CHƯƠNG 2 : TẬP LỆNH CỦA PLC S7-300

1.1 CẤU TRÚC LỆNH VÀ TRẠNG THÁI KẾT QUẢ:

Trong tài liệu ĐKLT 1 đã trình bày về các phương pháp lập trình cho PLC, gồm có ngôn ngữ lập trình dạng STL, LAD và FBD. Phần này sẽ trình bày chủ yếu về cấu trúc và kết quả của lệnh dạng STL.

Một lệnh STL của PLC S7-300 gồm có: "Tên lệnh" + "Toán hạng".

Ví dụ: A I0.0 là lệnh nạp giá trị ngõ vào có địa chỉ I0.0

Trong đó: A là "Tên lệnh"

I0.0 là "Toán hạng"

Lưu ý: toán hạng có thể là dữ liệu hoặc là địa chỉ của một vùng nhớ nào đó.

- **Toán hạng là dữ liệu:**
  - Dữ liệu logic
  - Số nhị phân
  - Số thập lục phân
  - Số nguyên kiểu INT (2 byte)
  - Số nguyên kiểu DINT (4 byte)
  - Số thực kiểu REAL
  - Dữ liệu về thời gian
  - Dữ liệu của bộ đếm, định thời
  - Dữ liệu kiểu ký tự
- **Toán hạng là địa chỉ nhớ:**

  Địa chỉ trong bộ nhớ PLC S7-300 gồm 2 phần: phần chữ và phần số.

  Ví dụ: địa chỉ ngõ vào I0.5

  Trong đó:
  - Phần chữ: chỉ vị trí và kích thước của vùng nhớ.
  - Phần số: chỉ địa chỉ của vùng nhớ trong miền đã được xác định.
- **Thanh ghi trạng thái:**

  Khi thực hiện lệnh, CPU sẽ ghi lại trạng thái của phép tính trung gian cũng như ghi lại kết quả vào 1 thanh ghi đặc biệt 16 bit, gọi là thanh ghi trạng thái.

  Tuy nhiên chỉ có 9 bit thấp của thanh ghi này được sử dụng, có cấu trúc như sau:

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

Trong đó:

- FC (Fisrt Check): khi thực hiện các lệnh logic liên tiếp nhau gồm các phép tính ^ ***(VÀ), V (HOẶC), ĐẢO*** thì bit FC=1. Khi kết thúc các lệnh thì FC=0.

  Ví dụ:

  A I0.0 // FC=1

  AN I1.0 //FC=1

  = Q0.0 //FC=0

- RLO (Result of Logic Operation): thể hiện kết quả tức thời của phép tính logic vừa thực hiện.

  Ví dụ:

  A I0.0

  Nếu trước khi thực hiện bit FC=0 thì có tác dụng chuyển nội dung ngõ vào I0.0 vào bit trạng thái RLO.

  Còn khi bit FC=1 thì có tác dụng thực hiện phép tính **VÀ** (RLO ^ I0.0), kết quả được ghi trở lại vào RLO.

- STA (Status Bit): bit trạng thái, luôn có giá trị logic của tiếp điểm được chỉ trong lệnh.

  Ví dụ: cả hai lệnh sau đều gán cho bit STA giá trị của ngõ vào I0.3.

  A I0.3

  AN I0.3

- OR: ghi lại giá trị của phép tính VÀ cuối cùng được thực hiện để thực hiện phép tính HOẶC (V) sau đó.
- OS (Overflow Store bit): ghi giá trị bit bị tràn.
- OV (Overflow bit): bit báo kết quả phép tính bị tràn.
- CCO và CC1 (Condition Code): hai bit báo trạng thái của kết quả phép tính với số nguyên, số thực, dịch chuyển hoặc phép tính logic trong ACCU.
- BR (Binary Result bit): bit trạng thái cho phép liên kết giữa hai ngôn ngữ STL và LAD. Cho phép người lập trình viết một khối chương trình FB hoặc FC bằng STL, nhưng có thể gọi và sử dụng chúng trong chương trình khác viết bằng LAD. Để có mối liên kết này, cần phải kết thúc trong chương trình trong FB, FC bằng lệnh ghi giá trị vào BR:

  1 nếu chương trình không có lỗi,

  0 nếu chương trình có lỗi.

Chú ý:

Một chương trình viết bằng STL có thể gồm nhiều network. Mỗi network chứa một đoạn chương trình phục vụ một việc cụ thể. Ở đầu mỗi network , thanh ghi trạng thái

nhận giá trị 0, chỉ sau khi thực hiện lệnh đầu tiên của network các bit trạng thái mới thay đổi theo phép tính.

1.2 NHÓM LỆNH LOGIC:

- **Lệnh gán:**
  - STL:

    Cú pháp = **<toán hạng>**

    Toán hạng là địa chỉ bit I, Q, M, L, D, T, C.

    Lệnh gán giá trị logic của RLO tới ô nhớ có địa chỉ được chỉ thị trong toán hạng.

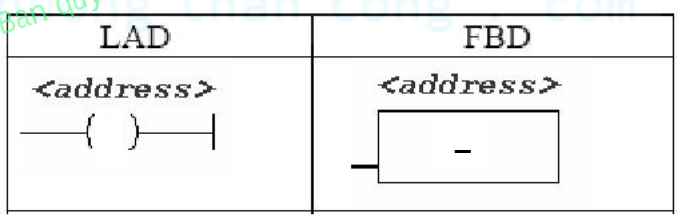
    Lệnh tác động vào thanh ghi trạng thái như sau:

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

    Ký hiệu: (-) Chỉ nội dung bit không bị thay đổi theo lệnh.

    (x) Chỉ nội dung bit bị thay đổi theo lệnh.

  - LAD:



Với:

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

Khi giá trị logic của bit tại <address> bằng 1 thì RLO có giá trị 1.

Khi giá trị logic của bit tại <address> bằng 0 thì RLO có giá trị bằng 0.

- **Lệnh gán có điều kiện giá trị 1:**
  - STL:

    Cú pháp **S <toán hạng>**

    Toán hạng là địa chỉ bit I, Q, L, M, D.

    Nếu RLO=1 lệnh sẽ ghi giá trị 1 váo ô nhớ có địa chỉ trong toán hạng.

    Lệnh tác động vào thanh ghi trạng thái như sau:

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

- LAD:

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

Nếu RLO = 1 thì địa chỉ cụ thể được đặt ở mức 1 và duy trì ở trạng thái này cho đến khi nó bị xóa về 0 bằng lệnh reset.

- **Lệnh gán có điều kiện giá trị 0:**
  - STL:

    Cú pháp **R <toán hạng>**

    Toán hạng là địa chỉ bit I, Q, M, L, D.

    Nếu RLO=0, lệnh sẽ ghi giá trị 0 vào ô nhớ có địa chỉ trong toán hạng.

    Lệnh tác động vào thanh ghi trạng thái như sau:

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

  - LAD:

LAD	FBD
<address> —(R)—	<address> R

Với:

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

Nếu RLO = 1 thì địa chỉ cụ thể được đặt ở mức 0 và duy trì ở trạng thái này cho đến khi nó đặt lên 1 bằng lệnh set.

- **Lệnh AND:**
  - STL:

Cú pháp **A <toán hạng>**

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

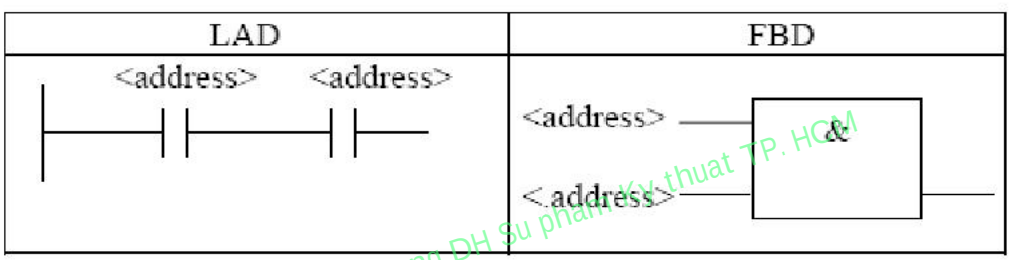
Nếu FC = 0 lệnh sẽ gán giá trị logic của toán hạng vào RLO.

Ngược lại khi FC = 1 lệnh sẽ thực hiện phép tính AND RLO với toán hạng và ghi lại kết quả vào RLO.

Lệnh tác động vào thanh ghi trạng thái như sau:

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

- LAD:



Với:

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

Khi giá trị logic hai địa chỉ <address> bằng 1 thì RLO có giá trị 1.

Nếu có ít nhất 1 trong 2 ngõ vào xuống mức 0 thì RLO có giá trị bằng 0.

- **Lệnh AND NOT:**
  - STL:

Cú pháp **AN <toán hạng>**

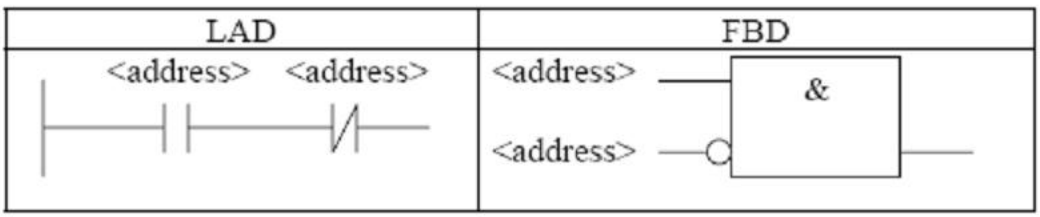
Toán hạng là dữ liệu kiểu BOOL hoặc địa chỉ bit I, Q, M, L, D, T, C.

