

LẬP TRÌNH PLC

S7-1200

Totally Integrated Automation Portal

Phần cơ bản

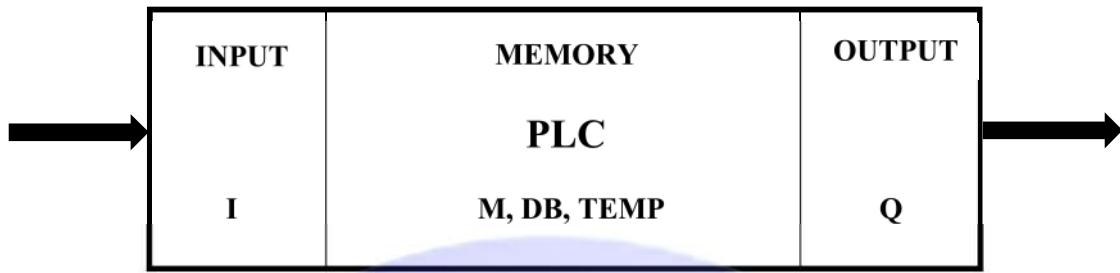


Mục lục

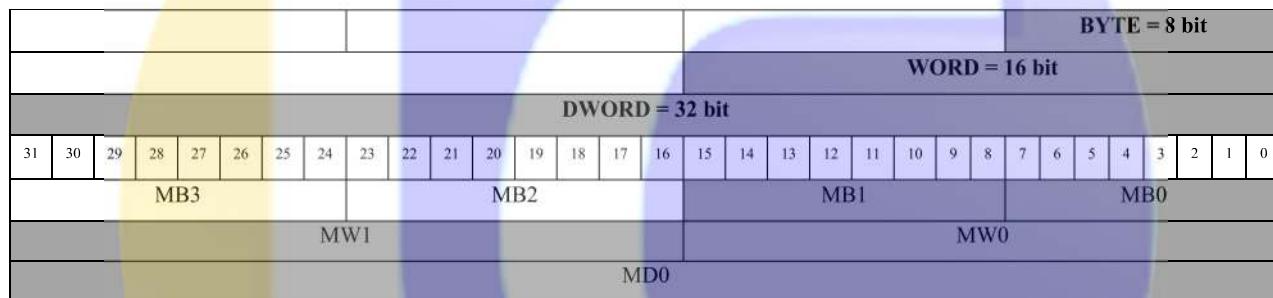
.....	1
I. CẤU TRÚC VÙNG NHỚ PLC	4
1.1. Cấu trúc thanh ghi	4
1.1.1. Bit	4
1.1.2. Byte	5
1.1.3. Word	6
1.1.4. Double Word	6
1.2. Kiểu dữ liệu	7
II. KHỐI BLOCK	9
2.1. Khối hàm OB	9
2.1.1. Program cycle	9
2.1.2. Startup	9
2.1.3. Cyclic Interrupt	9
2.1.4. Hardware Interrupt	9
2.2. Khối hàm FC	9
2.3. Khối hàm FB	9
2.4. Bộ nhớ DB	10
III. TẬP LỆNH CƠ BẢN	11
3.1. Tiếp điểm NO, NC	11
3.2. Lệnh SET và RESET	11
3.2.1. Lệnh Set	12
3.2.2. Lệnh RESET	12
3.3. Xung sườn lên (P) và xung sườn xuống (N)	13
3.4. Tập lệnh so sánh	14
3.4.1. So sánh bằng	14
3.4.2. So sánh hơn hoặc bằng	15
3.4.3. So sánh hơn	15
3.4.4. So sánh nhỏ hơn hoặc bằng	16
3.4.5. So sánh nhỏ hơn	16

3.4.6. So sánh không bằng	17
3.5. Tập lệnh IN_RANGE và OUT_RANGE.....	17
3.6. Tập lệnh tính toán Cộng, Trừ, Nhân, Chia và MOVE	19
3.6.1. Lệnh ADD	19
3.6.2. Lệnh SUB	20
3.6.3. Lệnh MUL.....	20
3.6.4. Lệnh DIV.....	21
3.6.5. Lệnh MOVE.....	22
3.7. Tập lệnh Timer	22
3.7.1. Lệnh TP	22
3.7.2. Lệnh TON	24
3.7.3. Lệnh TOF	25
3.8. Tập lệnh Counter.....	26
3.8.1. CTU (Counter Up)	26
3.8.2. CTD (Counter Down)	27
3.8.3. CTUD (Counter Up Down)	29

I. CẤU TRÚC VÙNG NHỚ PLC



Cấu trúc vùng nhớ trong PLC



1.1. Cấu trúc thanh ghi

- 1 đoạn thanh ghi = 1 byte (B) = 8 bit
- 1 word (W) = 2 byte = 16 bit
- 1 DoubleWord (DW) = 4 byte = 32bit

1.1.1. Bit

- Bit là đơn vị nhỏ nhất trong bộ nhớ PLC, 1 bit chỉ có giá trị 0 (**FALSE**) hoặc 1 (**TRUE**).
- Cách gọi bit:
 - I0.0 = bit số 0 của đoạn thanh ghi số 0 trong vùng nhớ I
 - I1.6 = bit số 6 của đoạn thanh ghi số 1 trong vùng nhớ I

- M10.7 = bit số 7 của đoạn thanh ghi số 10 trong vùng nhớ M
- Q2.0 = bit số 0 của đoạn thanh ghi số 2 trong vùng nhớ Q

Lưu ý : Không có bít số 8 của vùng nhớ bất kỳ nào.

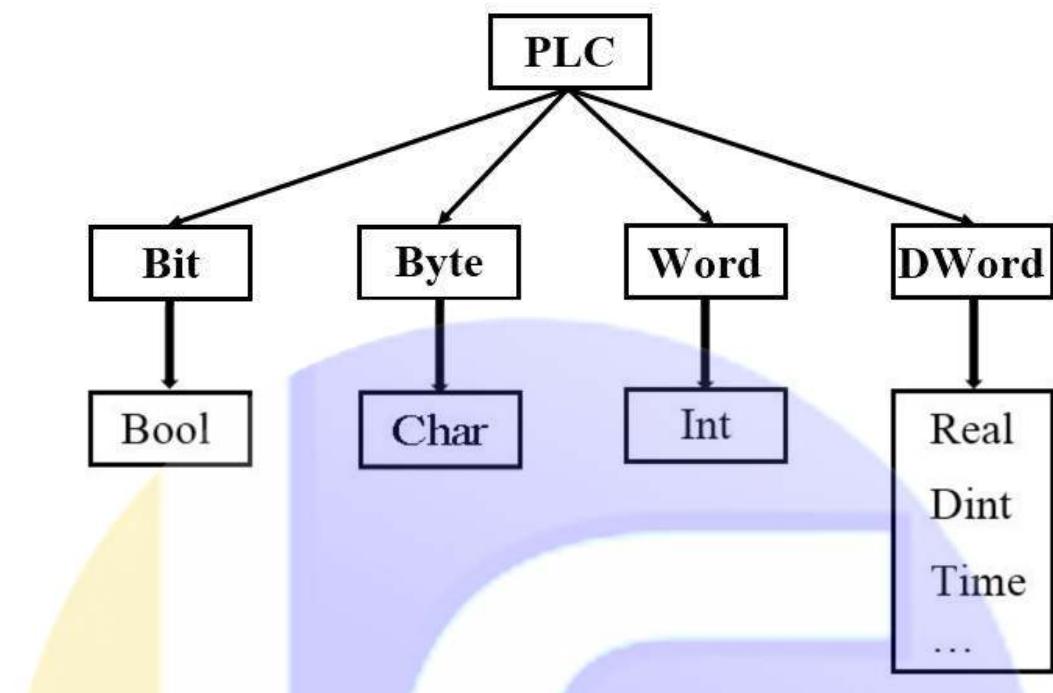
1.1.2. Byte

- 1 byte chứa 8 bit.

