Standardowe języki programowania PLC – LAD – instrukcje podstawowe i rozszerzone.

Dr inż. Patrycja Paduchowicz

Konsultacje: środa 9:15-11:00(TP)/11:15-13:00(TN) oraz piątek 11:15-13:00

Bloki

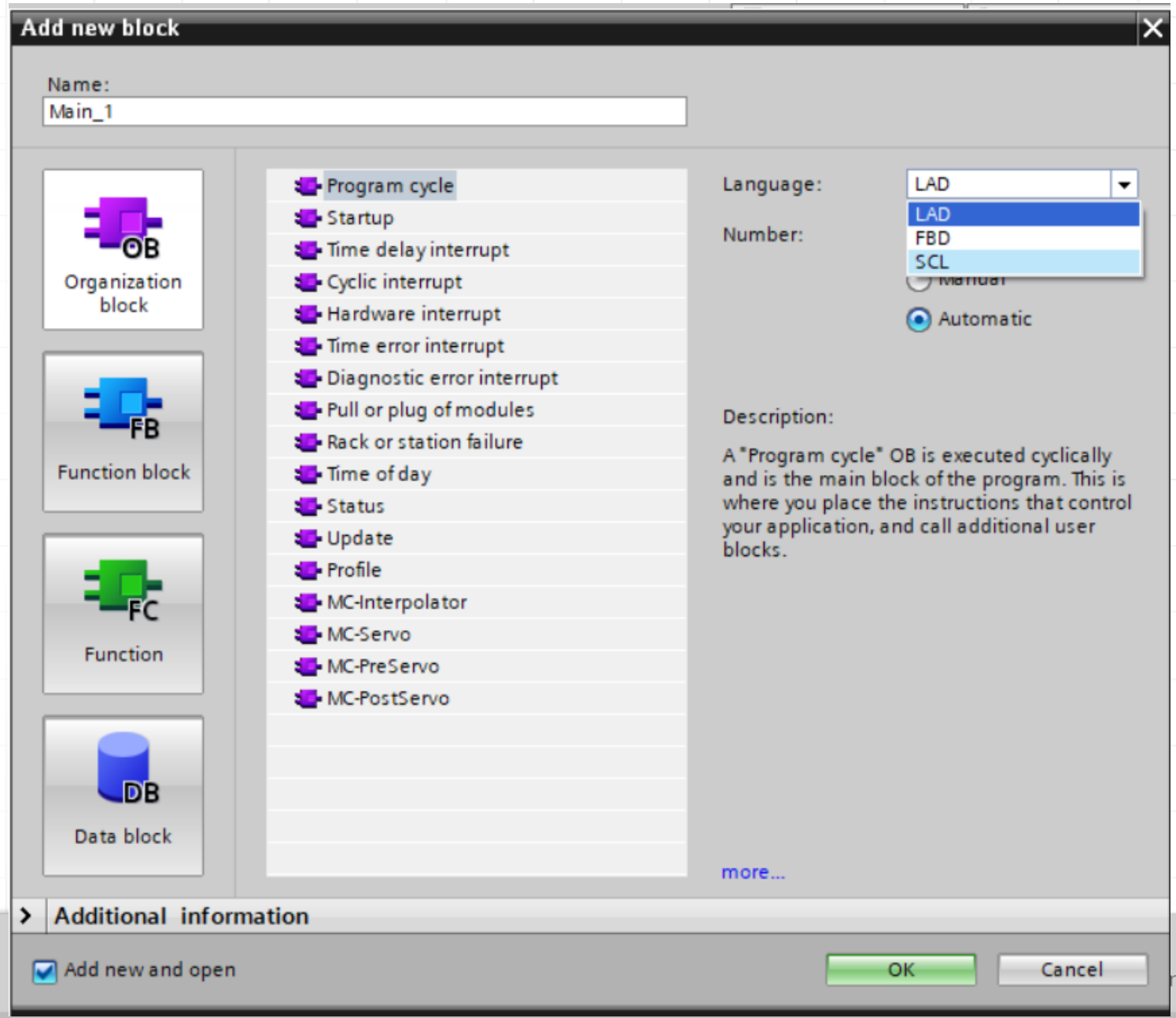
Przy programowaniu systemów automatyki w TIA Portal możemy skorzystać z następujących bloków:

Bloki organizacyjne (OB)	Bloki organizacyjne definiują strukturę programu użytkownika.
Funkcje (FC)	Funkcje zawierają procedury programu dla powtarzających się zadań. Nie mają „pamięci”.
Blok funkcyjny (FB)	Bloki funkcyjne to bloki kodu, które przechowują swoje wartości na stałe w blokach danych instancji, dzięki czemu pozostają dostępne nawet po wykonaniu bloku.
Bloki danych instancji	Bloki danych instancji są przypisywane do bloku funkcji, gdy jest on wywoływany w celu przechowywania danych programu.
Globalne bloki danych	Globalne bloki danych to obszary danych do przechowywania danych, które mogą być używane przez dowolne bloki.

Standardowe języki programowania PLC – LAD.

Bloki

Dla każdego z bloków możemy wybrać dowolny język programowania:



LAD – język drabinkowy

LAD jest graficznym językiem programowania. Jego wygląd jest oparty na schematach obwodów.

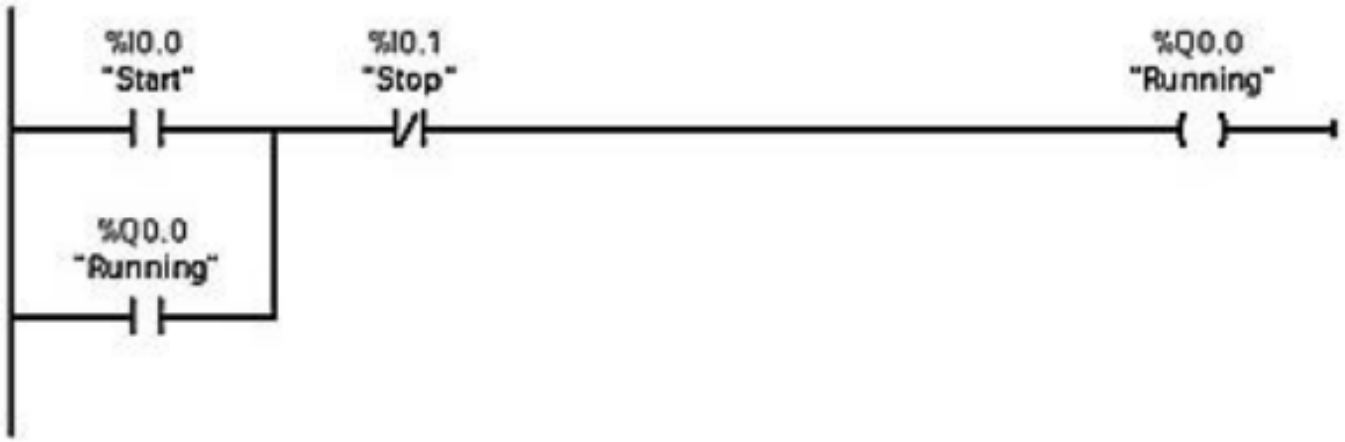
LAD – język drabinkowy

Elementy obwodu, takie jak styki normalnie zwarte lub normalnie rozwarte i cewki, są ze sobą łączone w celu utworzenia sieci.
W celu zaprojektowania logiki dla złożonych operacji, na schemacie można umieszczać gałęzie dla utworzenia logiki obwodów równoległych. Gałęzie równoległe są na początku rozwarte lub bezpośrednio łączone do szyny zasilania. Od strony końcowej gałęzi występują połączenia zakańczające.
LAD umożliwia stosowanie instrukcji dla różnych funkcji, takich jak arytmetyczne, czasowe, zliczające oraz związane z napędami.

Uwaga
Każda sieć LAD musi kończyć się cewką lub instrukcją.

LAD – logika bitowa

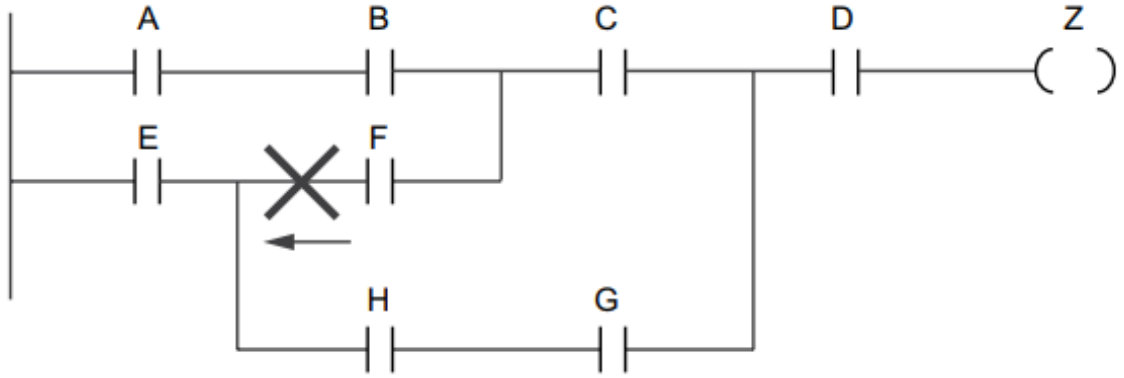
Podstawowymi elementami instrukcji logicznych są styki i cewki. Styki odczytują stan bitu, a cewki nadpisują stan operacji do bitu.
Styki testują stan logiczny przypisanego bitu. Styki są zwarte
dla 1 i rozwarte dla 0.
Stan cewek jest zgodny z przetworzona logiką układu.



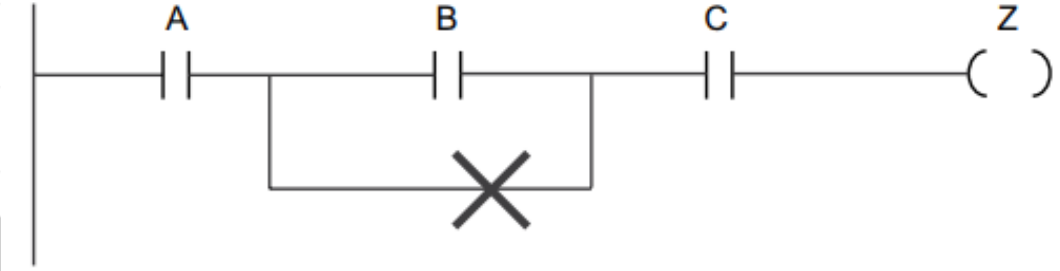
LAD – język drabinkowy

Podczas tworzenia schematu LAD, należy kierować się następującymi zasadami:

- Nie wolno tworzyć gałęzi, w której może nastąpić przepływ mocy w odwrotnym kierunku.



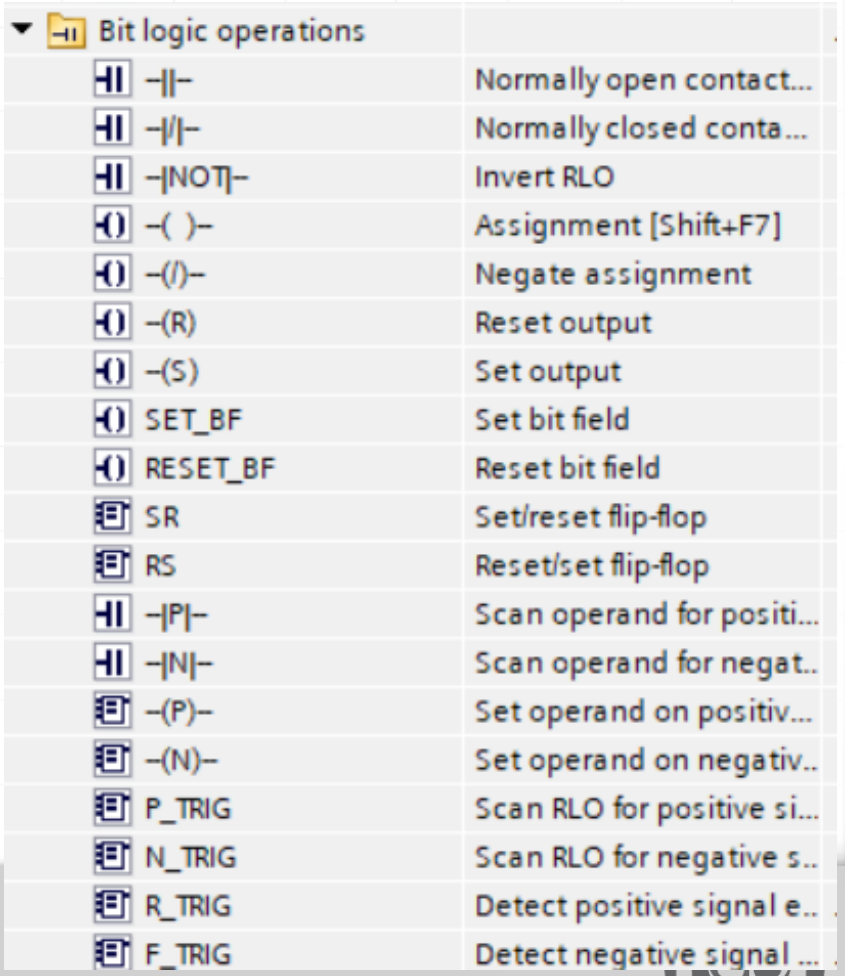
- Nie wolno tworzyć gałęzi, która mogłaby spowodować zwarcie.



LAD – logika bitowa

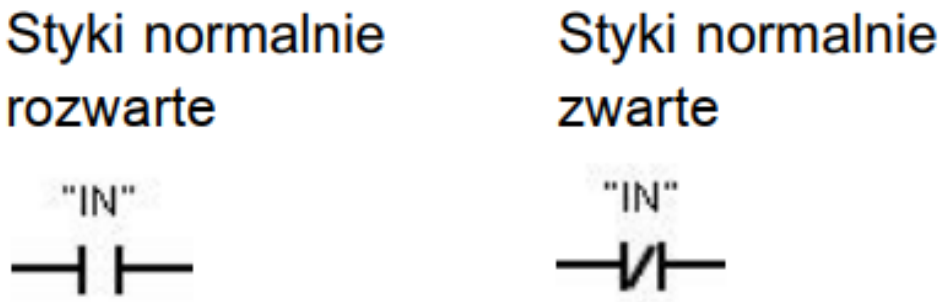
Instrukcja wyjściowa sterowania cewką ustala wartość bitu wyjściowego. Jeżeli zaprogramowany przez użytkownika bit wyjściowy ma identyfikator pamięci Q, to CPU włącza lub wyłącza ten bit w rejestrze obrazu procesu tak, by był zgodny ze stanem zasilania. Wyjściowe sygnały sterujące urządzeniami wykonawczymi są podłączone do zacisków wyjściowych sterownika PLC. W trybie RUN, system CPU w sposób ciągły skanuje sygnały wejściowe, przetwarza te sygnały zgodnie z logiką programu i w rezultacie ustala nowe wartości stanów wyjściowych w rejestrze wyjściowym obrazu procesu. Po zakończeniu każdego cyklu programu, CPU przesyła te nowe wartości stanów wyjściowych zapamiętane w rejestrze obrazu procesu do okablowanych zacisków wyjściowych.

LAD – Bit logic operations



LAD – styki

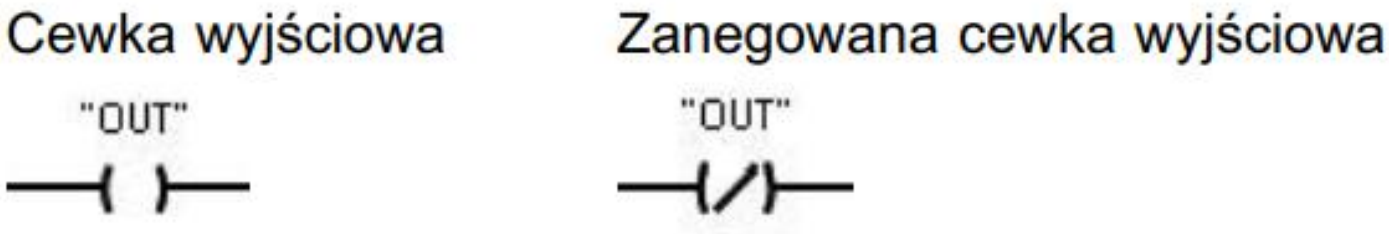
Styki normalnie rozwarte (*normally open*) są
zwarte (ON) wtedy, kiedy wartość przypisanego bitu jest równa 1.
Styki normalnie zwarte (*normally closed*) są
zwarte (ON) wtedy, kiedy wartość przypisanego bitu jest równa 0.



LAD – cewki

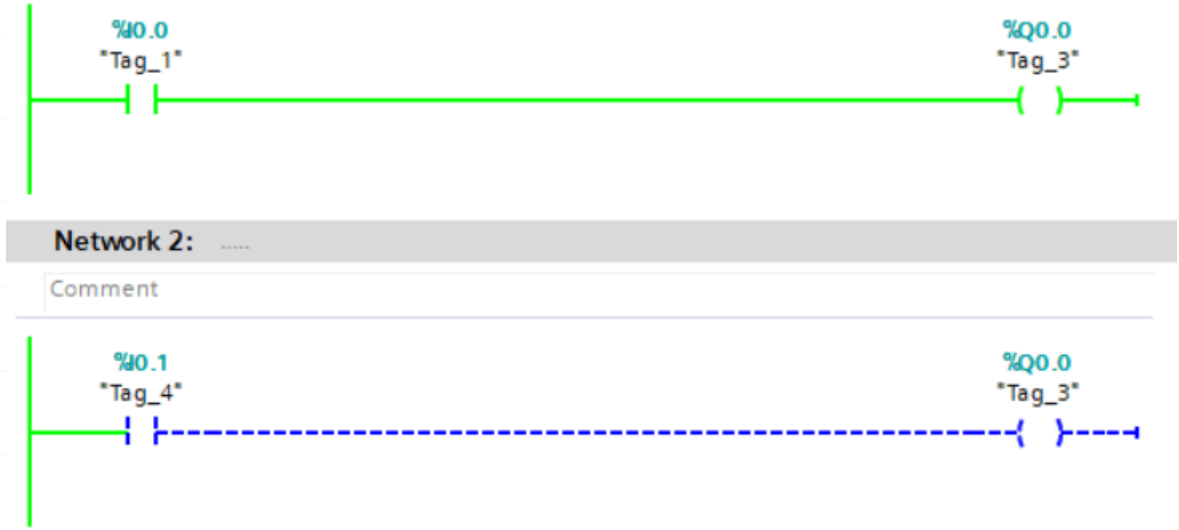
Należy zwrócić uwagę na następujące wartości bitów wyjściowych zależne od zasilania cewek wyjściowych oraz zanegowanych cewek wyjściowych:

- Jeżeli cewka wyjściowa jest zasilana, to wartość bitu wyjściowego jest ustalana na 1.
- Jeżeli cewka wyjściowa nie jest zasilana, to wartość bitu wyjściowego jest ustalana na 0.
- Jeżeli zanegowana cewka wyjściowa jest zasilana, to wartość bitu wyjściowego jest ustalana na 0.
- Jeżeli zanegowana cewka wyjściowa nie jest zasilana, to wartość bitu wyjściowego jest ustalana na 1.