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

Lệnh tác động vào thanh ghi trạng thái như sau:

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

- LAD:



Với:

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

- **Lệnh OR:**
  - STL:

    Cú pháp **O <toán hạng>**

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

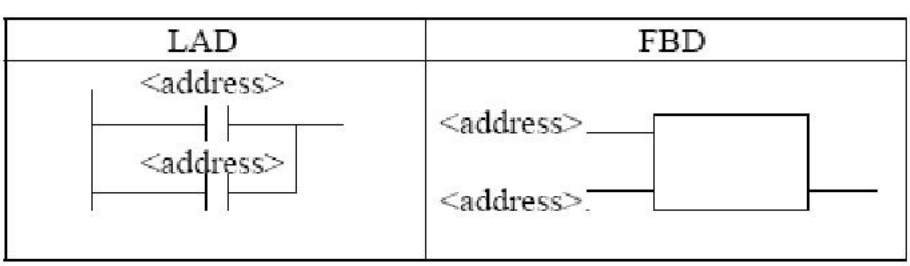
    Nếu FC = 0 lệnh sẽ gán giá trị logic của toán hạng vào RLO. Nếu FC = 1 nó thực hiện

    phép tính OR RLO với toán hạng và ghi lại kết quả vào RLO.

    Lệnh tác động vào thanh ghi trạng thái như sau:

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

  - LAD:



Với :

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

RLO có giá trị 1 khi có ít nhất một trong hai tín hiệu tại hai địa chỉ <address> ở mức 1.

RLO có giá trị 0 khi cả hai tín hiệu ngõ vào đều xuống mức 0.

- **Lệnh OR NOT:**
  - STL:

    Cú pháp **ON <toán hạng>**

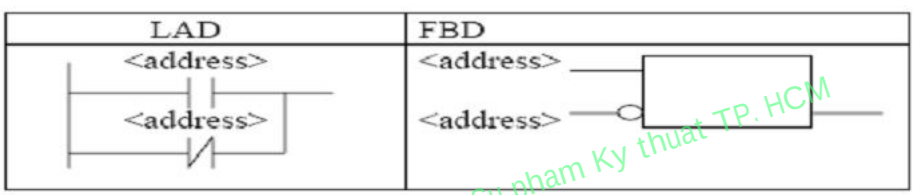
    Toán hạng là dữ liệu kiểu BOOL hoặc địa chỉ bit I, Q, M, L, D, T, C.

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

    Lệnh tác động vào thanh ghi trạng thái như sau:

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

  - LAD:



Với:

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

- **Lệnh AND biểu thức:**
  - STL:

    Cú pháp **A(**

)

    Lệnh không có toán hạng.

    Nếu FC = 0 lệnh sẽ gán giá trị logic của biểu thức trong dấu ngoặc sau nó vào RLO.

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

    Lệnh tác động vào thanh ghi trạng thái như sau:

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

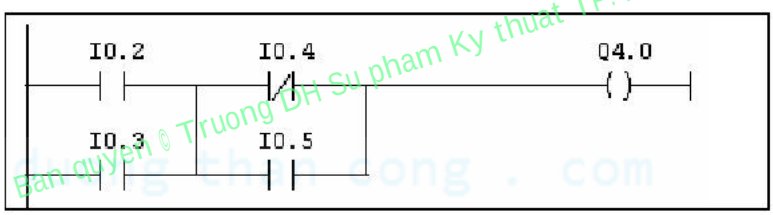
BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
-	-	-	-	-	x	1	x	1

Ví dụ:

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

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

- LAD:



- **Lệnh OR biểu thức:**
  - STL:

    Cú pháp **O(**

)

    Lệnh không có toán hạng.

    Nếu FC = 0 lệnh sẽ gán giá trị logic của biểu thức trong dấu ngoặc sau nó vào RLO.

    Nếu FC = 1 nó sẽ thực hiện phép tính OR giữa RLO với giá trị của biểu thức trong dấu ngoặc sau nó và ghi lại kết quả vào RLO.

    Lệnh tác động vào thanh ghi trạng thái như sau:

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:

Thực hiện Q4.0 = I0.2 v (I0.4 v I0.5)

A I0.2

O(

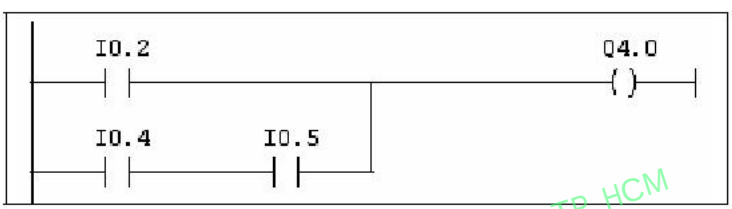
AN I0.4

A I0.5

)

= Q4.0

- LAD:



1.3 NHÓM LỆNH TIẾP ĐIỂM ĐẶC BIỆT:

- **Lệnh ghi giá trị logic 1 vào RLO:**
  - STL:

    Cú pháp **SET**

    Lệnh không có toán hạng và có tác dụng ghi 1 vào RLO.

    Lệnh tác động vào thanh ghi trạng thái như sau:

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

  - LAD: lệnh không thực hiện.

- **Lệnh ghi giá trị logic 0 vào RLO:**
  - STL:

    Cú pháp **CLR**

    Lệnh không có toán hạng và có tác dụng ghi 1 vào RLO.

    Lệnh tác động vào thanh ghi trạng thái như sau:

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

  - LAD: lệnh không thực hiện.

- **Lệnh đảo giá trị RLO:**
  - STL:

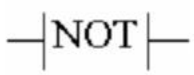
    Cú pháp **NOT**

    Lệnh không có toán hạng và có tác dụng đảo giá trị RLO.

    Lệnh tác động vào thanh ghi trạng thái như sau:

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

  - LAD:

LAD	FBD
	

- **Lệnh phát hiện xung cạnh lên:**
  - STL:

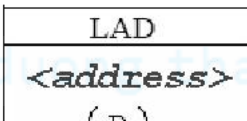
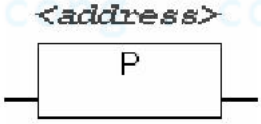
    Cú pháp **FP <Toán hạng>**

    Toán hạng là địa chỉ I, Q, M, L, D và được sử dụng như một biến cờ để ghi lại giá trị của RLO tại vị trí này trong chương trình. RLO sẽ có giá trị trong vòng quét khi có sườn lên trong RLO.

    Lệnh tác động váo thanh ghi trạng thái như sau:

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

  - LAD:

LAD	FBD
	

Với:

Thông số	Kiểu dữ liệu	Toán hạng	Mô tả
<address>	BOOL	I, Q, M, L, D	Địa chỉ bit lưu trữ trạng thái tín hiệu của RLO trước đó

Khi RLO thay đổi từ 0 lên 1 kết quả của lệnh kiểm tra FB ở trạng thái 1 trong một vòng quét. Để hệ thống phát hiện được sự thay đổi cạnh lên thì RLO phải được lưu trữ trong 1 bit nhớ FB hoặc bit dữ liệu <address>.

Nếu giá trị RLO trước đó lưu trữ trong <address > có giá trị 0 và RLO ở vòng quét hiện tại có giá trị 1 thì kết quả RLO của lệnh có giá trị 1 trong vòng quét.

- **Lệnh phát hiện xung cạnh xuống:**
  - STL:

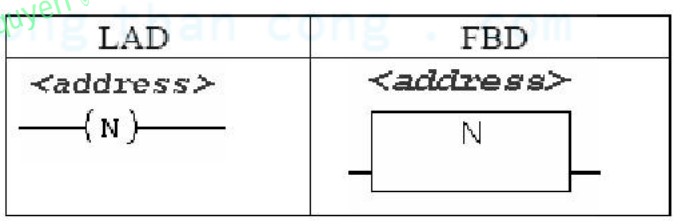
    Cú pháp **FN <Toán hạng>**

    Toán hạng là địa chỉ I, Q, M, L, D và được sử dụng như 1 biến cờ để ghi lại giá trị của RLO tại vị trí này trong chương trình. RLO sẽ có giá trị trong vòng quét khi có sườn xuống trong RLO.

    Lệnh tác động vào thanh ghi trạng thái như sau:

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

  - LAD:



Với:

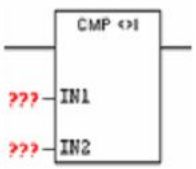
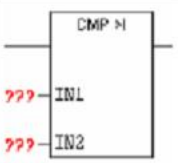
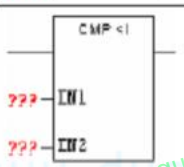
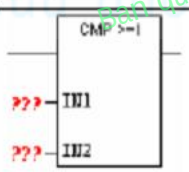
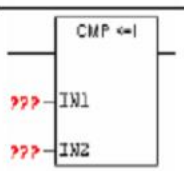
Thông số	Kiểu dữ liệu	Toán hạng	Mô tả
<address>	BOOL	I,Q,M,L,D	Địa chỉ bit lưu trữ trạng thái tín hiệu của RLO trước đó

Khi RLO thay đổi từ 1 xuống 0 kết quả của lệnh kiểm tra FB ở trạng thái trong 1 vòng quét. Để hệ thống phát hiện được sự thay đổi cạnh lên thì RLO phải được lưu trữ trong một bit nhớ FB hoặc bit dữ liệu <address>.

