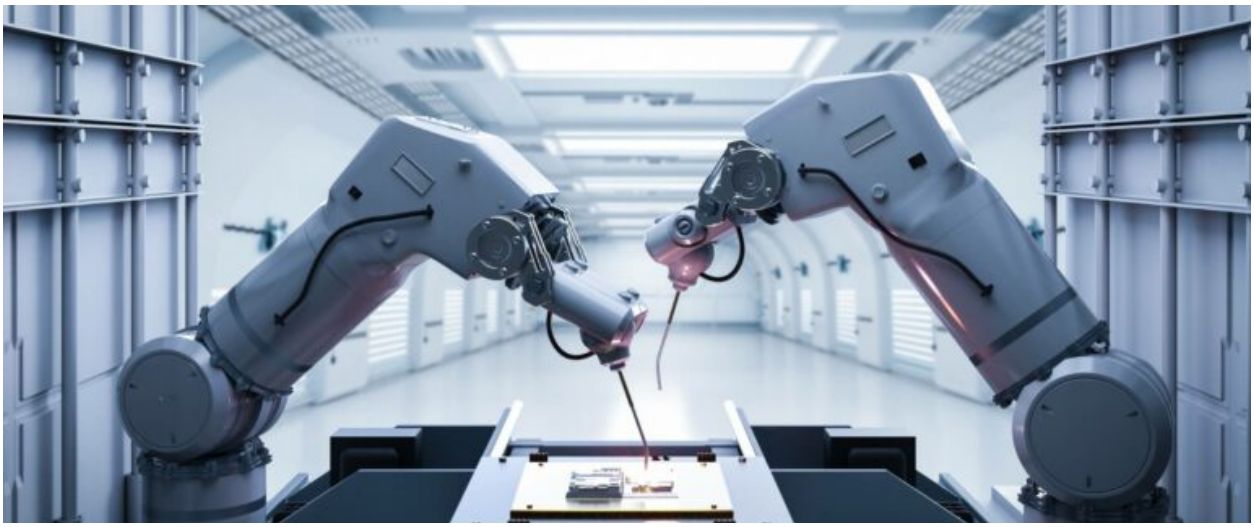


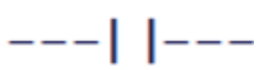
# Tia Portal Notes

## Bit logic operations

▼ Normally open contact

Sinyal ne zaman ‘1’ geldiğinde normalde açık kontak kapanır. Sinyal ‘0’ olduğunda ise açık pozisyonda durur.

İşareti:



▼ Normally close contact

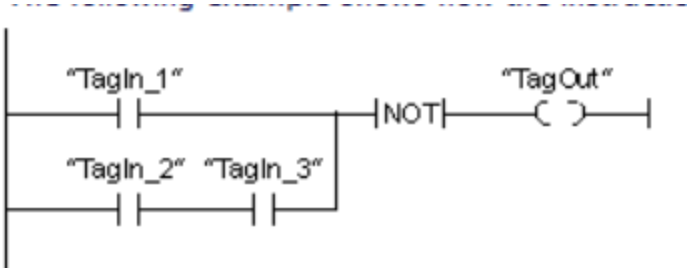
Sinyal ne zaman ‘1’ geldiğinde normalde kapalı kontak açılır. Sinyal ‘0’ olduğunda ise kapalı pozisyonda durur.

İşareti:

▼ Invert RLO

Eğer gelen sinyal ‘1’ ise onu ‘0’ yapar. Eğer ‘0’ ise ‘1’ yapar

ÖRNEK KULLANIM;




İşareti:

--|NOT|--

▼ Output(Assigment)

eğer coil'in inputu '1' durumunda ise sinyal durumu '1' olur röledeki. Eğer '0' ise röledeki sinyal '0' olur

İŞARETİ:

---( )---

▼ Negate Assigment

eğer coil'in inputu '0' durumunda ise sinyal durumu '1' olur röledeki. Eğer '1' ise röledeki sinyal '0' olur

İŞARETİ:

--( / )--

▼ Reset Output

rölenin çıkışını resetlemek istiyorsak kullanırız.İnput sinyali geldiği gibi röleyi '0' durumuna çeker.

---( R )---

▼ Set Output

Rölenin çıkışını setlemek istiyorsak kullanırız. İnput sinyali geldiği gibi röleyi ‘1’ durumuna getirir.

---( S )---

▼ SET_BF (Set bit field)

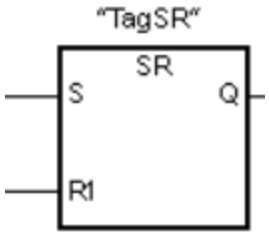
Birden fazla çıkışı set etmek için kullanılır.

▼ RESET_BF

Birden fazla çıkışı reset etmek için kullanılır.

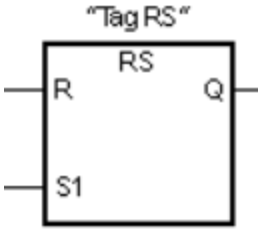
▼ SR (Set/reset flip-flop)

Set reset flip-flopudur. Eğer set kısmına ‘1’, Reset kısmına ‘0’ gelirse çıkışı set eder. Tam tersi ise çıkışı reset eder. NOT: R1 öncelikli inputtur.



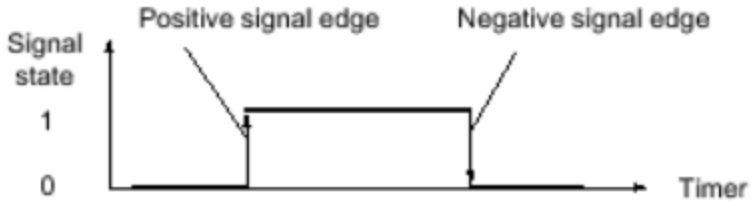

▼ RS(Reset/set flip-flop)

Reset set flip-flopudur. Eğer set kısmına ‘1’, Reset kısmına ‘0’ gelirse çıkışı set eder. Tam tersi ise çıkışı reset eder. NOT: S1 öncelikli inputtur.



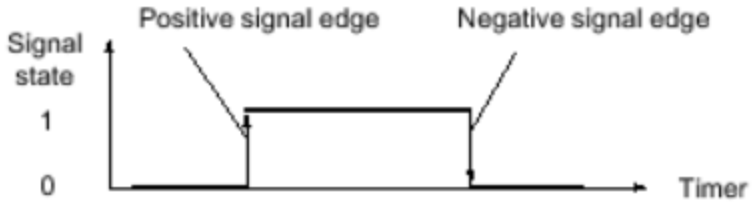
▼ |P| (Positive Signal Edge)

Sinyalin pozitif kenarı ile tetik alır ve çıkışa ‘1’ şeklinde bir pulse verir. Tekrar çıkış vermesi için bir daha pozitif kenarlı sinyal alması gerekir.



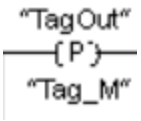

▼ |N| (Negative signal edge)

Sinyalin negatif kenarı ile tetik alır ve çıkışa ‘1’ şeklinde bir pulse verir. Tekrar çıkış vermesi için bir daha negatif kenarlı sinyal alması gerekir.




▼ —(P)— (Set operand on positive signal edge)

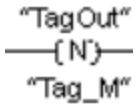
Pozitif tetiklemeli coildir. girişine ‘0’ dan ‘1’ e geçen bir sinyal geldiği anda . <Operand2> yi ‘1’ yapar İnput sinyali ‘1’ olduğu kadar <Operand2> ‘1’ olarak kalır ne zaman input ‘0’ olursa <Operand2>‘0’ a çeker. Fakat <Operand1> sadece ilk pozitif kenar sinyal geldiğinde çıkışa bir pulse verir onun dışında her zaman ‘0’ dır. Aşağıda  Operand1:Tag_Out / Operand2:Tag_M




▼ —(N)— (Set operand on negative signal edge)

negatif tetiklemeli coildir. girişine ‘0’ den ‘1’ a geçen bir sinyal geldiği anda . <Operand2> yi ‘1’ yapar İnput sinyali ‘1’ olduğu kadar <Operand2> ‘1’ olarak kalır ne zaman input ‘0’ olursa <Operand2>‘0’ a çeker. Fakat <Operand1>

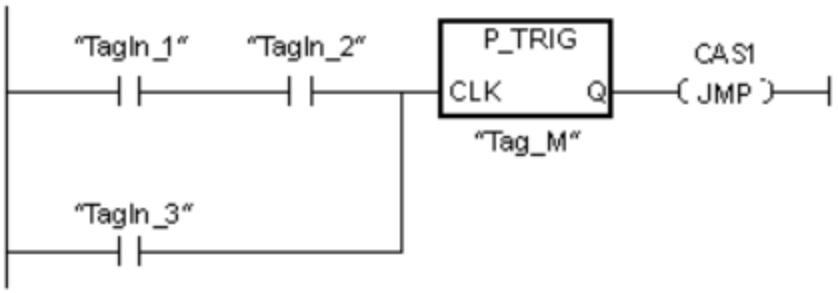
sadece ilk negatif kenar sinyal geldiğinde çıkışa bir pulse verir onun dışında her zaman ‘0’ dır. Aşağıda  Operand1:Tag_Out / Operand2:Tag_M




▼ P_TRIG (Scan RLO for positive signal edge)

CLK Girişine ‘1’ sinyali geldiği sürece <Operand> ‘1’ durumunda kalır. CLK girişine ‘0’ dan ‘1’ e geçen yani yükselen kenar sinyal geldiğinde Q dan bir kere pulse verir.

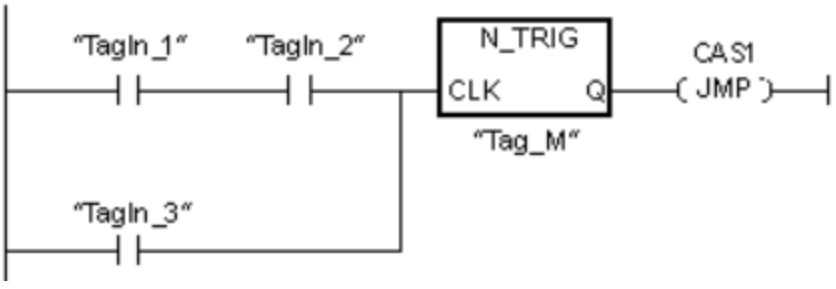
KULLANIM ÖRNEĞİ;




▼ N_TRIG (Scan RLO for negative signal edge)

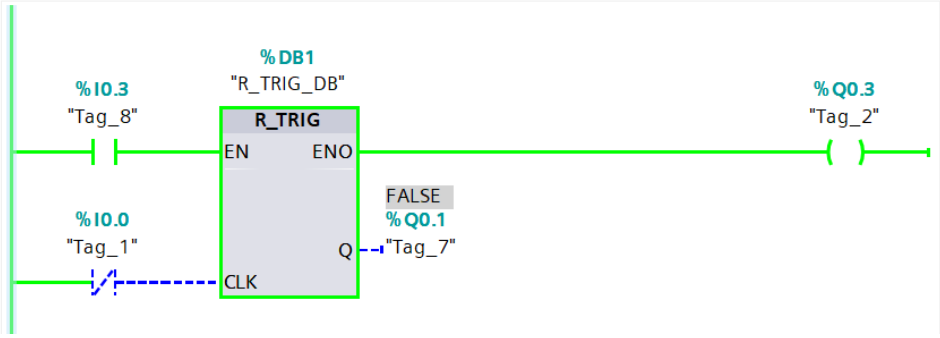
CLK Girişine ‘1’ sinyali geldiği sürece <Operand> ‘1’ durumunda kalır. CLK girişine ‘1’ den ‘0’ a geçen yani yükselen kenar sinyal geldiğinde Q dan bir kere pulse verir.