Suy ra, giá trị 1 byte trong khoảng: 0 - (2⁸-1)

Cách gọi byte :

- IB0 = byte số 0 của vùng nhớ M
= 8 bit của byte số 0 trong vùng nhớ I
- MB4 = byte số 4 của vùng nhớ M
= 8 bit của byte số 4 trong vùng nhớ M
- QB1 = byte số 1 của vùng nhớ Q
= 8 bit của byte số 1 trong vùng nhớ Q



1.1.3. Word

- 1 Word = 2 byte = 16 bit

Suy ra, giá trị 1 Word trong khoảng: $0 - (2^{16}-1)$

Cách gọi Word :

- $IW0 = IB0 + IB1$
= 8 bit của byte IB0 + 8 bit của IB1
- $MW7 = MB7 + MB8$
= 8 bit của byte MB7 + 8 bit của MB8
- $QW2 = QB2 + QB3$
= 8 bit của byte QB7 + 8 bit của QB8

1.1.4. Double Word

- 1 Dword = 2 Word = 4 byte = 32bit

Suy ra, giá trị 1 Double Word trong khoảng: $0 - (2^{32}-1)$

Cách gọi Double word :

- $ID0 = IW0 + IW2$
 $= IB0 + IB1 + IB2 + IB3$
- $MD2 = MW2 + MW4$
 $= MB2 + MB3 + MB4 + MB5$
- $QD1 = QW1 + QW3$
 $= QB1 + QB2 + QB3 + QB4$

1.2. Kiểu dữ liệu

Kiểu dữ liệu	Kích thước (bit)	Phạm vi
Bool	1	0 đến 1
Byte	8	16#00 đến 16#FF
Word	16	16#0000 đến 16#FFFF
DWord	32	16#0000 0000 đến 16#FFFF FFFF
Char	8	16#00 đến 16#FF
SInt	8	-128 đến 127
USint	8	0 đến 255
Int	16	-32 768 đến 32 767
UInt	16	0 đến 65535
DInt	32	-2 147 483 648 đến 2 147 483 647
UDint	32	0 đến 4 294 967 295
Real	32	$+- 1,18 \times 10^{-38}$ đến $+- 3,40 \times 10^{38}$
LReal	64	$+- 2,23 \times 10^{-308}$ đến $+- 1,79 \times 10^{308}$
Time	32	T# -24d_20h_31m_23s_648ms đến T#

		24d_20h_31m_23s_647ms Tương tự -2 147 483 648ms đến 2 147 483 647
DTL	12	DTL#1970-01-01-00:00:00.0 đến DTL#2554- 12-31-23:59:59.999



II. KHỐI BLOCK

2.1. Khối hàm OB

2.1.1. Program cycle

- Khối hàm chính của chương trình, thực thi vòng lặp của chương trình PLC theo chu kỳ.
- Khối hàm thực hiện vòng quét liên tục trong suốt quá trình.
- Là khối hàm có thể chứa các khối hàm bổ sung như FB, FC.

2.1.2. Startup

- OB Startup sẽ thực thi một lần khi chế độ hoạt động của PLC thay đổi từ STOP sang RUN.
- Sau khi hoàn thành, khối Program cycle sẽ bắt đầu thực thi.

2.1.3. Cyclic Interrupt

- Hàm ngắt theo chu kỳ thời gian đặt trước.

2.1.4. Hardware Interrupt

- Hàm ngắt theo tín hiệu phần cứng.

2.2. Khối hàm FC

- Là hàm con có cấu trúc tương tự hàm OB nhưng chỉ thực hiện khi có lệnh gọi ra từ hàm OB.
- Hàm FC không có bộ nhớ đệm, dữ liệu của hàm FB nếu không được lưu vào bộ nhớ riêng của PLC, dữ liệu sẽ không được giữ lại trong hàm FC.

2.3. Khối hàm FB

- Là hàm con có cấu trúc tương tự hàm OB nhưng chỉ thực hiện khi có lệnh gọi ra từ hàm OB.
- Hàm FB có kèm bộ nhớ đệm DB khi thực thi lệnh trên OB.
- Khi kết thúc lệnh FB thì dữ liệu vẫn được giữ lại trong bộ nhớ riêng của hàm FB.

2.4. Bộ nhớ DB

- Bộ nhớ DB có thể thực hiện lưu trữ giữ liệu như bộ nhớ M trong PLC.
- Bộ nhớ DB có thêm một số kiểu dữ liệu nâng cao mà bộ nhớ M không có (chi tiết xem thêm tại website : www.libcode.net).

Chú ý :

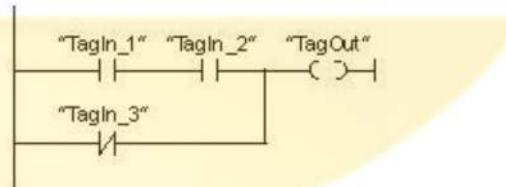
- *Cấu trúc các khối hàm OB, FC, FB như nhau nên khi thực hiện viết chương trình các lệnh viết là như nhau.*
- *Khối hàm FC, FB được sử dụng để làm gọn chương trình OB hơn.*
- *Bộ nhớ DB có thể sử dụng để làm gọn cấu trúc bộ nhớ của khối chương trình trong PLC thay vì sử dụng bộ nhớ M.*
- *Khối FB có bộ nhớ đệm DB riêng có thể sử dụng kèm với một số tập lệnh nâng cao mà chỉ khối hàm FB mới thực hiện được (chi tiết xem thêm tại website : www.libcode.net).*

III. TẬP LỆNH CƠ BẢN

3.1. Tiếp điểm NO, NC

- Tiếp điểm có 2 loại tiếp điểm : **tiếp điểm đầu vào** và **tiếp điểm đầu ra**.
 - ✓ **Tiếp điểm đầu vào:** là các yếu tố đầu vào như tiếp điểm đầu vào vật lý PLC, biến nhớ M, ...
 - **Tiếp điểm thường mở.**
 - **Tiếp điểm thường kín.**
 - ✓ **Tiếp điểm đầu ra:** là trạng thái logic đầu ra của 1 bit logic.
- Tiếp điểm chỉ có 2 trạng thái **True** và **False** tương ứng với mức logic **1** và **0**.
- Kiểu dữ liệu của tiếp điểm là kiểu **Bool**.
- Tiếp điểm thường mở NO (Normally Open) được đóng lại (ON) khi giá trị bit được gán bằng **True**.
- Tiếp điểm thường đóng NC (Normally Closed) được mở ra (ON) khi giá trị bit được gán bằng **True**.