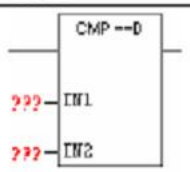
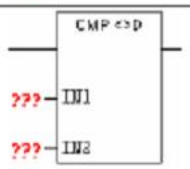
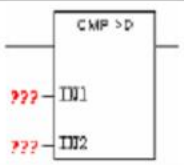
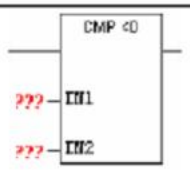
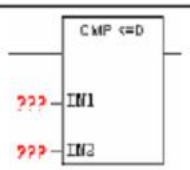
Nếu giá trị RLO trước đó lưu trữ trong <address > có giá trị 0 và RLO ở vòng quét hiện tại có giá trị 1 thì kết quả RLO của lệnh có giá trị 1 trong vòng quét.

1.4 NHÓM LỆNH SO SÁNH:

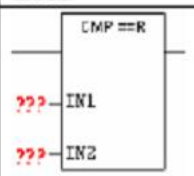
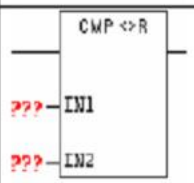
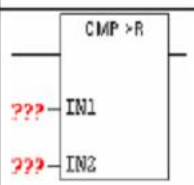
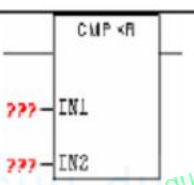
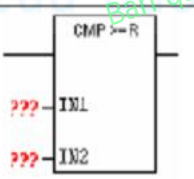
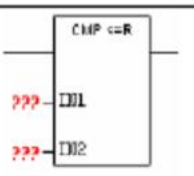
- **So sánh số nguyên 16 bit:**

	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.

- **So sánh số nguyên 32 bit:**

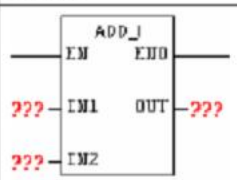
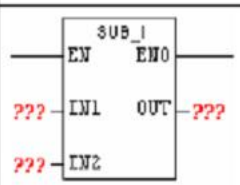
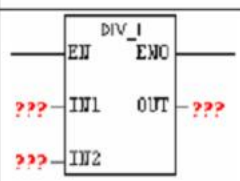
	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh so sánh bằng		==D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1=IN2 -Ngõ vào lên mức 1.
Lệnh so sánh không bằng		<>D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<>IN2 -Ngõ vào lên mức 1.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

- **So sánh số thực:**

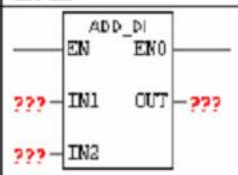
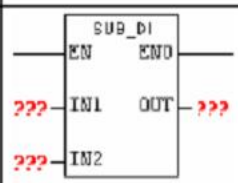
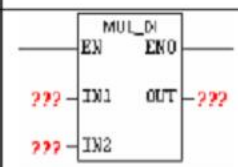
	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh so sánh bằng		==R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1=IN2 -Ngõ vào lên mức 1.
Lệnh so sánh không bằng		<>R	IN1 I,Q,L,M,D (REAL) const IN2 I,Q,M,L,D (REAL) const	Ngõ ra sẽ lên mức 1 nếu thỏa: -IN1<>IN2 -Ngõ vào lên mức 1.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

1.5 NHÓM LỆNH TOÁN HỌC:

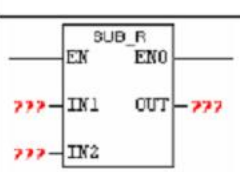
- **Thực hiện với số nguyên 16 bit:**

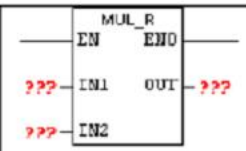
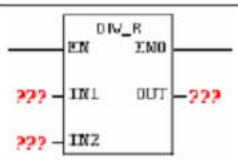
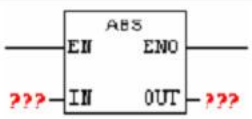
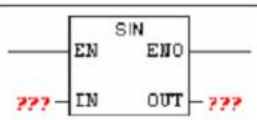
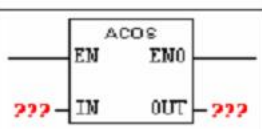
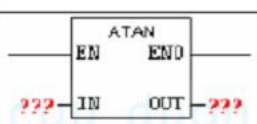
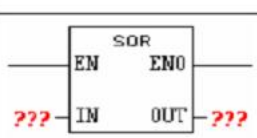
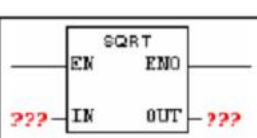
	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh cộng		+ I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const OUT I.Q.M.L.D, (INT) const	Lệnh cộng 2 số nguyên 16 bit trong IN 1 và IN 2. Kết quả cất vào OUT.
Lệnh trừ		- I	IN1 I,Q,L,M,D (INT) const IN2 I,Q,M,L,D (INT) const OUT I.Q.M.L.D, (INT) const	Lệnh trừ 2 số nguyên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

- **Thực hiện với số nguyên 32 bit:**

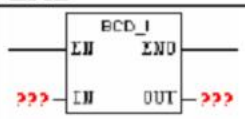
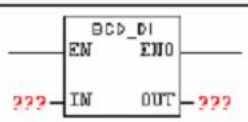
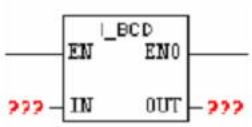
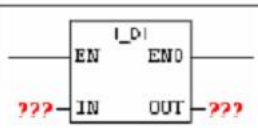
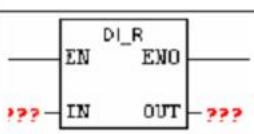
	LAD	STL	TOÁN HẠNG	MÔ TẢ
Lệnh cộng		+ D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const OUT I.Q.M.L.D, (DINT) const	Lệnh cộng 2 số nguyên 32 bit trong IN 1 và IN 2. Kết quả cất vào OUT.
Lệnh trừ		- D	IN1 I,Q,L,M,D (DINT) const IN2 I,Q,M,L,D (DINT) const OUT I.Q.M.L.D, (DINT) const	Lệnh trừ 2 số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

- **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 lấy giá trị tuyệt đối		ABS	IN I,Q,L,M,D (REAL) OUT I.Q.M.L.D, (REAL)	Lệnh lấy giá trị tuyệt đối trong IN. Kết quả cất vào OUT.
Lệnh tính sin		SIN	IN I,Q,L,M,D (REAL) OUT I.Q.M.L.D, (REAL)	Lệnh tính sin trong IN. Kết quả cất vào OUT.
Lệnh tính cos		Cos	IN I,Q,L,M,D (REAL) OUT I.Q.M.L.D, (REAL)	Lệnh tính cos trong IN. Kết quả cất vào OUT.
Lệnh tính tg		TAN	IN I,Q,L,M,D (REAL) OUT I.Q.M.L.D, (REAL)	Lệnh tính tg trong IN. Kết quả cất vào OUT.
Lệnh tính arsin		ASIN	IN I,Q,L,M,D (REAL) OUT I.Q.M.L.D, (REAL)	Lệnh tính arsin trong IN. Kết quả cất vào OUT.
Lệnh tính arcos		ACOS	IN I,Q,L,M,D (REAL) OUT I.Q.M.L.D, (REAL)	Lệnh tính arcos trong IN. Kết quả cất vào OUT.
Lệnh tính artg		ATAN	IN I,Q,L,M,D (REAL) OUT I.Q.M.L.D, (REAL)	Lệnh tính artg trong IN. Kết quả cất vào OUT.
Lệnh tính bình phương		SQR	IN I,Q,L,M,D (REAL) OUT I.Q.M.L.D, (REAL)	Lệnh tính bình phương trong IN. Kết quả cất vào OUT.
Lệnh tính căn bậc 2		SQRT	IN I,Q,L,M,D (REAL) OUT I.Q.M.L.D, (REAL)	Lệnh tính căn trong IN. Kết quả cất vào OUT.

1.6 LỆNH CHUYỂN ĐỔI BCD – SỐ NGUYÊN:

	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.

1.7 LỆNH VỀ TIMER:

1.7.1 Giới thiệu Timer:

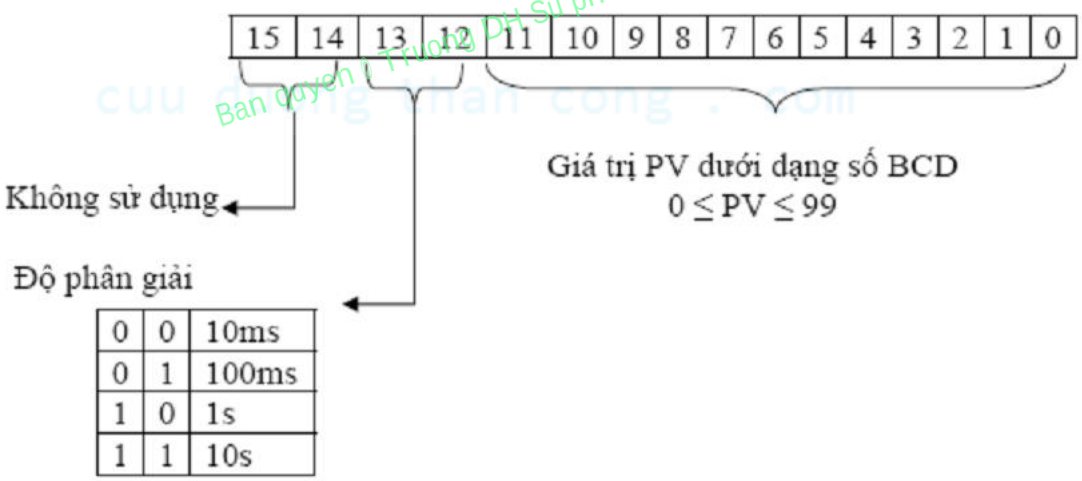
Bộ thời gian Timer là bộ tạo thời gian trễ T mong muốn giữa tín hiệu logic ngõ vào và tín hiệu logic ngõ ra.