KULLANIM ÖRNEĞİ;



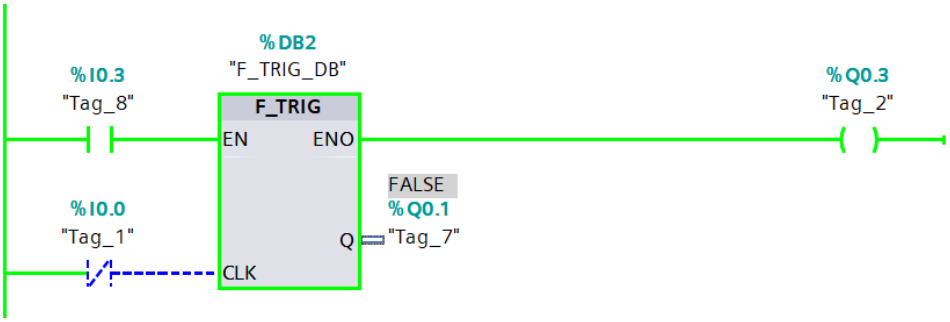

▼ R_TRIG (Detect Positive signal edge)

EN girişinde ‘1’ sinyali geldiği sürece çalışan bir fonksiyondur. Enable girişine ‘1’ geldiği sürece ENO çıkışı ‘1’ olur. CLK girişine ‘0’ dan ‘1’ e geçen bir sinyal gönderildiğinde ,Q çıkışı bir tarama süresince ‘1’ olur.




▼ F_TRIG (Detect negative signal edge)

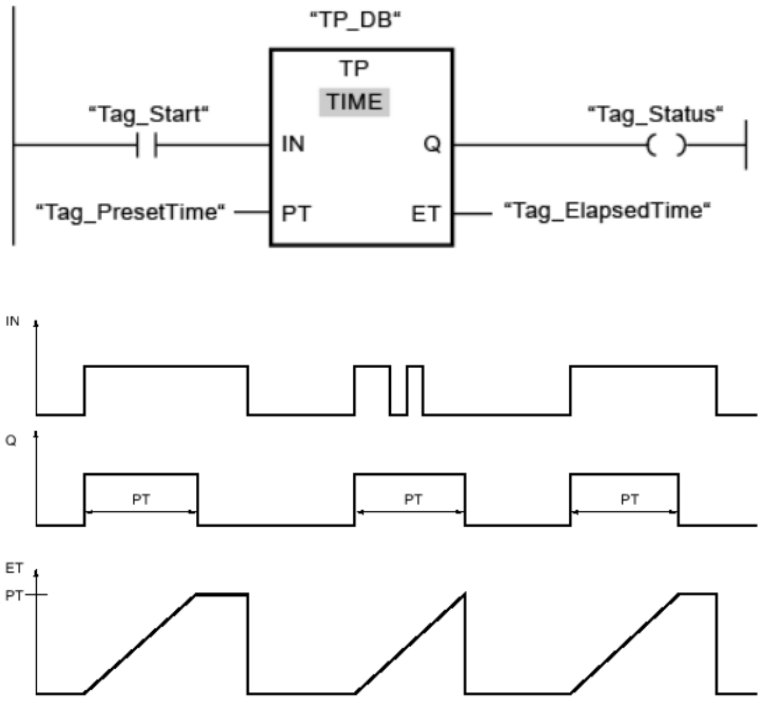
EN girişinde ‘1’ sinyali geldiği sürece çalışan bir fonksiyondur. Enable girişine ‘1’ geldiği sürece ENO çıkışı ‘1’ olur. CLK girişine ‘1’ den ‘0’ a geçen bir sinyal gönderildiğinde ,Q çıkışı bir tarama süresince ‘1’ olur.




# Timer operations

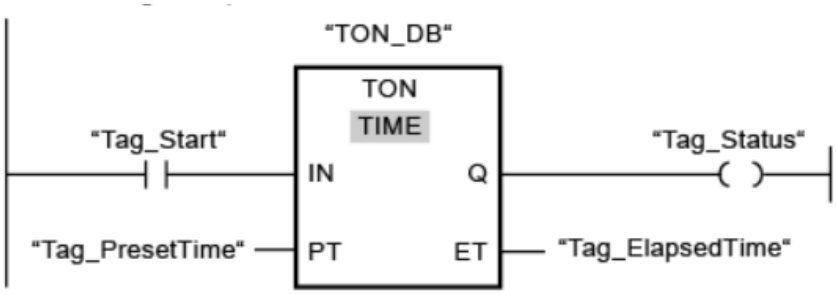
▼ TP (Generate Pulse timer)

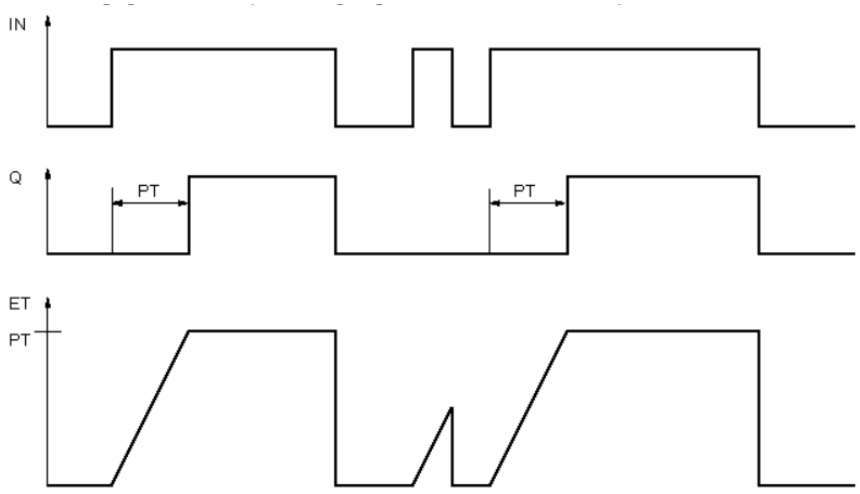
IN girişine ‘0’ dan ‘1’ e geçen bir sinyal geldiğinde aktif olur aktif olduğu süre boyunca çıkışını ‘1’ yapar.  IN girişi sürekli ‘1’ gelirse kendini tekrarlamaz sadece o süre boyunca çıkışı ‘1’ yapar. Tekrar çalışması için IN girişinin ‘0’ dan ‘1’ e geçmesi gerekiyor.




▼ TON (Generate on-delay timer)

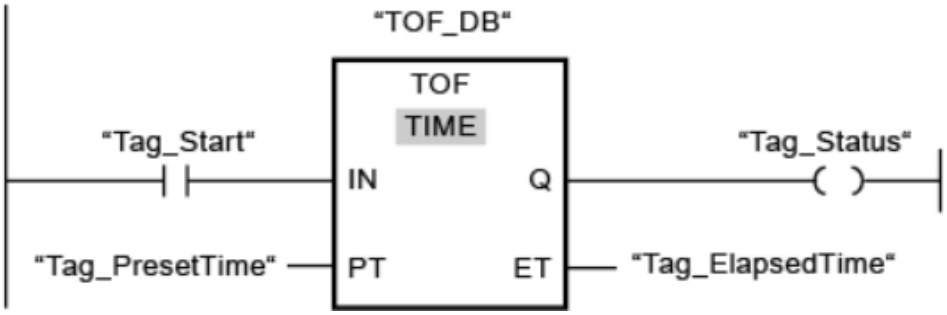
IN girişine ‘1’ geldiği sürece aktif olur ve sayaç saymaya başlar eğer IN girişi ‘0’ olursa sayaç sıfırlanır ve devre dışı kalır. Eğer sayaç boyunca giriş aktif olduysa sayaç bittiğinde Q çıkışı ‘1’ olur ve IN girişi ‘0’ olana kadar Q çıkışı ‘1’ olarak kalır ne zaman IN girişi ‘0’ olur o zaman resetlenir.

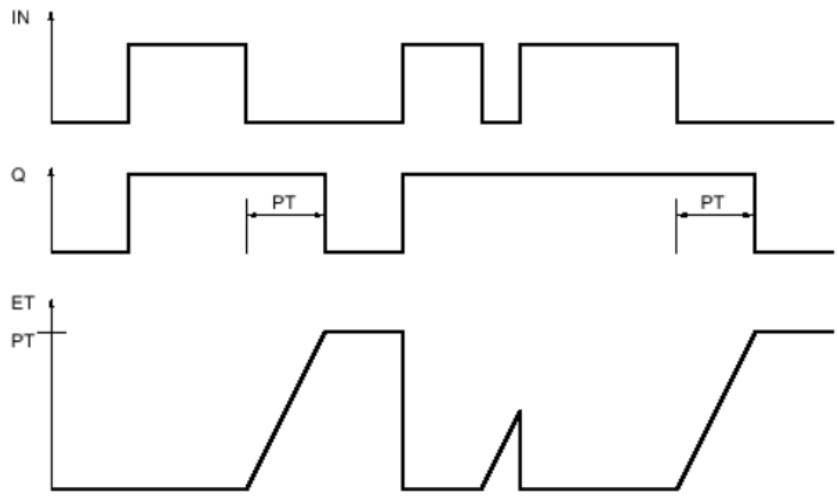





▼ TOF(Generate off-delay timer)

