: giá trị khối lượng trong đọc đầu cân Loadcell, giá trị áp suất trong đọc tín hiệu từ cảm biến áp suất.....)
- Phương pháp đọc Analog trong trường hợp này ta sẽ không cần quan tâm nhiều đến chế độ đơn cực hay lưỡng cực,mà chỉ cần xác định được 2 điểm,từ đó lập được phương trình đường thẳng (Giá trị mong muốn đọc theo AIW)
- Ví dụ: Để đọc khối lượng từ đầu cân :Ta xây dựng hàm Khối lượng theo AIW(là tín hiệu đọc vào)
- Bước 1: Ta cần xác định 2 điểm:

  Điểm 1: Ta online trên máy tính,đọc giá trị AIW0 là x1,trong trường hợp ở điểm 1 (Điểm 1 là điểm ta đặt quả cân chuẩn 1:có khối lượng m1 lên bàn cân) ,Tương tự ta có thể xác định được điểm 2 (tương ứng x2 và m2).

  Từ đó ta có 2 điểm : Điểm 1 (x1,m1) , Điểm 2 (x2,m2).

  Phương trình đường thẳng đi qua 2 điểm 1,2 có dạng:

  (X-X1/X2-X1) = (Y-Y1/Y2-Y1),Từ đó rút Y theo X

  Đó chính là phương trình khối lượng theo AIW.

- Ví dụ cụ thể: Điểm 1 (0,0), điểm 2 (32000,1000)

Phương trình lập:

(X-0/32000-0) = (Y-0/1000-0) Từ đó suy ra:

Y= 1xX/ 32

Vậy : Khối lượng = AIW / 32

b/Xuất tín hiệu analog qua modul EM232:

Các tín hiệu có thể xuất ra Modul EM232(tuỳ thuộc việc chọn các Switch trên modul):

- Tín hiệu đơn cực (Tín hiệu dòng điện): 0-20mA
- Tín hiệu lưỡng cực (tín hiệu điện áp): -10VDC – 10VDC

Tín hiệu 0 -20mA tương ứng với giá trị 0 – 32000

Tín hiệu -10VDC – 10VDC tương ứng -32000 – 32000

Giá trị xuất ra Modul EM232 được đưa vào ô nhớ AQW tương ứng.

c/ Modul EM235:

- Các tín hiệu có thể đọc được thông qua Modul EM235 (Tuỳ theo Switch chọn trên Modul):

Đơn cực : 0 – 50mV , 0 – 100mV , 0 – 500mV , 0 – 1V , 0 – 5VDC , 0 – 20mA , 0 – 10VDC.

Lưỡng cực : +-25mV , +-50mV , +-100mV , +-250mV , +-500mV , +-1VDC ,+-2.5VDC , +-5VDC ,+-10VDC

Giá trị tương ứng cho chế độ đơn cực : Từ 0 – 64000

Giá trị tương ứng cho chế độ lưỡng cực : -32000 – 32000

- Ngoài ra Modul EM235 còn có 2 Ngõ ra Analog output tương ứng : +-10VDC, 0 – 20mA

5/ Xuất dữ liệu và nhận dữ liệu qua Port giao tiếp (Port 0,Port 1):

S7_200 thông thường cho phép ta sử dụng các Port giao tiếp để giao tiếp với các thiết bị bên ngoài,Trường hợp CPU sử dụng có 2 Port giao tiếp thì ta cũng có thể sử dụng cả 2 Port giao tiếp để có thể giao tiếp với các thiết bị bên ngoài (Như : Giao tiếp 485 với đầu cân,giao tiếp với các đầu đo điện.....).

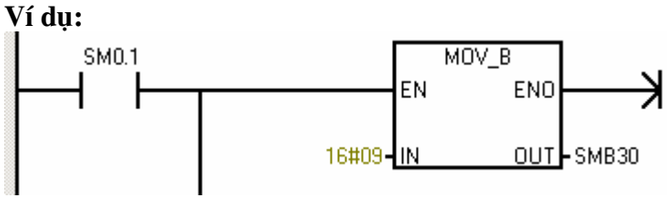
a/Xuất dữ liệu ra Port giao tiếp:

Để thực hiện việc xuất dữ liệu ra Port giao tiếp ta thực hiện như sau:

Bước 1: Định dạng cho việc giao tiếp qua Port (Tốc độ Baud,số Bit dữ liệu...) thông qua 2 byte **SMB30** (cho Port 0),**SMB130** (cho Port 1)

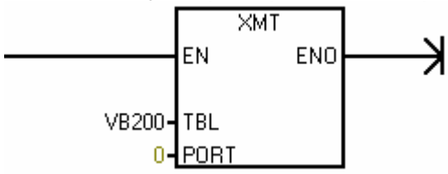
S7-200 Symbol Name	SM Address Port 0	SM Address Port 1	Bit Format	Bit Format MSB 7 … LSB 0 = p p d b b b m m
P0_Config	SMB30			
P1_Config		SMB130		
	SM30.6 – SM30.7	SM130.6 – SM130.7	pp:	0 0 = No parity 0 1 = Even parity 1 0 = No parity 1 1 = Odd parity
	SM30.0 – SM30.1	SM130.0 – SM130.1	d:	0 = 8 data bits per character 1 = 7 data bits per character
	SM30.2 – SM30.4	SM130.2 – SM130.4	bbb:	0 0 0 = 38,400 bps 0 0 1 = 19,200 bps 0 1 0 = 9,600 bps 0 1 1 = 4,800 bps 1 0 0 = 2,400 bps 1 0 1 = 1,200 bps 1 1 0 = 115,200 bps* 1 1 1 = 57,600 bps* * Requires S7-200 CPU version 1.2 or later
P0_Config_0	SM30.0		mm:	0 0 = Point-to-Point Interface protocol (PPI/slave mode) 0 1 = Freeport protocol 1 0 = PPI/master mode 1 1 = Reserved (defaults to PPI/slave mode) **Note:** When you select code mm = 10 (PPI master), the CPU becomes a master on the network and allows the NETR and NETW instructions to be executed. Bits 2 through 7 are ignored in PPI modes.
P1_Config_0		SM130.0		
	SM30.1	SM130.1		

Ví dụ:



Định dạng giao tiếp Port 0 (No Parity, 8 data bits per character,tốc độ baud 9600, chế độ Freeport protocol)

Bước 2: Thực hiện việc xuất dữ liệu



Xuất dữ liệu qua Port 0.
TBL = VB200

Ý nghĩa TBL: Số Byte được cho phép truyền đi

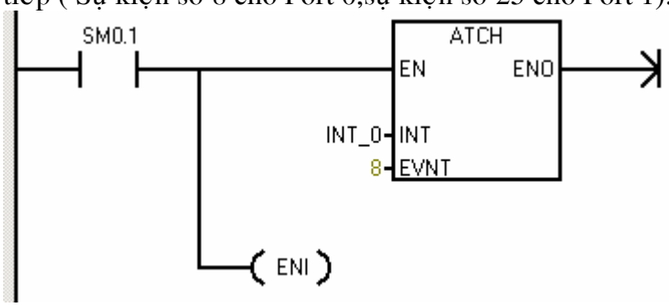
Vị trí Byte truyền bắt đầu bằng TBL + 1

Ví dụ: Ở lệnh trên Nếu VB200 = 5 ,lệnh trên sẽ thực hiện việc truyền 5 Byte (VB201,VB202,VB203,VB204,VB205)

b/Nhận dữ liệu qua Port giao tiếp:

Để thực hiện việc nhận dữ liệu qua Port giao tiếp ,trước hết ta cũng phải định dạng giao thức cho việc giao tiếp giống như phần xuất dữ liệu ra Port giao tiếp.

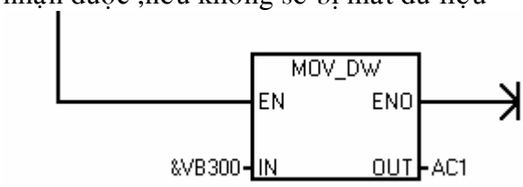
Để thực hiện việc nhận dữ liệu,ta thực hiện việc liên kết sự kiện nhận dữ liệu qua Port giao tiếp (Sự kiện số 8 cho Port 0,sự kiện số 25 cho Port 1).