S7 300 có 5 loại timer khác nhau**.** Tất cả 5 loại Timer này cùng bắt đầu tạo thời gian trễ tín hiệu kể từ thời điểm kích của tín hiệu đầu vào, tức là khi tín hiệu đầu vào chuyển trạng thái, được gọi là thời điểm timer được kích.

Thời gian trễ T mong muốn được khai báo với timer bằng một word 16 bit bao gồm 2 thành phần:

- Độ phân giải: timer của S7 300 có 4 chế độ phân giải: 10ms, 100ms, 1s và 10s.
- Một số nguyên BCD trong khoảng 0 ÷ 999 được gọi là PV (reset value _ giá trị đặt trước).

Thời gian trễ T mong muốn tính như sau: T = Độ phân giải *PV



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

Bit 14, 15 không sử dụng.

Bit 13, 12 dùng để đặt độ phân giải.

Bit 0 đến bit 11 là giá trị PV dưới dạng BCD (0< PV < 999).

Ngay tại thời điểm kích timer, giá trị PV được chuyển vào thanh ghi 16 bit của T_word (gọi là thanh ghi CV, viết tắt current value, giá trị tức thời). Timer sẽ ghi nhớ khoảng thời gian trôi qua kể từ khi được kích bằng cách giảm dần một cách tương ứng nội dung thanh ghi CV. Nếu nội dung thanh ghi trở về bằng 0 thì timer đã đạt được thời gian trễ mong muốn T và điều này sẽ được báo ra ngoài bằng cách đổi trạng thái tín hiệu ngõ ra.

CPU 314 có 128 timer được đánh số từ 0 đến 127. Một timer được đặt tên là Tx, trong đó x là số hiệu của timer ($0 \leq x \leq 127$). Ký hiệu Tx cũng đồng thời là địa chỉ hình thức của thanh ghi CV (T- word) và của đầu ra T-bit của timer đó. Tuy chúng có cùng địa chỉ hình thức, song T-word và T-bit vẫn được phân biệt với nhau nhờ kiểu lệnh sử dụng với toán hạng Tx .Khi dùng lệnh làm việc với từ, Tx được hiểu là địa chỉ của Tword, ngược lại khi sử dụng lệnh làm việc với tiếp điểm Tx sẽ được hiểu là địa chỉ của T-bit.

Một timer đang trong chế độ làm việc (sau khi được kích) có thể được đưa về chế độ chờ khởi động ban đầu, tức là chờ sườn lên của tín hiệu đầu vào. Công việc này gọi là reset timer. Tín hiệu reset timer được gọi là tín hiệu xoá và khi tín hiệu xoá có giá trị bằng 1 timer sẽ không làm việc. Tại thời điểm xuất hiện sườn lên của tín hiệu xoá, T_word và T-bit được xoá về 0, tức là thanh ghi CV được đặt về 0 và tín hiệu đầu ra có trạng thái 0**.**

1.7.2 Khai báo sử dụng Timer:

Khai báo sử dụng timer gồm có 5 bước:

- Khai báo tín hiệu enable nếu muốn sử dụng tín hiệu chủ động kích.
- Khai báo tín hiệu đầu vào.
- Khai báo tín hiệu trễ mong muốn.
- Khai báo loại timer được sử dụng.
- Khai báo tín hiệu xoá timer nếu muốn.

- **Khai báo tín hiệu enable:**

  Cú pháp **A <Địa chỉ bit>**

  FR <Tên timer>

  Toán hạng thứ nhất "Địa chỉ bit" xác định tín hiệu sẽ được sử dụng làm tín hiệu chủ động kích cho timer có tên trong toán hạng thứ hai.

- **Khai báo tín hiệu đầu vào**:

  Cú pháp **A <Địa chỉ bit>**

  "Địa chỉ bit" trong toán hạng xác định tín hiệu đầu vào cho timer.

  Ví dụ:

  A I2.0

  FR T1

  A I2.1

- **Khai báo thời gian trễ mong muốn:**

  Cú pháp **L <hằng số>**

  "Hằng số " trong toán hạng xác định thời gian trễ T đặt trước cho timer. Hằng số này có hai dạng:

  - Dạng dữ liệu thời gian trực tiếp: S5T#h_m_s_ms

    L S5T#00h05m20s00ms có thời gian trễ là 5 phút 20 giây.

  - Dạng khai báo theo độ phân giải:

    L W#16#2127 có thời gian trễ là 127 giây.

- **Khai báo loại timer**:

  S7-300 có 5 loại timer được khai báo theo các lệnh:

  - **SD: Timer đóng mạch chậm**

  Cú pháp **SD <Tên timer>**

  Thời gian giữ trễ được bắt đầu tính từ khi có sườn lên của tín hiệu đầu vào (hoặc khi có sườn lên của tín hiệu enable đồng thời tín hiệu vào bằng 1), tức là ở ngay thời điểm đó giá trị PV (giá trị đặt trước) được chuyển vào thanh ghi T-word (CV giá trị tức thời). Trong khoảng thời gian trễ T-bit có giá trị 0. Khi hết thời gian trễ, T-bit có giá trị bằng 1. Như vậy T-bit có giá trị 1 khi T-word = 0 hay CV = 0.

  Khoảng thời gian trễ chính là khoảng thời gian giữa thời điểm xuất hiện sườn lên của tín hiệu đầu vào và sườn lên của T-bit.

  Khi tín hiệu vào bằng 0, T-bit và T-word cùng nhận giá trị 0.

  - **SS: Timer đóng mạch chậm có nhớ**

  Cú pháp : **SS <tên timer>**

  Thời gian giữ trễ được bắt đầu tính từ khi có sườn lên của tín hiệu đầu vào (hoặc khi có sườn lên của tín hiệu enable đồng thời tín hiệu vào bằng 1), tức là ở ngay thời điểm đó giá trị PV (giá trị đặt trước) được chuyển vào thanh ghi T-word (CV giá trị tức thời). Trong khoảng thời gian trễ T-bit có giá trị 0. Khi hết thời gian trễ, tức là khi T-word = 0, T-bit có giá trị bằng 1.

  Khoảng thời gian trễ chính là khoảng thời gian giữa thời điểm xuất hiện sườn lên của tín hiệu đầu vào và sườn lên của T-bit.

  Với bộ timer trễ theo sườn lên có nhớ, thời gian trễ vẫn được tính cho dù lúc đó tín hiệu đầu vào đã về 0.

- **SP: Timer Xung**

Cú pháp **SP <tên timer>**

Thời gian giữ trễ được bắt đầu tính từ khi có sườn lên của tín hiệu đầu vào (hoặc khi có sườn lên của tín hiệu enable đồng thời tín hiệu vào bằng 1), tức là ở ngay thời điểm đó giá trị PV (giá trị đặt trước) được chuyển vào thanh ghi T-word (CV giá trị tức thời). Trong khoảng thời gian trễ ,tức là khi T-word có giá trị ≠ 0, T-bit có giá trị bằng 1. Ngoài thời gian trễ T-bit có giá trị bằng 0.

Nếu chưa hết thời gian trễ mà tín hiệu đầu vào về 0 thì giá trị T-bit và t-word cũng về 0.

- **SE: Timer giữ độ rộng xung**

Cú pháp **SE <Tên timer>**

Thời gian giữ trễ được bắt đầu tính từ khi có sườn lên của tín hiệu đầu vào (hoặc khi có sườn lên của tín hiệu enable đồng thời tín hiệu vào bằng 1), tức là ở ngay thời điểm đó giá trị PV (giá trị đặt trước) được chuyển vào thanh ghi T-word (CV giá trị tức thời). Trong khoảng thời gian trễ, tức là khi T-word có giá trị ≠ 0, T-bit có giá trị bằng 1. Ngoài thời gian trễ T-bit có giá trị bằng 0.

Nếu chưa hết thời gian trễ mà tín hiệu đầu vào về 0 thì thời gian trễ vẫn được tính tiếp tục, tức là T-bit và T-word không về 0 theo tín hiệu đầu vào.

- **SF: Timer mở mạch chậm**

Cú pháp **SF <Tên timer>**

Thời gian giữ trễ được bắt đầu tính từ khi có sườn lên của tín hiệu đầu vào (hoặc khi có sườn lên của tín hiệu enable đồng thời tín hiệu vào bằng 1), tức là ở ngay thời điểm đó giá trị PV (giá trị đặt trước) được chuyển vào thanh ghi T-word (CV giá trị tức thời). Trong khoảng thời gian trễ, tức là khi T-word có giá trị ≠ 0, T-bit có giá trị bằng 1. Ngoài thời gian trễ T-bit có giá trị bằng 0.

- **Khai báo tín hiệu xóa (reset)**

Cú pháp **A <Địa chỉ bit>**

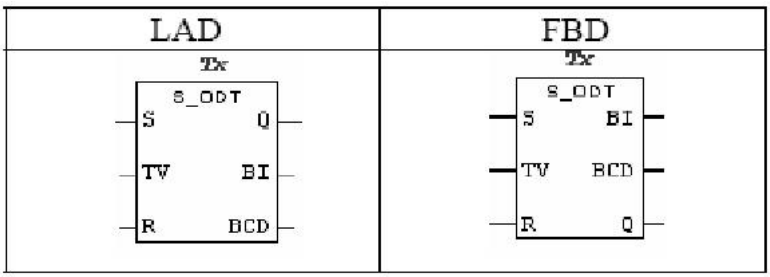
R <Tên timer>

Toán hạng thứ nhất "Địa chỉ bit" xác định tín hiệu sẽ được sử dụng làm tín hiệu chủ động xóa cho timer có tên trong toán hạng thứ hai. Khi tín hiệu xóa bằng 1, T-word (thanh ghi CV) và T-bit cùng đồng thời được đưa về 0.