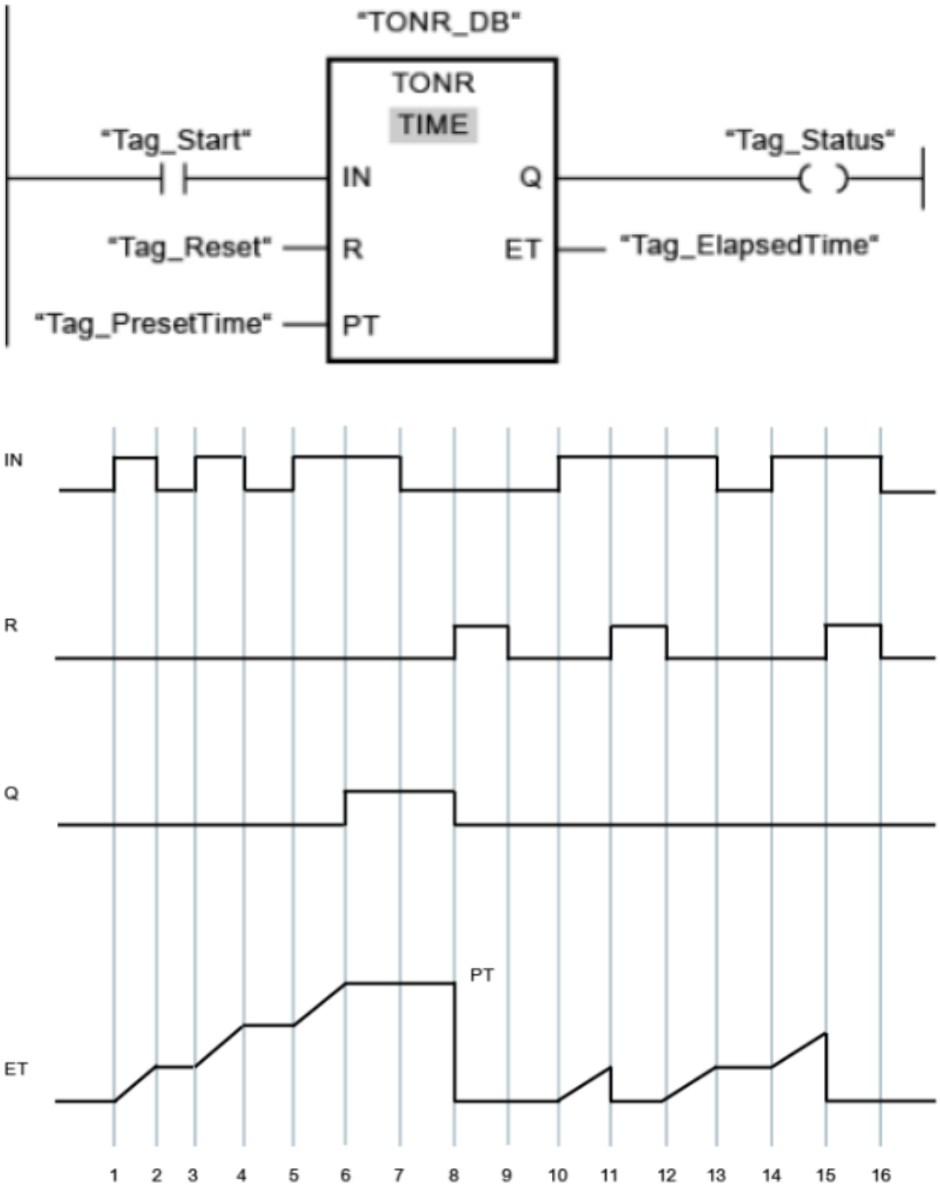
IN girişi ‘1’ olduğu anda Q çıkışı ‘1’ olur. IN girişi ‘1’ kaldığı sürece Q çıkışı yine ‘1’ kalır, ne zaman IN çıkışı ‘0’ olursa sayaç başlar sayaç bitene kadar yine Q çıkışı ‘1’ kalır. Sayaç bitince Q çıkışı ‘0’ a geçer.



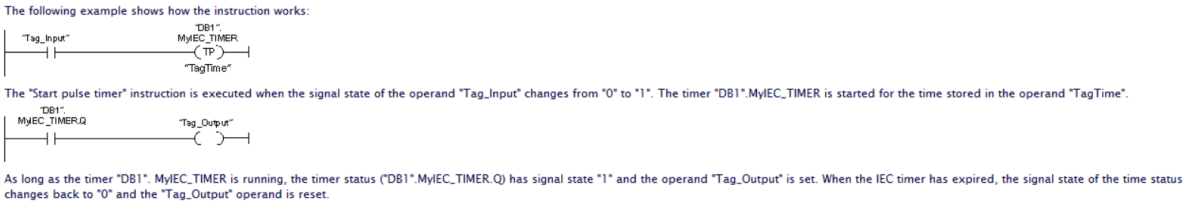
▼ TONR (Time accumulator)

IN girişine ‘1’ sinyali geldiği anda sayaç saymaya başlar. IN tekrar ‘0’ bile olsa sayaç resetlenmez aynı yerinde kalır. Sayaç tamamlandığında Q çıkışı ‘1’ olarak kalır (IN girişi ‘0’ olsa bile). Q’yu ve sayacı sıfırlamak için ise R girişine ‘1’ gelmesi gerekir.




▼ -(TP)- (Start pulse timer)

TP Timer'ın datablock oluşturup bağlanmış halidir. Outputunu başka bir network yapısı içerisinde kullanabiliriz.

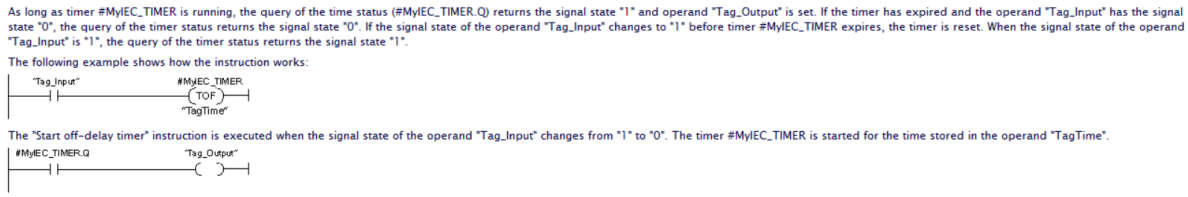



▼ -(TON)- (Start on-delay timer)

TON Timer'ın datablock oluşturup bağlanmış halidir. Outputunu başka bir network yapısı içerisinde kullanabiliriz.

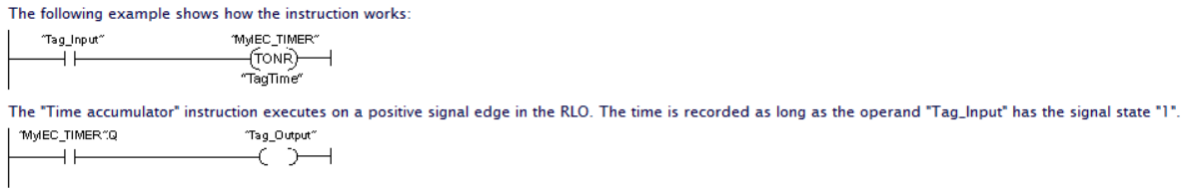

▼ -(TOF)- (Start off-delay timer)

TOF Timer'ın datablock oluşturup bağlanmış halidir. Outputunu başka bir network yapısı içerisinde kullanabiliriz.



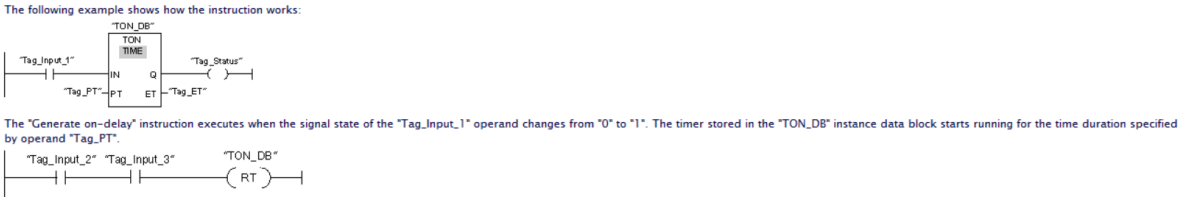

▼ -(TONR)- (Time accumulator)

TONR Timer'ın datablock oluşturup bağlanmış halidir. Outputunu başka bir network yapısı içerisinde kullanabiliriz.



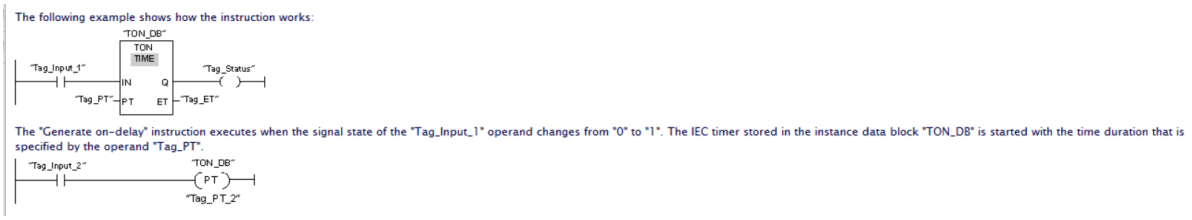

▼ -(RT)- (Reset timer)

[RT] fonkisyonu direkt bir datablock'a bağlıdır inputundan 1 geldiği zaman direkt o data block'un PT değerini sıfıra çeker. Yani bu data block bir TON timer'a da bağlı ise TON timer direkt çıkışı input kesilmediği sürece '1' yapar veya TOF timera bağlı ise giriş '0' olduğu anda çıkışta '0' olur. Bir nevi Timer'ı etkisiz bir eleman yapar.




▼ -(PT)- (Load time duration)

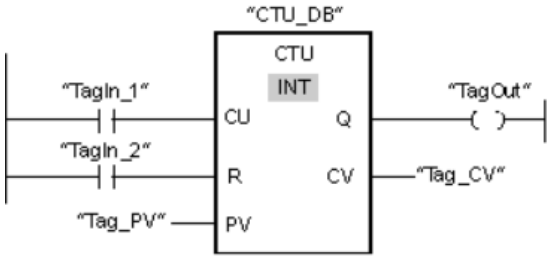
Hangi TIMER data bloğuna bağlı ise o data bloktaki PT değerini değiştirmemizi sağlar. INPUT geldiği anda data blocktaki PT değeri değişir ve öyle kalır.




# Counter operations

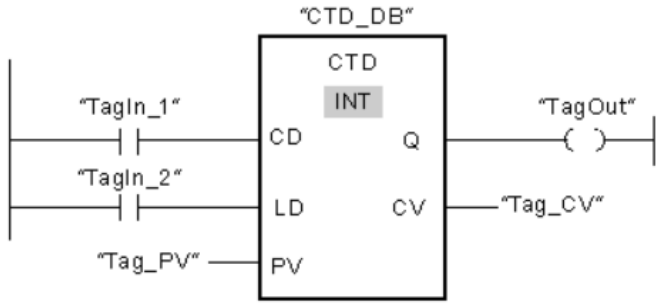
▼ CTU (Count up)

Yukarı sayıcı görevi görür. Data bloğa bağlıdır. CU girişine her sinyal geldiğinde bir artar PV değerine ulaştığı zaman çıkış verir. R girişinden sıfırlanabilir.



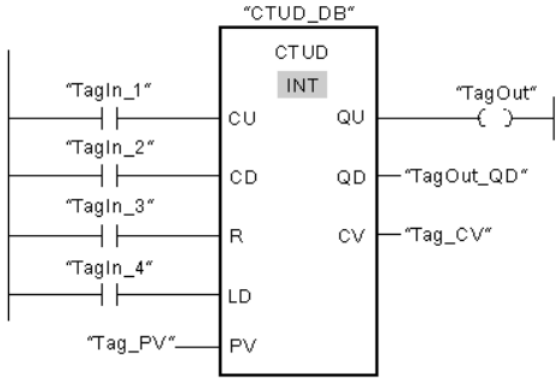

▼ CTD (Count down)

aşağı sayıcı görevi görür. Data bloğa bağlıdır. CU girişine her sinyal geldiğinde bir azalır sıfır değerine ulaştığı zaman çıkış verir. Başlangıç değeri PV'ye yazılır, LD girişinden tekrar PV değerine yüklenir.