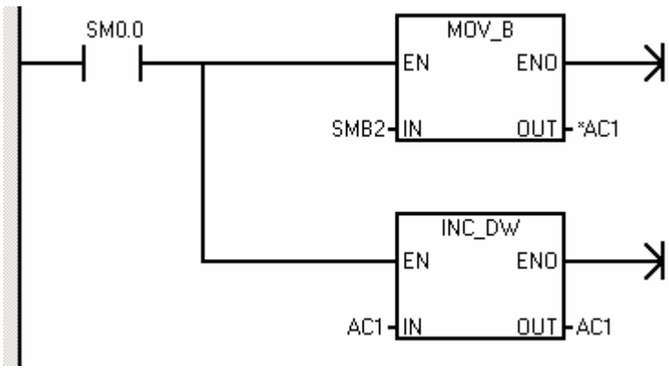
Liên kết sự kiện ngắt số 8 với chương trình ngắt INT_0 (sự kiện nhận dữ liệu qua Port giao tiếp)

Cho phép ngắt (ENI)

SMB2 là byte chứa dữ liệu nhận được từ Port 0 và Port 1 trong quá trình giao tiếp,Nghĩa là dữ liệu nhận được sẽ đẩy vào SMB2,do vậy trong chương trình ngắt ta phải lưu lại dữ liệu nhận được ,nếu không sẽ bị mất dữ liệu



Con trỏ trỏ tới ô nhớ VB300



Dữ liệu sau khi nhận được đưa vào con trỏ AC1 (tức là đưa vào VB300),sau đó tăng con trỏ lên 1 Con trỏ đang ở VB301

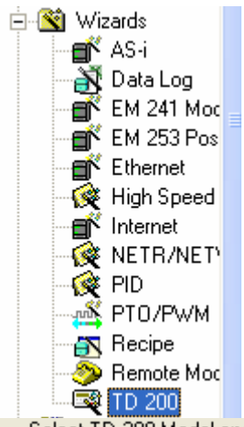
Tăng con trỏ lên 1

Sau đó sẽ gọi chương trình con để xử lí chuỗi dữ liệu vừa mới nhận đó.

6/Kết nối TD200:

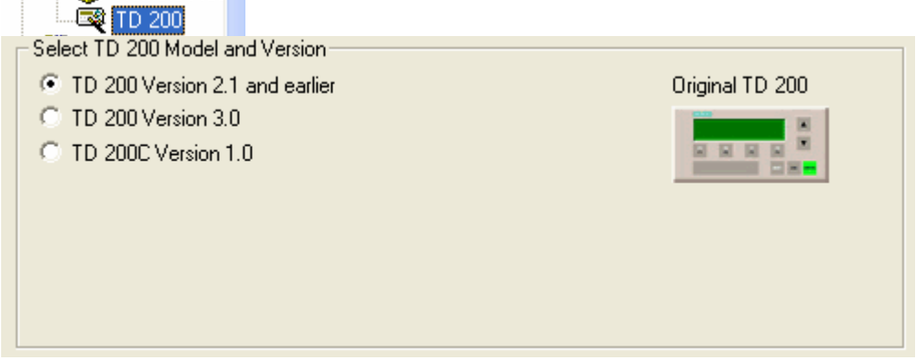
TD200 là màn hình giao tiếp với CPU S7_200,màn hình TD200 là màn hình dạng Text cho phép người sử dụng thay đổi dữ liệu,cảnh báo khi gặp sự cố.....Tuy nhiên loại màn hình này không có phần mềm chuyên biệt cho việc lập trình,mà việc liên kết với nó phải thông qua chương trình S7_200, Nghĩa là để có thể liên kết với TD200 ,Trong chương trình S7_200 ta phải thực hiện việc định dạng bằng Wizard.

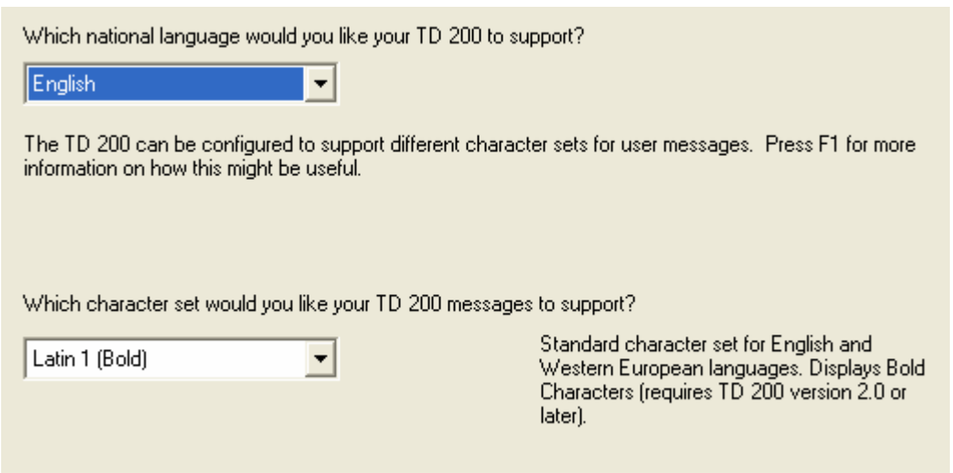
Các bước thực hiện Wizard:



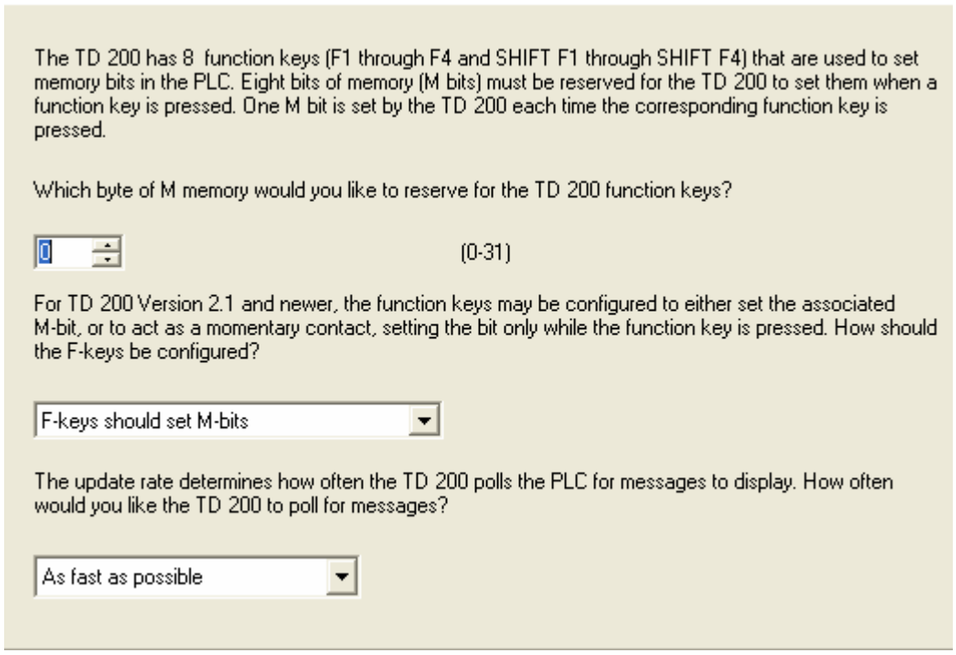
Chọn Wizard TD200,bằng cách Double click vào TD 200,rồi chọn next

Sau đó chọn loại TD200 cần dùng (TD200 V2.1,TD 200 V3.0 ,TD200C)





Chọn ngôn ngữ và loại Font chữ cho phù hợp. Sau đó tiếp tục chọn next,để qua trang kế tiếp.

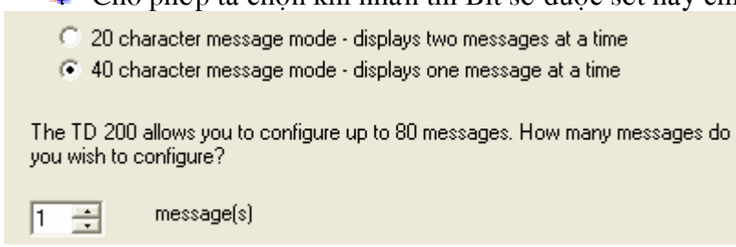


TD200 có 8 nút nhấn từ F1 – F4 , SHIFT F1 – SHIFT F4,

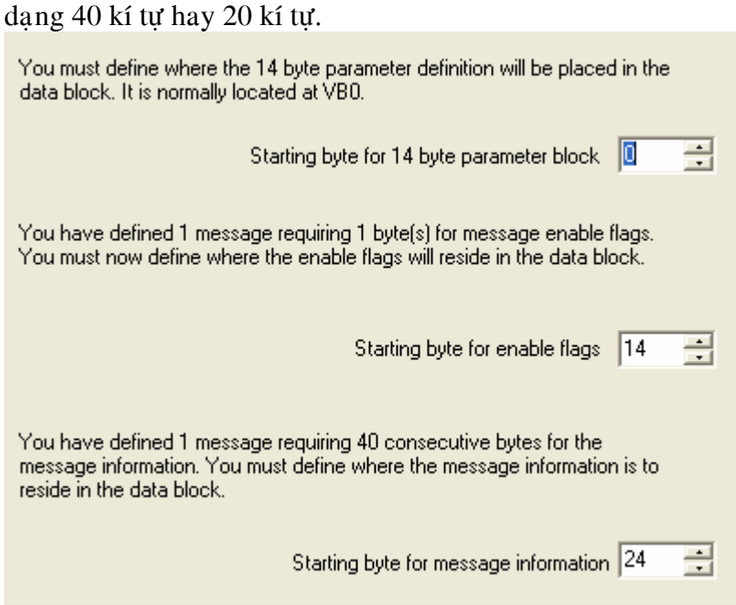
- Các nút nhấn này cho phép ta chọn địa chỉ Byte cho 8 nút nhấn này.