LAD – cewki

Jeśli cewka o tym samym adresie jest użyta w więcej niż jednym miejscu programu, to podczas odświeżania fizycznych wyjść zostanie ustawiony stan odpowiadający stanowi ostatniej cewki.



"Tag_1":P	%I0.0:P	Bool	TRUE
"Tag_4":P	%I0.1:P	Bool	FALSE
"Tag_3"	%Q0.0	Bool	FALSE

LAD – negacja

Inwerter stykowy NOT języka LAD neguje wejściowy stan logiczny zasilania.

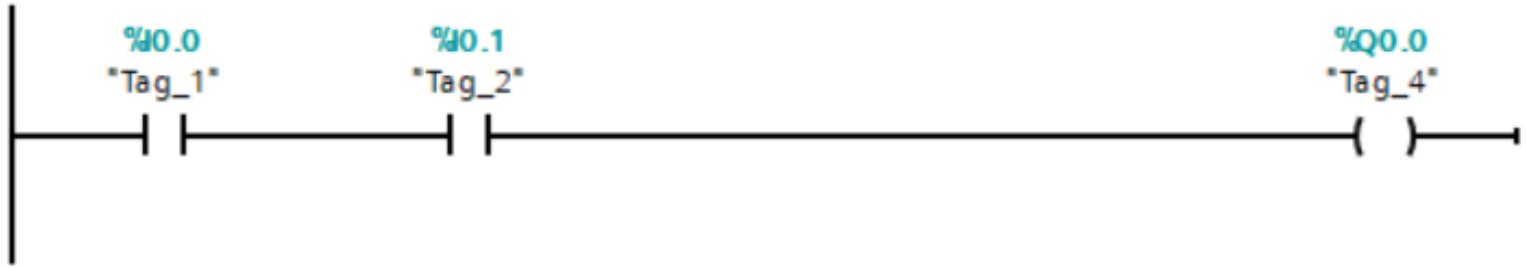
- Jeżeli na wejściu styku NOT nie ma zasilania, to na wyjściu zasilanie występuje.
- Jeżeli na wejściu styku NOT jest zasilanie, to na wyjściu zasilanie nie występuje.

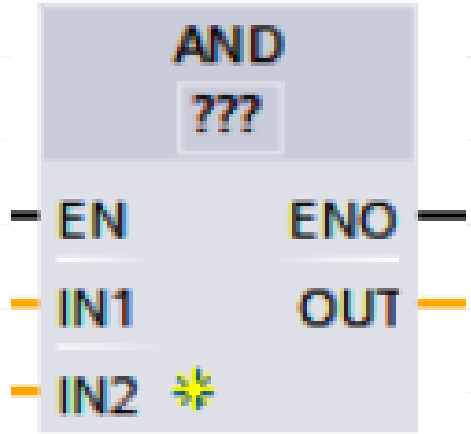
Inwerter stykowy

NOT (LAD)

LAD – funkcja logiczna AND

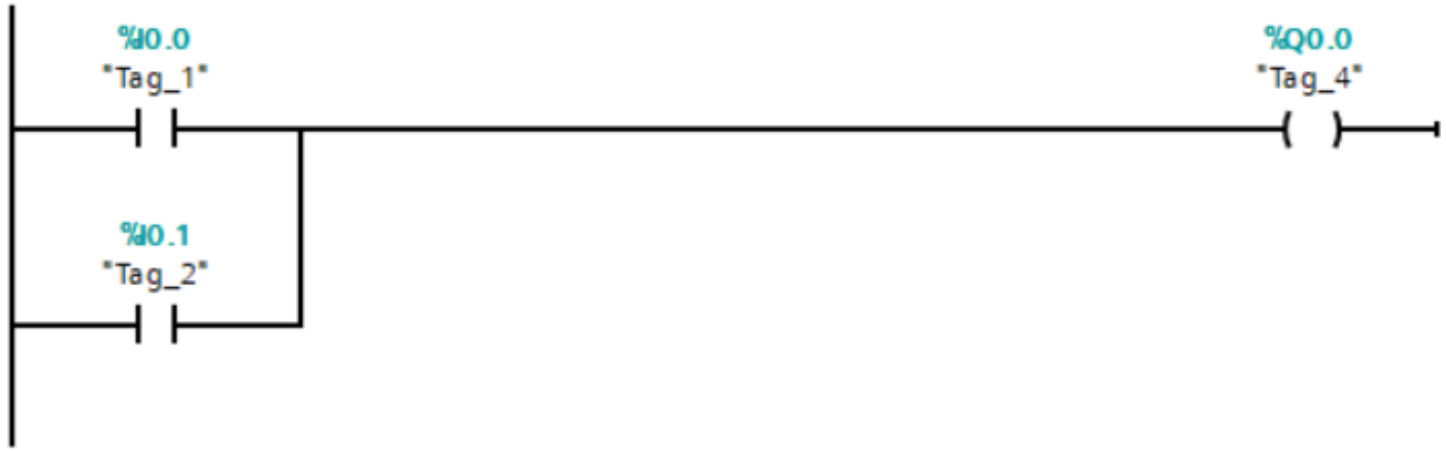
Aby stan wyjścia funkcji AND był TRUE, wszystkie wejścia muszą być w stanie TRUE.

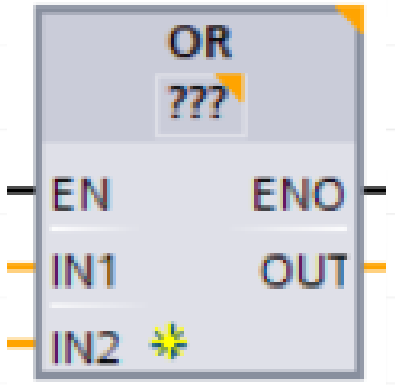




LAD – funkcja logiczna OR

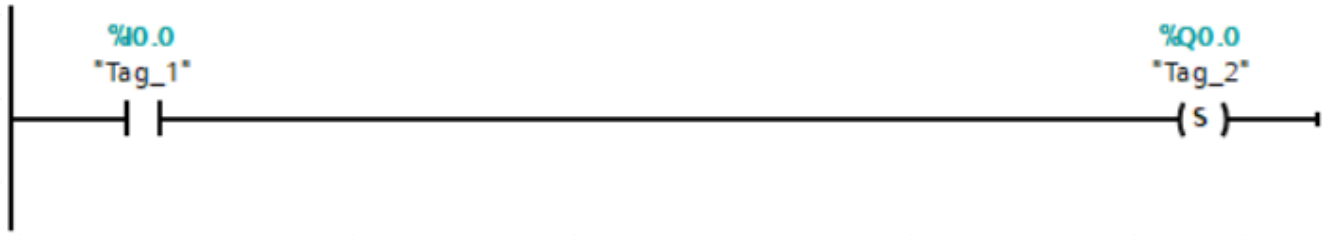
Aby stan wyjścia funkcji OR był TRUE, dowolne wejście musi być w stanie TRUE.





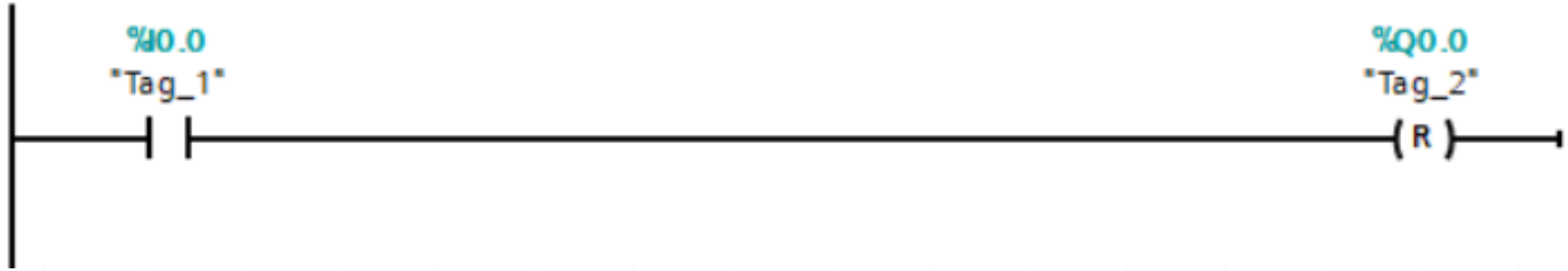
LAD – cewka SET

Stan wysoki na styku I0.0 powoduje wpisanie na cewce Q0.0 stałego stanu wysokiego. Stan wysoki będzie się utrzymywał nawet jeśli stan I0.0 zmieni się na niski. Cewka Q0.0 będzie w stanie wysokim dopóki nie zostanie zresetowana.



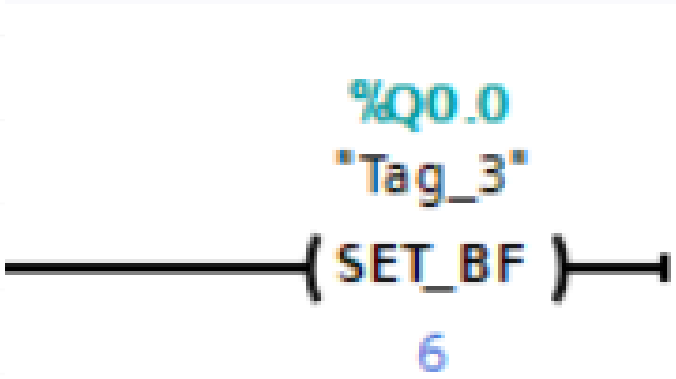
LAD – cewka RESET

Stan wysoki na styku I0.0 powoduje wpisanie na cewce Q0.0 stałego stanu niskiego. Stan niski będzie się utrzymywał nawet jeśli stan I0.0 zmieni się na niski. Cewka Q0.0 będzie w stanie niskim dopóki nie zostanie zasetowana.



LAD – funkcja logiczna SET_BF

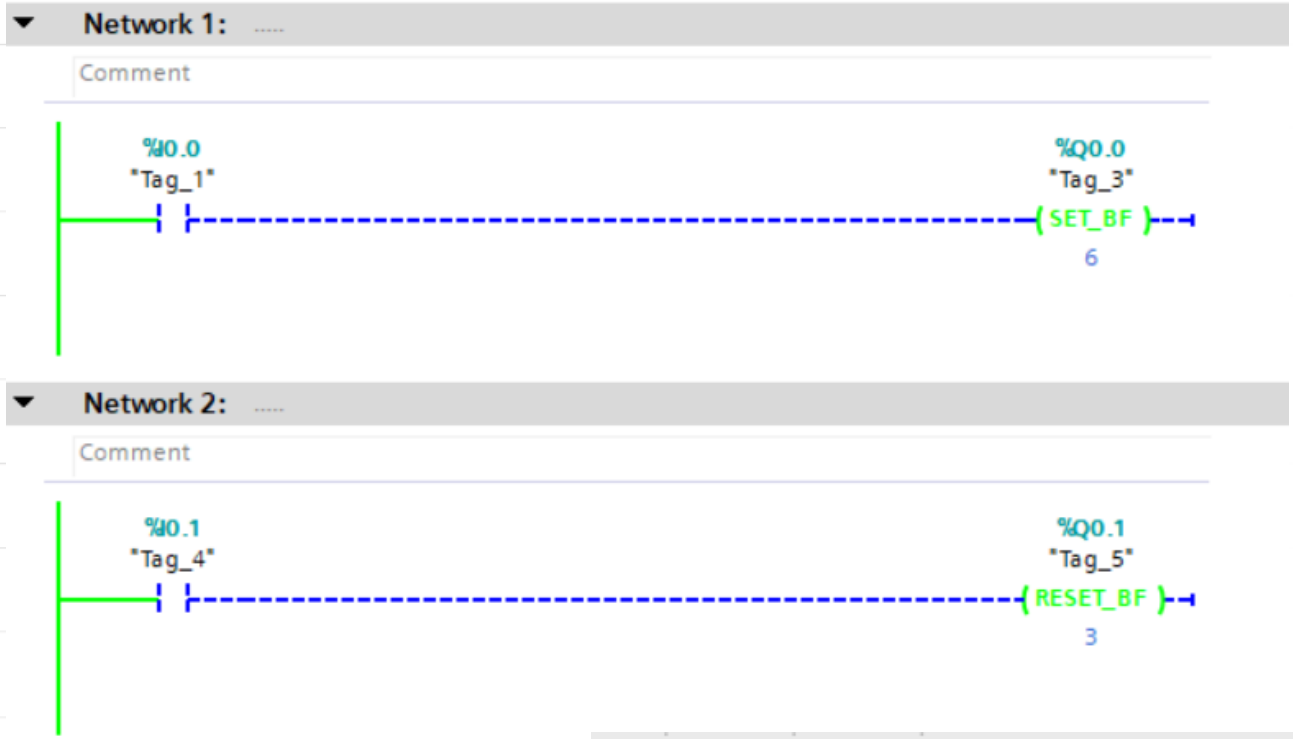
Funkcja SET_BF ustawia stan wysoki na określonej liczbie bitów (6), zaczynając od zadanego adresu (Q0.0).



LAD – funkcja logiczna RESET_BF

Funkcja RESET_BF ustawia stan niski na określonej liczbie bitów (3), zaczynając od zadanego adresu (Q0.1).

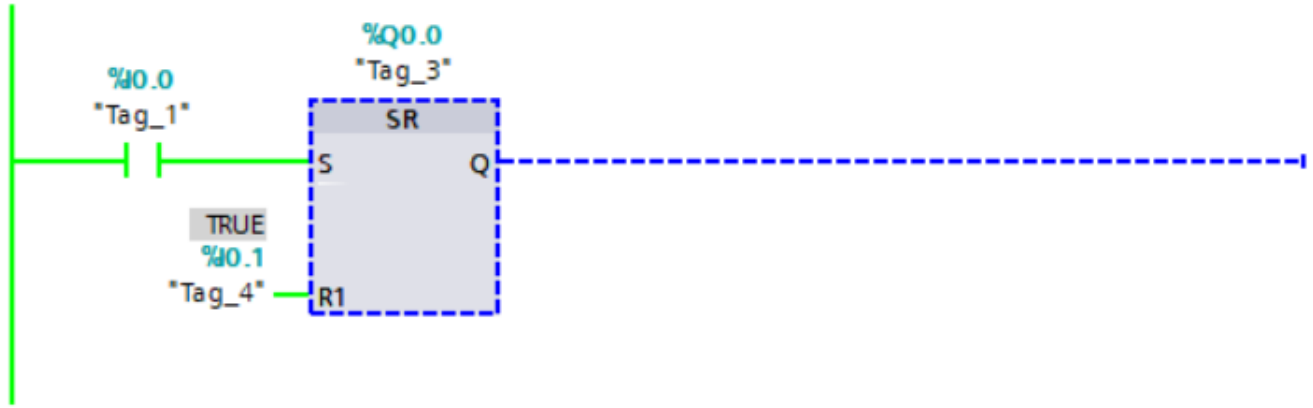
LAD – SET_BF i RESET_BF



Po prawej przedstawiono przykład działania funkcji SET_BF oraz RESET_BF po jednym cyklu pracy programu.

Name	Address	Display format	Monitor/Modify value	Bits	Consistent modify
"Tag_1":P	%I0.0:P	Bool	FALSE	☐	FALSE
"Tag_4":P	%I0.1:P	Bool	FALSE	☐	FALSE
"Tag_2"	%QB0	Hex	16#31	☐☐☑☑☐☐☐☑	16#00
"Tag_3"	%Q0.0	Bool	TRUE	☑	FALSE
"Tag_5"	%Q0.1	Bool	FALSE	☐	FALSE

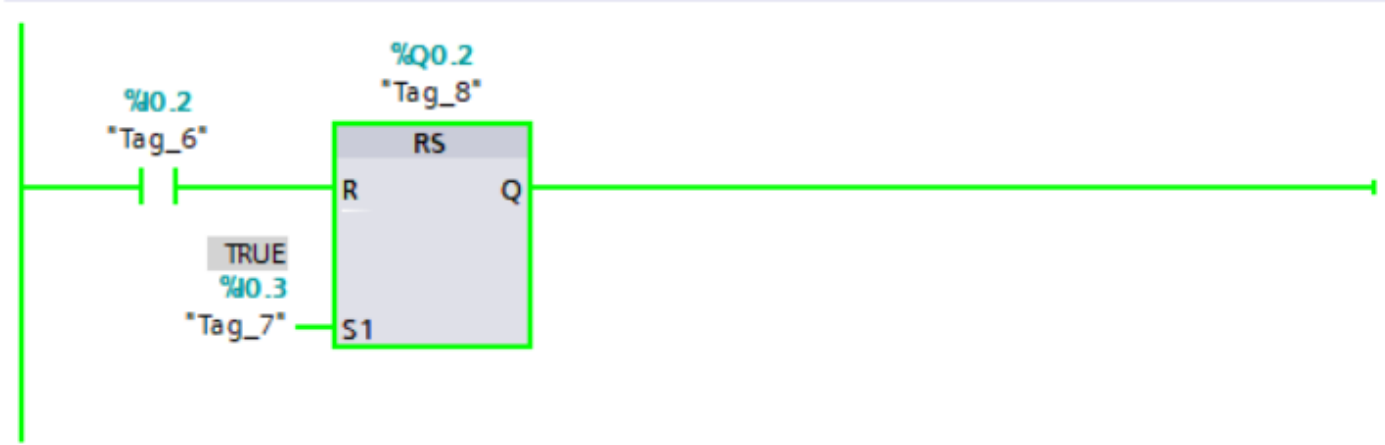
LAD – SR set/reset flip-flop



Po prawej przedstawiono przykład działania funkcji SR z wejściami aktywnymi. W tym przypadku dominująca jest funkcja R.