▼ CTUD (Count up and down)

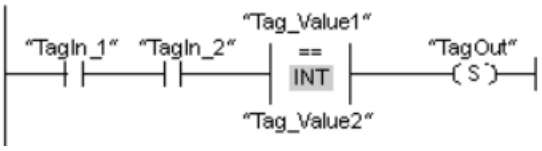
Hem aşağı hem yukarı sayacıdır. 4 girişi, 2 çıkışı vardır. PV değeri girildikten sonra CU girişine sinyal geldiğinde yukarı CD girişine sinyal geldiğinde ise aşağı sayar. R girişinden sıfırlanır. LD girişinden PV değerini sayıcıya yükler yeniden. Sayaç sıfıra geldiğinde QD çıkışından '1' verirken sayaç PV sayısına ulaştığında QU çıkışından '1' verir.




# Comperator Operation

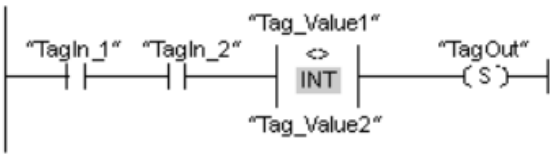
▼ CMP == (Equal)

<Operand1> e yazdığımız ifade ile <Operand2> ye yazdığımız eşit ise girişine gelen sinyali geçirir aksi halde geçirmez.



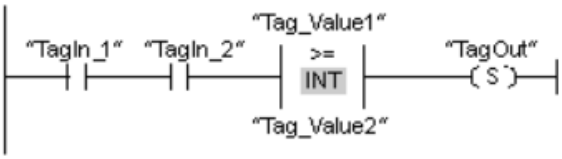

▼ CMP <> (Not equal)

<Operand1> e yazdığımız ifade ile <Operand2> ye yazdığımız eşit değil ise girişine gelen sinyali geçirir aksi halde geçirmez.



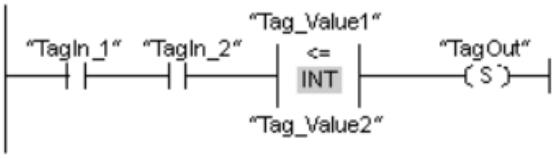

▼ CMP >= (Greater or equal)

<Operand1> e yazdığımız ifade <Operand2> den büyük veya eşit ise girişine gelen sinyali geçirir aksi halde geçirmez.



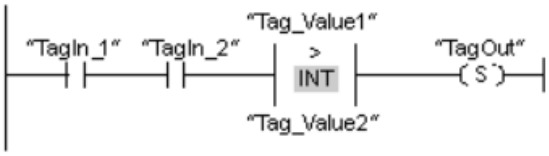

▼ CMP <= (Less or equal)

<Operand1> e yazdığımız ifade <Operand2> den küçük veya eşit ise girişine gelen sinyali geçirir aksi halde geçirmez.



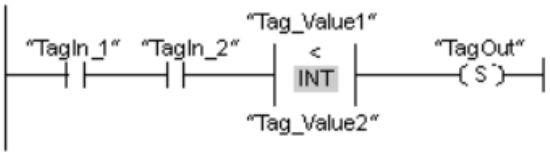

▼ CMP > (Greater than)

<Operand1> e yazdığımız ifade <Operand2> den büyük ise girişine gelen sinyali geçirir aksi halde geçirmez.



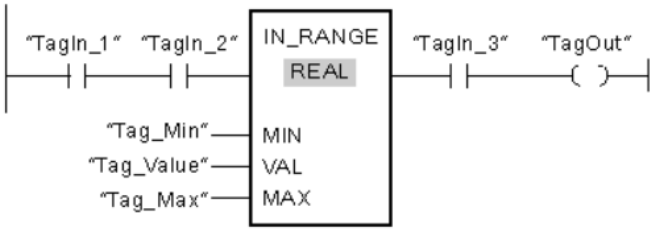

▼ CMP< (Less than)

<Operand1> e yazdığımız ifade <Operand2> den küçük ise girişine gelen sinyali geçirir aksi halde geçirmez.



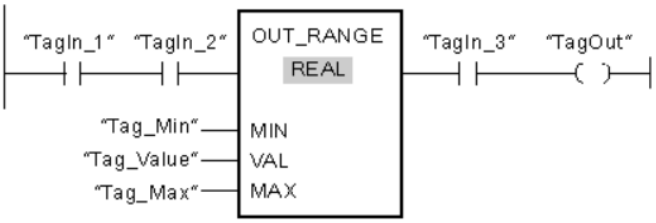

▼ IN_RANGE (Value within range)

VALUE girişine yazılan sayı. MIN ve MAX girişlerine girilen sayılar arasında ise çıkış verir.



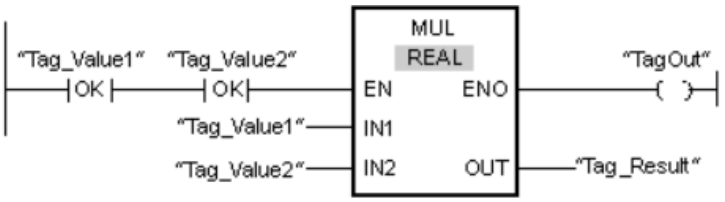

▼ OUT_RANGE (Value outside range)

VALUE girişine yazılan sayı. MIN ve MAX girişlerine girilen sayılar arasında değil ise çıkış verir.



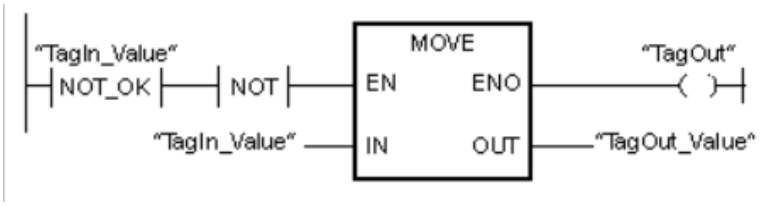

▼ -|OK|- (Check validity)

Sayı floating-number ise sinyali geçirir diğer durumlarda geçirmez. (HEX,DEC,BIN...)




▼ -|NOT_OK|- (Check invalidity)

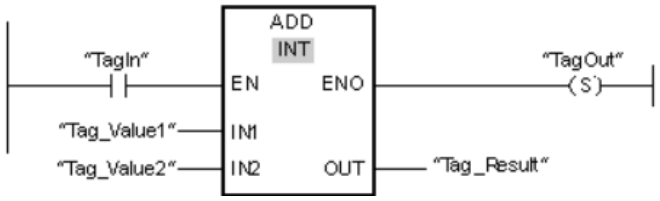
Sayı floating-number ise sinyali geçirmez diğer durumlarda geçirir. (HEX,DEC,BIN...)




# Math Function

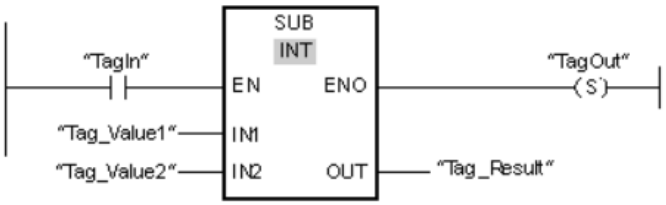
▼ ADD (Add)

IN1 girişine yazılan sayı ile IN2 girişine yazılan sayıyı toplayıp OUT çıkışına yazar.



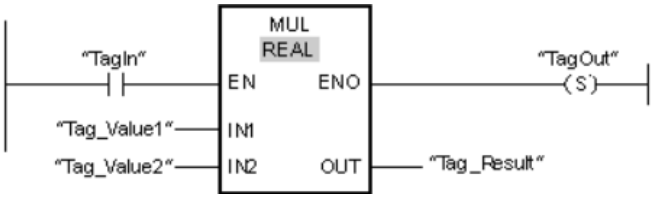

▼ SUB (Substract)

IN1 girişine yazılan sayıdan IN2 girişine yazılan sayıyı çıkarıp OUT çıkışına yazar.



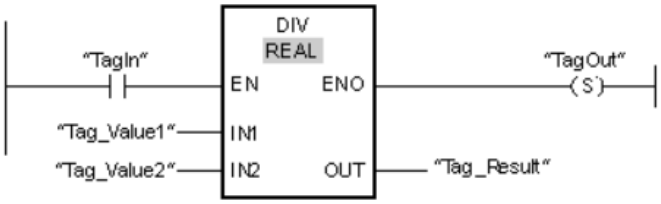

▼ MUL (Multiply)

IN1 girişine yazılan sayı ile IN2 girişine yazılan sayıyı çarpıp OUT çıkışına yazar.



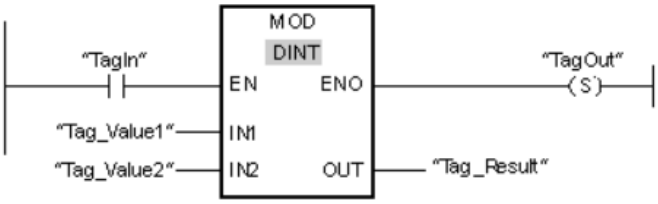

▼ DIV (Divide)

IN1 girişine yazılan sayıyı IN2 girişine yazılan sayıya böler OUT çıkışına yazar.



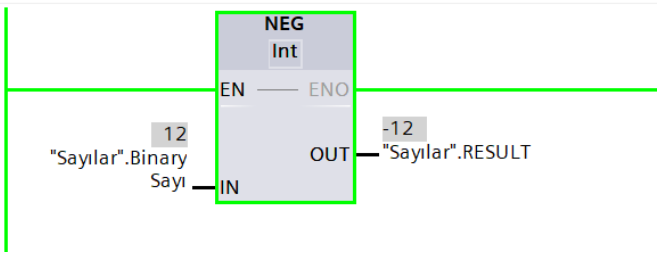

▼ MOD (Return Remainder of division)

IN1 girişine yazılan sayıynın IN2 girişine yazılan sayıya göre modunu alıp OUT çıkışına yazar.



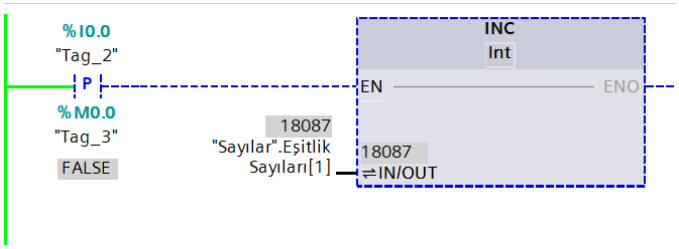

▼ NEG (Create twos complement)

IN girişine girilen sayının negatifini alır ve OUT'a yazar



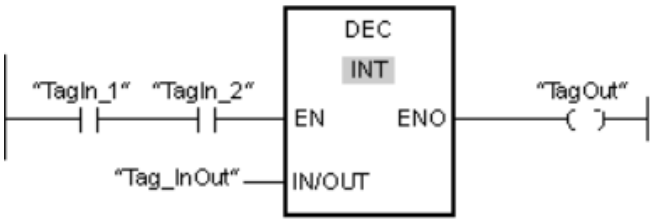

▼ INC (Increment)

EN girişine sinyal geldiği sürece IN/OUT girişindeki sayıyı bir arttırarak üstüne yazar.