Mặc định (Byte M0 ,khi đó : F1:M0.0 , F2:M0.1.....SHIFT F4 : M0.7)

- Cho phép ta chọn khi nhấn thì Bit sẽ được set hay chỉ ON/OFF



TD200 cho phép ta định dạng khoảng tối đa 80 màn hình ,mỗi màn hình cho phép ta định dạng 40 kí tự hay 20 kí tự.



Định dạng vị trí bắt đầu cho 14 Byte dành cho vùng Data Block (mặc định VB0)

Định dạng Byte dành cho Bit cho phép của trang màn hình cần hiển thị (Mặc định VB14)

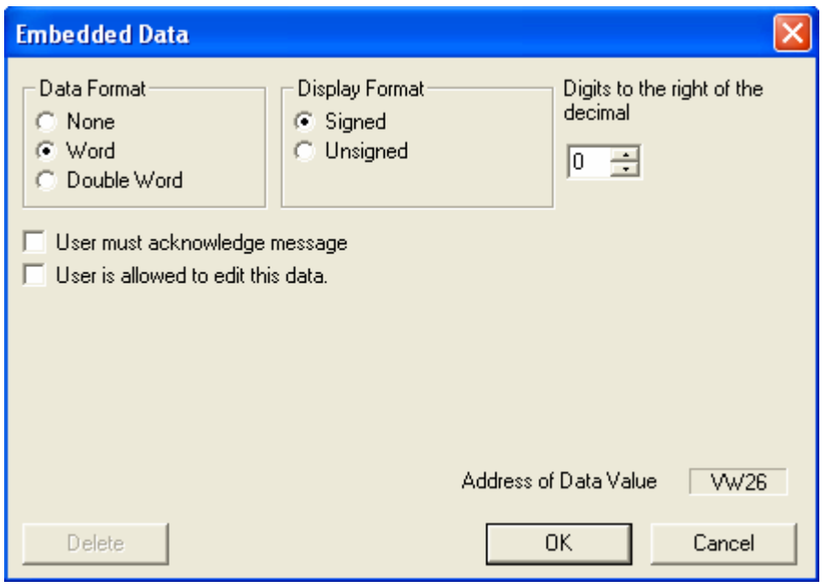
Vị trí Byte đầu tiên cho 40 Byte dữ liệu kí tự của màn hình. (Mặc định VB24)

Bit cho phép của màn hình là V14.7
Địa chỉ Byte bắt đầu VB24: Do vậy
VB24 = “T” ,VB25=”A" VB26 = “I” .....

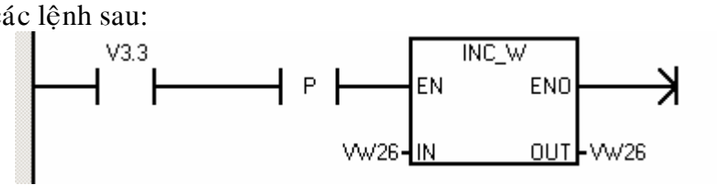
Sau đó chọn Finish cho việc hoàn thành định dạng Wizard,khi đó sau khi Download chương trình xuống PLC thì PLC sẽ hiểu TD200 khi CPU liên kết với màn hình.
Ngoài việc định dạng Wizard ta còn cần phải viết lệnh trong chương trình S7_200 để có thể tăng hoặc giảm các dữ liệu trong S7_200
Ngoài ra một số nút nhấn tăng giảm:

Nút tăng : V3.3
Nút giảm: V3.2
Nút Enter : V3.1

Ngoài ra ta còn có thể chọn :



Dữ liệu dạng Word,hoặc Double Word cho các ô nhớ tương ứng.
Ví dụ: Khi nhấn nút tăng,muốn dữ liệu tăng lên 1,thì trong chương trình PLC ta phải thực hiện các lệnh sau:

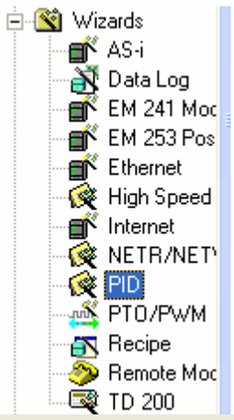


7/ Điều khiển PID:

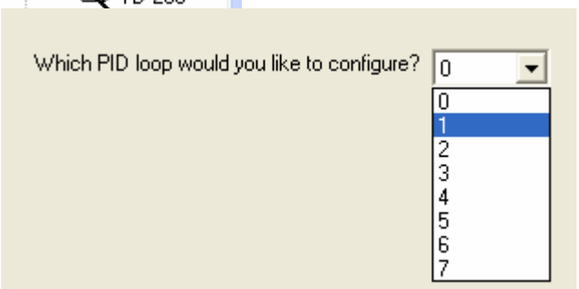
Một hệ thống điều khiển mong muốn : Đạt giá trị xác lập với thời gian và sai số nhỏ nhất có thể.Để có thể đạt được yêu cầu này thì trước tiên hệ thống điều khiển phải là hệ thống điều khiển vòng kín (Nghĩa là phải có vòng hồi tiếp cho hệ thống điều khiển).
Yêu cầu của hệ thống:

- Giá trị đặt (Là giá trị do người sử dụng mong muốn)
- Giá trị đo (Giá trị đo về từ cảm biến)

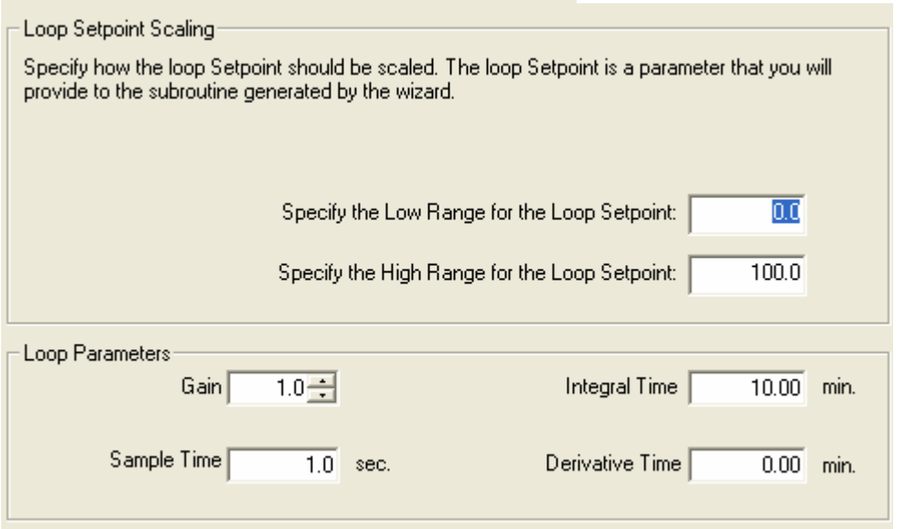
Từ sự chênh lệch sai số giữa giá trị đặt và giá trị đo từ đó có phương pháp hiệu chỉnh (điều khiển kịp thời)
Các bước thực hiện Wizard cho việc điều khiển PID:



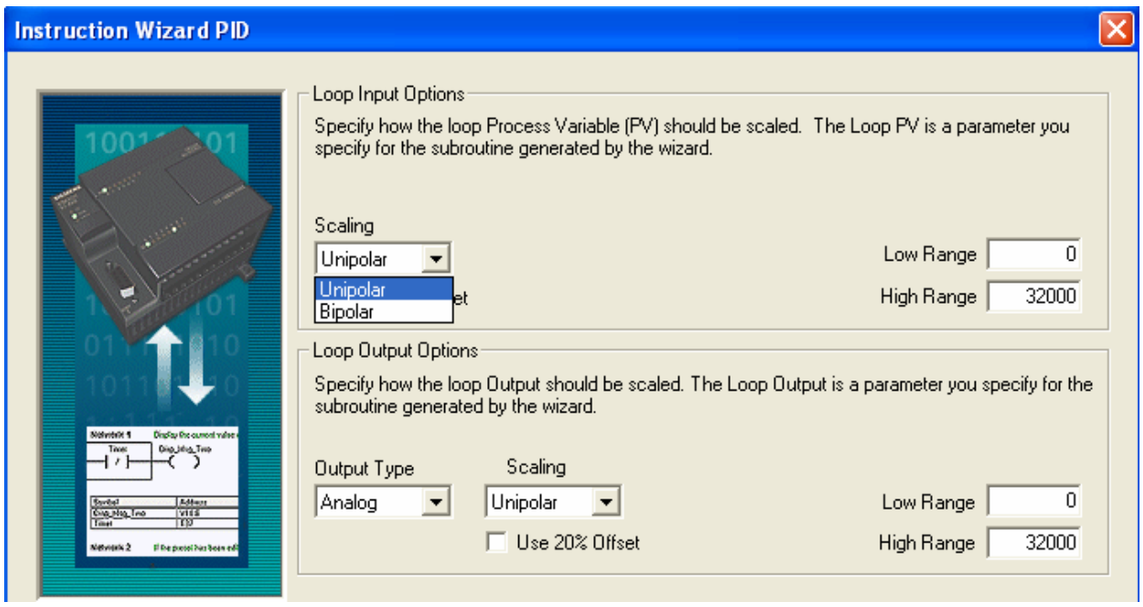
Chọn Wizard PID,Double click vào PID để chọn việc định dạng cho Wizard PID



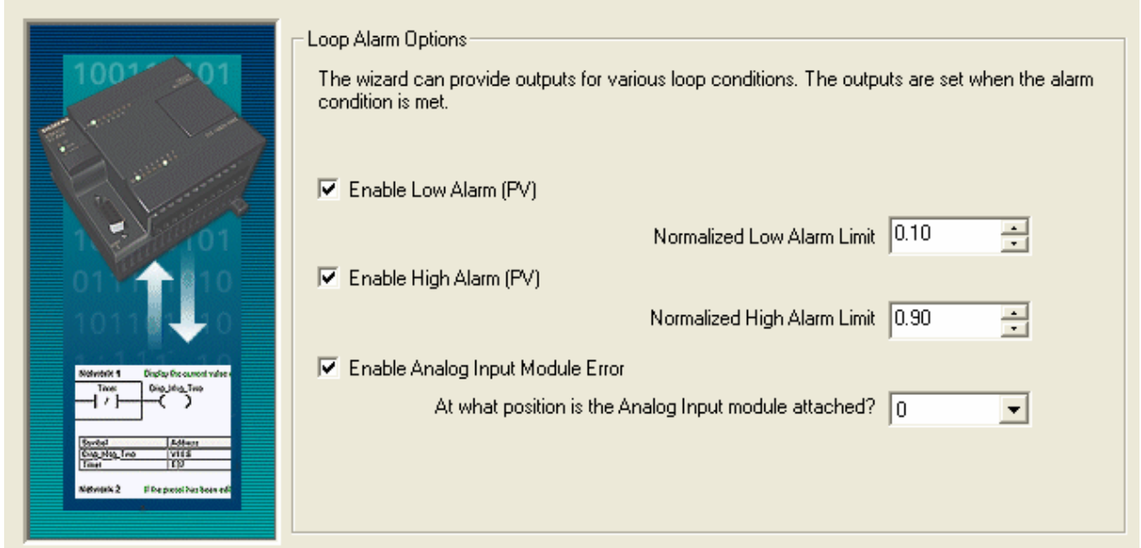
Chọn số vòng (LOOP) cho việc điều khiển PID,số vòng tối đa 3



Chọn giá trị nhỏ nhất cũng như lớn nhất của giá trị Setpoint,chọn hệ số Gain,Sample time ,Integral Time,Derivative Time. Các giá trị này phải chọn phù hợp thì thời gian xác lập mới nhanh,và sai số tốt.



- Chọn loại tín hiệu đơn cực hay lưỡng cực
- Chọn loại tín hiệu điều khiển,tín hiệu analog hay tín hiệu Digital
- Chọn có dùng Off set 20% hay không



- Chọn các tín hiệu cho phép Alarm mức thấp (Giá trị chọn tương ứng)
- Chọn tín hiệu cho phép Alarm mức cao (Giá trị chọn tương ứng)
- Chọn chế độ Alarm lỗi

Chọn next và Finish để kết thúc việc định dạng wizard.

Chương trình sẽ tạo ra 2 chương trình con PID0_INIT và PID_EXE,ta có thể sử dụng 2 chương trình con này trong chương trình ứng dụng cho phù hợp.

Chương trình con PID0_INIT được thực hiện trong chương trình chính,còn chương trình PID_EXE sẽ được thực hiện khi quá trình PID đã được xác lập.

8/ Sử dụng Memory Catridge:

S7_200 có thêm một công cụ thật lí thú đó là Memory Catridge

Memory Catridge là option gắn thêm cho CPU S7_200 khi người dùng có yêu cầu những ứng dụng liên quan đến thiết bị này,thông thường thì tại mỗi CPU vị trí của Memory Catridge sẽ được che kín bởi 1 thiết bị phụ trợ tránh trường hợp bụi xâm nhập vào,mỗi khi cần dùng thêm thiết bị Memory Catridge thì ta thay thế thiết bị che chắn đó bằng Memory Catridge.

Các công dụng có thêm của Memory Catridge:

- Mở rộng dung lượng nhớ cho chương trình
- Thiết lập Recipe
- Thiết lập Data Log
- Lưu trữ chương trình khi cần thiết

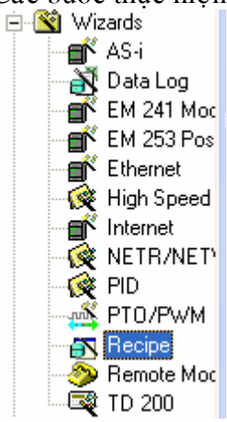
a/Mở rộng dung lượng bộ nhớ chương trình: Mỗi CPU chỉ có một dung lượng cho bộ nhớ chương trình nhất định,trong trường hợp chương trình quá dài,vượt quá dung lượng của bộ nhớ chương trình ,chỉ còn một cách duy nhất là sử dụng Memory Catridge để chia sẻ bớt chương trình cần thiết.

Ví dụ: Dung lượng bộ nhớ chương trình thông thường khoảng 8KB,nếu chương trình ứng dụng có dung lượng lớn hơn 8KB ta phải sử dụng Memory Catridge gắn thêm.(Memory Catridge có thể là 64KB,128KB,256KB...

b/Thiết lập Recipe: Chương trình S7_200 cho phép ta thiết lập những công thức có sẵn trong chương trình S7_200,chương trình này sẽ được lưu trong Memory Catridge khi Download.

Ứng dụng này thường được dùng trong những hệ thống cần sử dụng nhiều công thức có sẵn biết trước mà không cần phải sử dụng màn hình nhập từ bên ngoài.

Các bước thực hiện thiết lập Recipe:



Chọn Wizard Recipe bằng cách Double click vào Recipe

Sau đó chọn Next

	Field Name	Data Type	Default Value	Comment
1	Cat	WORD	0	
2	Da	WORD	0	
3	Ximang	WORD	0	
4	Nuoc	WORD	0	
5	Phugia	WORD	0	
6				

Click 'Next' to edit the recipes for this configuration.

Chọn các mục cần thiết cho việc thiết lập công thức như : Cát,Đá,Ximăng,Nước,Phụgia

There are currently 5 recipe(s) defined for this configuration.

	Field Name	Data Type	DEF0_RCP1	DEF0_RCP2	DEF0_RCP3	DEF0_RCP4	DEF0_RCP0
1	Cat	WORD	1000	1200	900	1500	1600
2	Da	WORD	1500	1600	1400	2000	2100
3	Ximang	WORD	200	250	180	290	300
4	Nuoc	WORD	100	120	80	140	150
5	Phugia	WORD	10	12	8	14	16

Thiết lập các công thức cho ứng dụng cần thiết

Ví dụ: Công thức 1:

Cát = 1600Kg

Đá = 2100Kg

Ximăng = 300Kg

Nước = 150Kg

Phụ gia = 16Kg

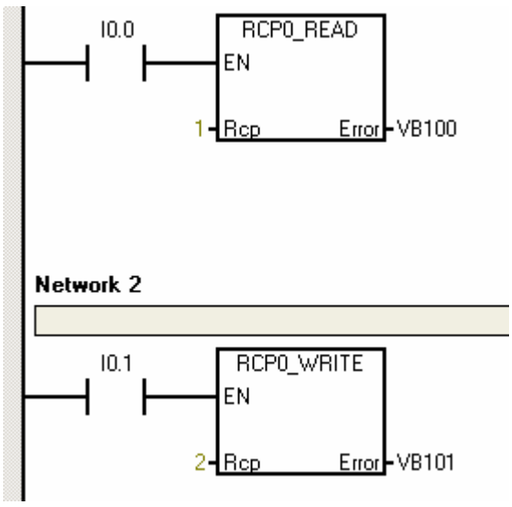
Sau đó chọn Next,rồi chọn Finish

Khi đó chương trình sẽ tự động tạo ra 2 chương trình con RCP0_Read và RCP0_Write,ta phải sử dụng 2 chương trình này trong chương trình ứng dụng tương ứng.