"Tag_1":P		%I0.0:P	Bool	TRUE	FALSE	
"Tag_4":P		%I0.1:P	Bool	TRUE	FALSE	
"Tag_3"		%Q0.0	Bool	FALSE	FALSE	

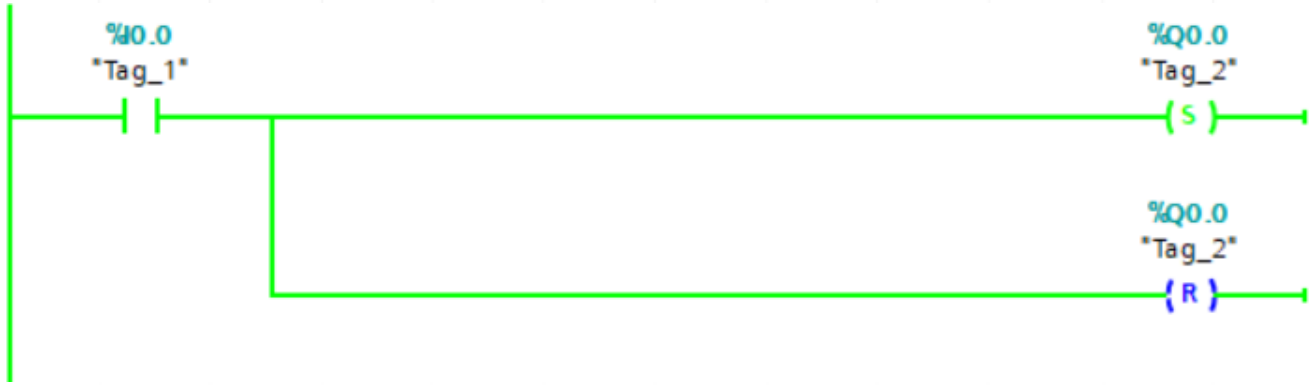
LAD – RS reset/set flip-flop



Po prawej przedstawiono przykład działania funkcji RS z wejściami aktywnymi. W tym przypadku dominująca jest funkcja S.

"Tag_6":P	%I0.2:P	Bool	TRUE		FALSE	
"Tag_7":P	%I0.3:P	Bool	TRUE		FALSE	
"Tag_8"	%Q0.2	Bool	TRUE		FALSE	

LAD – jednoczesny Set i Reset cewki



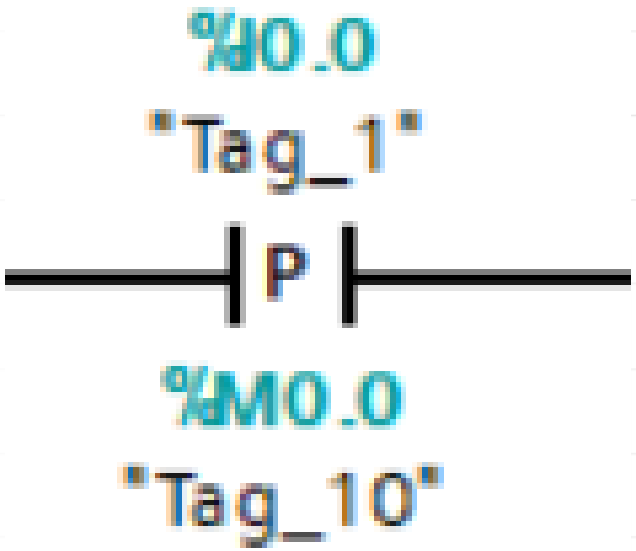
"Tag_1":P	%I0.0:P	Bool	TRUE
"Tag_2"	%Q0.0	Bool	FALSE

Po prawej przedstawiono przykład jednoczesnego Setowania i Resetowania wartości cewki przy aktywnym wejściu I0.0. Jeśli cewka o tym samym adresie jest użyta w więcej niż jednym miejscu programu, to podczas odświeżania fizycznych wyjść zostanie ustawiony stan odpowiadający stanowi ostatniej cewki.

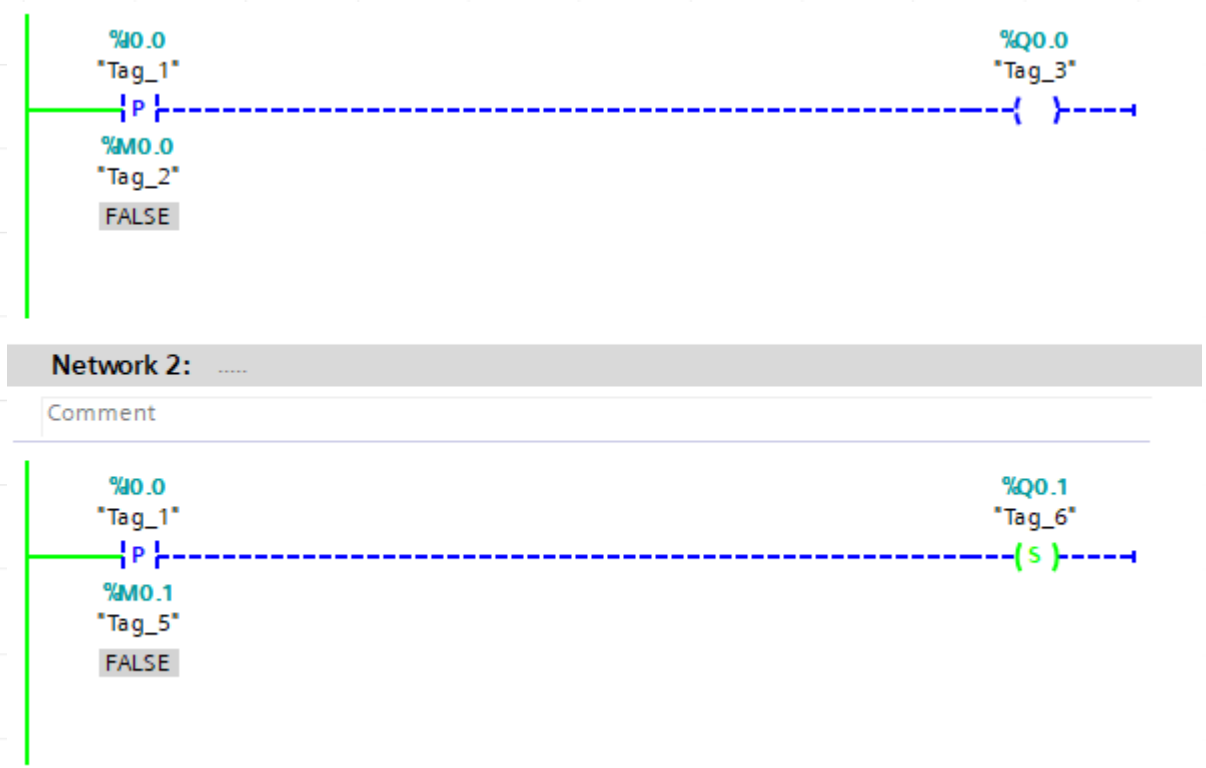
LAD – |P| scan operand for positive signal edge

Detekcja zbocza narastającego:

- Operand 1 (I0.0) sygnał jest porównywany z sygnałem znajdującym się w komórce pamięci (M0.0)
- Operand 2 (M0.0) poprzednia wartość operandu 1



LAD – |P| scan operand for positive signal edge

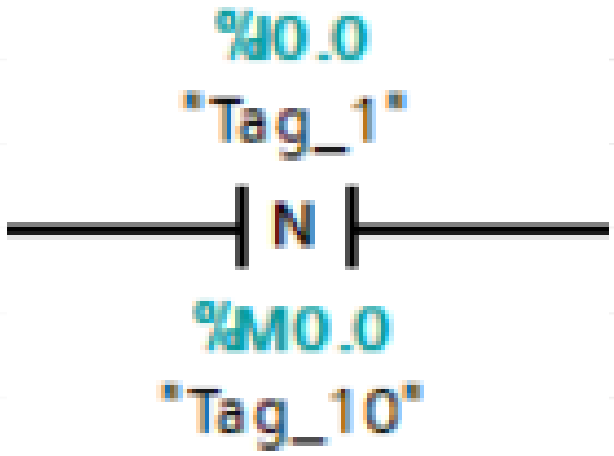


W momencie zmiany stanu na I0.0 z niskiego na wysoki przez jeden cykl programu pojawia się stan wysoki na wyjściu Q0.0.

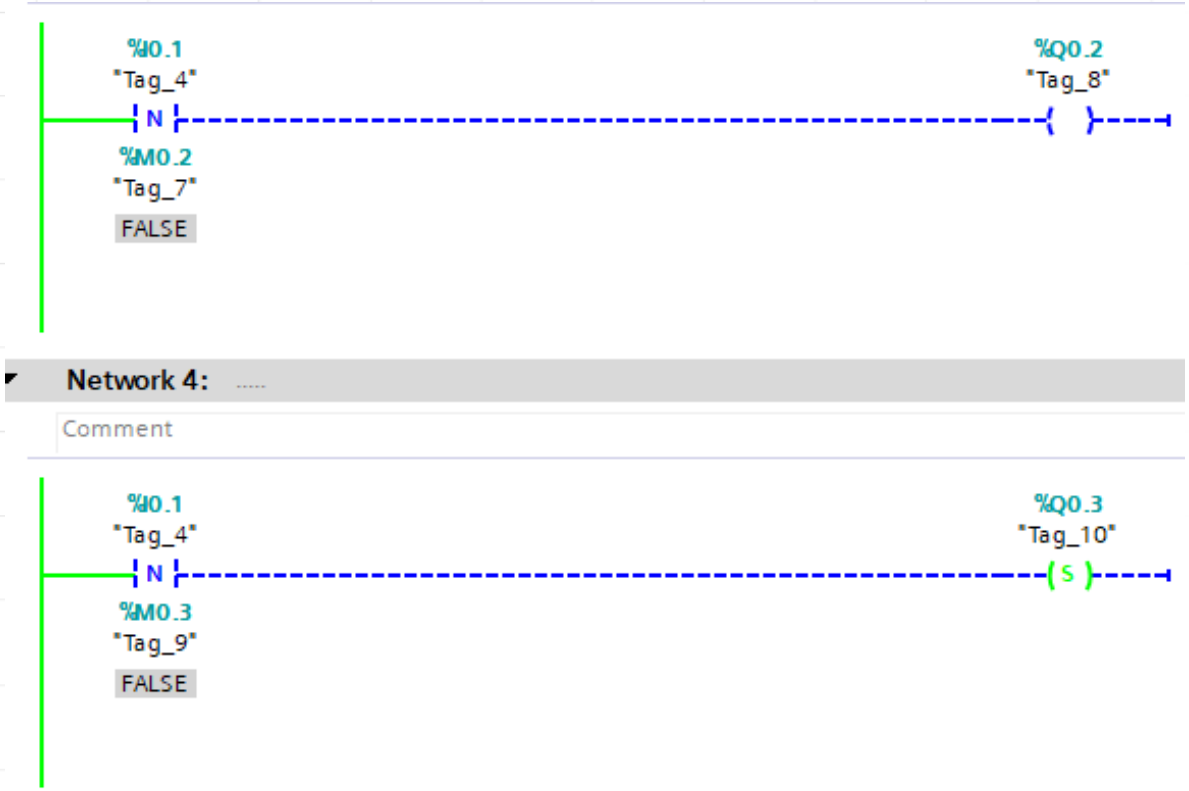
LAD – |N| scan operand for negative signal edge

Detekcja zbocza opadającego:

- Operand 1 (I0.0) sygnał jest porównywany z sygnałem znajdującym się w komórce pamięci (M0.0)
- Operand 2 (M0.0) poprzednia wartość operandu 1



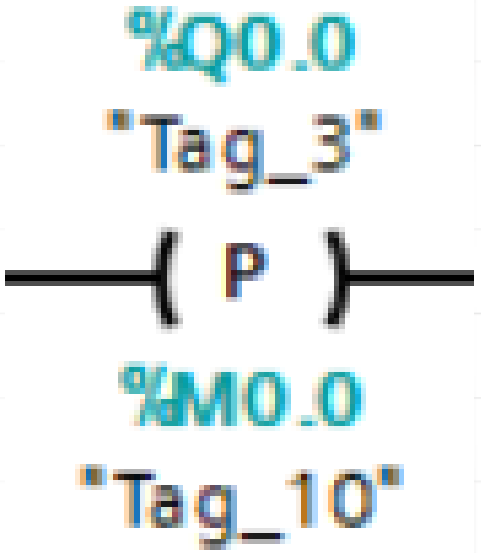
LAD – |N| scan operand for negative signal edge



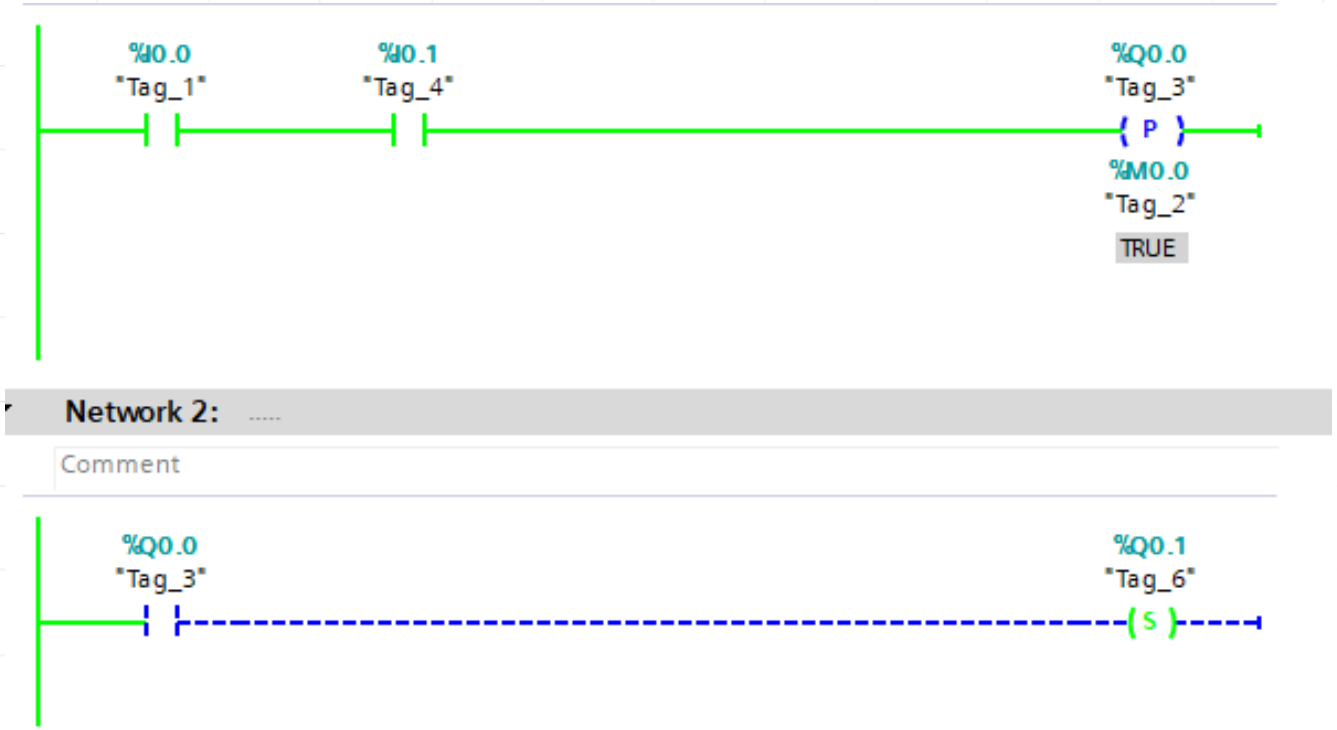
W momencie zmiany stanu na I0.1 z wysokiego na niski przez jeden cykl programu pojawia się stan wysoki na wyjściu Q0.2.

LAD – (P) set operand on positive signal edge

- Operand 1 (Q0.0) sygnał jest porównywany z sygnałem znajdującym się w komórce pamięci (M0.0)
- Operand 2 (M0.0) poprzednia wartość operandu 1



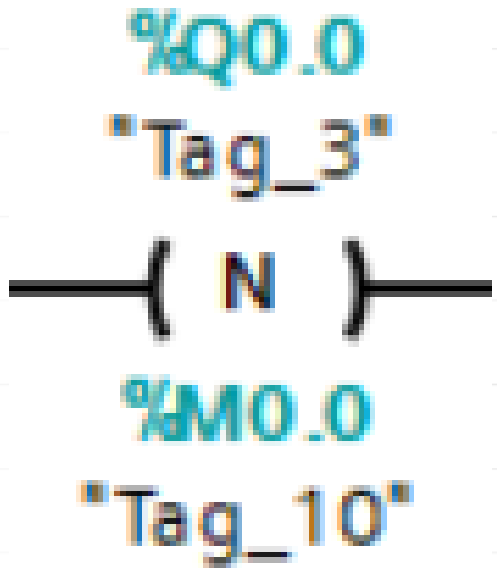
LAD – (P) set operand on positive signal edge



W momencie zmiany stanu na I0.0 i I0.1 z niskiego na wysoki przez jeden cykl programu pojawia się stan wysoki na wyjściu Q0.0.