Nếu tín hiệu xóa bằng 0, timer sẽ chờ được kích lại.

1.7.3 Khai báo Timer trong LAD và FBD:

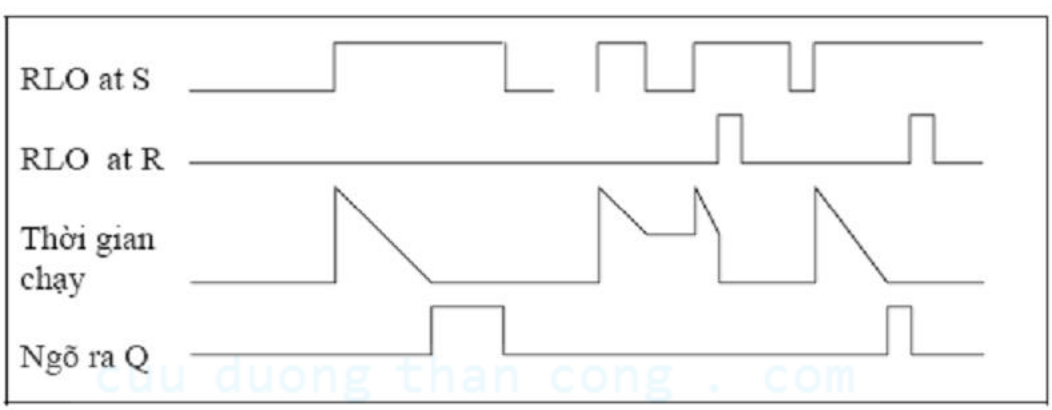
- **Timer đóng mạch chậm (SD):**



Bảng khai báo thông số Timer đóng chậm:

Thông số	Kiểu dữ liệu	Toán hạng
X (Số hiệu của timer,tuỳ loại CPU)	TIMER	I,Q,L,M,D
S (ngõ vào khởi động)	BOOL	I,Q,M,L,D
TV (Giá trị đặt trước:từ 0 đến 999)	S5TIME	I,Q,M,L,D
R (Ngõ vào reset)	BOOL	I,Q,M,L,D
Q (Ngõ ra)	BOOL	I,Q,M,L,D
BI (Giá trị hiện hành timer dạng integer)	WORD	I,Q,M,L,D
BCD (Giá trị hiện hành timer dạng BCD)	WORD	I,Q,M,L,D

Giản đồ thời gian Timer đóng chậm:



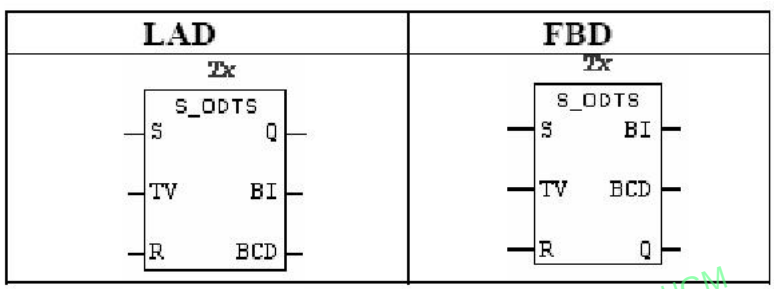
Khởi động: Timer khởi động khi RLO tại ngõ vào S thay đổi từ 0 lên 1. Timer bắt đầu chạy với giá trị thời gian rõ ràng đặt tại ngõ vào TV miễn là trạng thái ngõ vào S =1.

Xoá: Khi RLO reset ngõ vào "R" là 1, thì giá trị thời gian hiện hành và độ phân giải bị xoá và ngõ ra Q ở trạng thái Reset.

Ngõ ra digital: Giá trị thời gian hiện hành có thể đọc như một số nhị phân tại ngõ ra BI và BCD. Giá trị thời gian hiện hành là giá trị ban đầu của TV trừ đi giá trị thời gian đã hoạt động của timer, tính từ khi timer được khởi động.

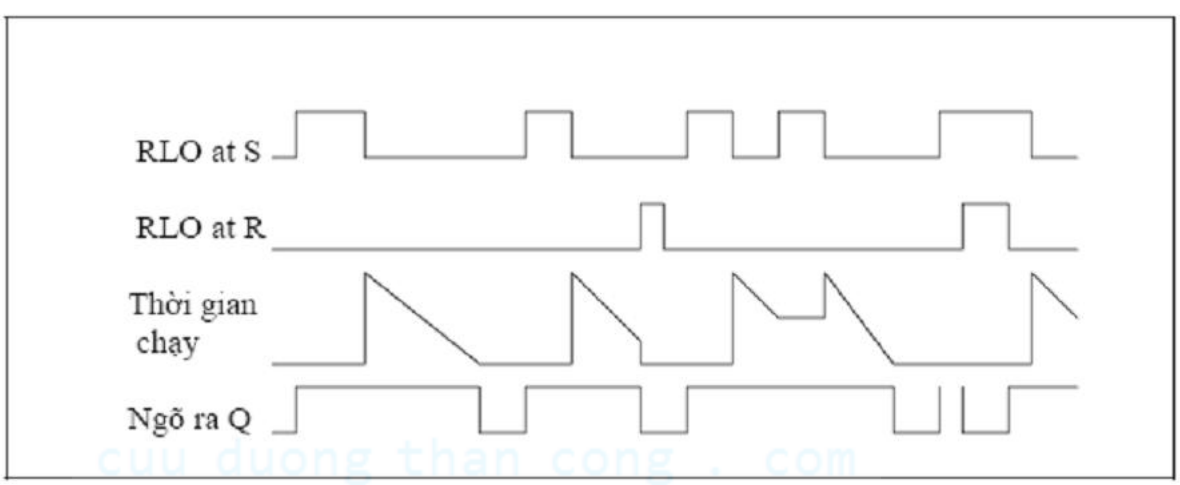
Ngõ ra Binary: Tín hiệu tại ngõ ra "Q" là "1", sau khi timer đã chạy hết, không có lỗi và ngõ vào "S" có tín hiệu ở trạng thái "1". Khi timer đang hoạt động, nếu tín hiệu ở ngõ vào "S" thay đổi từ "1" xuống "0", thì timer ngưng hoạt động. Trong trường hợp này ngõ ra Q có trạng thái tín hiệu 0.

- **Timer đóng mạch chậm có nhớ (SS):**



Với các thông số, kiểu dữ liệu và toán hạng khai báo giống như dang LAD và FBD của timer đóng mạch chậm (SD).

Giản đồ thời gian Timer đóng chậm có nhớ:

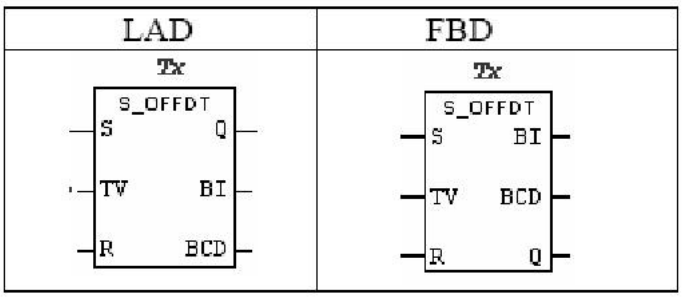


Khởi động: Timer khởi động khi RLO ở ngõ vào S thay đổ từ 0 đến 1. Timer bắt đầu hoạt động với giá trị thời gian xác định rõ ràng tại ngã vào TV và tiếp tục hoạt động thậm chí nếu tín hiệu ngõ vào S thay đổi thành 0 trong suốt thời gian đó. Nếu tín hiệu tại ngõ vào S thay đổi từ 0 đến 1 trong khi timer đang hoạt động, thì timer sẽ khởi động mới lại.

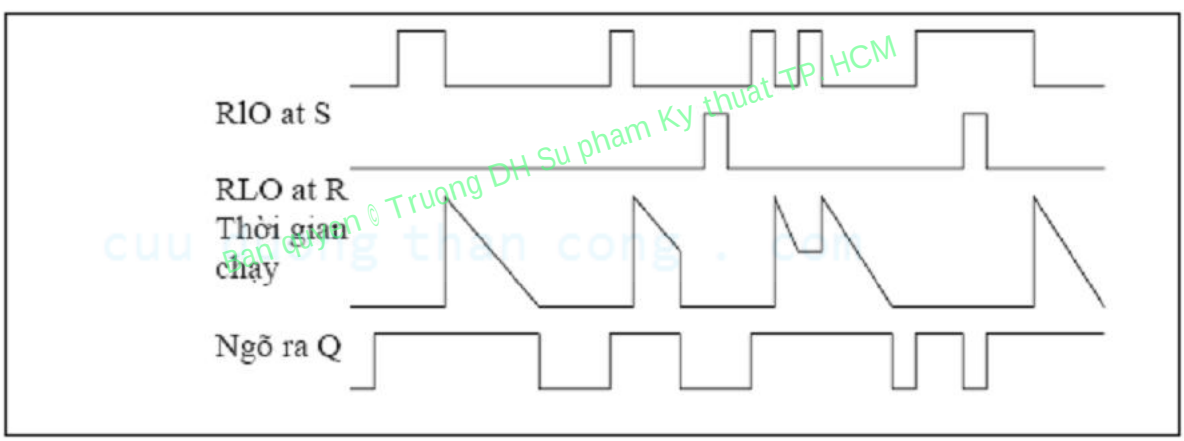
Reset: Khi RLO tại ngõ vào R là 1 thì giá trị thời gian hiện hành và độ phân giải bị xoá và ngõ ra Q ở trạng thái Reset.