Ví dụ:



Mô tả : “**TagOut**” đạt giá trị **True** khi thỏa mãn 1 trong 2 trường hợp:

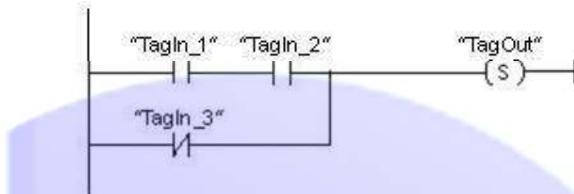
- Khi “**TagIn_1**” và “**TagIn_2**” cùng bằng **True**.
- Khi “**TagIn_3**” bằng **False**.

3.2. Lệnh SET và RESET

3.2.1. Lệnh Set

- Khi lệnh S (Set) được kích hoạt, giá trị dữ liệu ở địa chỉ OUT được đặt lên 1.

Ví dụ:



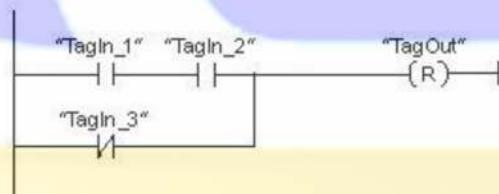
Mô tả : “TagOut” được Set giá trị True khi thỏa mãn 1 trong 2 trường hợp:

- Khi “TagIn_1” và “TagIn_2” cùng bằng True.
- Khi “TagIn_3” bằng False.

3.2.2. Lệnh RESET

- Khi lệnh R (Reset) được kích hoạt, giá trị dữ liệu ở địa chỉ OUT được đặt về 0.

Ví dụ:

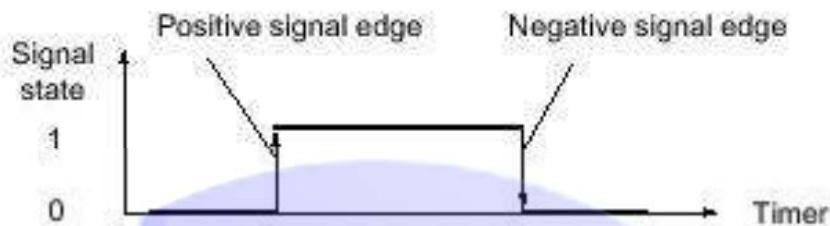


Mô tả : “TagOut” được Reset giá trị False khi thỏa mãn 1 trong 2 trường hợp:

- Khi “TagIn_1” và “TagIn_2” cùng bằng True.
- Khi “TagIn_3” bằng False.

Chú ý : Trong chương trình, khi cả 2 điều kiện SET và RESET 1 bit đều thỏa mãn thì sẽ ưu tiên lệnh RESET.

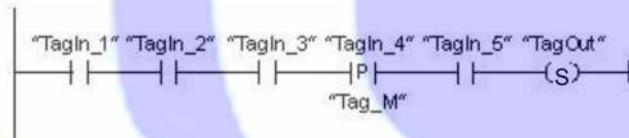
3.3. Xung sườn lên (P) và xung sườn xuống (N).



- Xung sườn lên P:

- Trạng thái của tiếp điểm này là “**TRUE**” khi có sự thay đổi tín hiệu từ OFF sang ON được phát hiện trên **bit** được gán.
- Trạng thái logic của tiếp điểm sau đó được kết hợp với dòng tín hiệu trong mạch để thiết lập trạng thái ngõ ra của dòng tín hiệu.

Ví dụ 1:



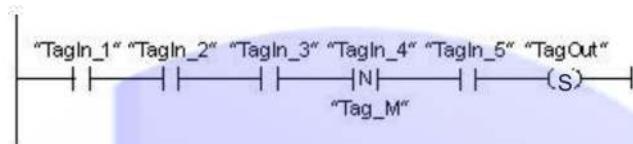
Mô tả : “*TagOut*” đạt giá trị **True** khi thỏa mãn tất cả các trường hợp sau:

- “*TagIn_1*”, “*TagIn_2*”, “*TagIn_3*” cùng bằng **True**.
- “*TagIn_4*” thay đổi từ **False** sang **True**. Trạng thái tín hiệu “*TagIn_4*” của lần quét trước được lưu vào bit bộ nhớ cạnh “*Tag_M*”
- “*TagIn_5*” bằng **True**.

- Xung sườn xuống N:

- Trạng thái của tiếp điểm này là “**TRUE**” khi một sự thay đổi tín hiệu từ ON sang OFF được phát hiện trên **bit** được gán.
- Trạng thái logic của tiếp điểm sau đó được kết hợp với dòng tín hiệu trong mạch để thiết lập trạng thái ngõ ra của dòng tín hiệu.

Ví dụ 2:



Mô tả : “*TagOut*” đạt giá trị **True** khi thỏa mãn tất cả các trường hợp sau:

- “*TagIn_1*”, “*TagIn_2*”, “*TagIn_3*” cùng bằng **True**.
- “*TagIn_4*” thay đổi từ **True** sang **False**. Trạng thái tín hiệu “*TagIn_4*” của lần quét trước được lưu vào bit bộ nhớ cạnh “*Tag_M*”
- “*TagIn_5*” bằng **True**.

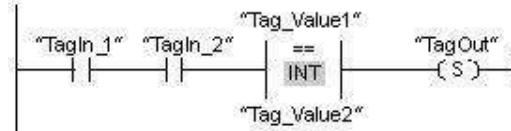
3.4. Tập lệnh so sánh

Thông số	Kiểu dữ liệu	Mô tả
Tag_Value_1 và Tag_Value_2	Int, Dint, Real, Time...	Các giá trị để so sánh

3.4.1. So sánh bằng

- Khối logic cho tín hiệu qua khi giá trị so sánh của khối logic thỏa mãn.

Ví dụ:



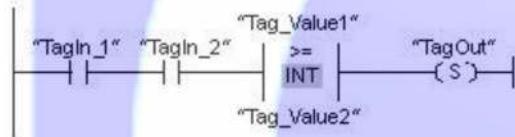
Mô tả : “TagOut” đạt giá trị **True** khi thỏa mãn tất cả các trường hợp sau:

- “TagIn_1”, “TagIn_2” cùng bằng **True**.
- “Tag_Value_1” = “Tag_Value_2”

3.4.2. So sánh hơn hoặc bằng

- Khối logic cho tín hiệu qua khi giá trị so sánh của khối logic thỏa mãn.