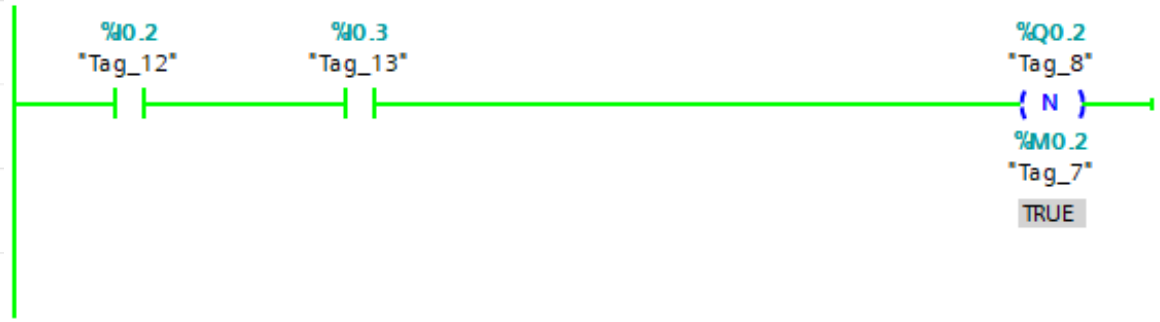
LAD – (N) set operand on negative signal edge

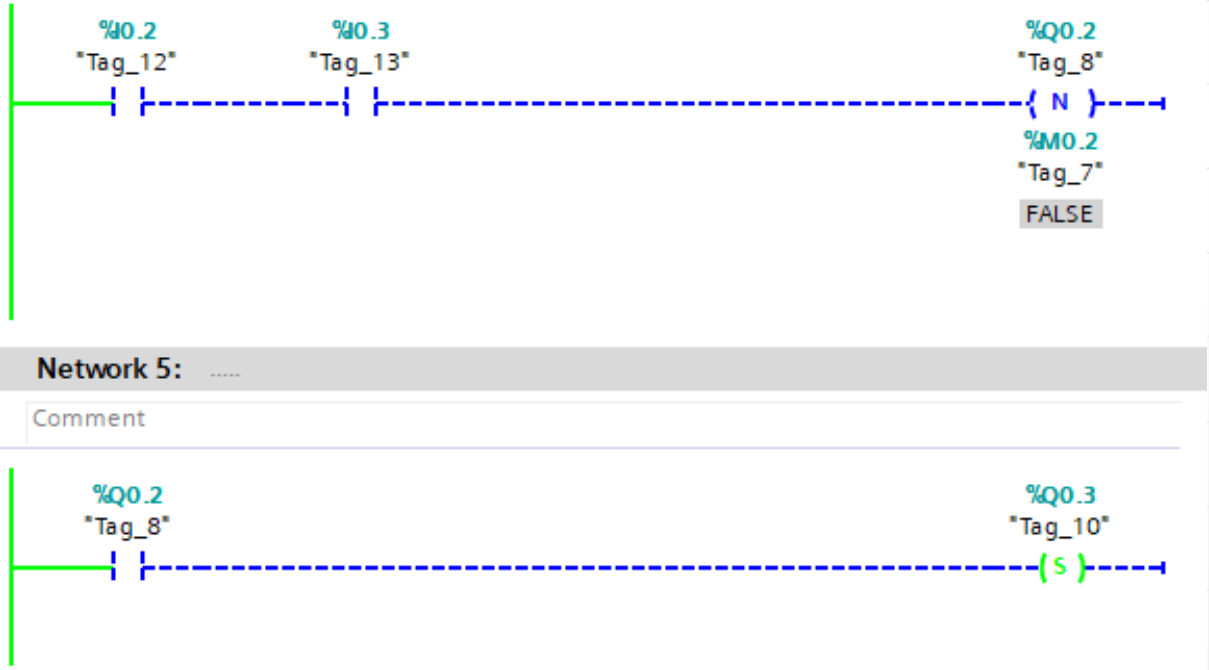
- Operand 1 (Q0.0) sygnał jest porównywany z sygnałem znajdującym się w komórce pamięci (M0.0)
- Operand 2 (M0.0) poprzednia wartość operandu 1



LAD – (N) set operand on negative signal edge

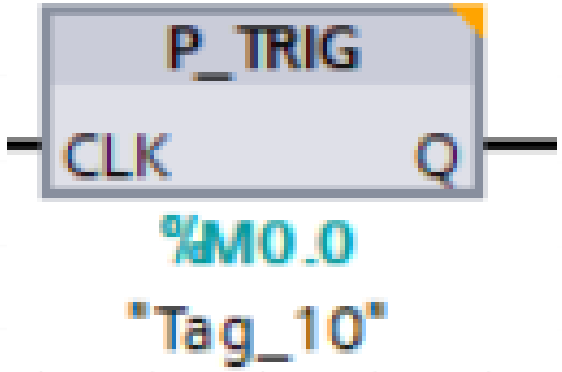
W momencie zmiany stanu na I0.2 i I0.3 z wysokiego na niski przez jeden cykl programu pojawia się stan wysoki na wyjściu Q0.2.



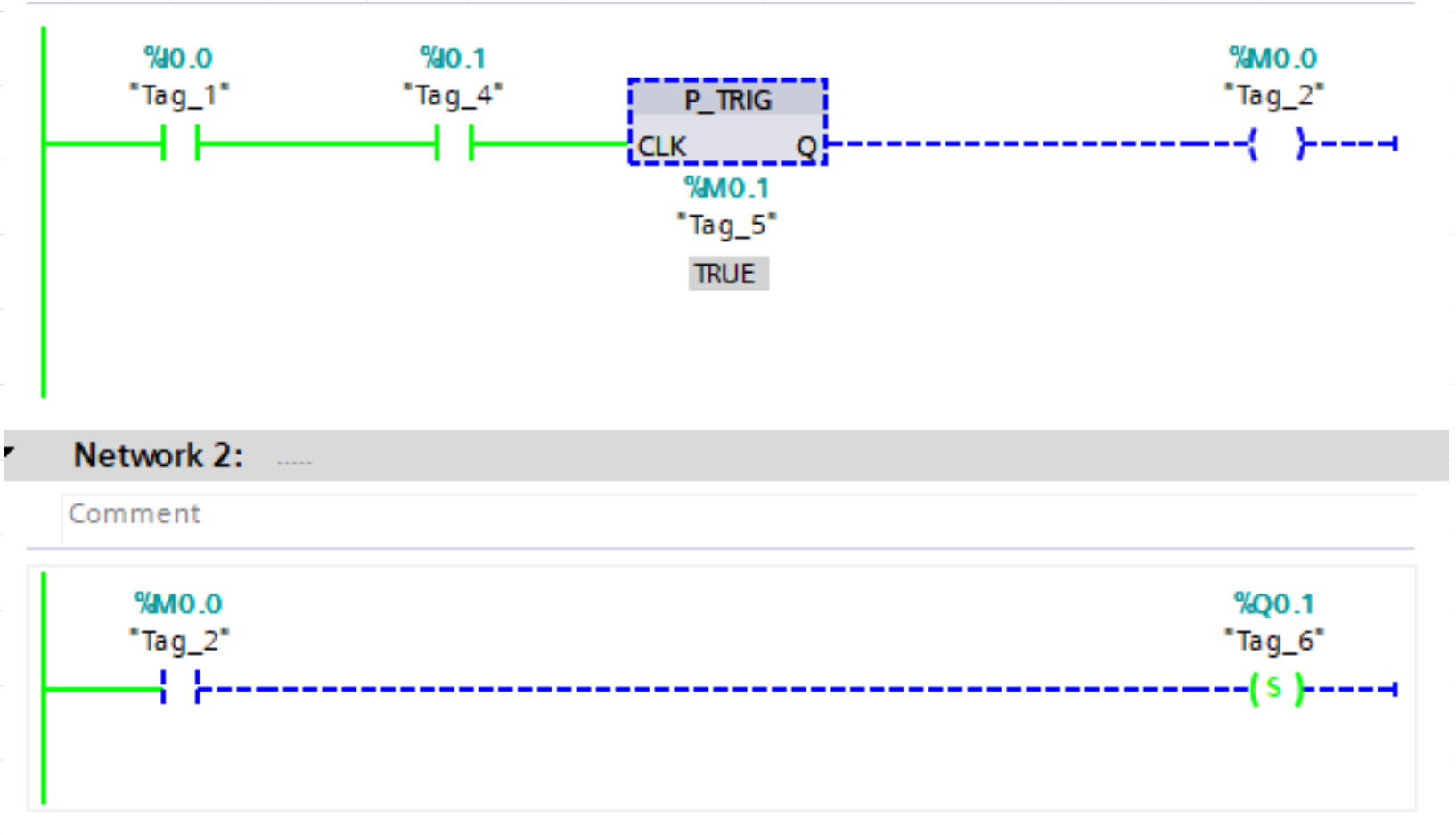


LAD – P_TRIG scan RLO for positive signal edge

- CLK – wejście elementu – podłączamy tutaj element typu prawda/fałsz,
- Mem (%M0.0) – poprzednia wartość na wejściu CLK,
- Q – wyjście elementu – krótki impuls „1” o czasie trwania jednego cyklu w przypadku wykrycia zbocza narastającego na wejściu CLK.



LAD – P_TRIG scan RLO for positive signal edge



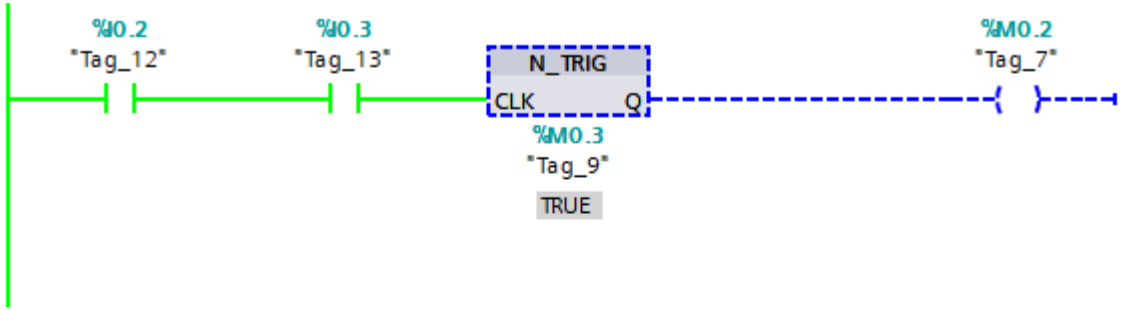
W momencie zmiany stanu na I0.0 i I0.1 z niskiego przez jeden cykl programu pojawia się stan wysoki na M0.0.

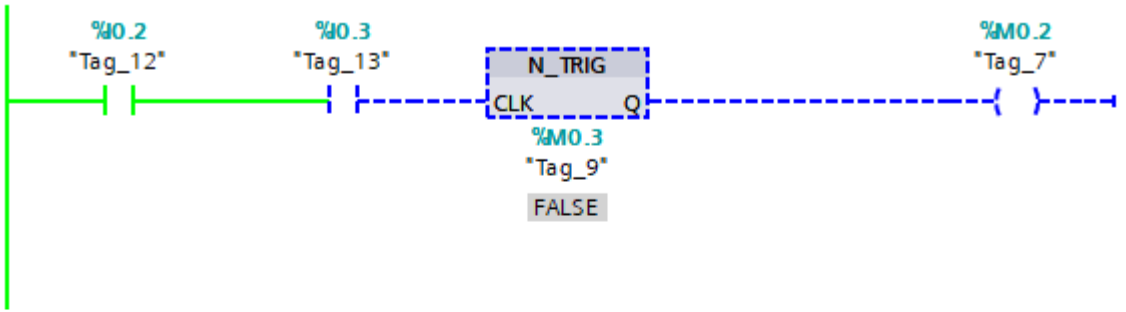
LAD – N_TRIG scan RLO for negative signal edge

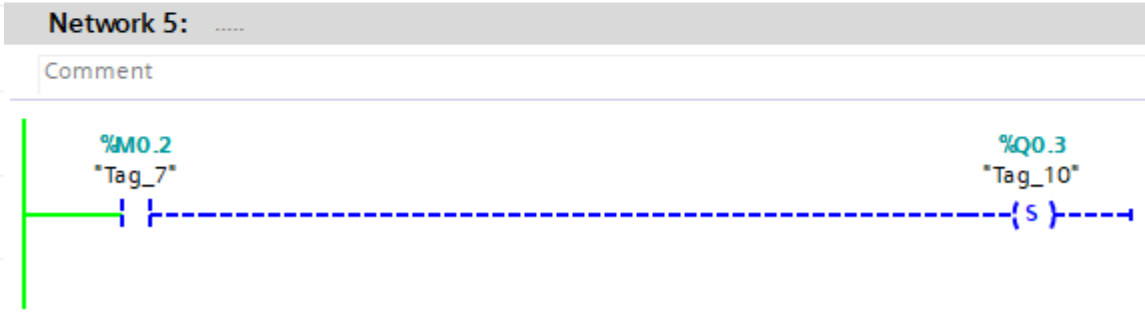
- CLK – wejście elementu – podłączamy tutaj element typu prawda/fałsz,
- Mem (%M0.0) – poprzednia wartość na wejściu CLK,
- Q – wyjście elementu – krótki impuls „1” o czasie trwania jednego cyklu w przypadku wykrycia zbocza opadającego na wejściu CLK.

LAD – N_TRIG scan RLO for negative signal edge

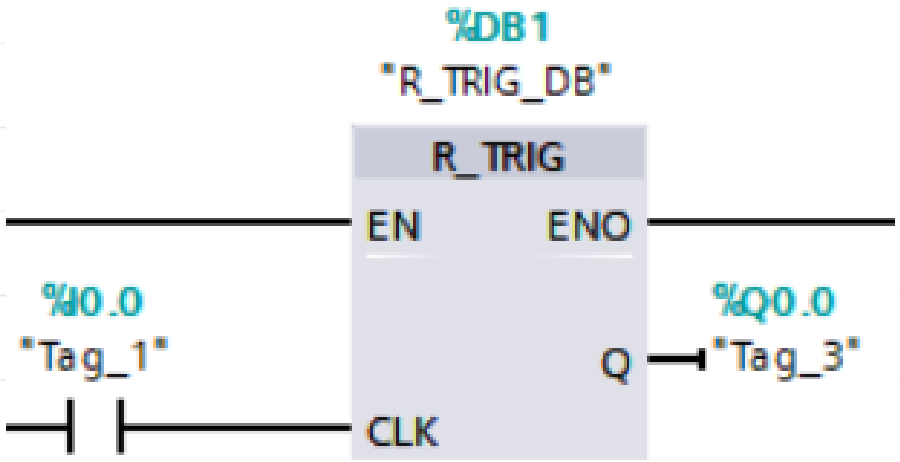
W momencie zmiany stanu na I0.2 i I0.3 z wysokiego na niski przez jeden cykl programu pojawia się stan wysoki na wyjściu M0.2.





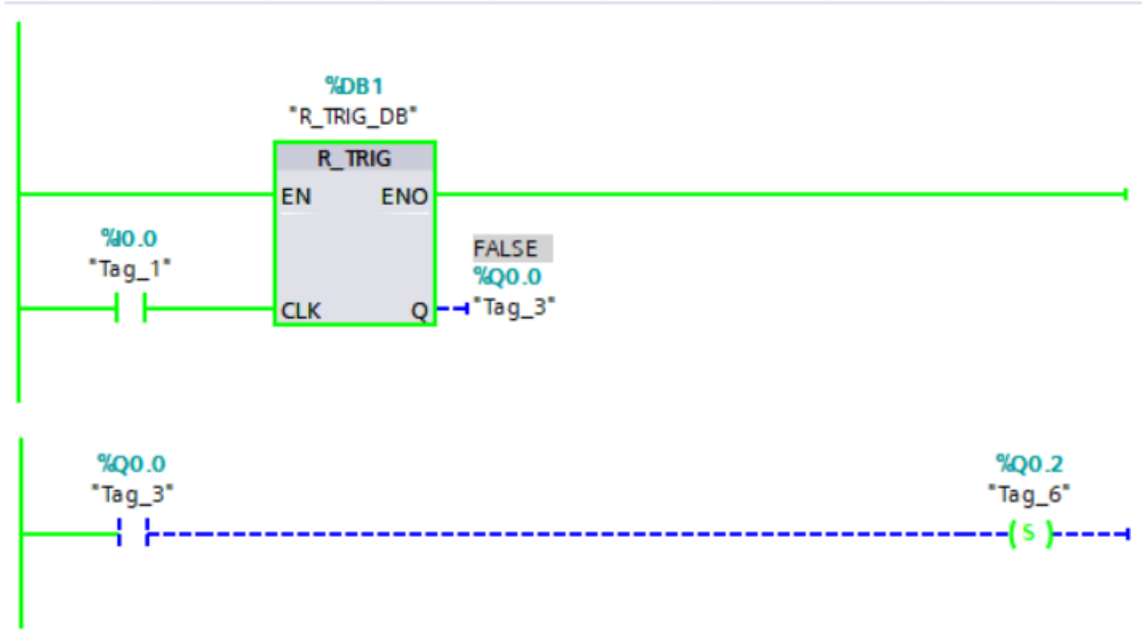


LAD – R_TRIG detect positive signal edge



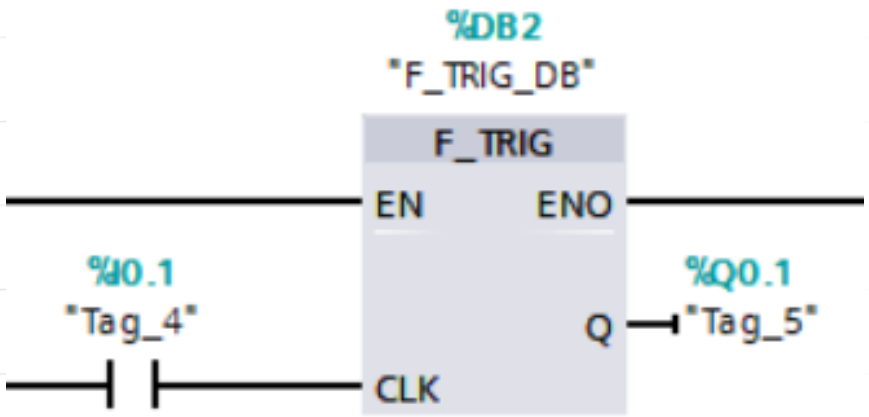
- CLK – wejście elementu – podłączamy tutaj element typu prawda/fałsz,
- Q – wyjście elementu – krótki impuls „1” o czasie trwania jednego cyklu w przypadku wykrycia zbocza narastającego na wejściu CLK.