RCP0_Read : dùng để đọc công thức từ PLC

RCP0_Write: dùng để viết các công thức lên PLC.

Ví dụ:



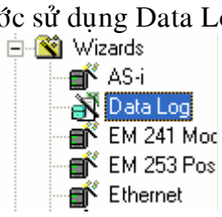
Khi có I0.0,chương trình sẽ đọc công thức 1,byte lỗi sẽ được lưu vào VB100

Khi có I0.1 chương trình sẽ viết công thức 2 ,Byte lỗi sẽ được đưa vào VB101

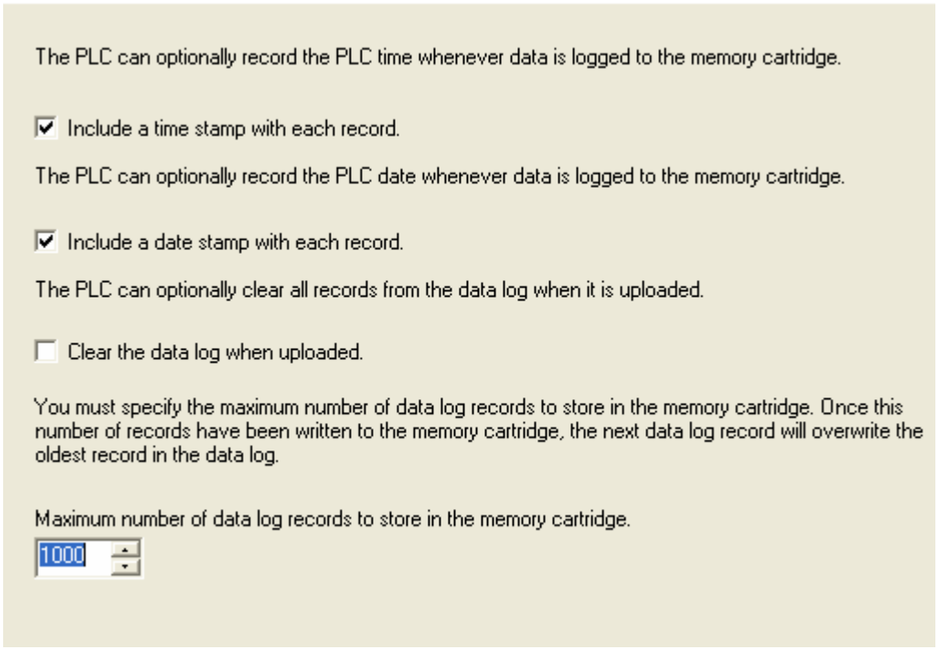
c/Thiết lập Data Log:

Chương trình cho phép ta thiết lập Data Log trong trường hợp người sử dụng muốn theo dõi sự hoạt động của một hệ thống theo thời gian,khi đó người sử dụng phải có Option Memory Catridge ,đồng thời phải sử dụng công cụ Data Log.Để đọc được Data Log ta phải sử dụng S7-200 Explorer,chương trình này sẽ đọc Data Log tương ứng có trong Memory Catridge

Các bước sử dụng Data Log:



Chọn Data Log trong Wizard bằng cách Double click vào Data Log
Sau đó chọn Next



Chọn các mục tương ứng:

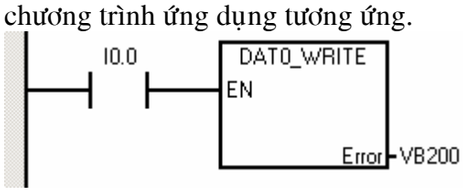
- Bao gồm kèm theo thời gian cho mỗi Record

- Bao gồm kèm theo ngày tháng cho mỗi Record
- Xoá Data Log khi Upload
- Chọn Số lượng Record được lưu trữ tối đa

	Field Name	Data Type	Comment
1	Nuoc1	WORD	
2	Nuoc2	WORD	
3	Nuoc3	WORD	
4			
5			

Click 'Next' to allocate the memory for this data log configuration.

Chọn Tên và loại dữ liệu cho việc thực hiện Data Log
Chọn Next và Finish cho việc hoàn thành định dạng Wizard.
Chương trình sẽ tạo ra chương trình con DAT0_Write,ta sẽ gọi chương trình con này trong chương trình ứng dụng tương ứng.



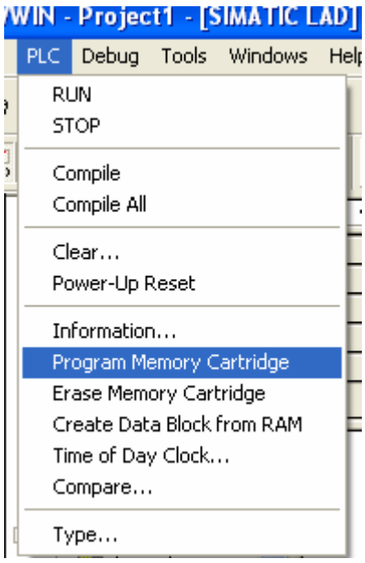
Khi có I0.0,chương trình sẽ thực thi việc ghi Data Log,Byte lỗi là VB200

- Khi thực hiện việc sử dụng Memory Catridge ,ta phải chọn mục Download to Memory Catridge khi Download chương trình ứng dụng.

Options

☑ Program Block	To : PLC
☑ Data Block	To : PLC
☑ System Block	To : PLC
☑ Recipes	To : Memory Cartridge
☑ Data Log Configurations	To : Memory Cartridge

e/Lưu chương trình ứng dụng:
Khi sử dụng Memory Catridge ta có thể lưu chương trình ứng dụng khi cần thiết.Việc lợi thế của lưu chương trình ứng dụng là ta không cần phải download lại chương trình ứng dụng khi thay thế CPU (trong trường hợp CPU bị hư hỏng),mà ta chỉ cần thay thế CPU rồi gắn Memory Catridge vào CPU,CPU sẽ tự động cập nhật toàn bộ chương trình đã có sẵn trong Memory Catridge.
Để thực hiện được điều này thì khi DownLoad chương trình xuống CPU ta phải thực hiện thêm 1 bước nữa là lưu chương trình vào trong Memory Catridge.



Ta vào PLC ,chọn Program Memory Catridge (Trong trương hợp phải Online với giữa máy tính và PLC) Hoặc ta có thể xoá Memory Catridge bằng cách chọn Erase Memory Catridge.

8/Một số ô nhớ đặc biệt sử dụng trong S7_200:

- **SMB0 : Status Bits**
  - **SM0.0 :** Bit này luôn luôn ON
  - **SM0.1** : Bit này ON trong chu kì quét đầu tiên của chương trình,hoặc ON khi bật từ Stop sang Run
  - **SM0.2** : Bit này ON trong 1 chu kì quét nếu dữ liệu của ô nhớ có khả năng nhớ bị mất.
  - **SM0.3 :** Bit này ON trong 1 chu kì quét khi có điện và đang ở trạng thái RUN
  - **SM0.4** : Bit này xung nhịp chu kì 1 phút, 30S ON, 30S OFF
  - **SM0.5** :Bit này xung nhịp chu kì 1giây , 0.5s ON , 0.5S OFF
  - **SM0.6** :Bit này xung nhịp chu kì 1 vòng quét , Vòng quét này ON,vòng Quét kế tiếp OFF.
  - **SM0.7** :Bit phản ánh vị trí của Switch chế độ : On khi Switch ở chế độ **RUN**, OFF khi Switch ở chế độ **TERM**
- **SMB1 : Status Bits**
  - **SM1.0 :** Bit này ON khi việc thực thi lệnh cho kết quả là Zero
  - **SM1.1** : Bit này ON khi kết quả thu được bị tràn ô nhớ hoặc kết quả thu được không hợp lệ.
  - **SM1.2** : Bit này ON khi kết quả thu được là số âm.
  - **SM1.3 :** Bit này ON khi thực hiện phép chia cho số 0
  - **SM1.4** : Bit này ON khi việc thêm dữ liệu vào một bảng bị tràn.
  - **SM1.5** :Bit này ON khi lệnh LIFO và FIFO thực hiện việc đọc từ 1 bảng trống.
  - **SM1.6** :Bit này ON khi lệnh chuyển đổi không phải số BCD sang số BIN được thực thi.
  - **SM1.7** :Bit Này ON khi việc thực hiện chuyển đổi số ASCII sang số Decimal không hợp lệ.
- **SMB2 : Nhận dữ liệu thông qua cổng FreePort**