Ví dụ:



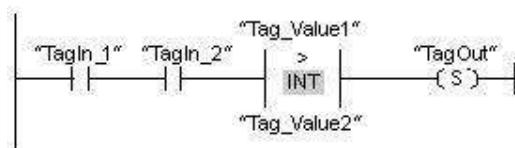
Mô tả : “TagOut” đạt giá trị **True** khi thỏa mãn tất cả các trường hợp sau:

- “TagIn_1”, “TagIn_2” cùng bằng **True**.
- “Tag_Value_1” \geq “Tag_Value_2”

3.4.3. So sánh hơn

- Khối logic cho tín hiệu qua khi giá trị so sánh của khối logic thỏa mãn.

Ví dụ:



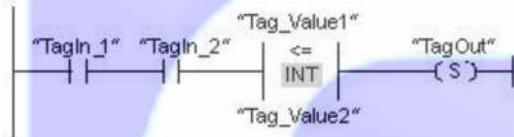
Mô tả : “TagOut” đạt giá trị **True** khi thỏa mãn tất cả các trường hợp sau:

- “TagIn_1”, “TagIn_2” cùng bằng **True**.
- “Tag_Value_1” > “Tag_Value_2”

3.4.4. So sánh nhỏ hơn hoặc bằng

- Khối logic cho tín hiệu qua khi giá trị so sánh của khối logic thỏa mãn.

Ví dụ:



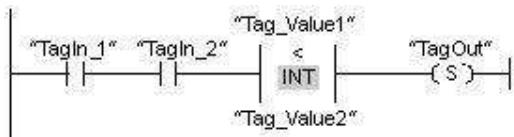
Mô tả : “TagOut” đạt giá trị **True** khi thỏa mãn tất cả các trường hợp sau:

- “TagIn_1”, “TagIn_2” cùng bằng **True**.
- “Tag_Value_1” <= “Tag_Value_2”

3.4.5. So sánh nhỏ hơn

- Khối logic cho tín hiệu qua khi giá trị so sánh của khối logic thỏa mãn.

Ví dụ:



Mô tả : “TagOut” đạt giá trị **True** khi thỏa mãn tất cả các trường hợp sau:

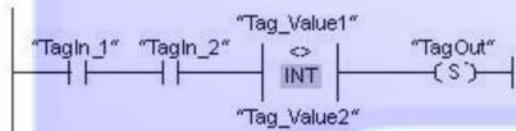
- “TagIn_1”, “TagIn_2” cùng bằng **True**.

- “Tag_Value_1” < “Tag_Value_2”

3.4.6. So sánh không bằng

- Khối logic cho tín hiệu qua khi giá trị so sánh của khối logic thỏa mãn.

Ví dụ:



Mô tả : “TagOut” đạt giá trị **True** khi thỏa mãn tất cả các trường hợp sau:

- “TagIn_1”, “TagIn_2” cùng bằng **True**.
- “Tag_Value_1” **không bằng** “Tag_Value_2”

3.5. Tập lệnh IN_RANGE và OUT_RANGE

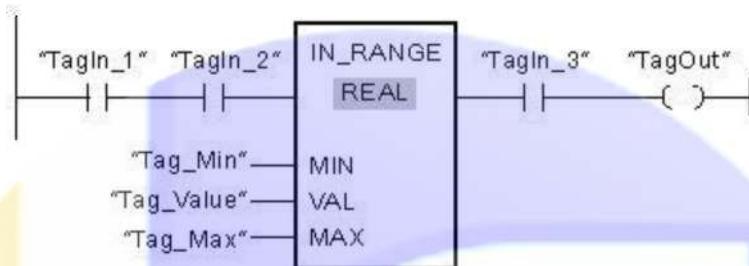
- Sử dụng các lệnh **IN_RANGE** hoặc **OUT_RANGE** kiểm tra trong một giá trị ngõ vào nằm **trong dải** hay **ngoài dải** giá trị định trước.
- Nếu sự so sánh thỏa mãn điều kiện thì ngõ ra là **“TRUE”**.
- Các thông số MIN, MAX là giá trị nhỏ nhất và lớn nhất của dải cần so sánh.
- VAL là giá trị đầu vào để so sánh với dải MIN, MAX.
- Các thông số MIN, VAL, MAX phải cùng kiểu dữ liệu.

Tập lệnh	Sự so sánh là đúng nếu:
IN_RANGE	MIN <= VAL <= MAX
OUT_RANGE	VAL < MIN hoặc VAL > MAX

Thông số	Kiểu dữ liệu	Mô tả

MIN, VAL, MAX	Int Dint Real	Các ngõ vào phần tử so sánh
---------------	---------------------	-----------------------------

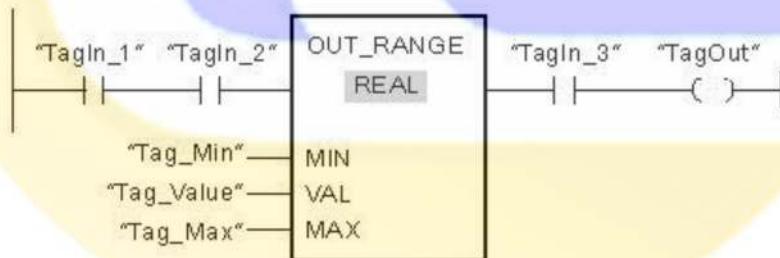
Ví dụ 1:



Mô tả : “TagOut” đạt giá trị **True** khi thỏa mãn tất cả các trường hợp sau:

- “TagIn_1”, “TagIn_2” cùng bằng **True**.
- “Tag_Value” nằm **trong** **dải** từ “Tag_Min” đến “Tag_Max”
- “TagIn_3” bằng **True**.

Ví dụ 2:



Mô tả : “TagOut” đạt giá trị **True** khi thỏa mãn tất cả các trường hợp sau:

- “TagIn_1”, “TagIn_2” cùng bằng **True**.
- “Tag_Value” nằm **ngoài** **dải** từ “Tag_Min” đến “Tag_Max”
- “TagIn_3” bằng **True**.

3.6. Tập lệnh tính toán Cộng, Trừ, Nhân, Chia và MOVE

- Lệnh **ADD** và **MUL** có thể thêm nhiều ngõ đầu vào bằng các click vào * trong khôi lệnh.
- Khi được cho phép **EN** = 1, lệnh phép toán thực hiện. Sau một sự hoàn tất thành công phép toán, lệnh sẽ đặt **ENO** = 1.