LAD – R_TRIG detect positive signal edge



W momencie zmiany stanu na I0.0 z niskiego na wysoki przez jeden cykl programu pojawia się stan wysoki na Q0.0.

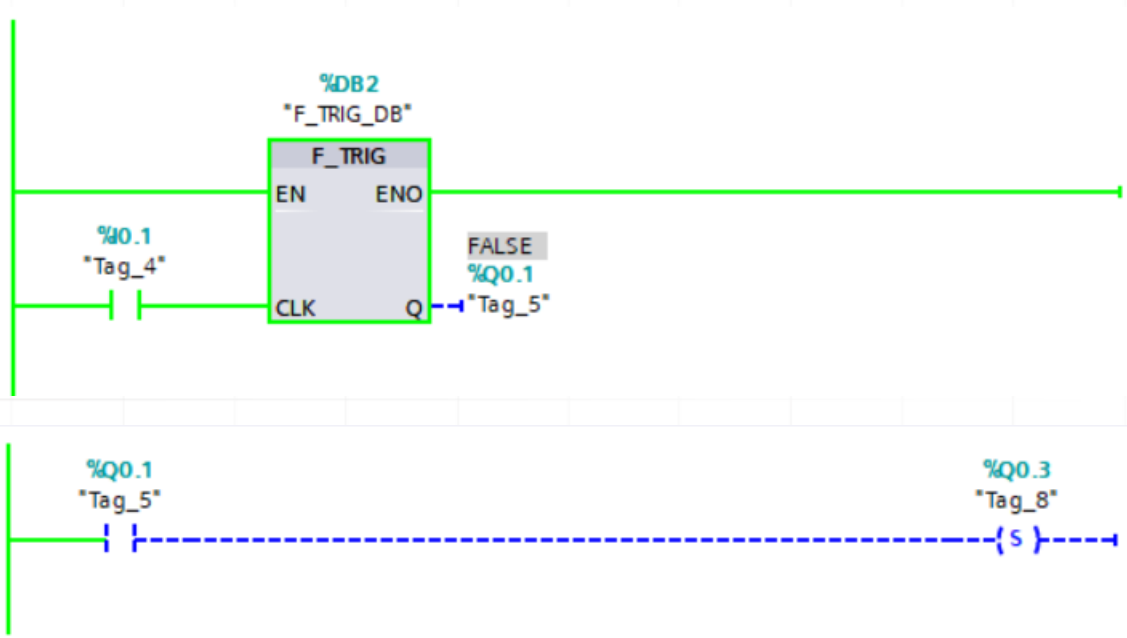
LAD – F_TRIG detect negative signal edge

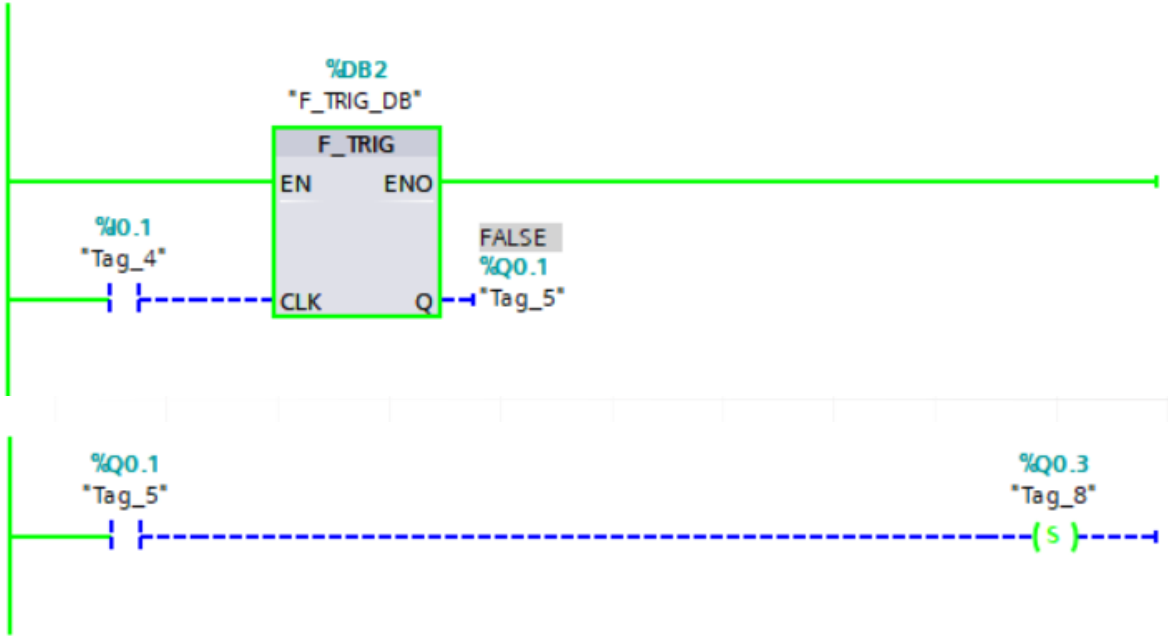


- CLK – wejście elementu – podłączamy tutaj element typu prawda/fałsz,
- Q – wyjście elementu – krótki impuls „1” o czasie trwania jednego cyklu w przypadku wykrycia zbocza opadającego na wejściu CLK.

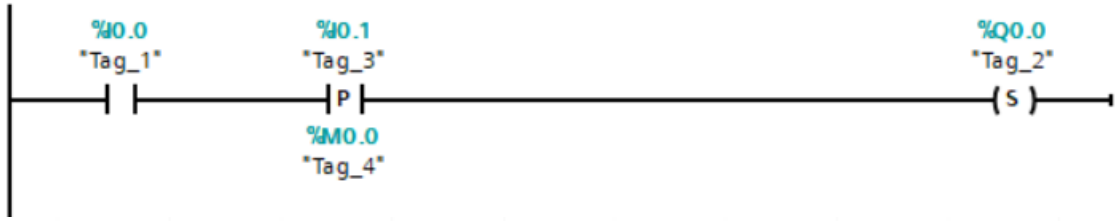
LAD – F_TRIG detect negative signal edge

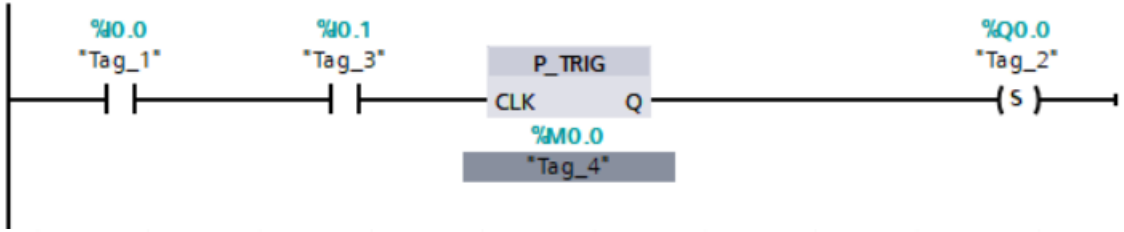
W momencie zmiany stanu na I0.1 z wysokiego na niski przez jeden cykl programu pojawia się stan wysoki na wyjściu Q0.1.





LAD – Detekcja zbocza porównanie





Na wyjściu Q0.0 zostanie ustawiony stan „1” gdy na wejściu I0.0 będzie stan „1” i na wejściu I0.1 pojawi się zbocze narastające.

Na wyjściu Q0.0 zostanie ustawiony stan „1” gdy iloczyn logiczny wejść I0.0 i I0.1 zmieni stan z „0” na „1”.

LAD – Timer operations

Timer operations	
TP	Generate pulse
TON	Generate on-delay
TOF	Generate off-delay
TONR	Time accumulator
–(TP)–	Start pulse timer
–(TON)–	Start on-delay timer
–(TOF)–	Start off-delay timer
–(TONR)–	Time accumulator
–(RT)–	Reset timer
–(PT)–	Load time duration

Typy danych czasowych

Timery			
Time	32 bity	≈T#-24d…≈T#24d	P:T#1 h_10 min
Ltime*	64 bity	≈LT#-106 751d…≈LT#106 751d	P:LT#5s_12ns

* Dostępne tylko w sterownikach Siemens serii S7-1500

Typy danych czasowych

Data i czas		
Date	2 bajty	D#1990-01-01…D#2169-06-06
Time_of_Day (TOD)	4 bajty	TOD#00:00:000 … TOD#23:59:59.999
LTime_of_Day (LTOD)*	8 bajtów	LTOD#00:00:00.000000000… LTOD#23:59:59.999999999 P: LTOD#10:20:30.400 365 215
Date_and_time (DT)*	8 bajtów	DT#1990-01-01-00:00:00.000… DT#2089-12-31-23:59:59.999 P: DT#2022-01-27-08:14:59.999
DTL	12 bajtów	DTL#1970-01-01-00:00:00.000000000… DTL#2262-04-11-23:47:16.854775807 P: DTL#2019-03-09-20:30:20.250

* Dostępne tylko w sterownikach Siemens serii S7-1500

Struktura DTL

Bajt	Składnik	Typ danych	Zakres
0	Year	UINT	Od 1970 do 2262
1			
2	Month	USINT	Od 1 do 12
3	Day	USINT	Od 1 do 31
4	Weekday	USINT	Od 1 (niedziela) do 7 (sobota)
5	Hour	USINT	Od 0 do 23
6	Minute	USINT	Od 0 do 59
7	Second	USINT	Od 0 do 59
8	Nanosecond	UDINT	Od 0 do 999 999 999
9			
10			
11			

LAD – Timery

S7-1200 obsługuje następujące timery:

- TP: Układ czasowy *Pulse timer* generuje impuls o ustalonym czasie trwania.
- TON: Układ czasowy *ON-delay timer* ustawia stan swojego wyjścia Q na ON (włączony) po upływie zadanego czasu opóźnienia.
- TOF: Układ czasowy *OFF-delay timer* kasuje stan swojego wyjścia Q na OFF (wyłączony) po upływie zadanego czasu opóźnienia.
- TONR: Układ czasowy *ON-delay Retentive timer* ustawia stan swojego wyjścia na ON (włączony) po upływie zadanego czasu opóźnienia. Upływający czas jest naliczany przez wiele okresów, aż do chwili, gdy zliczany upływ czasu zostanie wyzerowany za pomocą wejścia R.

LAD – Timery

Dla LAD i FBD timery te są dostępne jako blok instrukcji lub jako wyjściowa cewka. TIA Portal zapewnia także następujące cewki czasowe dla LAD:

- PT (*preset timer*) – cewka ładująca nowe nastawy czasu do określonego timera.
- RT (*reset timer*) – cewka kasująca określony timer.

Liczba timerów użytych w programie użytkownika jest ograniczona tylko rozmiarem pamięci CPU. Każdy timer używa 16 bajtów pamięci.
Każdy timer używa struktury przechowywanej w bloku danych. Dla FBD tworzone są automatycznie DB, gdy tylko tworzona jest instrukcja.

LAD – Timery

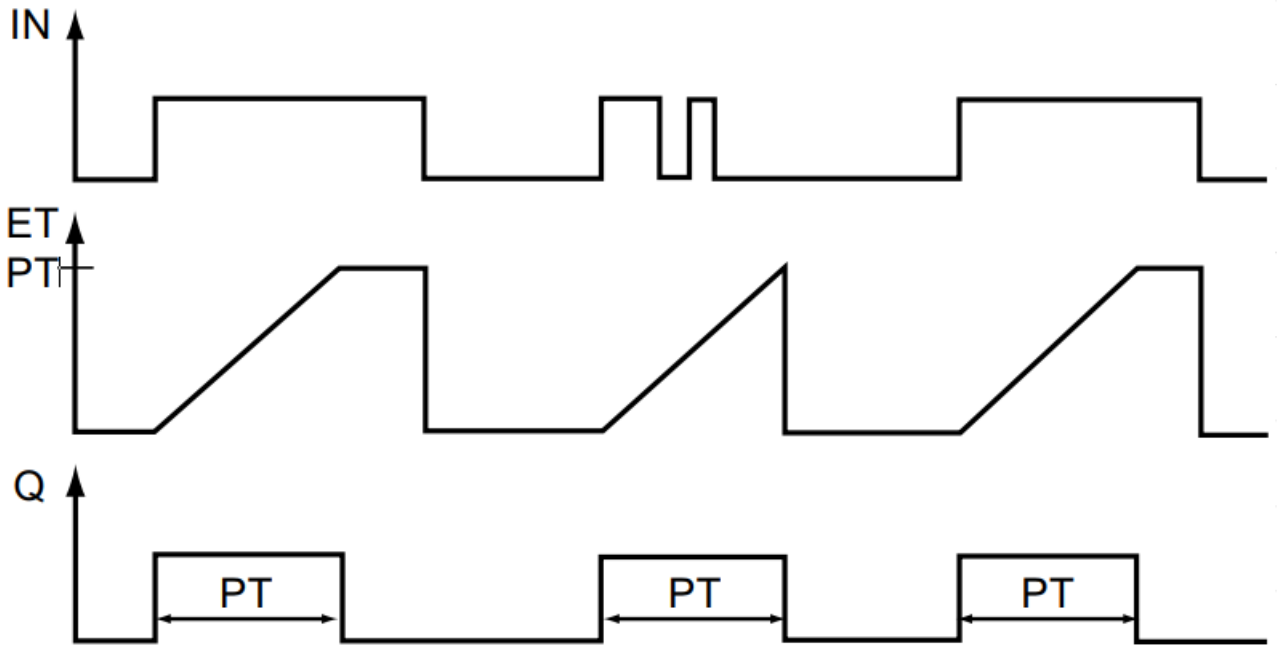
Kiedy instrukcja dotycząca timera zostaje umieszczona w bloku funkcji, to można wybrać opcję wielokrotnej instancji bloku danych. Ponieważ dane timera są zawarte w jednym bloku danych i nie są wymagane oddzielne bloki danych dla każdego timera, to redukuje czas przetwarzania. Nie występuje żadna interakcja pomiędzy strukturami danych timerów we współdzielonych wielokrotnych instancjach bloków danych.

Podczas deklarowania timerów, spotykamy się z następującymi zmiennymi:

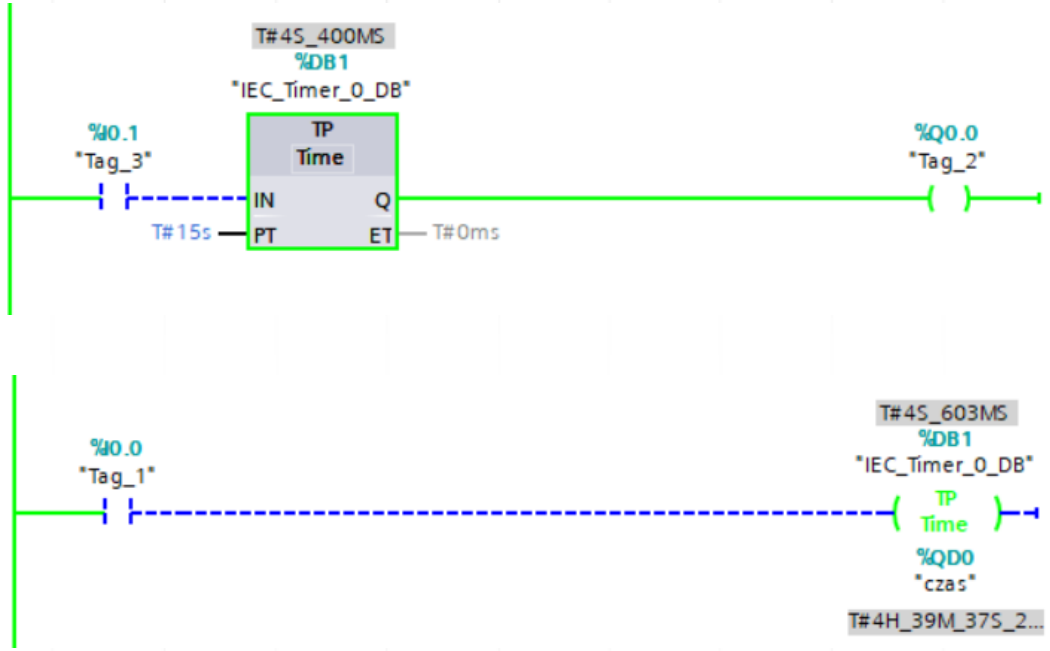
- **IN** – wejście timera (zezwolenie na odliczanie czasu)
- **PT** – czas trwania opóźnienia (zmienna typu TIME)
- **ET** – czas, który upłynął (zmienna typu TIME)
- **Q** – wyjście Timera
- **R** – kasowanie czasu ET

LAD – Timer TP

TP: Układ czasowy *Pulse timer* generuje impuls o ustalonym czasie trwania



LAD – Start pulse timer

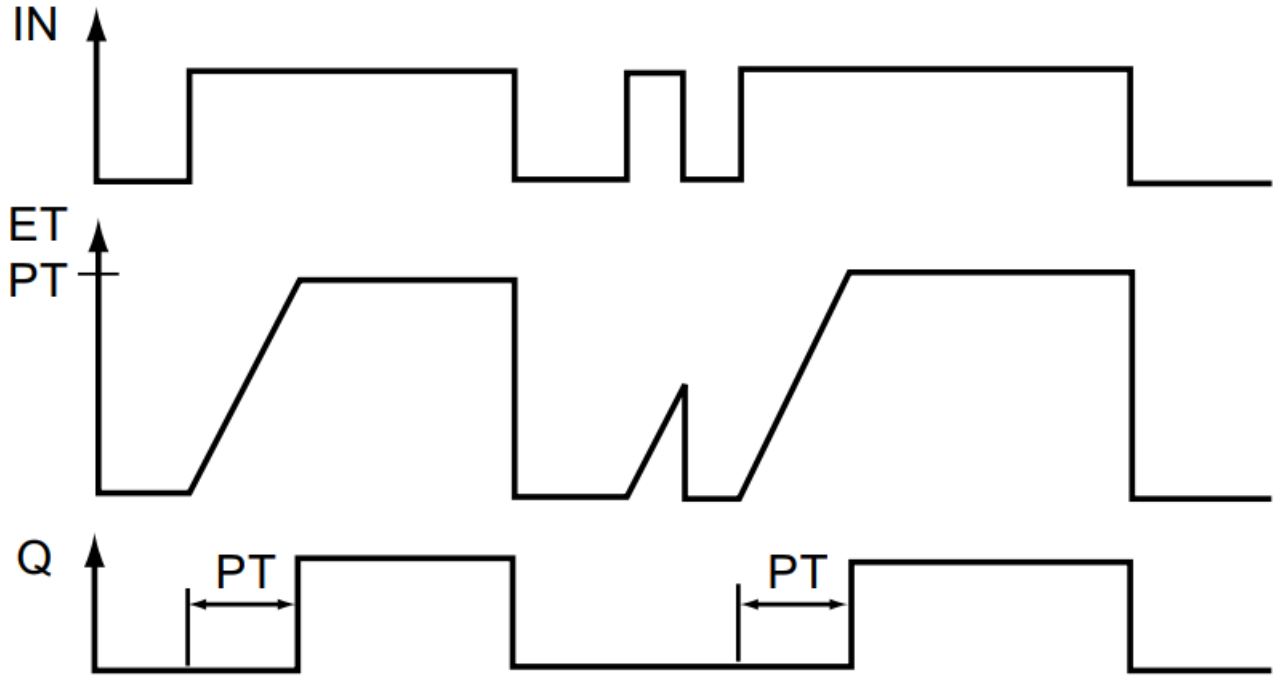


Start pulse timer pozwala na uruchomienie timera IEC o zadanym czasie trwania.