Ngõ ra nhị phân: Trạng thái tín hiệu ngõ ra Q là 1 sau khi timer đã hoạt động không bị lỗi, thì không cần chú ý đến trạng thái tín hiệu ngõ vào S là 1 hay 0.

- **Timer mở mạch chậm (OFF Delay, SF):**



Giản đồ thời gian của Timer mở chậm:



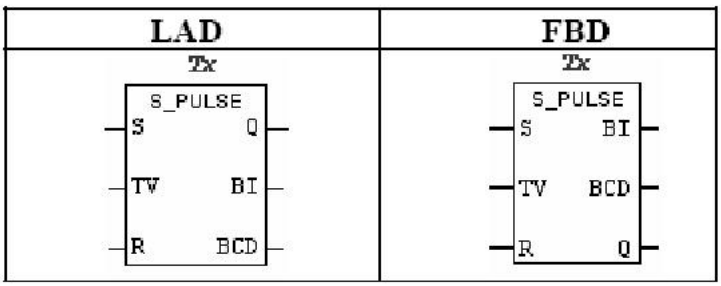
Khởi động: Timer khởi động khi RLO ở ngõ vào "S" thay đổi từ "1" đến "0". Sau khi timer đã hoạt động xong, thì ngõ ra Q sẽ chuyển đổi về "0". Nếu trạng thái tín hiệu ngõ vào "S" thay đổi từ 0 đến 1 trong khi timer đang hoạt động, thì timer sẽ dừng và thời gian kế tiếp trạng thái tín hiệu của S thay đổi từ 1 thành 0 nó sẽ bắt đầu lại từ đầu.

Reset : Khi RLO ngõ vào R là 1 thì giá trị thời gian hiện hành và độ phân giải bị xoá và ngõ ra Q bị reset. Nếu cả hai ngõ vào (S và R) có cùng trạng thái tín hiệu 1, thì ngõ ra Q không được set cho đến khi ngõ reset trở về 0.

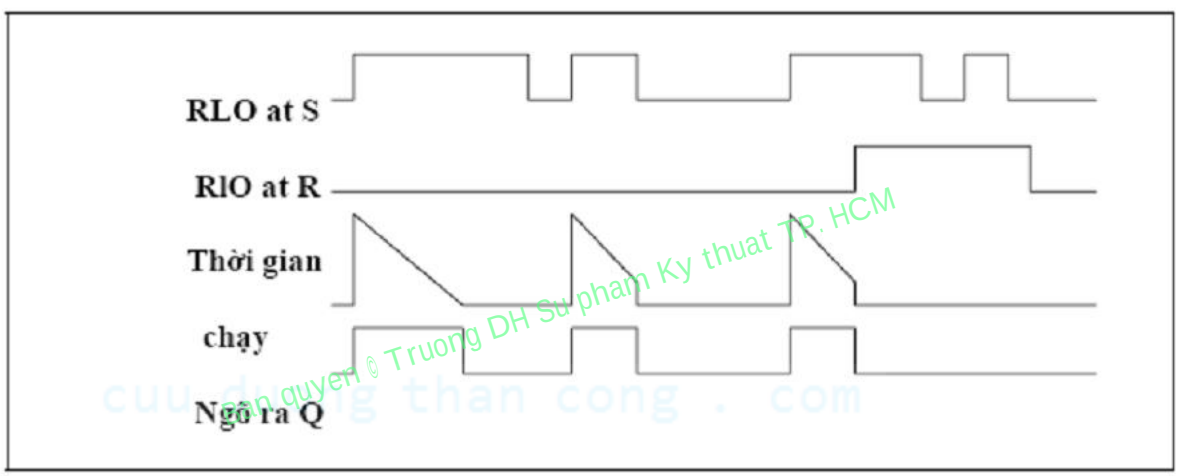
Ngõ ra nhị phân: Ngõ ra Q được kích hoạt khi RLO tại ngõ vào S thay đổi từ 0 đến 1.

Nếu ngõ vào S không được kích hoạt thì ngõ ra Q vẫn có trạng thái tín hiệu 1 cho đến khi thời gian lập trình được hoàn thành.

- **Timer Xung (Pulse, SP):**



Giản đồ thời gian của Timer xung:



Khởi động: Timer khởi động khi RLO tại ngõ vào S thay đổi từ 0 đến 1. Ngõ ra Q cũng đặt thành 1.

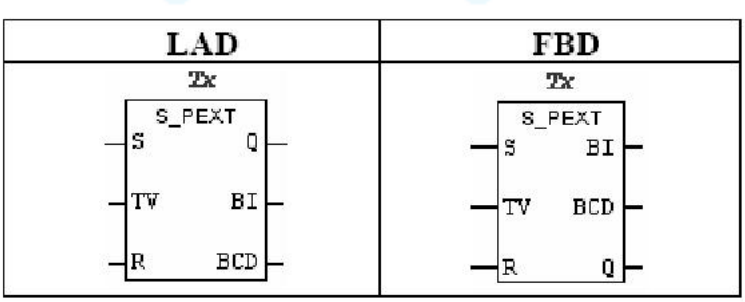
Reset: Ngõ ra Q bị reset khi:

Timer đã hoạt động xong, hoặc

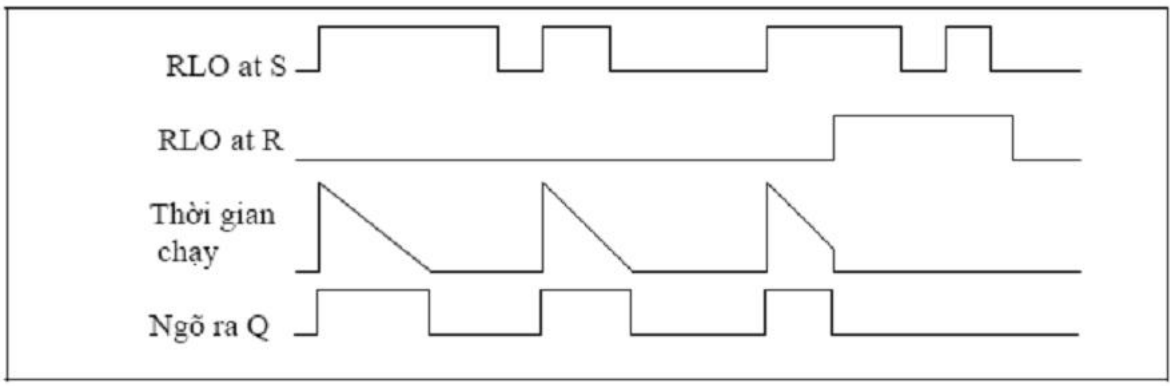
Tín hiệu start chuyển đổi từ 1 đến 0, hoặc

Ngõ vào reset R có trạng thái tín hiệu 1.

- **Timer giữ độ rộng xung (SE):**



Giản đồ thời gian Timer giữ độ rộng xung:



Khởi động : Timer hoạt động khi RLO tại ngõ vào S thay đổi từ 0 đến 1. Ngõ ra Q cũng được set thành 1. Trạng thái tín hiệu ngõ ra Q cũng vẫn là 1, mặc dù tín hiệu ngõ vào S thay đổi thành 0. Nếu tín hiệu ngõ vào start lại thay đổi từ 0 đến 1 trong khi timer đang hoạt động, thì timer sẽ khởi động lại.

Reset: Ngõ ra Q bị reset khi:

Timer đã hoạt động xong, hoặc

Ngõ vào reset R có trạng thái tín hiệu 1.

1.7.4 Sử dụng Timer theo lệnh bit:

Tất cả những chức năng timer cũng có thể được khởi động với những lệnh bit đơn giản. Sự giống nhau và khác nhau giữa phương pháp và những chức năng timer được đưa ra như sau:

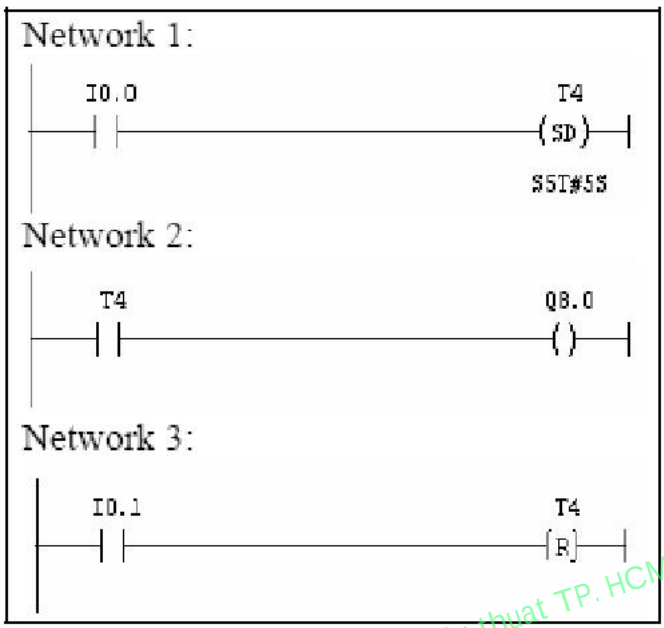
- **Giống nhau:**
  - Điều kiện khởi động ngõ vào S.
  - Đặt trước giá trị thời gian.
  - Điều kiện reset ngõ vào R .
  - Đáp ứng tín hiệu tại ngõ ra Q.