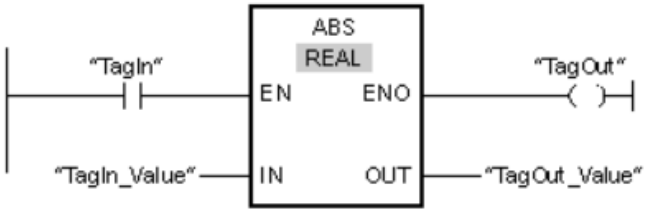

▼ DEC (Decrement)

EN girişine sinyal geldiği sürece IN/OUT girişindeki sayıyı bir azaltarak üstüne yazar.



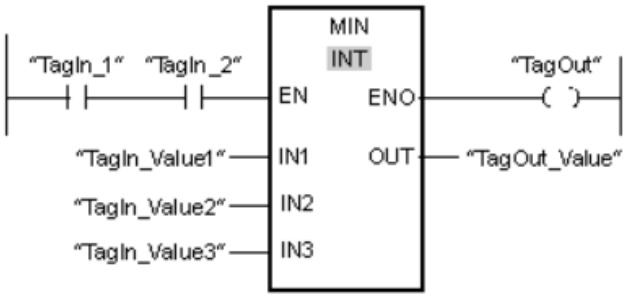

▼ ABS (Form absolute value)

IN girişine girilen sayının mutlak değerini alır ve OUT çıkışına yazar.



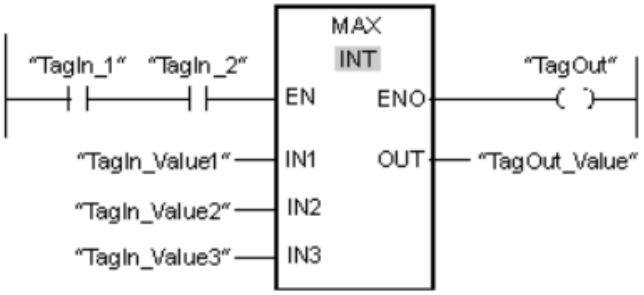
▼ MIN (Get minimum)

IN1, IN2, IN3 girişlerinden hangisi daha küçükse onu OUT çıkışına yazar.



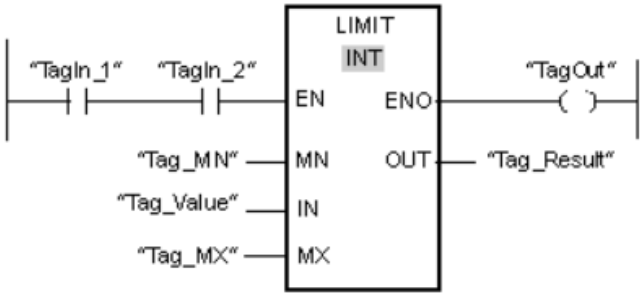

▼ MAX (Get maximum)

IN1, IN2, IN3 girişlerinden hangisi daha büyükse onu OUT çıkışına yazar.



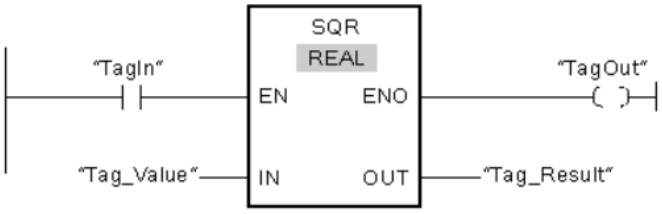

▼ LIMIT (Set limit value)

IN girişine yazılan değeri limitler. MIN ve MAX girişlerine sayılar yazılır eğer IN girişi MIN girişine yazılan sayıdan küçükse OUT çıkışına MIN girişi yazılır. Eğer IN girişi MAX girişindeki sayıdan büyükse OUT çıkışına MAX girişindeki sayı yazılır.



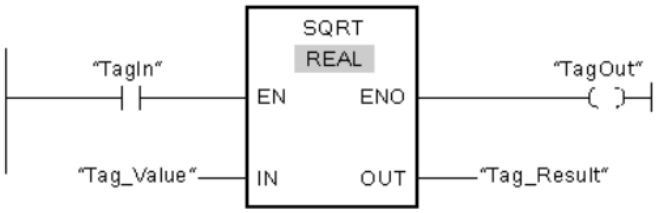

▼ SQR (Form square)

IN girişine yazılan sayının karesini alır.



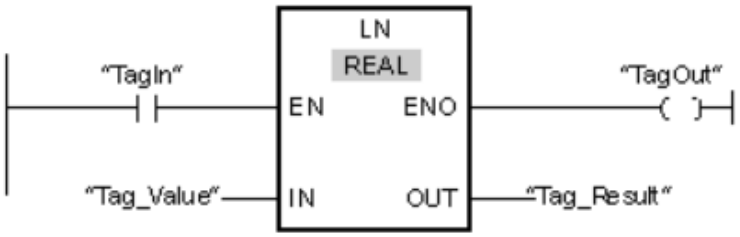

▼ SQRT (Form square root)

IN girişine girilen sayının kökünü alır.



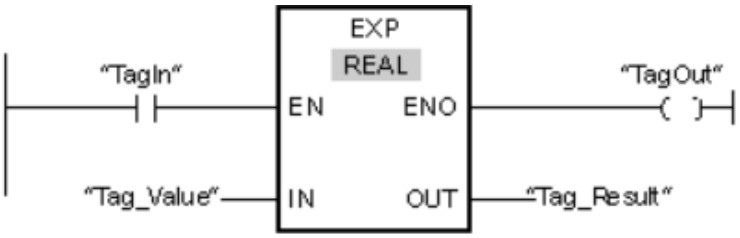

▼ LN (Form Natural Logarithm)

IN girişine girilen sayının Natural Log. unu alır.



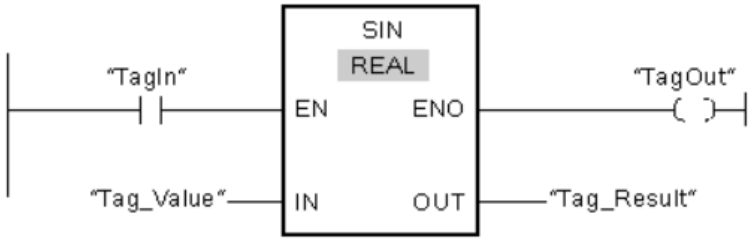

▼ EXP (Form exponantial value)

IN girişine girilen sayının Exponansiyalini alır.



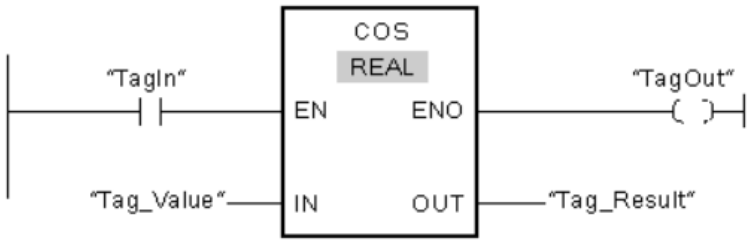

▼ SIN (Form sine value)

IN girişine girilen sayının sinüs karşılığını yazdırır.



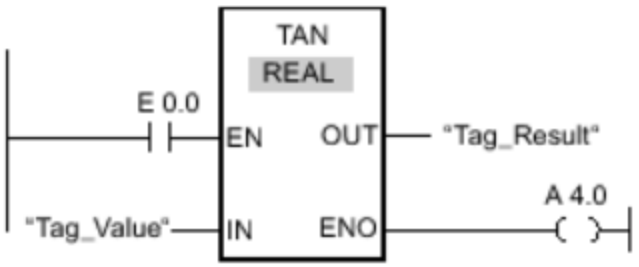

▼ COS (Form cosine value)

IN girişine girilen sayının cos karşılığını yazdırır.



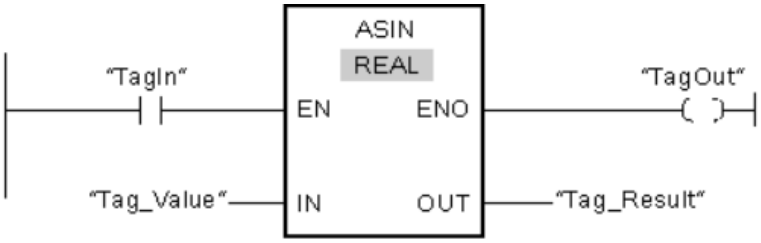

▼ TAN (Form tangent value)

IN girişine girilen sayının tanjant karşılığını yazdırır.



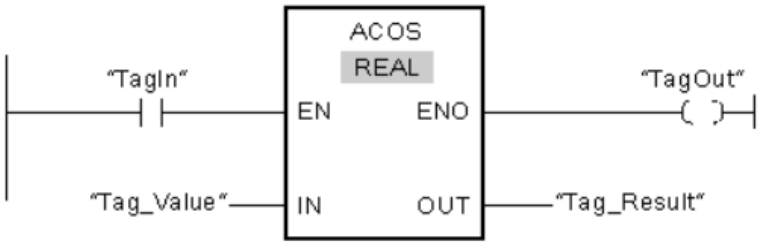

▼ ASIN (Form arcsine value)

IN girişine girilen sayının ters sinüs karşılığını yazdırır.



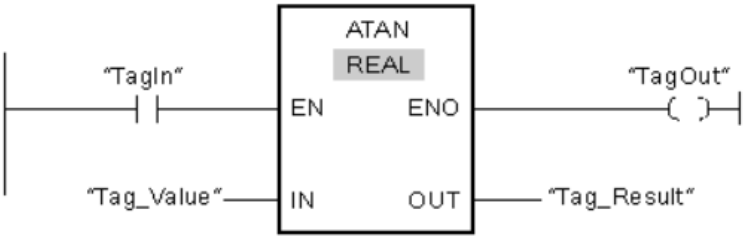
▼ ACOS (Form arccosine value)

IN girişine girilen sayının ters cos karşılığını yazdırır.



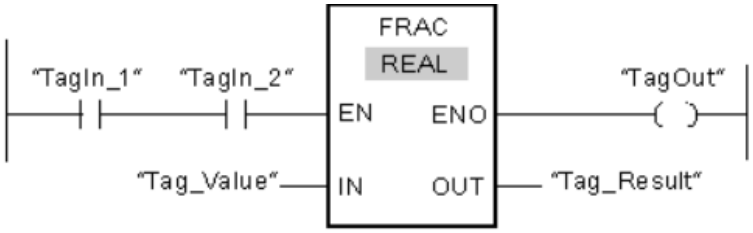

▼ ATAN (Form arctangent value)

IN girişine girilen sayının ters tan karşılığını yazdırır.