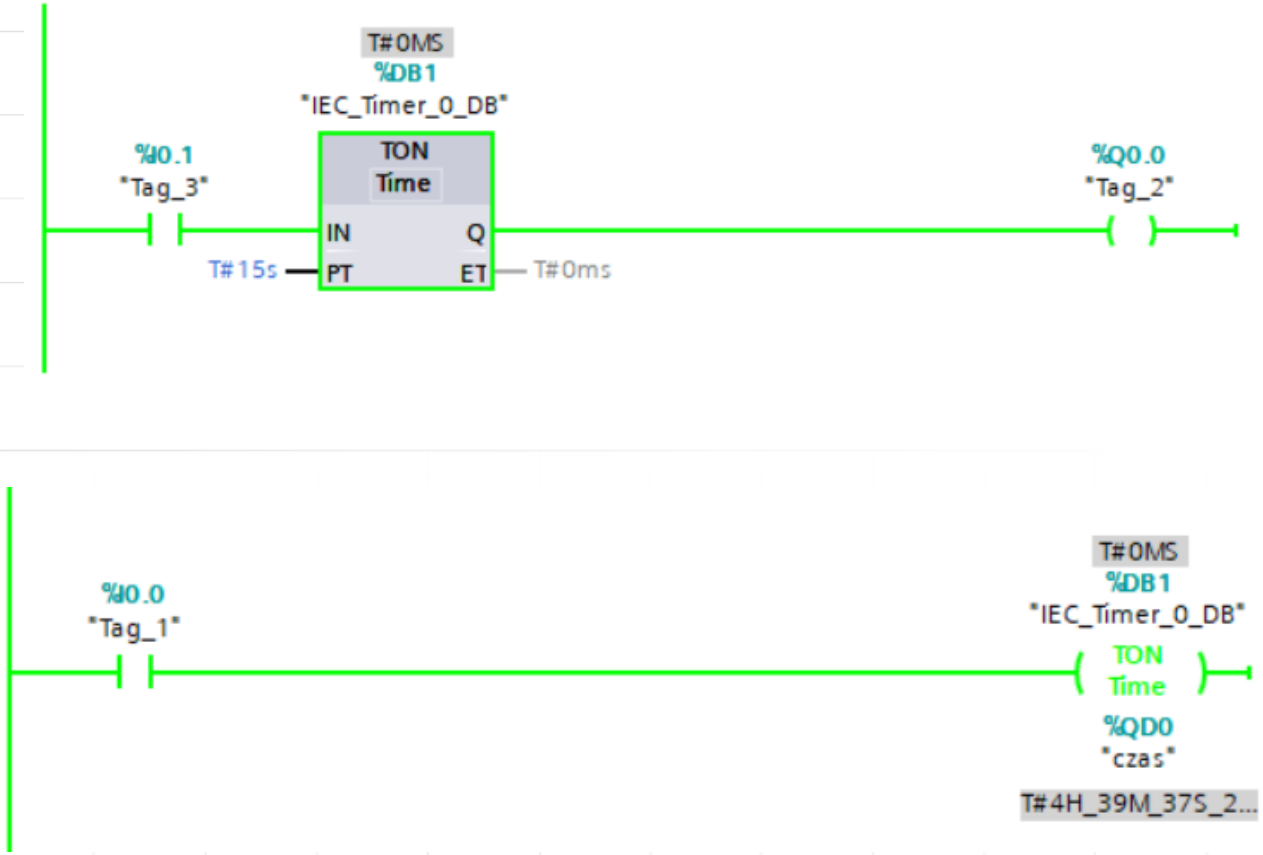
	Name	Address	Display format	Monitor/Modify value
	"IEC_Timer_0_DB".PT		Time	T#15S
	"IEC_Timer_0_DB".ET		Time	T#3S_32MS
	"IEC_Timer_0_DB".IN		Bool	FALSE
	"IEC_Timer_0_DB".Q		Bool	TRUE
	"Tag_1":P	%I0.0:P	Bool	FALSE
	"Tag_3":P	%I0.1:P	Bool	FALSE
	"Tag_2"	%Q0.0	Bool	TRUE
	"czas"	%QD0	Time	T#4H_39M_37S_216MS

LAD – Timer TON

TON: Układ czasowy *ON-delay timer* ustawia stan swojego wyjścia Q na ON (włączony) po upływie zadanego czasu opóźnienia.



LAD – Start on-delay timer

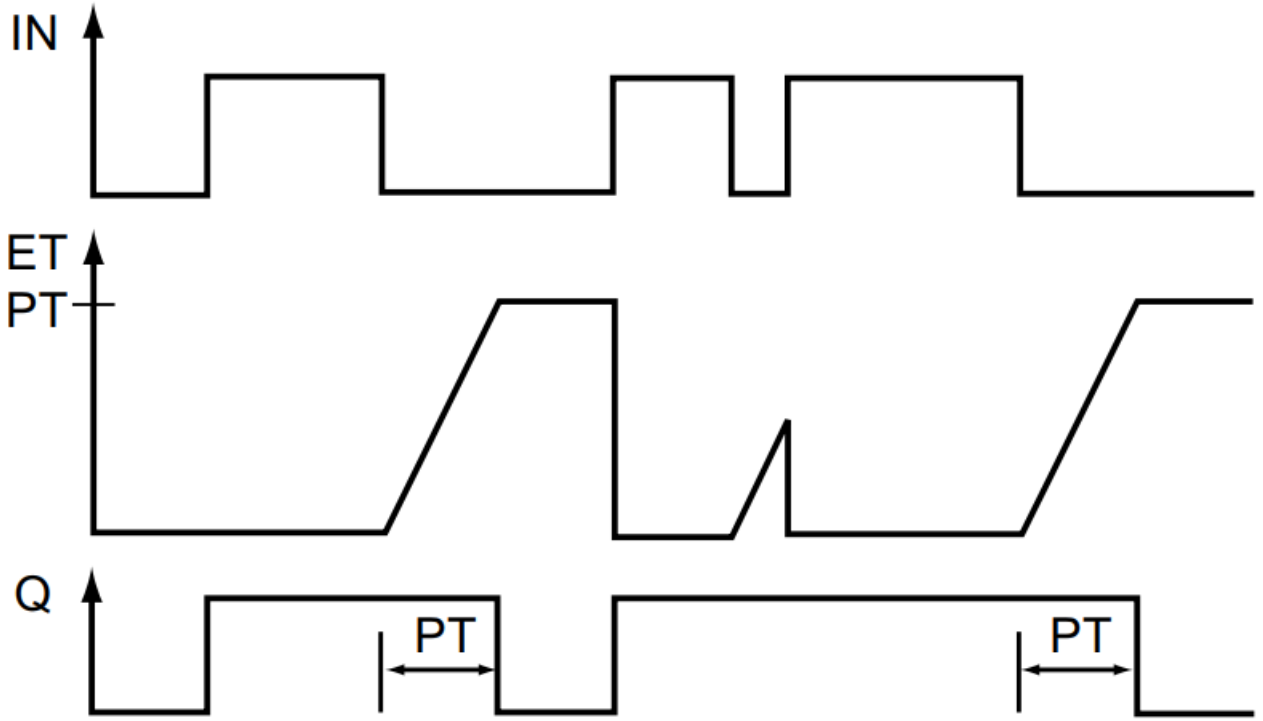


Instrukcja uruchamia timer IEC. Timer jest uruchamiany gdy na stykach I0.0 i I0.1 pojawi się stan wysoki. Cewka Q0.0 przechodzi w stan wysoki po upływie określonego czasu.

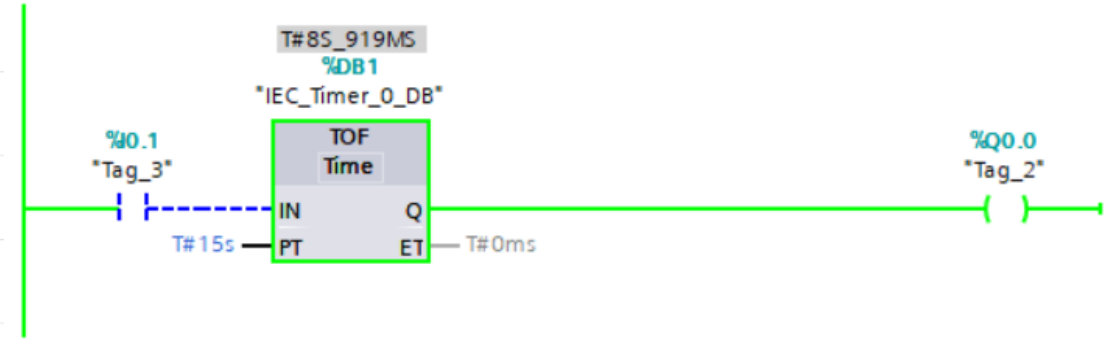
"IEC_Timer_0_DB".PT		Time	T#0MS
"IEC_Timer_0_DB".ET		Time	T#0MS
"IEC_Timer_0_DB".IN		Bool	TRUE
"IEC_Timer_0_DB".Q		Bool	TRUE
"Tag_1":P	%I0.0:P	Bool	TRUE
"Tag_3":P	%I0.1:P	Bool	TRUE
"Tag_2"	%Q0.0	Bool	TRUE
"czas"	%QD0	Time	T#4H_39M_37S_216MS

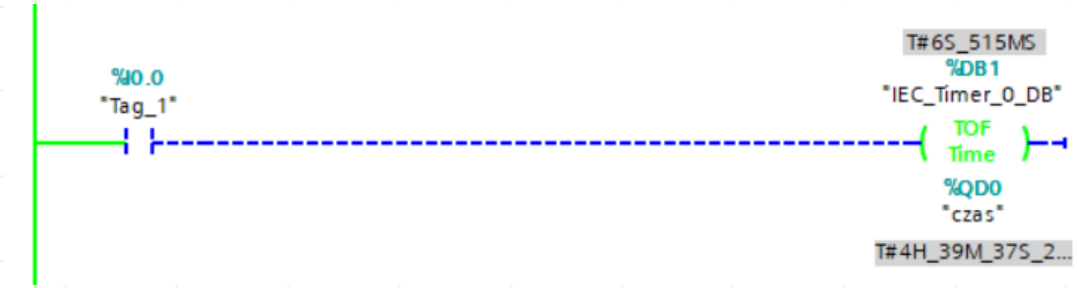
LAD – Timer TOF

TOF: Układ czasowy *OFF-delay timer* kasuje stan swojego wyjścia Q na OFF (wyłączony) po upływie zadanego czasu opóźnienia.



LAD – Start off-delay timer



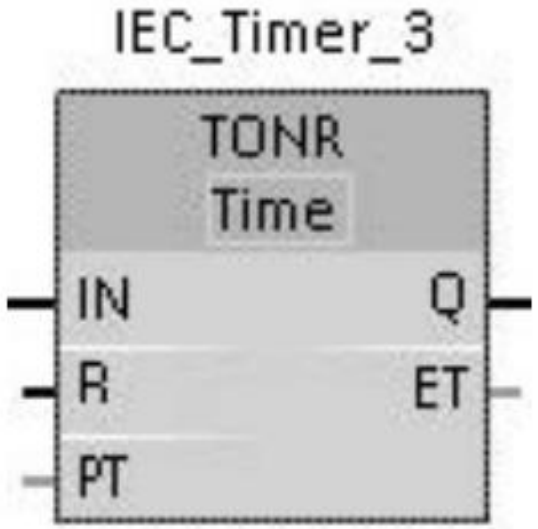


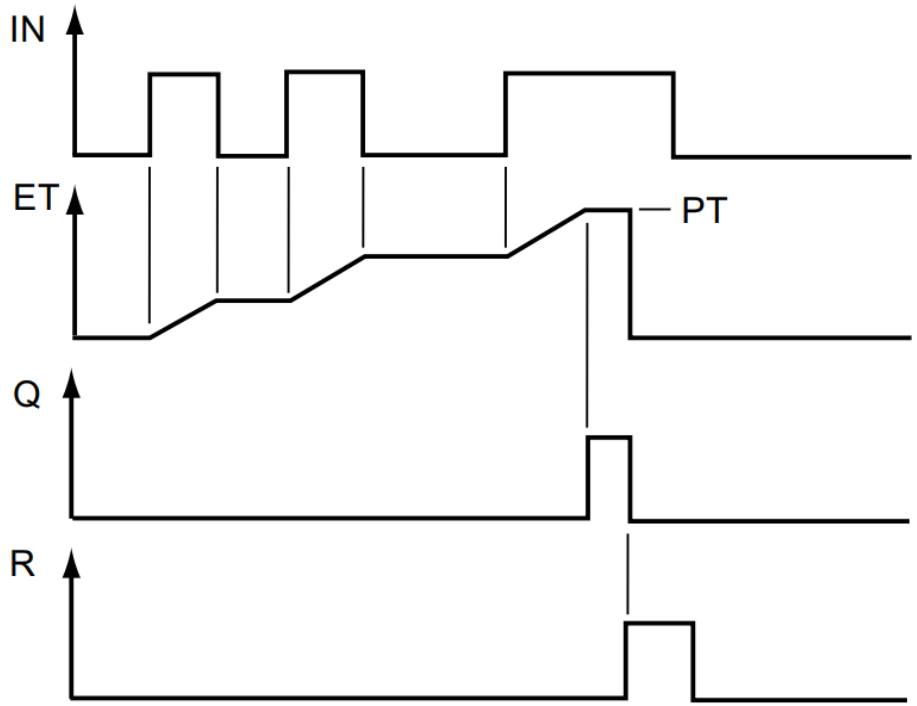
Zmiana sygnału I0.0 z wysokiego na niski powoduje uruchomienie Timera, który podtrzymuje stan wysoki na wyjściu Q0.0 przez zadany czas.

	"IEC_Timer_0_DB".PT		Time	T#15S
	"IEC_Timer_0_DB".ET		Time	T#3S_446MS
	"IEC_Timer_0_DB".IN		Bool	FALSE
	"IEC_Timer_0_DB".Q		Bool	TRUE
	"Tag_1":P	%I0.0:P	Bool	FALSE
	"Tag_3":P	%I0.1:P	Bool	FALSE
	"Tag_2"	%Q0.0	Bool	TRUE
	"czas"	%QD0	Time	T#4H_39M_37S_216MS

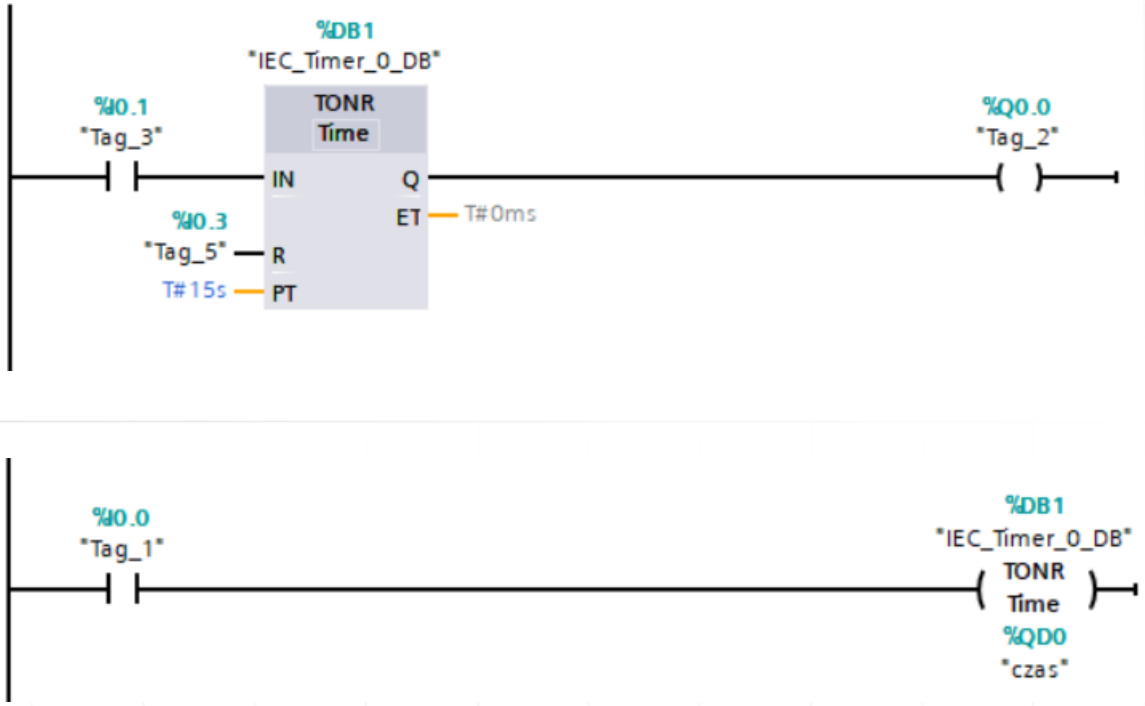
LAD – Timer TONR

TONR: Układ czasowy *ON-delay Retentive timer* ustawia stan swojego wyjścia na ON (włączony) po upływie zadanego czasu opóźnienia. Upływający czas jest naliczany przez wiele okresów, aż do chwili, gdy zliczany upływ czasu zostanie wyzerowany za pomocą wejścia R.