Dữ liệu được nhận qua cổng FreePort sẽ được đưa vào SMB2

- **SMB3 : Lỗi Parity thông qua cổng Freeport**
  - SM3.0 : Parity lỗi từ Port 0 hay Port 1 (0 = No Error , 1= Error được phát hiện)

- **SMB4 : Tràn dữ liệu**
  - **SM4.0 :** Bit này ON khi ngắt giao tiếp bị tràn.
  - **SM4.1** : Bit này ON khi ngắt ngõ vào bị tràn.
  - **SM4.2** : Bit này ON khi Ngắt thời gian bị tràn.
  - **SM4.3 :** Bit này ON khi thời gian thực hiện chương trình gặp vấn đề.
  - **SM4.4** : Bit này ON khi việc ngắt được cho phép.
  - **SM4.5** :Bit này ON khi việc truyền dữ liệu qua Port 0 không được thực thi
  - **SM4.6** : Bit này ON khi việc truyền dữ liệu qua Port 1 không được thực thi
  - **SM4.7** :Bit Này ON khi một số giá trị bị ép .(Bị Force)
- **SMB5 : Trạng thái I/O**
  - **SM5.0 :** Bit này ON khi có ngõ vào ra bị lỗi.
  - **SM5.1** : Bit này ON khi quá nhiều I/O được nối vào I/O Bus
  - **SM5.2** : Bit này ON khi quá nhiều Analog I/O được nối vào I/O Bus
  - **SM5.3 :** Bit này ON khi quá nhiều Modul I/O được kết nối vào I/O Bus.
- **SMB6 : ID của CPU**
  - **SM6.4 – SM6.7:**

    0000 : CPU 222

    0010 : CPU 224

    0110 : CPU 221

    1001 : CPU 226/CPU 226XM
- **SMB8 đến SMB21 : I/O Modul ID và lỗi thanh ghi**

❖ **Định dạng Modul ID:**

m	t	t	a	i	i	q	q

m: Modul tồn tại 0 : tồn tại

1 : Không tồn tại

tt : Loại Modul

00 : Không phải Modul I/O thông minh

01 : Modul thông minh

a: Loại I/O

0 : Loại I/O số

1 : Loại Analog

ii : Ngõ vào

00 : Không có ngõ vào

01 : 2AI hoặc 8DI

10 : 2AI hoặc 16DI

11 : 8AI hoặc 32DI

qq : Ngõ ra

00 : Không có Output

01 : 2AQ hoặc 8DQ

10 : 2AQ hoặc 16DQ

11 : 8AQ hoặc 32DQ

❖ **Định dạng Modul Error:**

c	0	0	b	r	p	f	t

c : Định dạng lỗi 0 : không lỗi

1 : Lỗi

b : Lỗi đường Bus hay lỗi Parity

r : Lỗi vượt tầm

p : Lỗi nguồn người sử dụng

SMB8 : Modul0 ID

SMB9 : Modul0 Error

SMB10 : Modul1 ID

SMB11 : Modul1 Error

SMB12 : Modul2 ID

SMB13 : Modul2 Error

SMB14 : Modul3 ID

SMB15 : Modul3 Error

SMB16 : Modul4 ID

SMB17 : Modul4 Error

SMB18 : Modul5 ID

SMB19 : Modul5 Error

SMB20 : Modul6 ID

SMB21 : Modul6 Error

- SMW22 đến SMW26 : Thời gian quét.
  - SMW22 : Thời gian quét chu kì cuối (millisecond)
  - SMW24 : Thời gian quét nhỏ nhất (millisecond)
  - SMW26 : Thời gian quét lớn nhất (millisecond)
- SMB28 đến SMB29 : Điều chỉnh giá trị Analog
  - SMB28 : Giá trị điều chỉnh ngõ thứ 1 của Analog,khi chương trình chuyển từ Stop/Run
  - SMB29 : Giá trị điều chỉnh ngõ thứ 2 của Analog,khi chương trình chuyển từ Stop/Run
- SMB30 và SMB130 : Thanh ghi điều khiển **FreePort**

S7-200 Symbol Name	SM Address		Bit Format	Bit Format (MSB 7 … LSB 0)
	Port 0	Port 1		
P0_Config	SMB30			
P1_Config		SMB130		
	SM30.6 – SM30.7	SM130.6 – SM130.7	pp:	0 0 = No parity 0 1 = Even parity 1 0 = No parity 1 1 = Odd parity
	SM30.0 – SM30.1	SM130.0 – SM130.1	d:	0 = 8 data bits per character 1 = 7 data bits per character
	SM30.2 – SM30.4	SM130.2 – SM130.4	bbb:	0 0 0 = 38,400 bps 0 0 1 = 19,200 bps 0 1 0 = 9,600 bps 0 1 1 = 4,800 bps 1 0 0 = 2,400 bps 1 0 1 = 1,200 bps 1 1 0 = 115,200 bps* 1 1 1 = 57,600 bps* * Requires S7-200 CPU version 1.2 or later
P0_Config_0	SM30.0		mm:	0 0 = Point-to-Point Interface protocol (PPI/slave mode) 0 1 = Freeport protocol 1 0 = PPI/master mode 1 1 = Reserved (defaults to PPI/slave mode) **Note:** When you select code mm = 10 (PPI master), the CPU becomes a master on the network and allows the NETR and NETW instructions to be executed. Bits 2 through 7 are ignored in PPI modes.
P1_Config_0		SM130.0		
	SM30.1	SM130.1		

- SMB34 và SMB35 : Thanh ghi điều khiển ngắt thời gian
  - SMB34 : Ngắt thời gian cho INT_0 (ms)
  - SMB35 : Ngắt thời gian cho INT_1 (ms)
- SMB34 đến SMB65 : Thanh ghi dành cho HSC0,HSC1 và HSC2

S7-200 Symbol Name	SM Address	Function
HSC0_Status	SMB36	**HSC0 Counter Status** **Note:** Counter Status bits are valid only while an interrupt routine triggered by a high-speed counter event is being executed.
	SM36.0 – SM36.4	Reserved
HSC0_Status_5	SM36.5	HSC0 current counting direction status bit: 1 = counting up
HSC0_Status_6	SM36.6	HSC0 current value equals preset value status bit: 1 = equal
HSC0_Status_7	SM36.7	HSC0 current value is greater than preset value status bit: 1 = greater than
HSC0_Ctrl	SMB37	**HSC0 Control** **Note:** When you are using an HSC external reset interrupt event, do not attempt to load a new current value or disable, then re-enable the high-speed counter from within the interrupt routine attached to that event. A fatal error condition can result.
HSC0_Reset_Level	SM37.0	Active level control bit for Reset: 0 = Reset is active high:; 1 = Reset is active low
	SM37.1	Reserved
HSC0_Rate	SM37.2	HSC0 count rate selection for Quadrature counters: 0 = 4x counting rate; 1= 1x counting rate
HSC0_Dir	SM37.3	HSC0 direction control bit: 1 = count up
HSC0_Dir_Update	SM37.4	HSC0 update direction: 1 = update direction
HSC0_PV_Update	SM37.5	HSC0 update preset value: 1 = write new preset value to HSC0 preset
HSC0_CV_Update	SM37.6	HSC0 update current value: 1 = write new current value to HSC0 current
HSC0_Enable	SM37.7	HSC0 enable bit: 1 = enable
HSC0_CV	SMD38	**HSC0 New Current Value** Double Word data: SMD38 is used to set HSC0 current value to any value you choose. Write SMD38 with the desired new current value, then write SM37.6 to 1. The new current value is then written to HSC0 current.
HSC0_PV	SMD42	**HSC0 New Preset Value** Double Word data: SMD42 is used to set HSC0 preset value to any value you choose. Write SMD42 with the desired new preset value, then write SM37.5 to 1. The new preset value is then written to HSC0 preset.
HSC1_Status	SMB46	**HSC1 Counter Status** **Note:** Counter Status bits are valid only while an interrupt routine triggered by a high-speed counter event is being executed.
	SM46.0 – SM46.4	Reserved
HSC0_Status_5	SM46.5	HSC1 current counting direction status bit: 1 = counting up
HSC0_Status_6	SM46.6	HSC1 current value equals preset value status bit: 1 = equal
HSC0_Status_7	SM46.7	HSC1 current value is greater than preset value status bit: 1 = greater than
HSC1_Ctrl	SMB47	**HSC1 Control** **Note:** When you are using an HSC external reset interrupt event, do not attempt to load a new current value or disable, then re-enable the high-speed counter from within the interrupt routine attached to that event. A fatal error condition can result.
HSC1_Reset_Level	SM47.0	HSC1 active level control bit for Reset: 0 = Reset is active high; 1 = Reset is active low
HSC1_Start_Level	SM47.1	HSC1 active level control bit for Start: 0 = Start is active high; 1 = Start is active low
HSC1_Rate	SM47.2	HSC1 count rate selection for Quadrature counters: 0 = 4x counting rate; 1 = 1x counting rate
HSC1_Dir	SM47.3	HSC1 direction control bit: 1 = count up 0 = count down
HSC1_Dir_Update	SM47.4	HSC1 update direction: 1 = update direction
HSC1_PV_Update	SM47.5	HSC1 update preset value: 1 = write new preset value to HSC1 preset
HSC1_CV_Update	SM47.6	HSC1 update current value: 1 = write new current value to HSC1 current
HSC1_Enable	SM47.7	HSC1 enable bit: 1 = enable HSC 0 = disable HSC
HSC1_CV	SMD48	**HSC1 New Current Value** Double Word data: SMD48 is used to set HSC1 current value to any value you choose. Write SMD48 with the desired new current value, then write SM47.6 to 1. The new current value is then written to HSC1 current.
HSC1_PV	SMD52	**HSC1 New Present Value** Double Word data: SMD52 is used to set HSC1 preset value to any value you choose. Write SMD52 with the desired new preset value, then write SM47.5 to 1. The new preset value is then written to HSC1 preset.