Thông số	Kiểu dữ liệu	Mô tả
IN1, IN2,...	Int, Dint, Real, Const...	Dữ liệu đầu vào của phép toán
OUT	Int, Dint, Real	Kết quả của phép toán

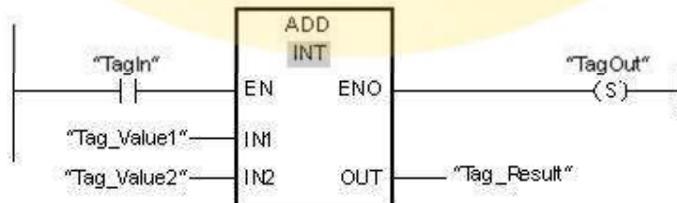
- Lưu ý: Các thông số IN1, IN2... và OUT phải cùng kiểu dữ liệu.

3.6.1. Lệnh ADD

- Khôi logic thực hiện lệnh **Cộng** khi khôi có tín hiệu vào chân **EN**. Khôi lệnh sẽ thực hiện Cộng tất cả các giá trị ở chân **IN** và xuất ra giá trị ở chân **OUT**.

Công thức : OUT = IN1 + IN2 + ... + INn

Ví dụ:



Mô tả : "Tag_Result" = "Tag_Value1" + "Tag_Value2"

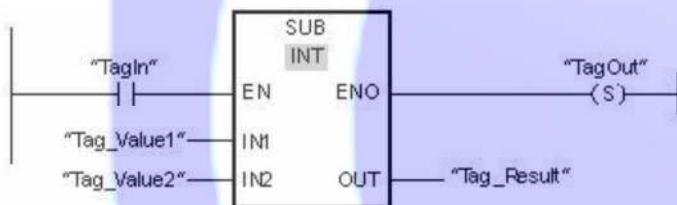
- Nếu “TagIn” bằng **True**, khôi lệnh logic **ADD** thực hiện cộng: “Tag_Value1” + “Tag_Value2”.
- Giá trị sẽ được lưu vào “Tag_Result”.
- Khi thực thi xong lệnh ADD, “TagOut” bằng **True**.

3.6.2. Lệnh SUB

- Khối logic thực hiện lệnh **Trừ** khi khôi có tín hiệu vào chân EN. Khôi lệnh sẽ thực hiện **Trừ** giá trị IN1 cho IN2 và xuất ra giá trị ở chân OUT.

Công thức : **OUT = IN1 - IN2**

Ví dụ:



Mô tả : “Tag_Result” = “Tag_Value1” - “Tag_Value2”

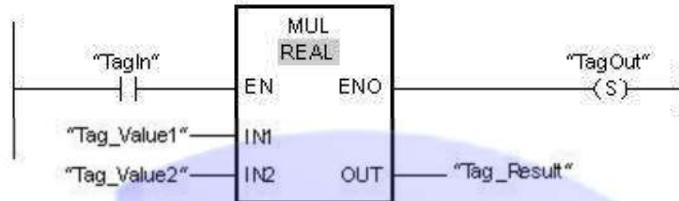
- Nếu “TagIn” bằng **True**, khôi lệnh logic **SUB** thực hiện trừ: “Tag_Value1” - “Tag_Value2”.
- Giá trị sẽ được lưu vào “Tag_Result”.
- Khi thực thi xong lệnh SUB, “TagOut” bằng **True**.

3.6.3. Lệnh MUL

- Khối logic thực hiện lệnh **Nhân** khi khôi có tín hiệu vào chân EN. Khôi lệnh sẽ thực hiện Nhân tất cả các giá trị ở chân IN và xuất ra giá trị ở chân OUT.

Công thức : **OUT = IN1 * IN2 * ... * INn**

Ví dụ:



Mô tả : $\text{Tag_Result} = \text{Tag_Value1} * \text{Tag_Value2}$

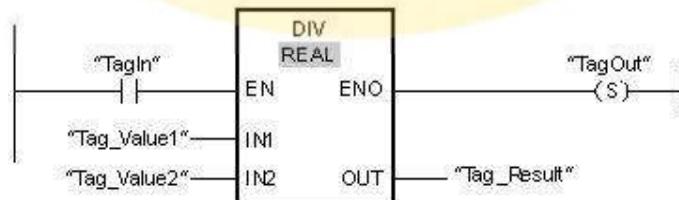
- Nếu “TagIn” bằng **True**, khối lệnh logic **MUL** thực hiện nhân: $\text{Tag_Value1} + \text{Tag_Value2}$.
- Giá trị sẽ được lưu vào “Tag_Result”.
- Khi thực thi xong lệnh **MUL**, “TagOut” bằng **True**.

3.6.4. Lệnh DIV

- Khối logic thực hiện lệnh **Chia** khi khối có tín hiệu vào chân EN. Khối lệnh sẽ thực hiện Chia giá trị IN1 cho IN2 và xuất ra giá trị ở chân OUT.

Công thức : **OUT = IN1 / IN2**

Ví dụ:



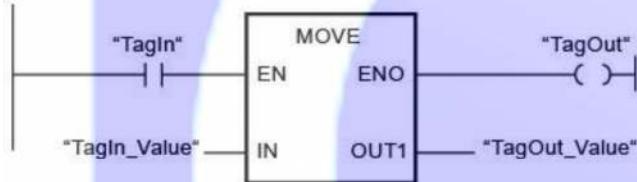
Mô tả : $\text{Tag_Result} = \text{Tag_Value1} / \text{Tag_Value2}$

- Nếu “TagIn” bằng **True**, khởi lệnh logic **DIV** thực hiện chia: “Tag_Value1” + “Tag_Value2”.
- Giá trị sẽ được lưu vào “Tag_Result”.
- Khi thực thi xong lệnh **DIV**, “TagOut” bằng **True**.

3.6.5. Lệnh MOVE

- Lệnh **MOVE** dùng để copy dữ liệu từ vùng nhớ này sang vùng nhớ khác
- Lệnh **MOVE** sao chép dữ liệu địa chỉ nguồn được xác định bởi thông số **IN** đến địa chỉ đích được xác định bởi thông số **OUT**

Ví dụ:

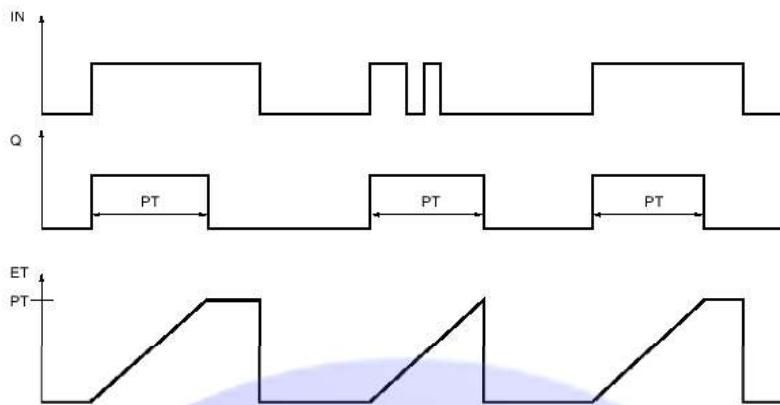


Mô tả : Lệnh **MOVE** thực hiện khi thỏa mãn:

- Nếu “TagIn” bằng **True**, khởi lệnh logic **MOVE** thực hiện sao chép dữ liệu từ địa chỉ “TagIn_Value” đến địa chỉ “TagOut_Value”.
- Khi thực thi xong lệnh **MOVE**, “TagOut” bằng **True**.