LAD – Time Accumulator



Przejście wejścia I0.0 w stan wysoki powoduje start Timera. Przejście wejścia w stan niski powoduje jego zatrzymanie, a ponowne przejście w stan wysoki – kontynuację liczenia czasu. Wyjście Q0.0 przechodzi w stan wysoki, jeżeli stan wysoki był na wejściu I0.0 przez zadany czas (sumarycznie), chyba, że Timer został wcześniej zresetowany wejściem I0.3.

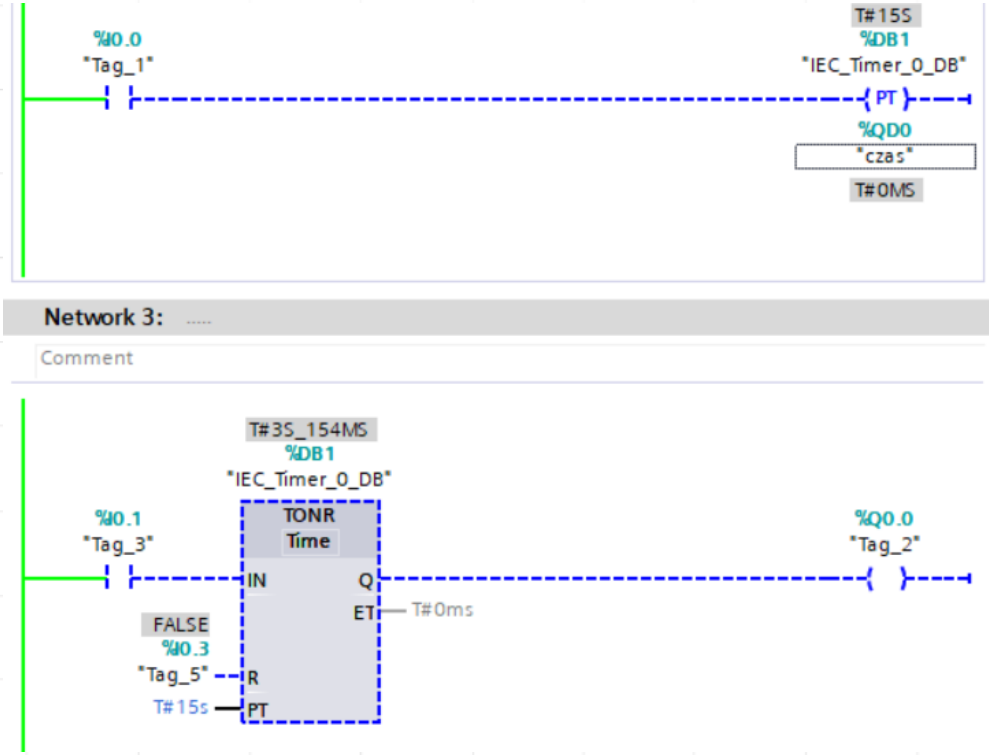
LAD – Timery PT i RT

Instrukcje-PT oraz RT jako cewki lub bloki. Cewki instrukcji mogą zostać wstawione w pozycji środkowej. Stan zasilania wyjścia cewki jest zawsze taki sam jak stan zasilania jej wejścia.

- Kiedy cewka -(PT)- jest aktywna, wartość elementu PRESET time w określonym polu IEC_Timer bloku DB jest ustawiana na czas trwania PRESET_Tag.
- Kiedy cewka RT jest aktywna, wartość elementu Elapsed time w określonym polu IEC_Timer bloku DB jest ustawiana na 0

LAD – Timer Preset Time

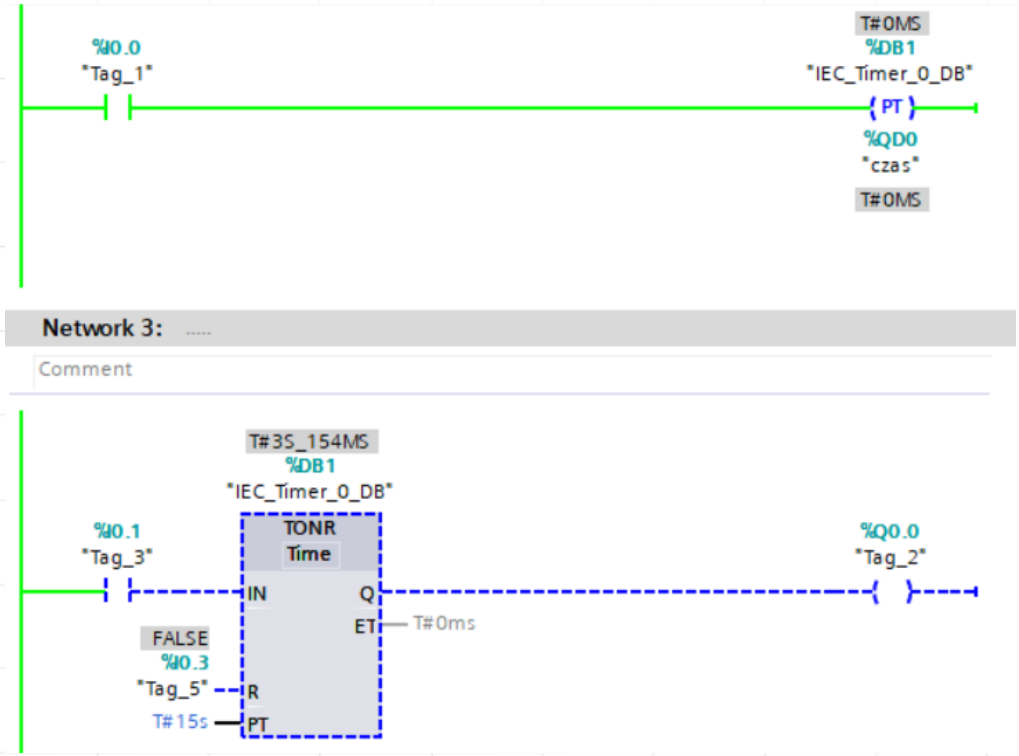
- Kiedy cewka -(PT)- jest aktywna, wartość elementu PRESET time w określonym polu IEC_Timer bloku DB jest ustawiana na czas trwania PRESET_Tag.



"IEC_Timer_0_DB".PT		Time		T#15S
"IEC_Timer_0_DB".ET		Time		T#3S_381MS
"IEC_Timer_0_DB".IN		Bool		FALSE
"IEC_Timer_0_DB".Q		Bool		FALSE
"Tag_1":P	%I0.0:P	Bool		FALSE
"Tag_3":P	%I0.1:P	Bool		FALSE
"Tag_5":P	%I0.3:P	Bool		FALSE
"Tag_2"	%Q0.0	Bool		FALSE
"czas"	%QD0	Time		T#0MS
"Tag_4"	%M0.0	Bool		FALSE

LAD – Timer Preset Time

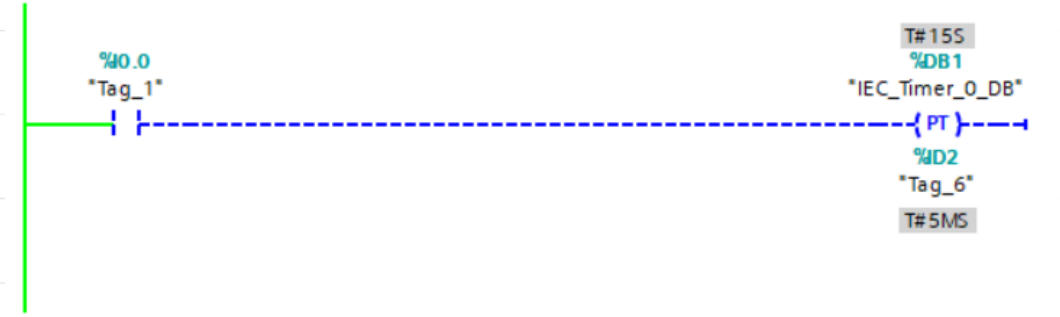
- Kiedy cewka -(PT)- jest aktywna, wartość elementu PRESET time w określonym polu IEC_Timer bloku DB jest ustawiana na czas trwania PRESET_Tag.

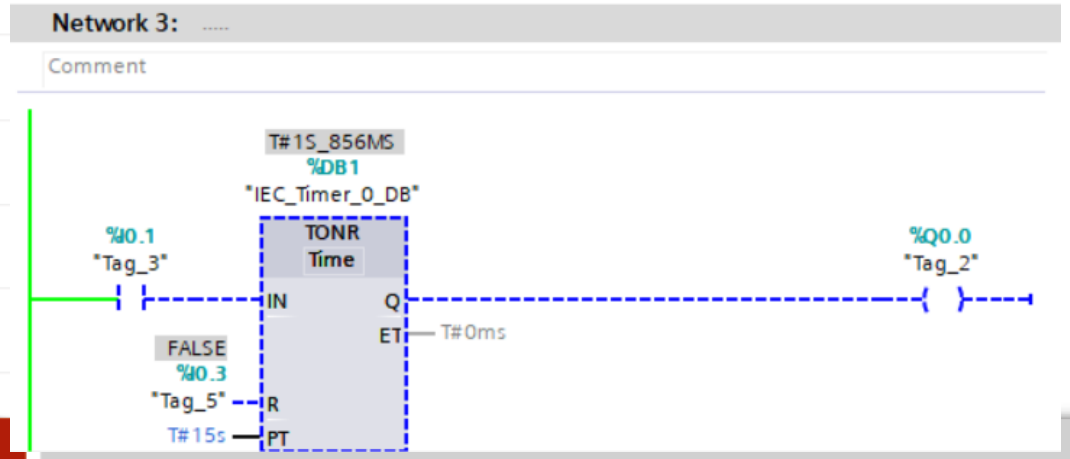


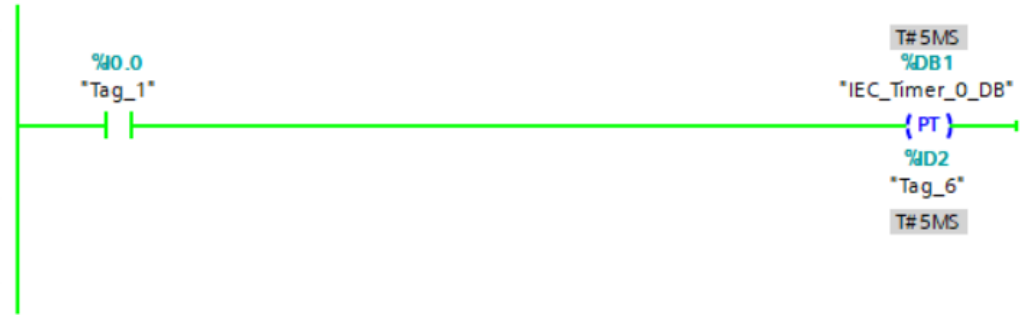
"IEC_Timer_0_DB".PT		Time	T#0MS
"IEC_Timer_0_DB".ET		Time	T#3S_154MS
"IEC_Timer_0_DB".IN		Bool	FALSE
"IEC_Timer_0_DB".Q		Bool	FALSE
"Tag_1":P	%I0.0:P	Bool	TRUE
"Tag_3":P	%I0.1:P	Bool	FALSE
"Tag_5":P	%I0.3:P	Bool	FALSE
"Tag_2"	%Q0.0	Bool	FALSE
"czas"	%QD0	Time	T#0MS
"Tag_4"	%M0.0	Bool	FALSE

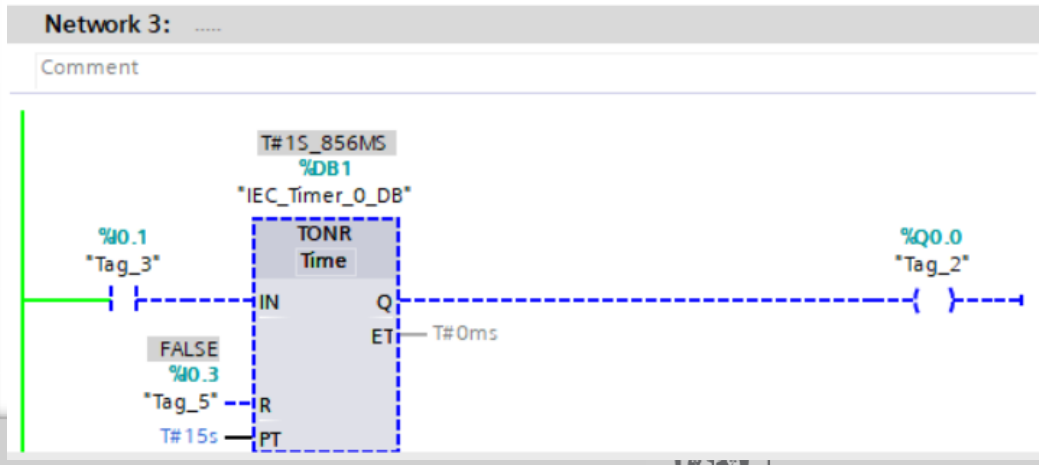
LAD – Timer Preset Time

- Kiedy cewka -(PT)- jest aktywna, wartość elementu PRESET time w określonym polu IEC_Timer bloku DB jest ustawiana na czas trwania PRESET_Tag.



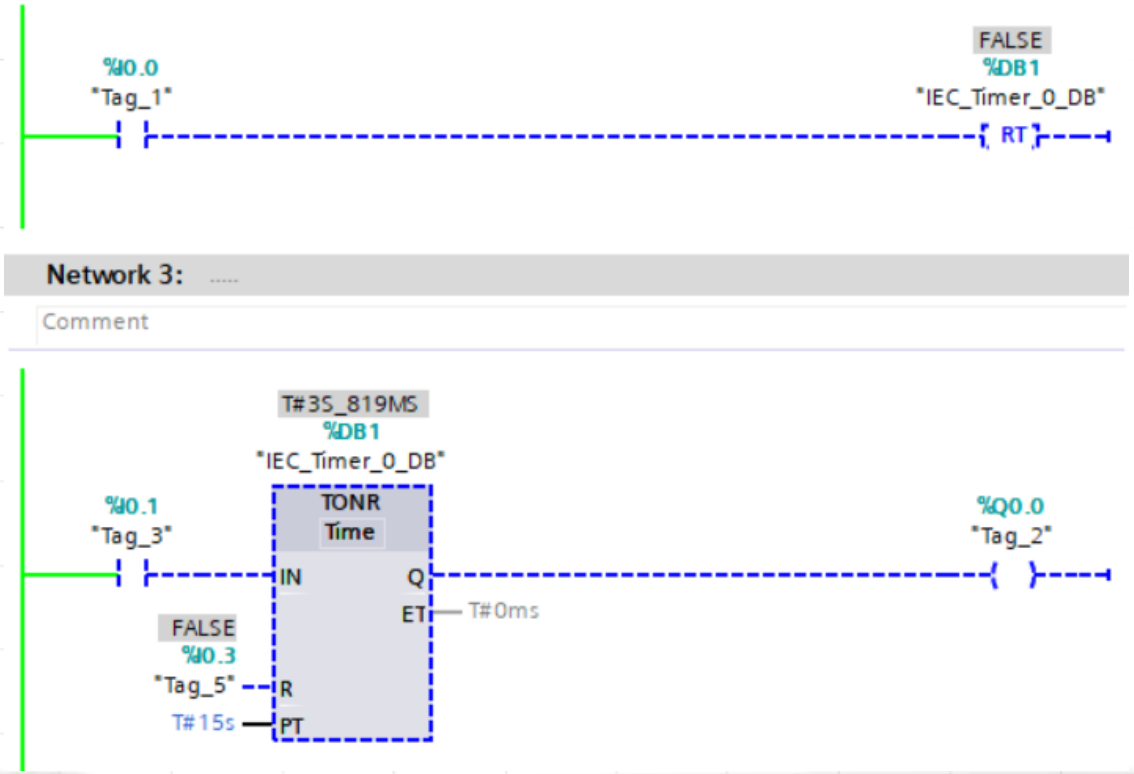






LAD – Timer Reset Time

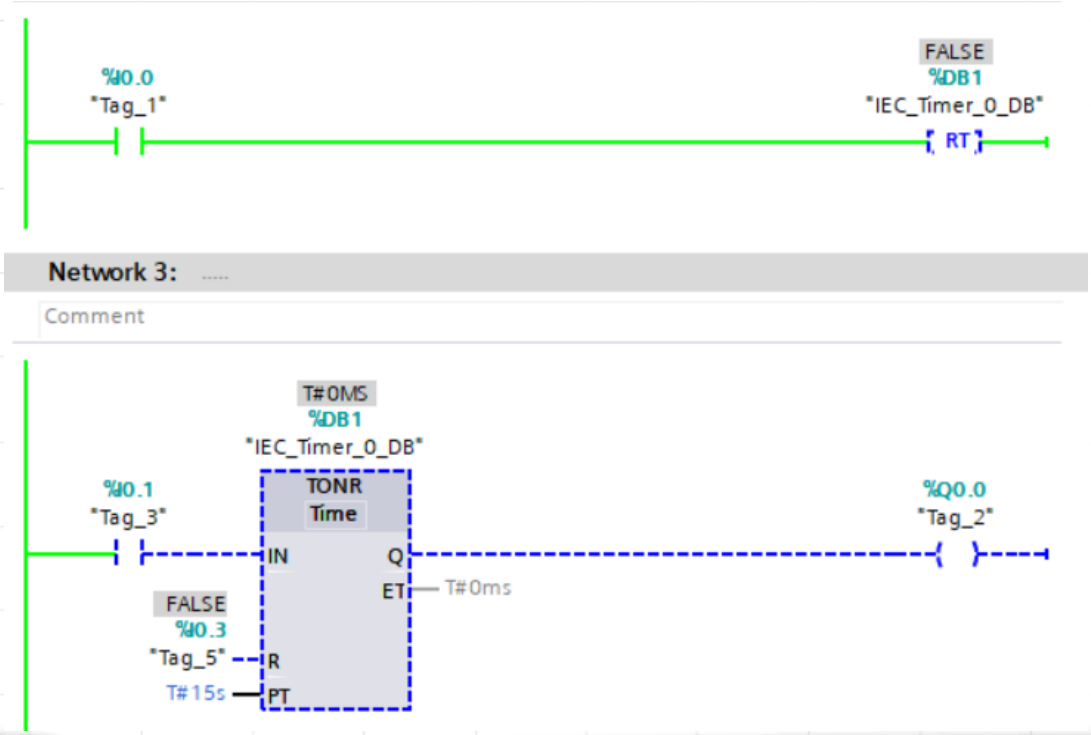
- Kiedy cewka [RT] jest aktywna, wartość elementu Elapsed time w określonym polu IEC_Timer bloku DB jest ustawiana na 0