- **Khác nhau (trong LAD và FBD)**

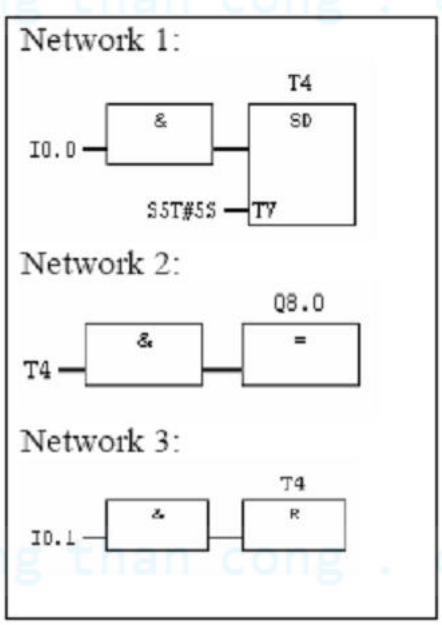
  Không có khả năng kiểm tra giá trị hiện hành, không có ngõ ra BI và BCD.

Ví dụ:

- Dạng LAD:



- Dạng FBD:



- Dạng STL:

Network 1:

```
A       I0.0
L S5T#5S
SD T4
```

Netwok 2:

```
A T4
= Q8.0
```

Network 3:

```
A I0.1
R T4
```

Timer T4 sẽ được kích nếu I0.0 lên mức 1. Sau 5s, T4 đóng làm Q8.0 lên mức 1. Timer được reset nếu I0.1 lên mức 1.

1.8 LỆNH VỀ COUNTER:

1.8.1 Giới thiệu Counter:

Counter là bộ đếm thực hiện chức năng đếm sườn xung của các tín hiệu đầu vào. S7-300 có tối đa 256 counter (tuỳ loại CPU), ký hiệu Cx, trong đó x là số nguyên trong khoảng từ 0 đến 255. Những bộ đếm của S7-300 đều có thể đồng thời đếm tiến theo sườn lên của một tín hiệu vào thứ nhất, được ký hiệu là CU (count up) và đếm tiến theo sườn lên của tín hiệu vào thứ hai, ký hiệu CD (count down).

Thông thường bộ đếm chỉ các sườn lên của tín hiệu CU và CD, song cũng có thể được mở rộng để đếm cả mức tín hiệu của chúng bằng cách sử dụng thêm tín hiệu enable.

Nếu có tín hiệu enable, bộ đếm sẽ đếm tiến khi xuất hiện sườn lên của tín hiệu enable đồng thời tại thời điểm CU có mức tín hiệu 1. Tương tự bộ đếm sẽ đếm lùi khi có sườn lên của tín hiệu enable và tại thời điểm CD có mức tín hiệu 1.

Số sườn xung đếm được ghi vào thanh ghi 2 byte của bộ đếm, gọi là thanh ghi C word. Nội dung của C-Word được gọi là giá trị đếm tức thời của bộ đếm và ký hiệu bằng CV (current value). Bộ đếm báo trạng thái của C-Word ra ngoài thông qua chân C-bit của nó. Nếu CV ≠ 0, C-Bit có giá trị 1. Ngược lại khi CV = 0 C-bit nhận giá trị 0.

CV luôn là 1 giá trị không âm. Bộ đếm sẽ không đếm lùi khi CV = 0.

Khác với timer, giá trị đặt trước PV của bộ đếm chỉ được chuyển vào C-Word tại thời điểm xuất hiện sườn lên của tín hiệu đặt (set S).

Bộ đếm có thể được xoá chủ động bằng tín hiệu xóa (reset). Khi bộ đếm được xóa, cả C-Word và C-bit đều nhận giá trị 0.

1.8.2 Khai báo sử dụng Counter:

Sử dụng Counter cần khai báo các thông số sau:

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

- **Khai báo tín hiệu enable:**

Cú pháp **A <Địa chỉ bit>**

FR <Tên counter>

Toán hạng thứ nhất "Địa chỉ bit" xác định tín hiệu sẽ được sử dụng làm tín hiệu kích đếm cho bộ đếm có tên trong toán hạng thứ hai.

Tên của bộ đếm có dạng Cx, với $0 \leq x \leq 255$.

- **Khai báo tín hiệu đầu vào CU**:

Cú pháp **A <Địa chỉ bit>**

CU <tên counter>

Toán hạng thứ nhất "Địa chỉ bit" xác định tín hiệu mà sườn lên của nó được bộ đếm với tên cho trong toán hạng thứ hai đếm tiến.

Mỗi khi xuất hiện một sườn lên của tín hiệu, bộ đếm sẽ tăng nội dung thanh ghi Cword (CV) lên 1 đơn vị.

- **Khai báo tín hiệu đầu vào CD**:

Cú pháp **A <Địa chỉ bit>**

CD <tên counter>

Toán hạng thứ nhất "Địa chỉ bit" xác định tín hiệu mà sườn lên của nó được bộ đếm với tên cho trong toán hạng thứ hai đếm lùi.

Mỗi khi xuất hiện mốt sườn lên của tín hiệu, bộ đếm sẽ giảm nội dung thanh ghi Cword (CV) xuống 1 đơn vị.

- **Khai báo tín hiệu đặt SET:**

Cú pháp **A <Địa chỉ bit>**

L C#<hằng số>

S <hằng số>

Toán hạng thứ nhất "Địa chỉ bit" xác định tín hiệu mỗi khi xuất hiện sườn.

Hằng số cho trong lệnh thứ hai dưới dạng BCD sẽ được chuyển vào thanh ghi C_word của bộ đếm có tên trong toán hạng thứ 3.

- **Khai báo tín hiệu đặt RESET:**

Cú pháp **A <Địa chỉ bit>**

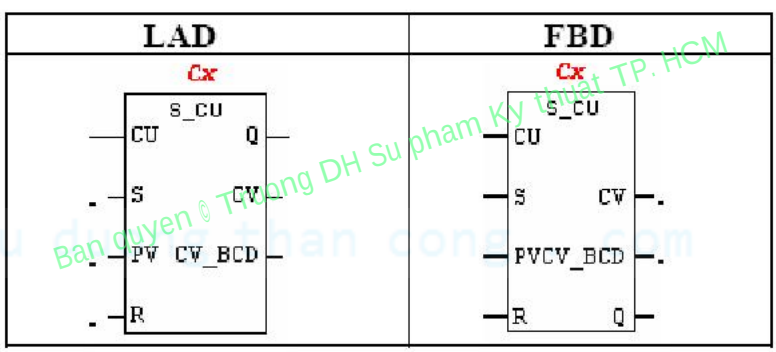
R <Tên counter>

Toán hạng thứ nhất "Địa chỉ bit" xác định tín hiệu mỗi khi xuất hiện sườn lên.

Thanh ghi C-word của bộ đếm có tên trong toán hạng thứ hai sẽ được xóa về 0.

1.8.3 Khai báo counter trong LAD và FBD:

- **Counter đếm lên:**



Bảng khai báo các thông số của Counter:

Thông số	Kiểu dữ liệu	Toán hạng
x (Số hiệu của counter, tùy vào loại CPU)	TIMER	T
CU (Ngõ vào bộ đếm lên)	BOOL	I,Q,M,L,D
S (Ngõ vào đặt giá trị đếm)	BOOL	I,Q,M,L,D,T,C
PV (Giá trị đặt trước: từ 0 đến 999)	WORD	I,Q,M,L,D,CONST
R (Ngõ vào Reset)	BOOL	I,Q,M,L,D,T,C
Q (Ngõ ra, trạng thái của counter)	BOOL	I,Q,M,L,D
CV (Giá trị hiện hành counter dạng integer)	WORD	I,Q,M,L,D
CV_BCD (Giá trị hiện hành counter dạng BCD)	WORD	I,Q,M,L,D

Đếm lên: Khi RLO tại ngõ vào CU thay đổi từ 0 đến 1 giá trị đếm hiện hành tăng lên 1. (tối đa = 999).

Set bộ đếm: Khi RLO tại ngõ vào S thay đổi từ 0 lên 1 bộ đếm được đặt với giá trị tại ngõ vào PV.

Reset bộ đếm: Khi RLO =1 counter được đặt về 0. Khi điều kiện reset được thoả mãn thì counter không thể đặt và không thể đếm.

PV: Giá trị đặt trước từ (0 ..999) được xác định tại ngõ vào PV ở dạng BCD, PV là hằng số đếm (C#...) qua giao tiếp dữ liệu dạng mã BCD.

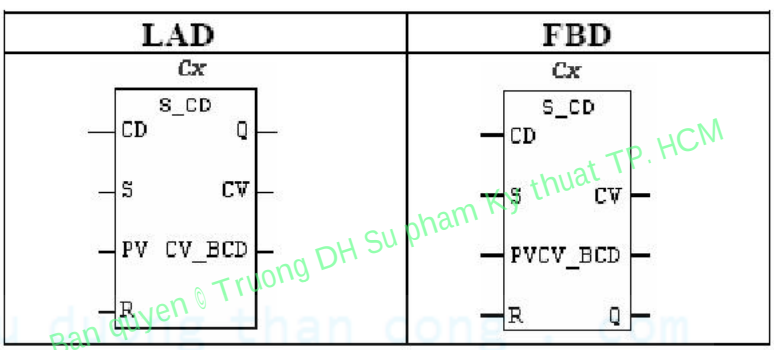
CV/CV-BCD: Giá trị counter có thể là một số nhị phân hoặc số BCD được nạp vào ô tích luỹ và từ đó chuyển tới các địa chỉ khác.

Q: Tình trạng tín hiệu của counter có thể kiểm tra tại ngõ ra Q.

Giá trị đếm bằng 0 suy ra Q = 0.

Giá trị đếm khác 0 suy ra Q = 1.