HSC2_Status	SMB56	HSC2 Counter Status Note: Counter Status bits are valid only while an interrupt routine triggered by a high-speed counter event is being executed
	SM56.0 – SM56.4	Reserved
HSC2_Status_5	SM56.5	HSC2 current counting direction status bit: 1 = counting up
HSC2_Status_6	SM56.6	HSC2 current value equals preset value status bit: 1 = equal
HSC2_Status_7	SM56.7	HSC2 current value is greater than preset value status bit: 1 = greater than
HSC2_Ctrl	SMB57	HSC2 Control Note: When you are using an HSC external reset interrupt event, do not attempt to load a new current value or disable, then re-enable the high-speed counter from within the interrupt routine attached to that event. A fatal error condition can result.
HSC2_Reset_Level	SM57.0	HSC2 active level control bit for Reset: 0 = Reset is active high; 1 = Reset is active low
HSC2_Start_Level	SM57.1	HSC2 active level control bit for Start: 0 = Start is active high; 1 = Start is active low
HSC2_Rate	SM57.2	HSC2 counting rate selection for Quadrature counters: 0 = 4x counting rate; 1 = 1x counting rate
HSC2_Dir	SM57.3	HSC2 direction control bit: 1 = count up
HSC2_Dir_Update	SM57.4	HSC2 update direction: 1 = update direction
HSC2_PV_Update	SM57.5	HSC2 update preset value: 1 = write new preset value to HSC2 preset
HSC2_CV_Update	SM57.6	HSC2 update current value: 1 = write new current value to HSC2 current
HSC2_Enable	SM57.7	HSC2 enable bit: 1 = enable
HSC2_CV	SMD58	HSC2 New Current Value Double Word data: SMD58 is used to set HSC2 current value to any value you choose. Write SMD58 with the desired new current value, then write SM57.6 to 1. The new current value is then written to HSC2 current.
HSC2_PV	SMD62	HSC2 New Preset Value Double Word data: SMD62 is used to set HSC2 preset value to any value you choose. Write SMD62 with the desired new preset value, then write SM57.5 to 1. The new preset value is then written to HSC2 preset.

Xem chi tiết trong hướng dẫn đọc High Speed Counter ở các mục trên.

SMB66 đến SMB85: PTO/PWM Registers

S7-200 Symbol Name	SM Address	Function
PTO0_Status	SMB66	PTO0 Status
	SM66.0 – SM66.3	Reserved
PLS0_Err_Abort	SM66.4	PTO0 profile aborted: 0 = no error, 1 = aborted due to a delta calculation error
PLS0_Cmd_Abort	SM66.5	PTO0 profile aborted: 0 = not aborted by user command; 1 = aborted by user command
PLS0_Ovr	SM66.6	PTO0 pipeline overflow (cleared by the system when using external profiles, otherwise must be reset by user): 0 = no overflow; 1 = pipeline overflow
PLS0_Idle	SM66.7	PTO0 idle bit: 0 = PTO in progress, 1 = PTO is idle
PLS0_Ctrl	SMB67	Control Pulse Train Output and Pulse Width Modulation for Q0.0
PLS0_Cycle_Update	SM67.0	PTO0/PWM0 update cycle time value: 1 = write new cycle time
PWM0_PW_Update	SM67.1	PWM0 update pulse width value: 1 = write new pulse width
PTO0_PC_Update	SM67.2	PTO0 update pulse count value: 1 = write new pulse count
PLS0_TimeBase	SM67.3	PTO0/PWM0 time base: 0 = 1 µs/tick, 1 = 1 ms/tick
PWM0_Sync	SM67.4	Update PWM0 synchronously: 0 = asynchronous update, 1 = synchronous update
PTO0_Op	SM67.5	PTO0 operation: 0 = single segment operation (cycle time and pulse count stored in SM memory); 1 = multiple segment operation (profile table stored in V memory).
PLS0_Select	SM67.6	PTO0/PWM0 mode select: 0 = PTO; 1 = PWM
PLS0_Enable	SM67.7	PTO0/PWM0 enable bit: 1 = enable
PLS0_Cycle	SMW68	Cycle Time Value, Pulse Train or Pulse Width Modulation Output 0 Word data: PTO0/PWM0 cycle time value (2 to 65535 units of the time base)
PWM0_PW	SMW70	Pulse Width Value of Pulse Width Modulation Output 0 Word data: PWM0 pulse width value (0 to 65535 units of the time base)
PTO0_PC	SMD72	Pulse Count Value of Pulse Train Output 0 Double Word data:PTO0 pulse count value (1 to 2^32 – 1)
PTO1_Status	SMB76	PTO1 Status
	SM76.0 – SM76.3	Reserved
PLS1_Err_Abort	SM76.4	PTO1 profile aborted: 0 = no error, 1 = aborted due to a delta calculation error
PLS1_Cmd_Abort	SM76.5	PTO1 profile aborted: 0 = not aborted by user command, 1 = aborted by user command
PLS1_Ovr	SM76.6	PTO1 pipeline overflow (cleared by the system when using external profiles, otherwise must be reset by user): 0 = no overflow; 1 = pipeline overflow
PLS1_Idle	SM76.7	PTO1 idle bit:0 = PTO in progress; 1 = PTO is idle

PLS1_Ctrl	**SMB77**	**Control Pulse Train Output and Pulse Width Modulation for Q0.1**
PLS1_Cycle_Update	SM77.0	PTO1/PWM1 update cycle time value: 1 = write new cycle time
PWM1_PW_Update	SM77.1	PWM1 update pulse width value: 1 = write new pulse width
PTO1_PC_Update	SM77.2	PTO1 update pulse count value: 1 = write new pulse count
PLS1_TimeBase	SM77.3	PTO1/PWM1 time base: 0 = 1 µs/tick, 1 = 1 ms/tick
PWM1_Sync	SM77.4	Update PWM1 synchronously: 0 = asynchronous update, 1 = synchronous update
PTO1_Op	SM77.5	PTO1 operation: 0 = single segment operation (cycle time and pulse count stored in SM memory); 1 = multiple segment operation (profile table stored in V memory)
PLS1_Select	SM77.6	PTO1/PWM1 mode select: 0 = PTO; 1 = PWM
PLS1_Enable	SM77.7	PTO1/PWM1 enable bit: 1 = enable

PLS1_Cycle	**SMW78**	**Cycle Time Value, Pulse Train or Pulse Width Modulation Output 1** Word data: PTO1/PWM1 cycle time value (2 to 65535 units of the time base)

PWM1_PW	**SMW80**	**Pulse Width Value of Pulse Width Modulation Output 1** Word data: PWM1 pulse width value (0 to 65535 units of the time base)

PTO1_PC	**SMD82**	**Pulse Count Value of Pulse Train Output 1** Double word data: PTO1 pulse count value (1 to 2^32 - 1)