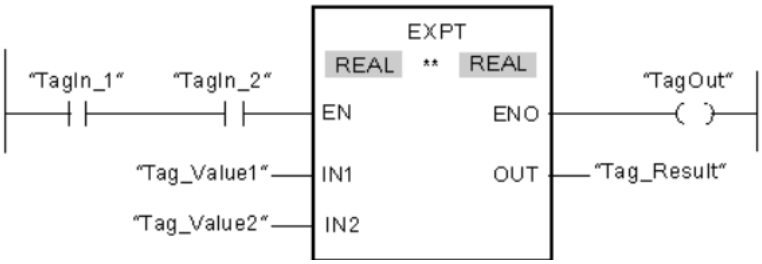

▼ FRAC (Return Fraction)

IN girişine girilen sayının sadece kesir kısmını yazdırır.




▼ EXPT (Exponentiate)

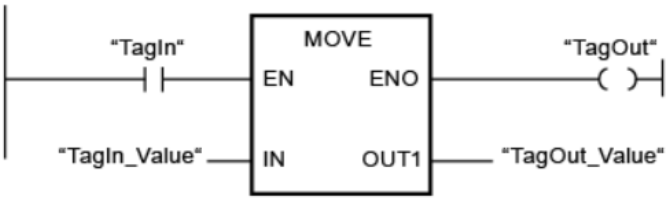
IN girişine girilen sayının Exponansiyalini alır.




# Move operation

▼ MOVE (Move value)

IN girişine yazılan değeri OUT çıkışına yazar.

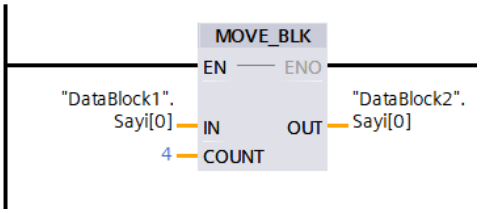



▼ Deserialize

▼ Serialize

▼ MOVE_BLK (Move block)

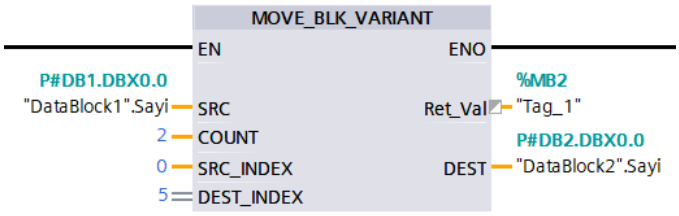
Bu komut bir data blocktaki arrayi diğer data bloağa taşımak için kullanılır.  IN kısmına data blockta hangi yerden başlayacağı girilir. COUNT kısmına kaç veri yazılacağı girilir. OUT kısmına ise yazılacak data block girilir.




▼ MOVE_BLK_VARIANT

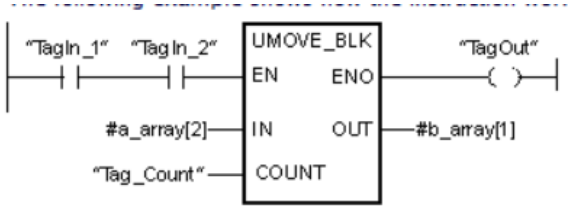
Bu komutta bir datablocktan diğerine array taşımayı sağlar fakat bu sefer arraydeki verileri diğer arrayin istediğimiz yerine koyabilme fırsatı sunar. SRC girişine taşıyacağımız array girilir, COUNT kaç adet veri taşıyacağımızdır, SRC INDEX taşıyacağımız arrayin hangi indexinden başlıyacağımızı seçeriz,

DEST_INDEX hedef arrayin hangi indexinden yazmaya başlayacağız onu seçeriz. DEST hedef datablock arrayi.



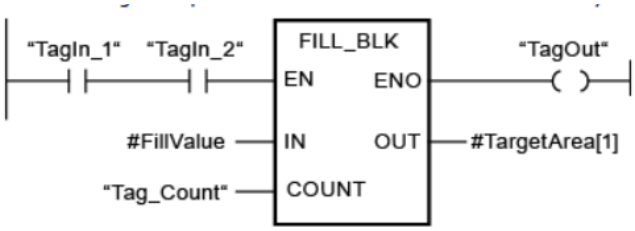

▼ UMOVE_BLK (Move block uninterruptible)

MOVE_BLK nin aynısıdır sadece programda interrupt varsa UMOVE_BLK etkilenmez.



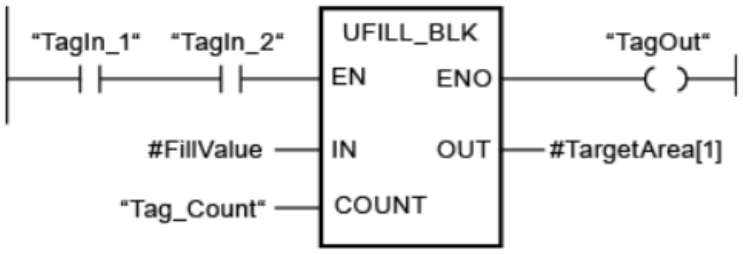

▼ FILL_BLK (Fill block)

IN kısmına girdiğimiz bool ifade , INT ifade Float, Char, Date... gibi olan ifadeyi COUNT sayısı kadar OUT' a girmiş olduğumuz array'e yazar.



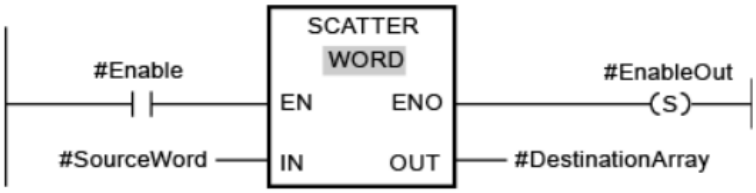

▼ UFILL_BLK (Fill block uninterruptible)

FILL_BLK'nın aynısıdır sadece interruptlardan etkilenmez.



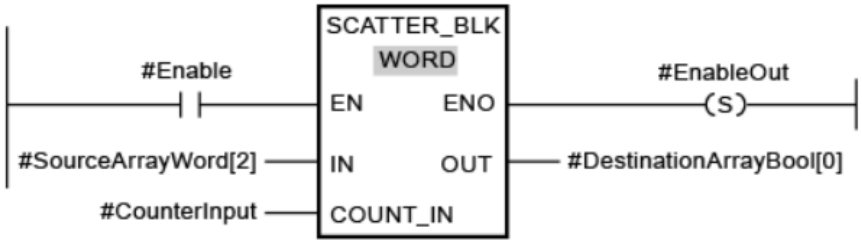

▼ SCATTER (Parse the bit sequence into individual bits)

IN kısmına BYTE, WORD, DWORD olarak girilen ifadeyi bool şeklinde bitlere ayıran fonksiyondur. Çalışması için IN kısmına WORD bir ifade yazdıysak diğer kısım için 15 elemanlı BOOL tipli bir array oluşturmak zorundayız.



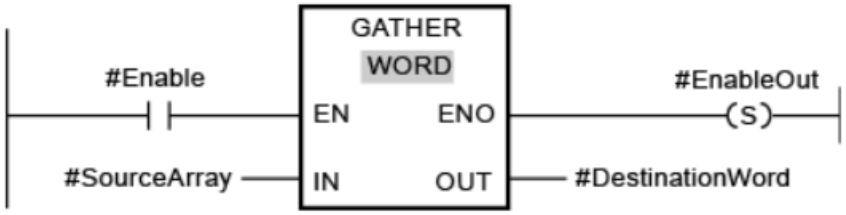

▼ SCATTER_BLK (Parse elements of an ARRAY of bit sequence into individual bits)

IN kısmına girilen BYTE, WORD, DWORD ifadeleri bool şeklinde bitlere ayırır. çalışması için IN kısmına 6 elemanlı WORD bir ifade girdiysek OUT kısmı için 96 elemanlı bir bool arrayi oluşturmamız gerekiyor. COUNT_IN kismına yazdığımız sayı kadar taşır mesela 3 yazarsak 3 WORD taşır.



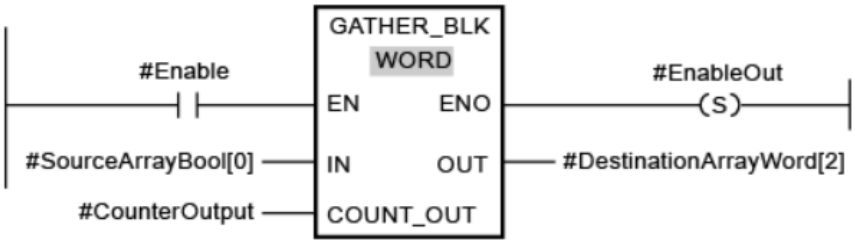

▼ GATHER (Merge individual bits into a bit sequence)

SCATTER'ın tersi işlev görür. Tek tek bool ifadeli arrayi WORD ifadesine çevirebilir. Çalışması için 16 elemanlı bool arrayine karşılık bir WORD ifadesi yazılması gerekir.



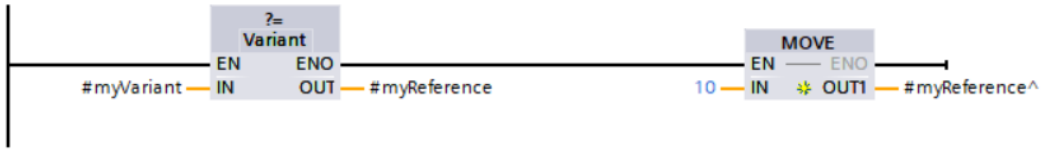

▼ GATHER_BLK (Merge individual bits into multiple elements of an ARRAY of bit sequence)

IN kısmına BOOL şeklinde yazdığımız ifadeyi birleştirip BYTE,WORD,DWORD şeklinde yazar. çalışması için 96 bool'luk bir arrayimiz var ise karşısında 6 elemanlı WORD arrayi olması gerekiyor. COUNT kısmına yazılan sayı kadar WORD oluşturur.




▼ AssigmentAttempt (Attempt assignment to a reference)

Fonksiyon bloklarda kullanılır IN kısmına yazılan değişkenin OUT kısmına yazılan değişken türü ile uyuşup uyuşmadığını kontrol eder.