- **Counter đếm xuống:**



Đếm xuống: Khi RLO tại ngõ vào CD thay đổi từ 0 lên 1 giá trị đếm hiện hành giảm xuống 1 (tối thiểu bằng 0).

Set bộ đếm: Khi RLO tại ngõ vào S thay đổi từ 0 lên 1 bộ đếm được đặt với giá trị tại ngõ vào CV.

Reset bộ đếm: Khi RLO =1 counter được đặt về 0. Khi điều kiện reset được thoả mãn thì counter không thể đặt và không thể đếm.

PV: Giá trị đặt trước từ (0 ..999) được xác định tại ngõ vào PV ở dạng BCD.

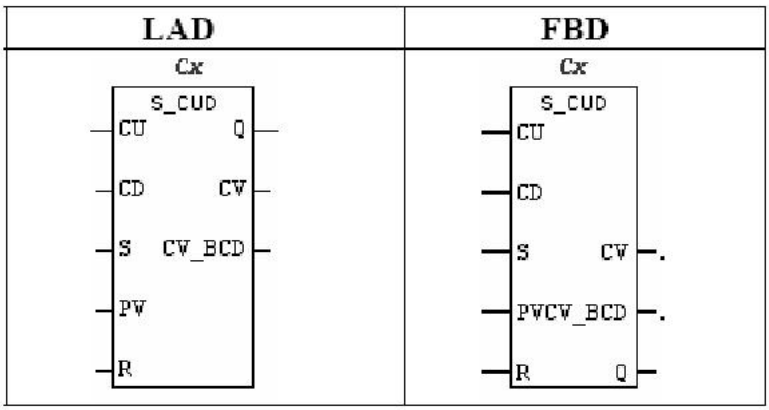
CV/CV-BCD: Giá trị counter có thể là một số nhị phân hoặc số BCD được nạp vào ô tích luỹ và từ đó chuyển tới các địa chỉ khác.

Q: Tình trạng tín hiệu của counter có thể kiểm tra tại ngõ ra Q

Giá trị đếm bằng 0 suy ra Q = 0.

Giá trị đếm khác 0 suy ra Q = 1

- **Counter đếm lên/xuống:**



Giá trị đếm: Mỗi một bộ đếm chiếm một word 16 bit trong vùng nhớ dữ liệu hệ thống, dùng lưu trữ giá trị đếm cho counter từ (0..999) trong mã nhị phân.

Đếm lên: Khi RLO tại ngõ vào CU thay đổi từ 0 đến 1 giá trị đếm hiện hành tăng lên 1, max = 999.

Đếm xuống: Khi RLO tại ngõ vào CD thay đổi từ 0 lên 1 giá trị đếm hiện hành giảm xuống 1, min=0.

Set bộ đếm: Khi RLO tại ngõ vào S thay đổi từ 0 lên 1 bộ đếm được đặt với giá trị tại ngõ vào CV.

Reset bộ đếm: Khi RLO =1 counter được đặt về 0. Khi điều kiện reset được thoả mãn thì counter không thể đặt và không thể đếm.

PV: Giá trị đặt trước từ (0 ..999) được xác định tại ngõ vào PV ở dạng BCD.

Giá trị đặt vào PV là hằng số đếm (C#...) qua giao tiếp dữ liệu dạng mã BCD.

CV/CV-BCD: Giá trị counter có thể là một số nhị phân hoặc số BCD được nạp vào ô tích luỹ và từ đó chuyển tới các địa chỉ khác.

Ngõ ra Q : Tình trạng tín hiệu của counter có thể kiểm tra tại ngõ ra Q.

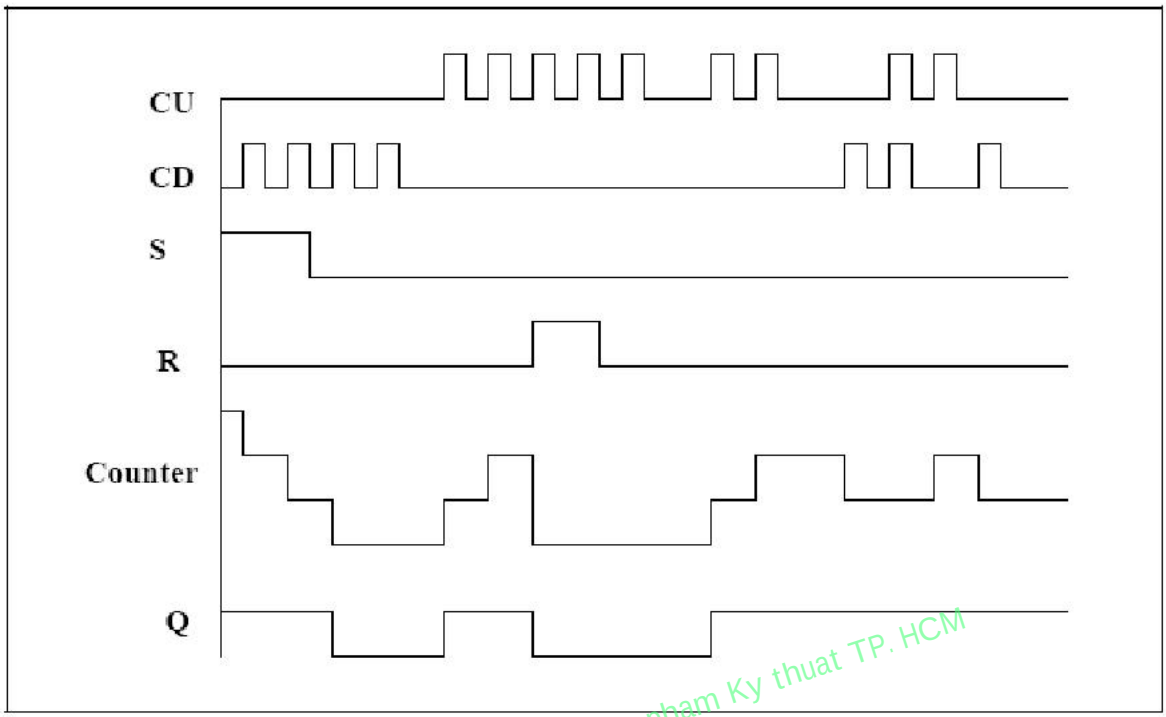
Giá trị đếm bằng 0 suy ra Q = 0.

Giá trị đếm khác 0 suy ra Q = 1.

Các loại bộ đếm:

- S_CU Bộ đếm lên (chỉ đếm lên).
- S_CD Bộ đếm xuống (chỉ đếm xuống).
- S_CUD Bộ đếm lên /đếm xuống.

- **Giản đồ thời gian của Counter:**



1.8.4 Sử dụng Counter theo lệnh bit:

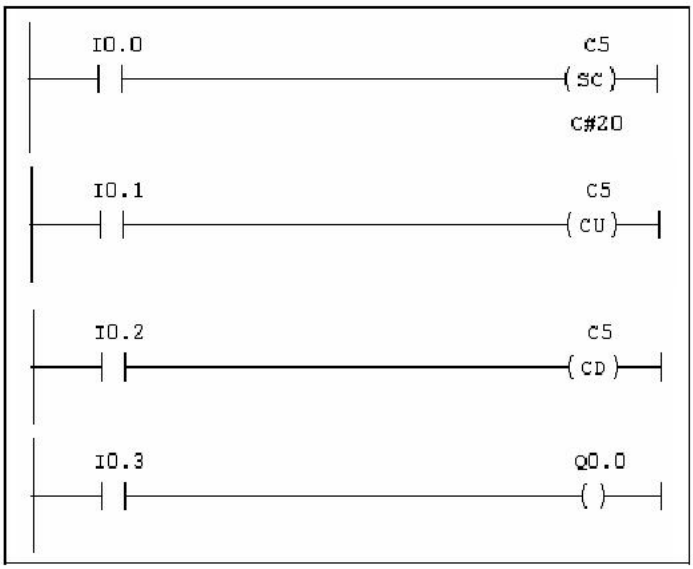
Tất cả những chức năng của counter cũng có thể thực hiện với những câu lệnh bit đơn giản. Sự giống nhau và khác nhau giữa phương pháp này và những chức năng counter được đưa ra trong các phần trên như sau:

- **Giống nhau:**
  - Điều kiện set ở ngõ vào SC.
  - Giá trị đặt trước của bộ đếm.
  - RLO thay đổi ngõ vào CU.
  - RLO thay đổi ngõ vào CD.

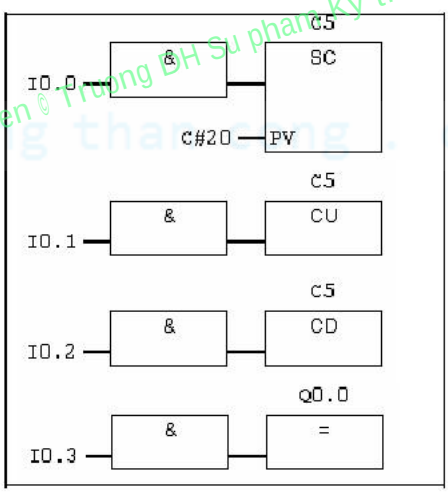
- **Khác nhau:**
  - Không có khả năng kiểm tra giá trị đếm hiện hành.
  - Ngõ ra nhị phân Q không thể hiện được bằng biểu đồ.

Ví dụ sau minh họa chương trình điều khiển counter theo bit.

- Dạng LAD:



- Dạng FBD:



- Dạng STL:

```
A     I     0.0
L     C#20
S     C     5

A     I     0.1
CU    C     5

A     I     0.2
CD    C     5

A     I     0.3
=     Q     0.0
```