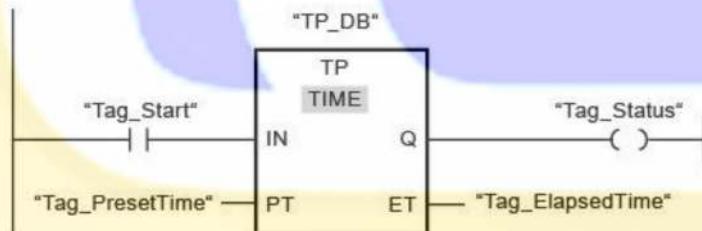
3.7. Tập lệnh Timer

3.7.1. Lệnh TP



- Khi tín hiệu đầu vào **IN** bằng **True** thì đầu ra **Q** của khối **TP** bằng **True** trong khoảng thời gian **PT** đặt trước.
- Thời gian của Timer được lưu vào **ET**.
- Khi giá trị **ET** đạt giá trị bằng **PT** thì đầu ra **Q** của khối **PT** bằng **False** mặc dù tín hiệu đầu vào **IN** bằng **True** hay **False**.
- Khi tín hiệu đầu vào **IN** bằng **False** thì giá trị **ET** trả về bằng **0ms**.
- **Giá trị ET là kiểu dữ liệu Time, được biểu diễn bởi 32 bit nhớ.**

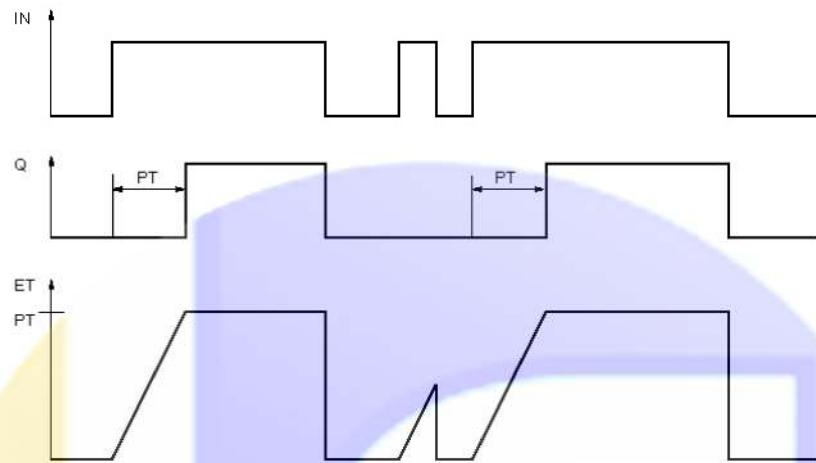
Ví dụ:



Mô tả : Lệnh TP thực hiện như sau:

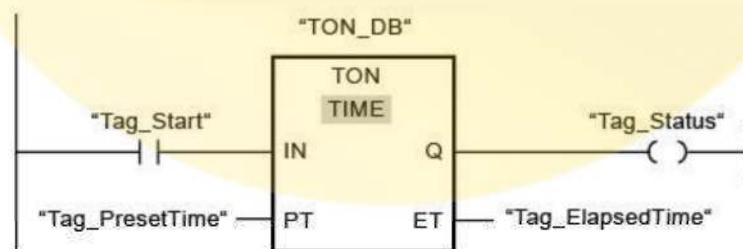
- Nếu “*Tag_Start*” bằng **True**.
- “*Tag_Status*” bằng **True**.
- Giá trị “*Tag_ElapsedTime*” thay đổi từ **0ms** đến “*Tag_PresetTime*”
- Khi giá trị “*Tag_ElapsedTime*” = “*Tag_PresetTime*” thì đầu ra “*Tag_Status*” bằng **False**.

3.7.2. Lệnh TON



- Khi tín hiệu đầu vào **IN** bằng **True** thì Timer **TON** bắt đầu tính thời gian và thời gian được lưu vào **ET**.
- Khi giá trị **ET** đạt giá trị bằng **PT** thì đầu ra **Q** của khối **PT** bằng **True**.
- Khi tín hiệu đầu vào **IN** bằng **False** thì đầu ra **Q** trả về bằng **False** và giá trị **ET** trả về bằng **0ms**.
- **Giá trị ET là kiểu dữ liệu Time, được biểu diễn bởi 32 bit nhớ.**

Ví dụ:

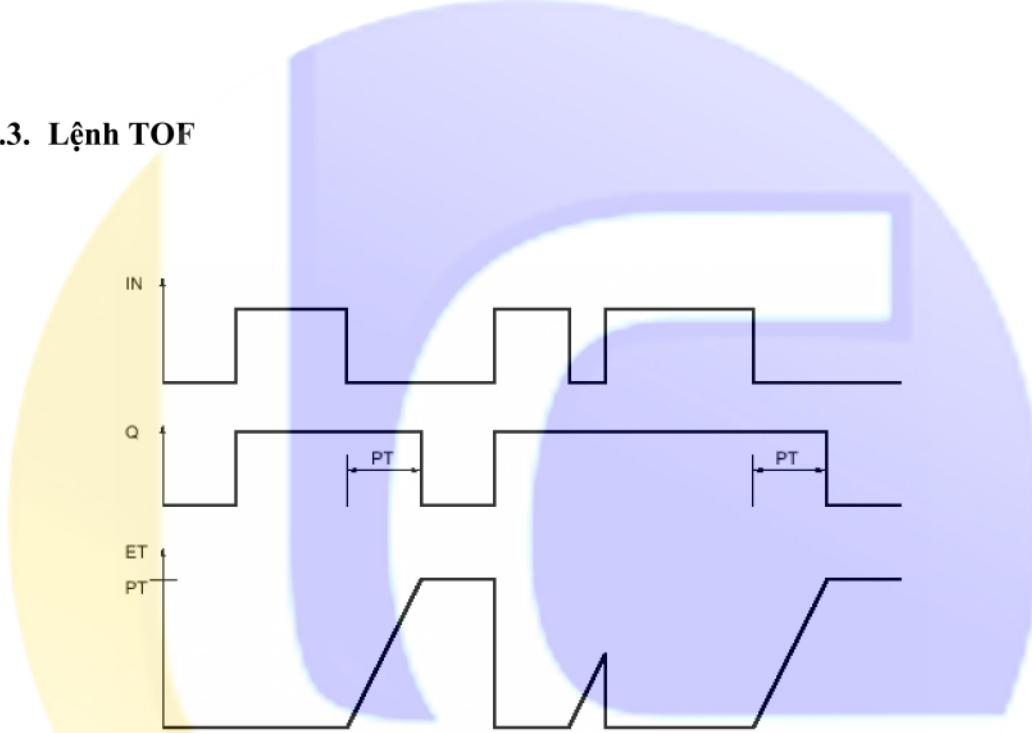


Mô tả : Lệnh TON thực hiện như sau:

- Nếu "Tag_Start" bằng **True**.