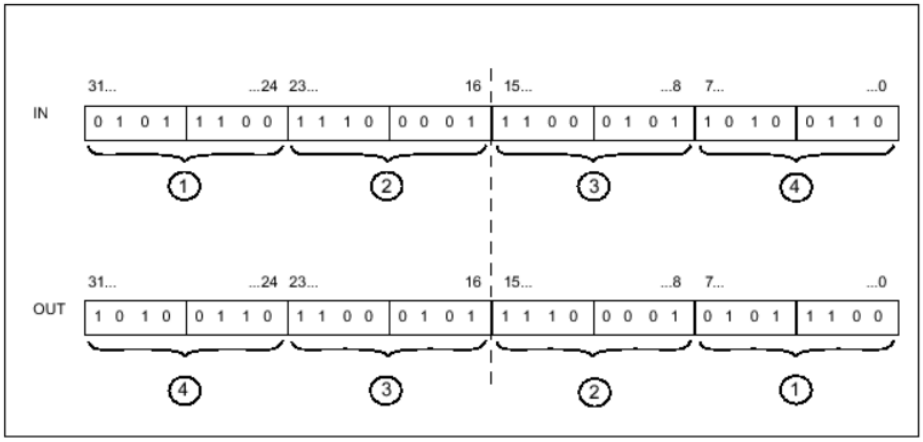



▼ SWAP (Swap)

**Description**

You can use the "Swap" instruction to change the order of the bytes at input IN and query the result at output OUT.

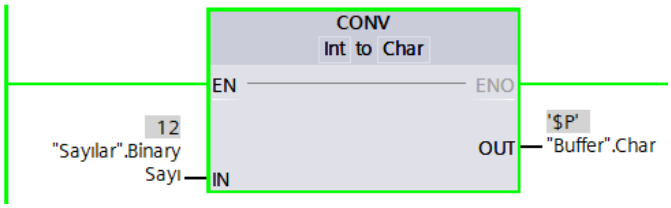
The following figure shows how the bytes of an operand of the DWORD data type are swapped using the "Swap" instruction:




# Conversion operations

▼ CONVERT (Convert)

IN kısmına yazılan parametreyi okur ve instruction boxta seçilen data tipine çevirir.

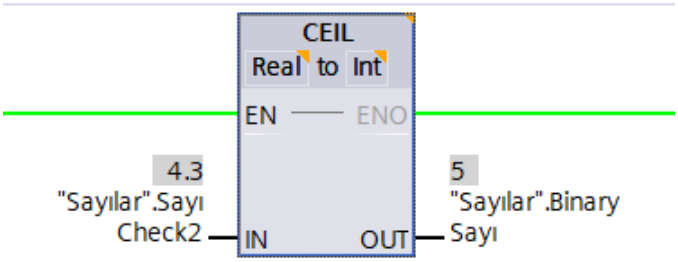



▼ ROUND (Round numerical value)

IN girişine girilen değeri yuvarlama yaparak OUT'a yazar.

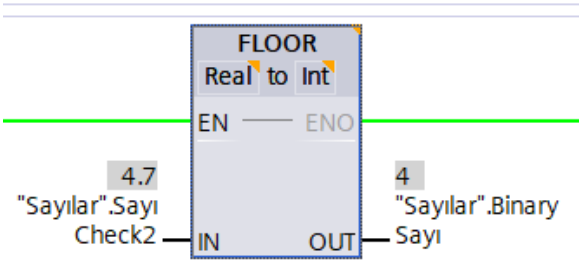

▼ CEIL (Generate next higher integer from floating-point number)

IN kısmına girilen sayıyı her türlü bir üst sayıya yuvarlayıp OUT kısmına yazar.



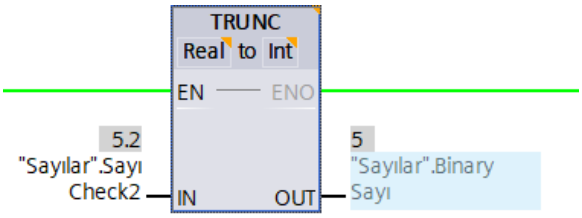

▼ FLOOR (Generate next lower integer from floating-point number)

IN kısmına girilen sayıyı her türlü bir alt sayıya yuvarlayıp OUT kısmına yazar.



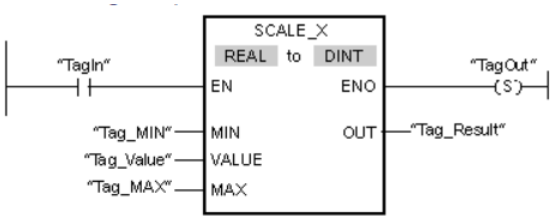

▼ TRUNC (Truncate numerical value)

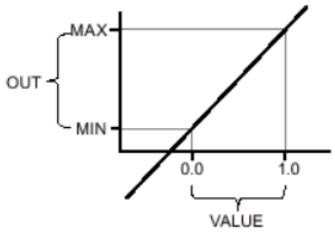
IN kısmına girilen sayının virgüllü kısmını çıkarır sadece tam sayı kısmını OUT kısmına yazar.




▼ SCALE_X (Scale)

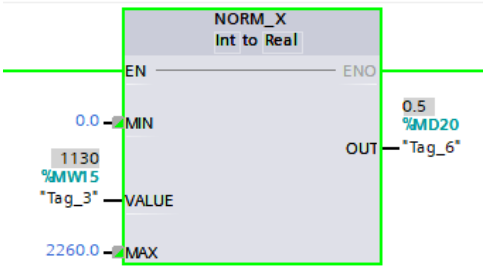
VALUE kısmına yazılan sayıyı MİN ve MAX kısmına yazılan sayılar arasına ölçeklendirir.

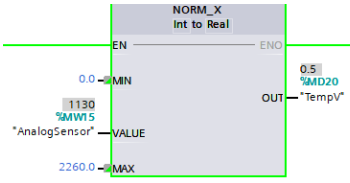


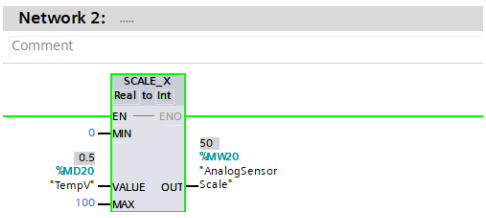




▼ NORM_X (Normalize)

VALUE kısmına girilen değeri OUT kısmında MIN ve MAX değerine göre ölçeklendirip 0-1 arasında değer gösterir.



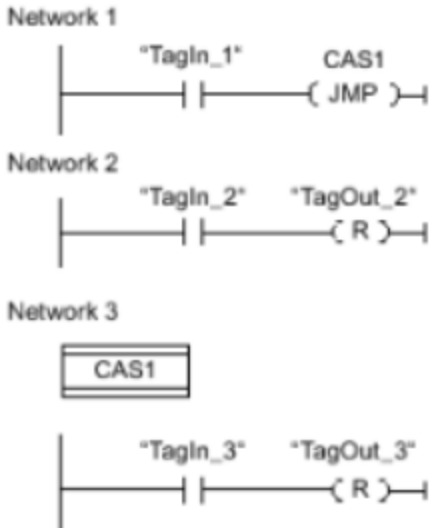




# Program control operation

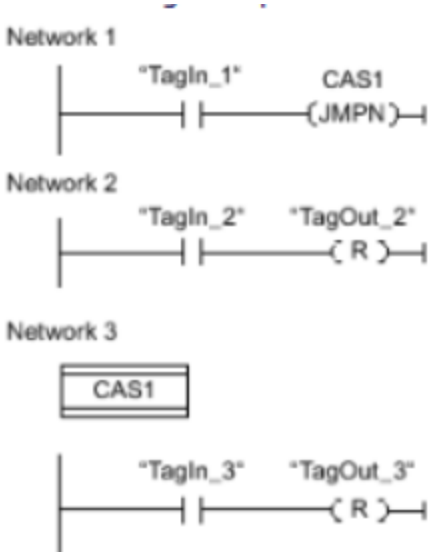
▼ ---( JMP ) (Jump if RLO = 1)

Eğer jump komutuna ‘1’ sinyali gelirse JMP komutunda belirtilen etikete atlama yapar eğer etiket ile JMP komutu arasında network varsa devre dışı kalır.




▼ ---( JMPN ) (Jump if RLO = 0)

Eğer jump komutuna ‘0’ sinyali gelirse JMP komutunda belirtilen etikete atlama yapar eğer etiket ile JMP komutu arasında network varsa devre dışı kalır.



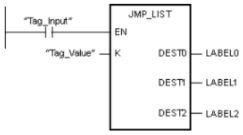

▼ LABEL  (Jump label)

Networklere LABEL koymak için kullanılır JMP komutu için gereklidir.

▼ JMP_LIST  (Define jump list)

Bu komut ile atlama listesi oluşturulabilir K değeri hangi etkieti gösteriyor ise EN kısmına ‘1’ geldiğinde o kısma atlar.

The following example shows how the instruction works:



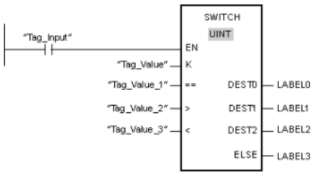

The following table shows how the instruction works using specific operand values:

| Parameters | Operand/jump label | Value |
|---|---|---|
| K | Tag_Value | 1 |
| Dest0 | LABEL0 | Jump to the network that is identified with the jump label "LABEL0". |
| Dest1 | LABEL1 | Jump to the network that is identified with the jump label "LABEL1". |
| Dest2 | LABEL2 | Jump to the network that is identified with the jump label "LABEL2". |

▼ SWITCH (Jump distributor)

K değerine göre atlama etiketi seçmemizi sağlayan komuttur.

The following example shows how the instruction works:




The following table shows how the instruction works using specific operand values:

| Parameters | Operand/jump label | Value |
|---|---|---|
| K | Tag_Value | 23 |
| == | Tag_Value_1 | 20 |
| > | Tag_Value_2 | 21 |
| < | Tag_Value_3 | 19 |
| DEST0 | LABEL0 | Jump to jump label "LABEL0", if the value of parameter K is equal to 20. |
| DEST1 | LABEL1 | Jump to jump label "LABEL1", if the value of parameter K is greater than 21. |
| DEST2 | LABEL2 | Jump to jump label "LABEL2", if the value of parameter K is less than 19. |
| ELSE | LABEL3 | Jump to jump label "LABEL3", if the none of the comparison conditions are fulfilled. |

If the operand "Tag_Input" changes to signal state "1", the instruction "Jump distributor" is executed. The execution of the program is continued in the network that is identified with the jump label "LABEL1".

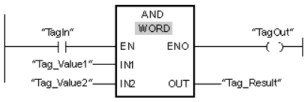
▼ -(RET) (Return)

RET komutu bir networkte çağrıldı ise o netwrok’ün altınaki networkler devre dışı kalır.


# Word logic operation

▼ AND (And logic operation)

IN1 ve IN2 ye girilen değerleri AND işlemine sokar.

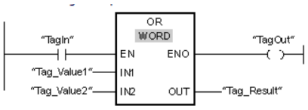



The following table shows how the instruction works using specific operand values:

| Parameter | Operand | Value |
|---|---|---|
| IN1 | Tag_Value1 | 0101 0101 0101 0101 |
| IN2 | Tag_Value2 | 0000 0000 0000 1111 |
| OUT | Tag_Result | 0000 0000 0000 0101 |

▼ OR (OR logic operation)

IN1 ve IN2 ye girilne değerleri OR işlemine sokar.