SMB86 đến SMB94,SMB186 đến SMB194 : Receive Message Control

S7-200 Symbol Name	SM Address		Receive Message Status Byte
	Port 0	Port 1	Bit Format: MSB 7 … 0 / n r e 0 0 t c P
P0_Stat_Rcv	**SMB86**		
P1_Stat_Rcv		**SMB186**	
P0_Stat_Rcv_7	SM86.7		n: 1 = Receive message was terminated by user disable command
P1_Stat_Rcv_7		SM186.7	
P0_Stat_Rcv_6	SM86.6		r: 1 = Receive message terminated: error in input parameters or missing start or end condition
P1_Stat_Rcv_6		SM186.6	
P0_Stat_Rcv_5	SM86.5		e: 1 = End character received
P1_Stat_Rcv_5		SM186.5	
P0_Stat_Rcv_2	SM86.2		t: 1 = Receive message terminated: timer expired
P1_Stat_Rcv_2		SM186.2	
P0_Stat_Rcv_1	SM86.1		c: 1 = Receive message terminated: maximum character count achieved
P1_Stat_Rcv_1		SM186.1	
P0_Stat_Rcv_0	SM86.0		p: 1 = Receive message was terminated because of a parity error
P1_Stat_Rcv_0		SM186.0	

			Receive Message Control Byte
	Port 0	Port 1	Bit Format: MSB 7 … LSB 0 / en sc ec il c/m Tmr bk 0
P0_Ctrl_Rcv	**SMB87**		
P1_Ctrl_Rcv		**SMB187**	
P0_Ctrl_Rcv_7	SM87.7		en: 0 = receive message function is disabled; 1 = receive message function is enabled
P1_Ctrl_Rcv_7		SM187.7	
P0_Ctrl_Rcv_6	SM87.6		sc: 0 = ignore SMB88 or SMB188; 1 = use the value of SMB88 or SMB188 to detect start of message
P1_Ctrl_Rcv_6		SM187.6	
P0_Ctrl_Rcv_5	SM87.5		ec: 0 = ignore SMB89 or SMB189; 1 = use the value of SMB89 or SMB189 to detect end of message
P1_Ctrl_Rcv_5		SM187.5	
P0_Ctrl_Rcv_4	SM87.4		il: 0 = ignore SMW90 or SMB190; 1 = use the value of SMW190 to detect an idle line condition
P1_Ctrl_Rcv_4		SM187.4	
P0_Ctrl_Rcv_3	SM87.3		c/m: 0 = timer is an inter-character timer; 1 = timer is a message timer
P1_Ctrl_Rcv_3		SM187.3	
P0_Ctrl_Rcv_2	SM87.2		tmr: 0 = ignore SMW92 or SMW192; 1 = terminate receive if the time period in SMW92 or SMW192 is exceeded
P1_Ctrl_Rcv_2		SM187.2	
P0_Ctrl_Rcv_1	SM87.1		bk: 0 = ignore break conditions; 1 = use break condition as start of message detection
P1_Ctrl_Rcv_1		SM187.1	

SMW98: Lỗi trên Modul mở rộng I/O:

SMW98 tăng mỗi khi bits parity lỗi được kiểm tra ở Modul mở rộng .Giá trị này sẽ được xoá mỗi khi bật nguồn hoặc có thể xoá bởi người sử dụng.

SMB131 đến SMB165:*HSC3,HSC4,HSC5*

S7-200 Symbol Name	SM Addr.	Description
HSC3_Status	SMB136	HSC3 Counter Status Note: Counter Status bits are valid only while an interrupt routine triggered by a high-speed counter event is being executed.
	SM136.0 – SM136.4	Reserved
HSC3_Status_5	SM136.5	HSC3 current counting direction status bit: 1 = counting up
HSC3_Status_6	SM136.6	HSC3 current value equals preset value status bit: 1 = equal
HSC3_Status_7	SM136.7	HSC3 current value is greater than preset value status bit: 1 = greater than
HSC3_Ctrl	SMB137	HSC3 Control
	SM137.0 – SM137.2	Reserved
HSC3_Dir	SM137.3	HSC3 direction control bit: 1 = count up
HSC3_Dir_Update	SM137.4	HSC3 update direction: 1 = update direction
HSC3_PV_Update	SM137.5	HSC3 update preset value: 1 = write new preset value to HSC3 preset
HSC3_CV_Update	SM137.6	HSC3 update current value: 1 = write new current value to HSC3 current
HSC3_Enable	SM137.7	HSC3 enable bit: 1 = enable
HSC3_CV	SMD138	HSC3 New Current Value Double word data: SMD138 is used to set HSC3 current value to any value you choose. Write SMD138 with the desired new current value, then write SM137.6 to 1. The new current value is then written to HSC3 current.
HSC3_PV	SMD142	HSC3 New Preset Value Double word data: SMD142 is used to set HSC3 preset value to any value you choose. Write SMD142 with the desired new preset value, then write SM137.5 to 1. The new preset value is then written to HSC3 preset.
HSC4_Status	SMB146	HSC4 Counter Status Note: Counter Status bits are valid only while an interrupt routine triggered by a high-speed counter event is being executed
	SM146.0 – SM146.4	Reserved
HSC4_Status_5	SM146.5	HSC4 current counting direction status bit: 1 = counting up
HSC4_Status_6	SM146.6	HSC4 current value equals preset value status bit: 1 = equal
HSC4_Status_7	SM146.7	HSC4 current value is greater than preset value status bit: 1 = greater than
HSC4_Ctrl	SMB147	HSC4 Control Note: When you are using an HSC external reset interrupt event, do not attempt to load a new current value or disable, then re-enable the high-speed counter from within the interrupt routine attached to that event. A fatal error condition can result.
HSC4_Reset_Level	SM147.0	HSC4 active level control bit for Reset: 0 = Reset is active high; 1 = Reset is active low
	SM147.1	Reserved
HSC4_Rate	SM147.2	HSC4 counting rate selection for Quadrature counters: 0 = 4x counting rate; 1 = 1x counting rate
HSC4_Dir	SM147.3	HSC4 direction control bit: 1 = count up
HSC4_Dir_Update	SM147.4	HSC4 update direction: 1 = update direction
HSC4_PV_Update	SM147.5	HSC4 update preset value: 1 = write new preset value to HSC4 preset
HSC4_CV_Update	SM147.6	HSC4 update current value: 1 = write new current value to HSC4 current
HSC4_Enable	SM147.7	HSC4 enable bit: 1 = enable
HSC4_CV	SMD148	HSC4 New Current Value Double word data: SMD148 is used to set HSC4 current value to any value you choose. Write SMD148 with the desired new current value, then write SM147.6 to 1. The new current value is then written to HSC4 current.
HSC4_PV	SMD152	HSC4 New Preset Value Double word data: SMD152 is used to set HSC4 preset value to any value you choose. Write SMD152 with the desired new preset value, then write SM147.5 to 1. The new preset value is then written to HSC4 preset.
HSC5_Status	SMB156	HSC5 Counter Status Note: Counter Status bits are valid only while an interrupt routine triggered by a high-speed counter event is being executed.
	SM156.0 – SM156.4	Reserved
HSC5_Status_5	SM156.5	HSC5 current counting direction status bit: 1 = counting up
HSC5_Status_6	SM156.6	HSC5 current value equals preset value status bit: 1 = equal
HSC5_Status_7	SM156.7	HSC5 current value is greater than preset value status bit: 1 = greater than

HSC5_Ctrl	**SMB157**	**HSC5 Control**
	SM156.0 – SM156.2	Reserved
HSC5_Dir	SM157.3	HSC5 direction control bit: 1 = count up
HSC5_Dir_Update	SM157.4	HSC5 update direction: 1 = update direction
HSC5_PV_Update	SM157.5	HSC5 update preset value: 1 = write new preset value to HSC5 preset
HSC5_CV_Update	SM157.6	HSC5 update current value: 1 = write new current value to HSC5 current
HSC5_Enable	SM157.7	HSC5 enable bit: 1 = enable

HSC5_CV	**SMD158**	**HSC5 New Current Value** Double word data: SMD158 is used to set HSC5 current value to any value you choose. Write SMD158 with the desired new current value, then write SM157.6 to 1. The new current value is then written to HSC5 current.

HSC5_PV	**SMD162**	**HSC5 New Preset Value** Double word data: SMD162 is used to set HSC5 preset value to any value you choose. Write SMD162 with the desired new preset value, then write SM157.5 to 1. The new preset value is then written to HSC5 preset.

9/ Điều khiển biến tần theo giao thức USS:

CPU S7_200 có thể điều khiển biến tần Siemens thông qua Port giao tiếp bằng giao thức USS