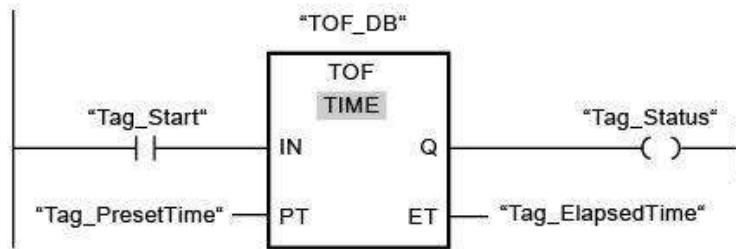
- Giá trị “*Tag_ElapsedTime*” thay đổi từ 0ms đến “*Tag_PresetTime*”
- Khi giá trị “*Tag_ElapsedTime*” = “*Tag_PresetTime*” thì đầu ra “*Tag_Status*” bằng **True**.
- Khi giá trị đầu vào “*Tag_Start*” bằng **False** thì giá trị đầu ra “*Tag_PresetTime*” bằng 0 và đầu ra “*Tag_Status*” bằng **False**.

3.7.3. Lệnh TOF



- Khi tín hiệu đầu vào **IN** bằng **True** thì đầu ra **Q** bằng **True**.
- Khi tín hiệu đầu vào **IN** thay đổi từ trạng thái **True** sang **False** thì Timer TOF được kích hoạt, giá trị thời gian được lưu vào **ET**.
- Khi giá trị **ET** đạt giá trị bằng **PT** thì đầu ra **Q** của khối **PT** bằng **False**.
- Khi tín hiệu đầu vào **IN** bằng **True** thì đầu ra **Q** trả về bằng **True** và giá trị **ET** trả về bằng **0ms**.
- **Giá trị ET là kiểu dữ liệu Time, được biểu diễn bởi 32 bit nhớ.**

Ví dụ:



Mô tả : Lệnh TON thực hiện như sau:

- Nếu “**Tag_Start**” bằng **True**.
- Giá trị “**Tag_Status**” bằng **True**.
- Khi giá trị “**Tag_Start**” bằng **False**.
- Giá trị “**Tag_ElapsedTime**” thay đổi từ 0ms đến “**Tag_PresetTime**”
- Khi giá trị “**Tag_ElapsedTime**” = “**Tag_PresetTime**” thì đầu ra “**Tag_Status**” bằng **False**.
- Khi giá trị đầu vào “**Tag_Start**” bằng **True** thì giá trị đầu ra “**Tag_PresetTime**” bằng 0 và đầu ra “**Tag_Status**” bằng **True**.

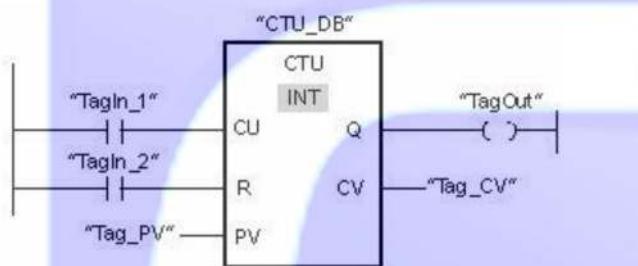
3.8. Tập lệnh Counter

3.8.1. CTU (Counter Up)

- Khi trạng thái tín hiệu đầu vào **CU** thay đổi từ **False** sang **True** thì lệnh được thực thi, và giá trị bộ đếm hiện tại ở đầu ra **CV** được tăng lên 1 đơn vị.
- Giá trị bộ đếm được tăng lên mỗi khi phát hiện thấy trạng thái tín hiệu đầu vào **CU** thay đổi từ **False** sang **True**.
- Khi đạt đến giới hạn **PV** đặt trước, trạng thái tín hiệu ở đầu vào **CU** không còn ảnh hưởng đến lệnh.
- Giá trị **CV** chỉ tăng đến giới hạn cao của kiểu dữ liệu đang chỉ định.

- Trạng thái tín hiệu ở đầu ra **Q** được xác định bởi tham số **PV**. Nếu giá trị bộ đếm **CV** hiện tại lớn hơn hoặc bằng giá trị của tham số **PV** thì đầu ra **Q** chuyển thành trạng thái tín hiệu **True**. Trong tất cả các trường hợp khác, đầu ra **Q** có trạng thái tín hiệu **False**.
- Giá trị ở đầu ra **CV** được đặt lại về 0 khi trạng thái tín hiệu ở đầu vào **R** thay đổi thành **True**. Miễn là đầu vào **R** có trạng thái tín hiệu **True** thì trạng thái tín hiệu ở đầu vào **CU** không ảnh hưởng đến lệnh.

Ví dụ:



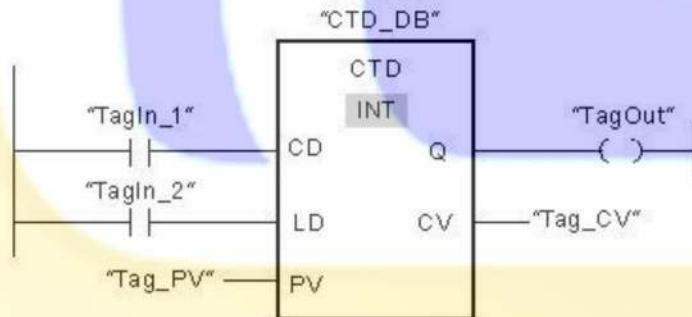
Mô tả : Lệnh **CTU** thực hiện như sau:

- Nếu trạng thái tín hiệu “**TagIn_1**” thay đổi từ **False** sang **True** thì giá trị “**Tag_CV**” tăng lên 1 đơn vị.
- Khi giá trị “**Tag_CV**” tăng dần đến khi lớn hơn hoặc bằng giá trị “**Tag_PV**”
- Khi giá trị “**Tag_CV**” lớn hơn hoặc bằng giá trị “**Tag_PV**” thì trạng thái tín hiệu “**TagOut**” bằng **True**.
- Khi trạng thái tín hiệu “**TagIn_2**” bằng **True** thì trạng thái tín hiệu “**TagOut**” bằng **False** và giá trị “**Tag_CV**” bằng 0.