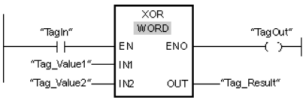



The following table shows how the instruction works using specific operand values:

| Parameter | Operand | Value |
|---|---|---|
| IN1 | Tag_Value1 | 0101 0101 0101 0101 |
| IN2 | Tag_Value2 | 0000 0000 0000 1111 |
| OUT | Tag_Result | 0101 0101 0101 1111 |

▼ XOR (Exclusive OR operation)

IN1 ve IN2 ye girilen değerleri XOR işlemine sokar.



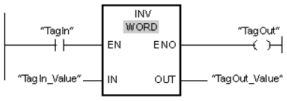

The following table shows how the instruction works using specific operand values:

| Parameter | Operand | Value |
|---|---|---|
| IN1 | Tag_Value1 | 0101 0101 0101 0101 |
| IN2 | Tag_Value2 | 0000 0000 0000 1111 |
| OUT | Tag_Result | 0101 0101 0101 1010 |

▼ INVERT (Create one’s complemet)

IN girişine girilen sayının 1’s complementini alır.

The following example shows how the instruction works:

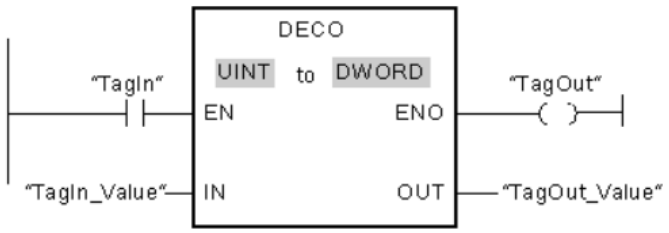



The following table shows how the instruction works using specific operand values:

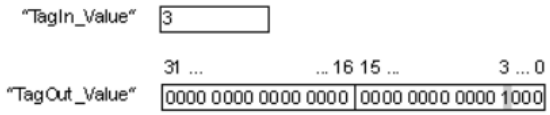
| Parameter | Operand | Value | |
|---|---|---|---|
| IN | TagIn_Value | W#16#000F | W#16#7E |
| OUT | TagOut_Value | W#16#FFF0 | W#16#81 |

▼ DECO (Decode)

OUT kısmındaki değerin IN kısmındaki bitini set eder.

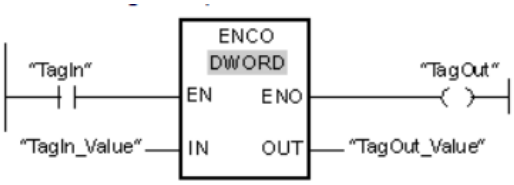



The following figure shows how the instruction works using specific operand values:

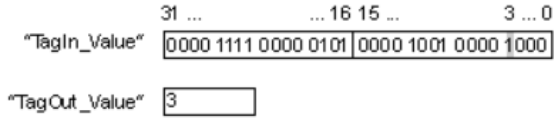



▼ ENCO (Encode)

"Encode" talimatı, IN girişindeki değerin en önemsiz bitini seçer ve bit numarasını OUT çıkışındaki etikete yazar. IN girişi DW#16#00000001 veya DW#16#00000000 değerine sahipse, OUT çıkışında "0" değeri verilir



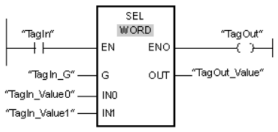

The following figure shows how the instruction works using specific operand values:




▼ SEL (Select)

IN1 veya IN2 girişini seçip OUT'a yazmamızı sağlayan fonksiyondur. Eğer G '0' ise IN1 yazılır. G '1' ise IN2 yazılır.

The following example shows how the instruction works:

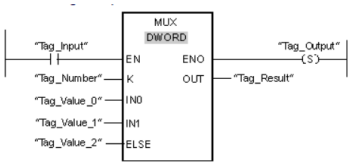



The following table shows how the instruction works using specific operand values:

| Parameter | Operand | Value | |
|---|---|---|---|
| G | TagIn_G | 0 | 1 |
| IN0 | TagIn_Value0 | W#16#0000 | W#16#4C |
| IN1 | TagIn_Value1 | W#16#FFFF | W#16#5E |
| OUT | TagOut_Value | W#16#0000 | W#16#5E |

▼ MUX (Multiplex)

SEL komutuyla aynı şekilde çalışır. SEL sadece 2 input için işlem yaparken MUX komutu 32 input için işlem yapabilir. K inputuna girilen sayıya göre inputu seçip OUT'a yazdırır.

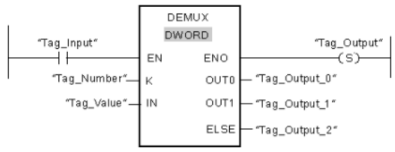



The following table shows how the instruction works using specific operand values:

| Parameters | Operand | Value |
|---|---|---|
| K | Tag_Number | 1 |
| IN0 | Tag_Value_0 | DW#16#00000000 |
| IN1 | Tag_Value_1 | DW#16#003E4A7D |
| ELSE | Tag_Value_2 | DW#16#FFFF0000 |
| OUT | Tag_Result | DW#16#003E4A7D |

▼ DEMUX (Demultiplex)

IN girişine girilen değeri K girişi ile seçilen OUT'a kopyalama işlemi yapar.




The following tables show how the instruction works using specific operand values:

Input values of the "Demultiplex" instruction before network execution

| Parameter | Operand | Values | |
|---|---|---|---|
| K | Tag_Number | 1 | 4 |
| IN | Tag_Value | DW#16#FFFFFFFF | DW#16#003E4A7D |

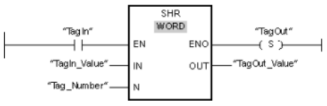
Output values of the "Demultiplex" instruction after network execution

| Parameter | Operand | Values | |
|---|---|---|---|
| OUT0 | Tag_Output_0 | Unchanged | Unchanged |
| OUT1 | Tag_Output_1 | DW#16#FFFFFFFF | Unchanged |
| ELSE | Tag_Output_2 | Unchanged | DW#16#003E4A7D |

# Shift and rotate

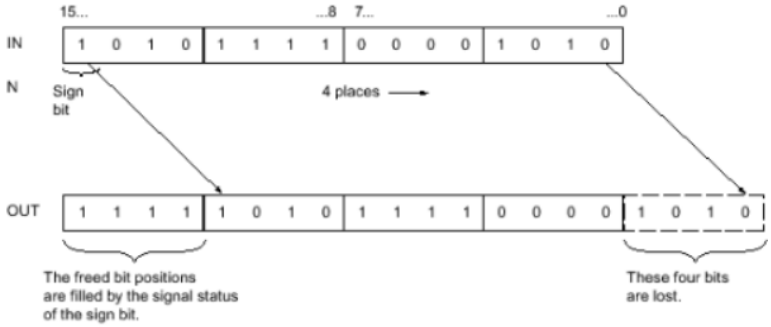
▼ SHR (Shift Right)

Girilen N değeri kadar IN kısmındaki değerin bitlerini sağa doğru kaydırır.



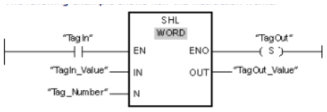

The following table shows how the instruction works using specific operand values:

| Parameters | Operand | Value |
|---|---|---|
| IN | TagIn_Value | 0011 1111 1010 1111 |
| N | Tag_Number | 3 |
| OUT | TagOut_Value | 0000 0111 1111 0101 |



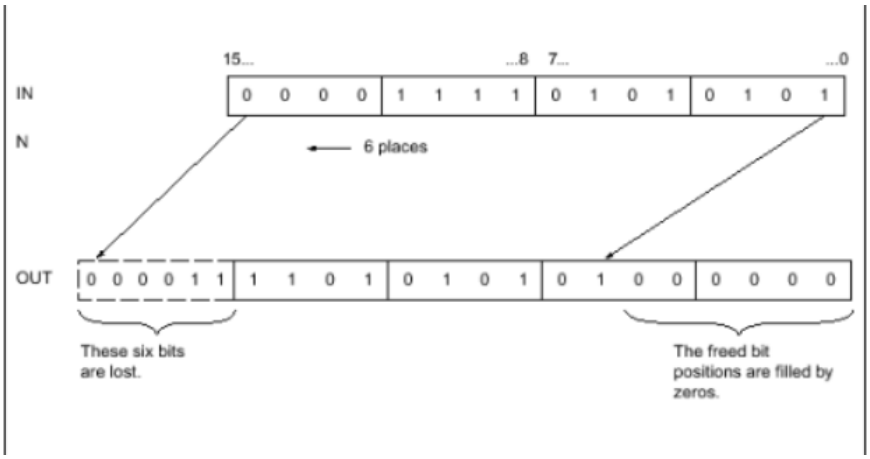

▼ SHL (Shift Left)

Girilen N değeri kadar IN kısmındaki değerin bitlerini sola doğru kaydırır.




The following table shows how the instruction works using specific operand values:

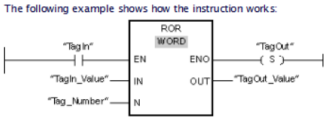
| Parameters | Operand | Value |
|---|---|---|
| IN | TagIn_Value | 0011 1111 1010 1111 |
| N | Tag_Number | 4 |
| OUT | TagOut_Value | 1111 1010 1111 0000 |




▼ ROR (Rotate Right)

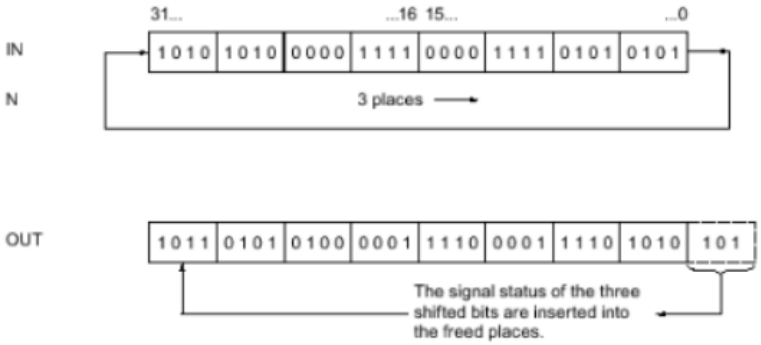
IN girişine girilen değerin en sağından N kadar bit alıp bunları değerin en soluna taşır.

The following example shows how the instruction works:



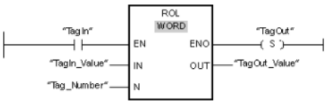

The following table shows how the instruction works using specific operand values:

| Parameters | Operand | Value |
|---|---|---|
| IN | TagIn_Value | 0000 1111 1001 0101 |
| N | Tag_Number | 5 |
| OUT | TagOut_Value | 1010 1000 0111 1100 |




▼ ROL (Rotate Left)

IN girişine girilen değerin en solundan N kadar bit alıp bunları değerin en sağına taşır.




The following table shows how the instruction works using specific operand values:

| Parameters | Operand | Value |
|---|---|---|
| IN | TagIn_Value | 1010 1000 1111 0110 |
| N | Tag_Number | 5 |
| OUT | TagOut_Value | 0001 1110 1101 0101 |