3.8.2. CTD (Counter Down)

- Khi trạng thái tín hiệu đầu vào **CD** thay đổi từ **False** sang **True** thì lệnh được thực thi, và giá trị bộ đếm hiện tại ở đầu ra **CV** được giảm đi 1 đơn vị.
- Giá trị bộ đếm được giảm đi mỗi khi phát hiện thấy trạng thái tín hiệu đầu vào **CD** thay đổi từ **False** sang **True**.
- Giá trị **CV** chỉ giảm đến giới hạn thấp của kiểu dữ liệu đang chỉ định.
- Khi giá trị **CV** giảm từ giá trị **PV** đặt trước về 0 thì trạng thái tín hiệu ở đầu vào **CD** không còn ảnh hưởng đến lệnh.
- Nếu giá trị bộ đếm hiện tại nhỏ hơn hoặc bằng 0, đầu ra **Q** được đặt thành trạng thái tín hiệu **True**. Trong tất cả các trường hợp khác, đầu ra **Q** có trạng thái tín hiệu **False**.
- Giá trị ở đầu ra **CV** được đặt lại về tham số **PV** khi trạng thái tín hiệu ở đầu vào **LD** thay đổi thành **True**. Miễn là đầu vào **LD** có trạng thái tín hiệu **True** thì trạng thái tín hiệu ở đầu vào **CD** không ảnh hưởng đến lệnh.

Ví dụ:



Mô tả : Lệnh **CTD** thực hiện như sau:

- Nếu trạng thái tín hiệu “**TagIn_1**” thay đổi từ **False** sang **True** thì giá trị “**Tag_CV**” giảm đi 1 đơn vị.
- Khi giá trị “**Tag_CV**” giảm dần đến khi nhỏ hơn hoặc bằng 0.
- Khi giá trị “**Tag_CV**” nhỏ hơn hoặc bằng 0 thì trạng thái tín hiệu “**TagOut**” bằng **True**.

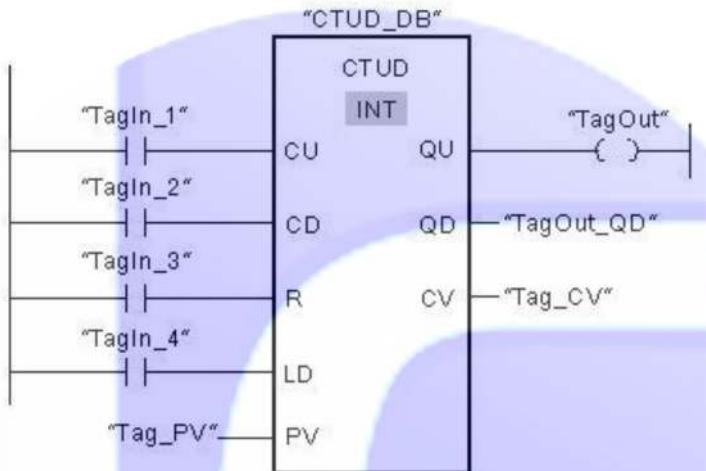
- Khi trạng thái tín hiệu “**TagIn_2**” bằng **True** thì trạng thái tín hiệu “**TagOut**” bằng **False** và giá trị “**Tag_CV**” bằng giá trị “**Tag_PV**”.

3.8.3. CTUD (Counter Up Down)

- Khi trạng thái tín hiệu ở đầu vào **CU** thay đổi từ **False** thành **True** thì giá trị bộ đếm hiện tại được tăng thêm một và được lưu trữ ở đầu ra **CV**.
- Nếu trạng thái tín hiệu ở đầu vào **CD** thay đổi từ **False** thành **True** thì giá trị bộ đếm ở đầu ra **CV** bị giảm đi một.
- Nếu có cạnh tín hiệu dương ở đầu vào **CU** và **CD** trong một chu kỳ chương trình, giá trị bộ đếm hiện tại ở đầu ra **CV** vẫn không thay đổi.
- Giá trị bộ đếm có thể được tăng lên cho đến khi đạt đến giá trị giới hạn cao của loại dữ liệu được chỉ định ở đầu ra **CV**. Khi đạt đến giới hạn cao, giá trị bộ đếm không còn tăng trên cạnh tín hiệu dương. Giá trị bộ đếm không còn giảm nữa sau khi đạt đến giới hạn thấp của loại dữ liệu đã chỉ định.
- Khi trạng thái tín hiệu ở đầu vào **LD** thay đổi thành **True**, giá trị bộ đếm ở đầu ra **CV** được đặt thành giá trị của tham số **PV**. Miễn là đầu vào **LD** có trạng thái tín hiệu **True**, trạng thái tín hiệu ở đầu vào **CU** và **CD** không ảnh hưởng đến lệnh.
- Giá trị bộ đếm được đặt thành **False** khi trạng thái tín hiệu ở đầu vào **R** thay đổi thành **True**. Miễn là đầu vào **R** có trạng thái tín hiệu **True**, thay đổi trạng thái tín hiệu của đầu vào **CU**, **CD** và **LD** không ảnh hưởng đến lệnh **CTUD**.
- Có thể truy vấn trạng thái của bộ đếm lên ở đầu ra **QU**. Nếu giá trị bộ đếm hiện tại lớn hơn hoặc bằng giá trị của tham số **PV**, đầu ra **QU** được đặt thành trạng thái tín hiệu **True**. Trong tất cả các trường hợp khác, đầu ra **QU** có trạng thái tín hiệu **False**.

- Có thể truy vấn trạng thái của bộ đếm xuống ở đầu ra **QD**. Nếu giá trị bộ đếm hiện tại nhỏ hơn hoặc bằng 0, đầu ra **QD** được đặt thành trạng thái tín hiệu **True**. Trong tất cả các trường hợp khác, đầu ra **QD** có trạng thái tín hiệu **False**.

Ví dụ:



Mô tả : Lệnh **CTUD** thực hiện như sau:

- Trạng thái tín hiệu ở đầu vào “**TagIn_1**” hoặc “**TagIn_2**” thay đổi từ **False** thành **True** thì lệnh **CTUD** được thực thi.
- Khi có tín hiệu **True** ở đầu vào “**TagIn_1**”, giá trị bộ đếm hiện tại được tăng thêm một và được lưu trữ ở đầu ra “**Tag_CV**”.
- Khi có tín hiệu **True** ở đầu vào “**TagIn_2**”, giá trị bộ đếm bị giảm đi một và được lưu trữ ở đầu ra “**Tag_CV**”.
- Khi có tín hiệu **True** ở đầu vào **CU**, giá trị bộ đếm được tăng lên cho đến khi đạt đến giới hạn cao.
- Nếu **CD** đầu vào có tín hiệu **True**, giá trị bộ đếm bị giảm cho đến khi đạt đến giới hạn thấp.
- Đầu ra “**TagOut**” có trạng thái tín hiệu **True** miễn là giá trị bộ đếm hiện tại lớn hơn hoặc bằng giá trị ở đầu vào “**Tag_PV**”.

Trong tất cả các trường hợp khác, đầu ra “TagOut” có trạng thái tín hiệu **False**.

- Đầu ra “TagOut_QD” có trạng thái tín hiệu **True** miễn là giá trị bộ đếm hiện tại nhỏ hơn hoặc bằng 0. Trong tất cả các trường hợp khác, đầu ra “TagOut_QD” có trạng thái tín hiệu **False**.