"IEC_Timer_0_DB".PT		Time	T#15S
"IEC_Timer_0_DB".ET		Time	T#3S_819MS
"IEC_Timer_0_DB".IN		Bool	FALSE
"IEC_Timer_0_DB".Q		Bool	FALSE
"Tag_1":P	%I0.0:P	Bool	FALSE
"Tag_3":P	%I0.1:P	Bool	FALSE
"Tag_5":P	%I0.3:P	Bool	FALSE
"Tag_2"	%Q0.0	Bool	FALSE
"czas"	%QD0	Time	T#0MS
"Tag_4"	%M0.0	Bool	FALSE

LAD – Timer Reset Time

- Kiedy cewka RT jest aktywna, wartość elementu Elapsed time w określonym polu IEC_Timer bloku DB jest ustawiana na 0



"IEC_Timer_0_DB".PT		Time	T#0MS
"IEC_Timer_0_DB".ET		Time	T#0MS
"IEC_Timer_0_DB".IN		Bool	FALSE
"IEC_Timer_0_DB".Q		Bool	FALSE
"Tag_1":P	%I0.0:P	Bool	TRUE
"Tag_3":P	%I0.1:P	Bool	FALSE
"Tag_5":P	%I0.3:P	Bool	FALSE
"Tag_2"	%Q0.0	Bool	FALSE
"czas"	%QD0	Time	T#0MS
"Tag_4"	%M0.0	Bool	FALSE

LAD – Liczniki

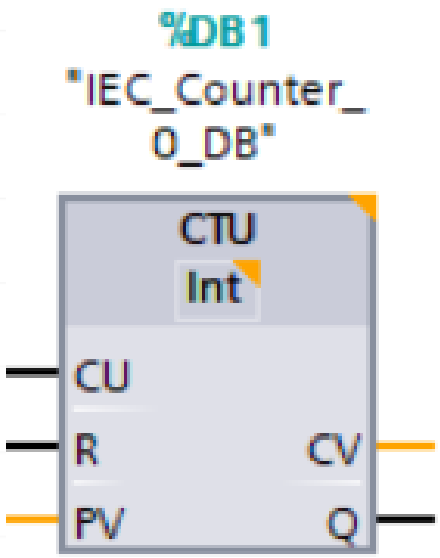
LAD – Liczniki

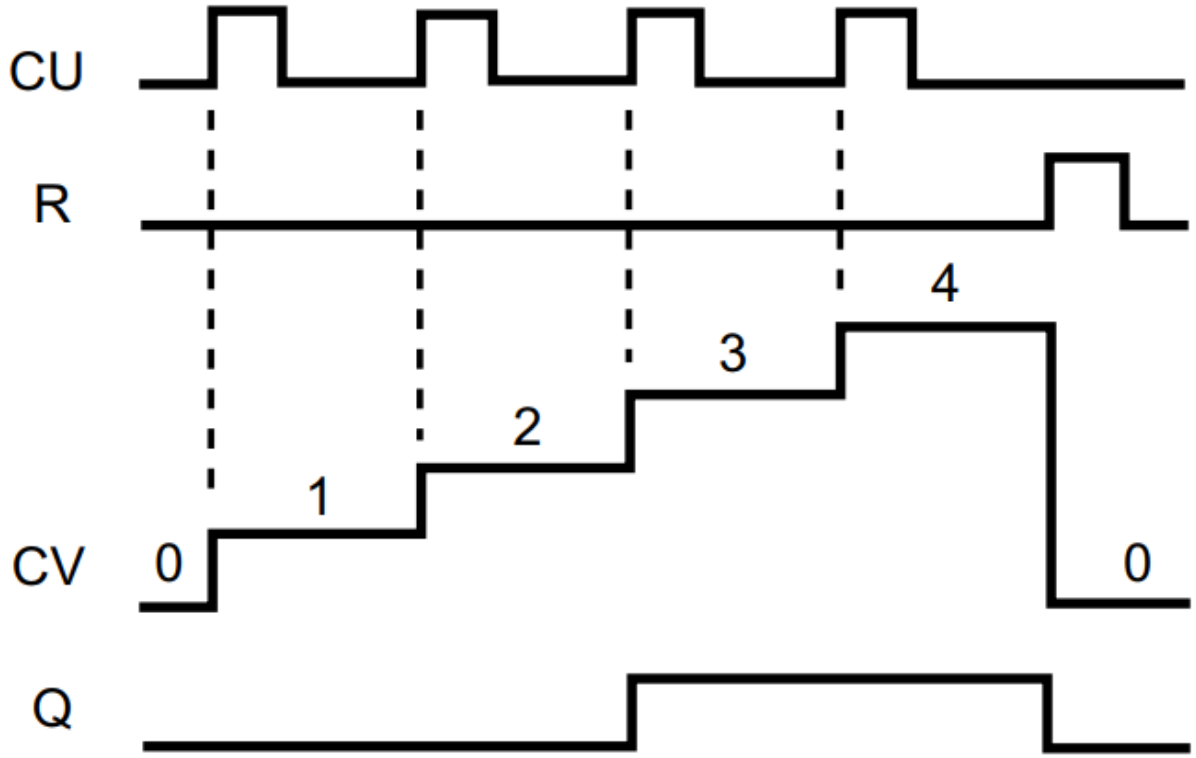
Każdy licznik wykorzystuje do pamiętania danych licznika strukturę przechowywaną w bloku danych. Dla LAD tworzone są automatycznie DB, gdy tylko tworzona jest instrukcja licznika.
Liczba liczników użytych w programie użytkownika jest ograniczona tylko rozmiarem pamięci CPU. Poszczególne liczniki używają 3 bajtów (dla SInt lub USInt), 6 bajtów (dla Int lub UInt) lub 12 bajtów (dla DInt lub UDInt).

LAD – Licznik CTU

CTU jest to licznik zliczający w górę o wartość 1, gdy wartość parametru wejściowego CU zmienia się z 0 na 1.

- **CU** – wejście licznika
- **PV** – ustalona wartość zliczeń (liczba całkowita)
- **CV** – bieżąca wartość zliczeń (liczba całkowita)
- **R** – kasowanie liczby zliczeń (na poziom)
- **Q** – wyjście licznika





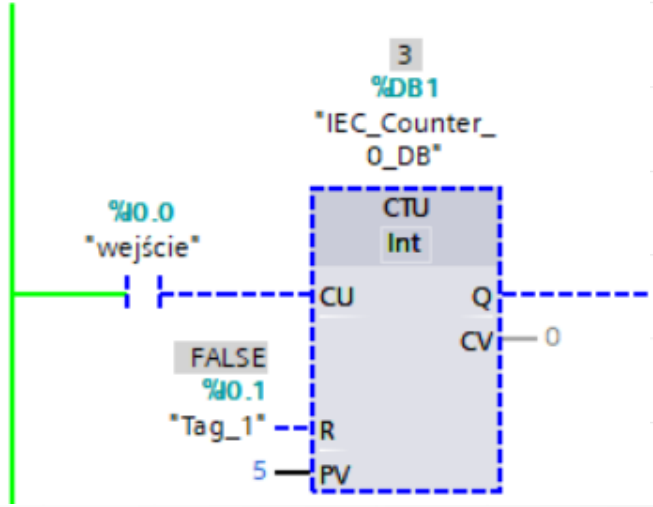
LAD – Licznik CTU

Na poprzednim slajdzie przedstawiono przebieg czasowy w przypadku licznika CTU zliczającego liczby całkowite bez znaku (dla PV = 3).

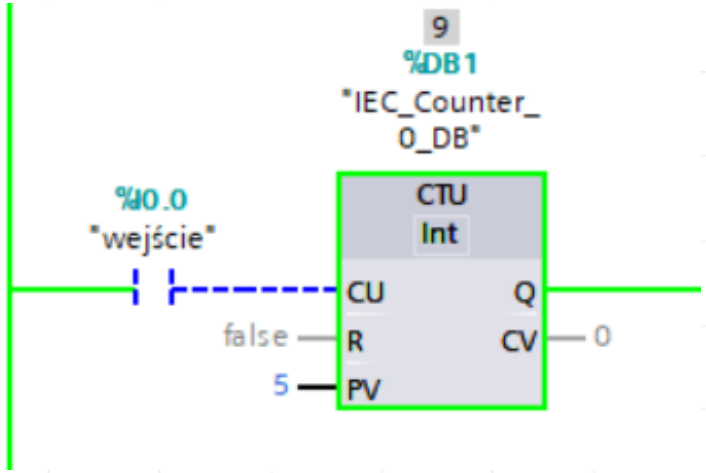
- Jeżeli wartość parametru CV (*current count value* – bieżąca wartość zliczeń) jest większa lub równa wartości parametru PV (*preset count value* – ustalona wartość zliczeń), to parametr wyjściowy licznika Q = 1.
- Jeżeli wartość parametru kasującego R zmienia się z 0 na 1, to bieżąca wartość zliczeń zostaje skasowana do 0.

LAD – Licznik CTU

"IEC_Counter_0_DB".QU		Bool		FALSE
"IEC_Counter_0_DB".QD		Bool		FALSE
"IEC_Counter_0_DB".PV		DEC+/-		5
"IEC_Counter_0_DB".CV		DEC+/-		3
"wejście":P	%I0.0:P	Bool		FALSE
"Tag_1":P	%I0.1:P	Bool		FALSE
"wyjście"	%Q0.0	Bool		FALSE



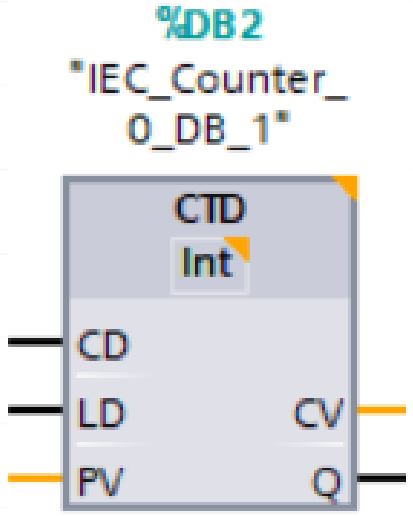
"IEC_Counter_0_DB".QU		Bool		TRUE
"IEC_Counter_0_DB".QD		Bool		FALSE
"IEC_Counter_0_DB".PV		DEC+/-		5
"IEC_Counter_0_DB".CV		DEC+/-		9
"wejście":P	%I0.0:P	Bool		FALSE
"wyjście"	%Q0.0	Bool		TRUE

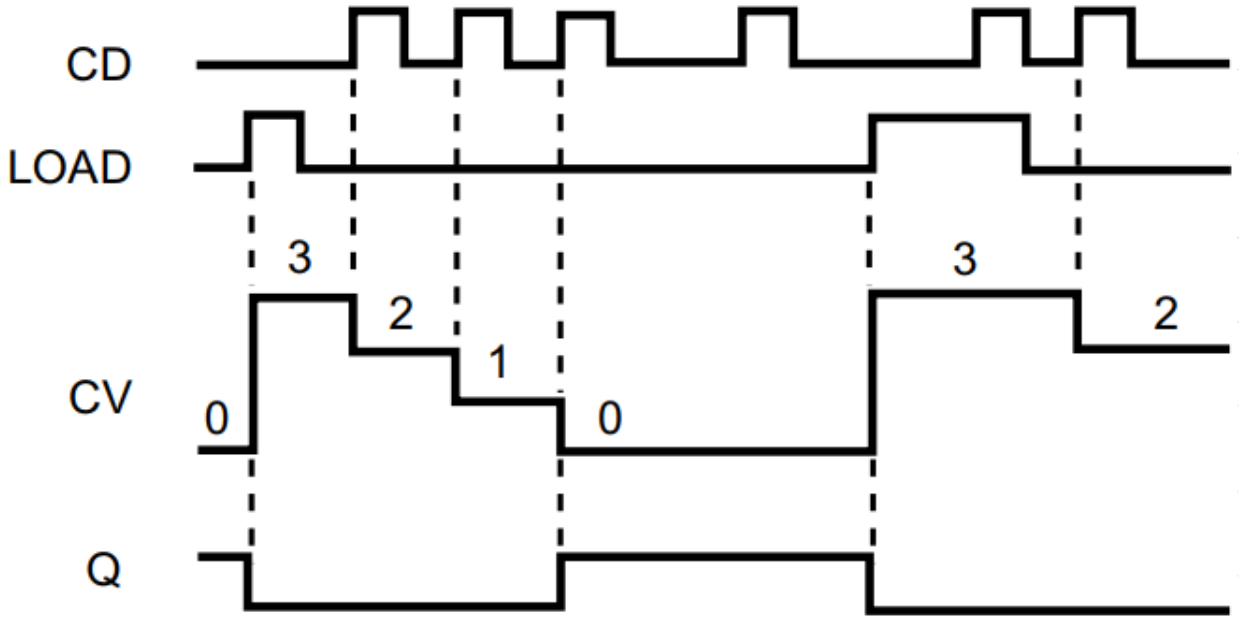


LAD – Licznik CTD

CTD jest to licznik zliczający w dół o wartość 1, gdy wartość parametru wejściowego CD zmienia się z 0 na 1.

- **CD** – wejście licznika
- **PV** – ustalona wartość zliczeń (liczba całkowita)
- **CV** – bieżąca wartość zliczeń (liczba całkowita)
- **LOAD** – wpis wartość **PV** do **CV** (na poziom)
- **Q** – wyjście licznika





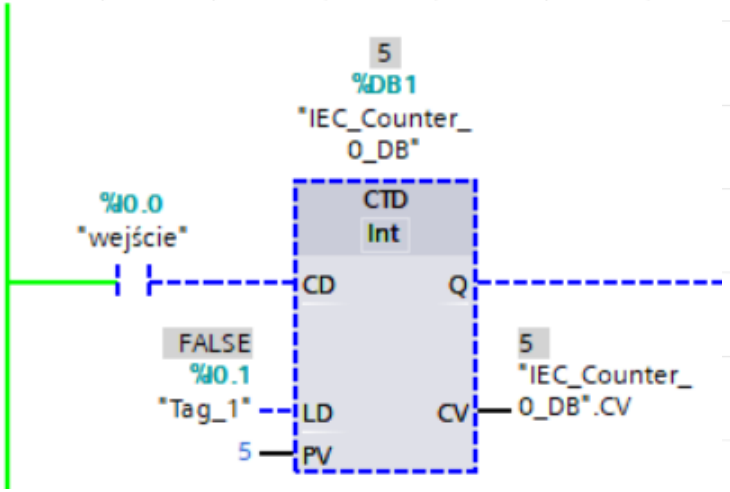
LAD – Licznik CTD

Na poprzednim slajdzie przedstawiono przebieg czasowy w przypadku licznika CTD zliczającego liczby całkowite bez znaku (dla PV = 3).

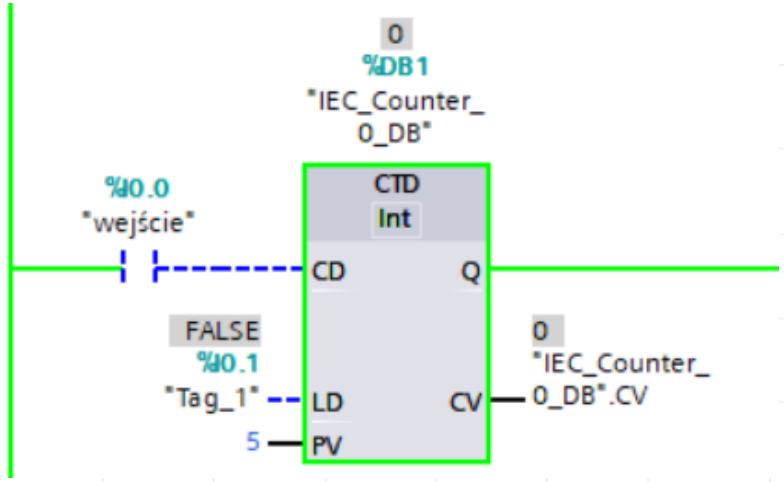
- Jeżeli wartość parametru CV (*current count value* – bieżąca wartość zliczeń) jest mniejsza lub równa 0, to parametr wyjściowy licznika Q = 1.
- Jeżeli wartość parametru LOAD zmienia się z 0 na 1, to wartość parametru PV (*preset count value* – ustalona wartość zliczeń) jest wpisywana do licznika jako nowa wartość CV (*current count value* – bieżąca wartość zliczeń).

LAD – Licznik CTD

"IEC_Counter_0_DB".PV		DEC+/-		5
"IEC_Counter_0_DB".CV		DEC+/-		5
"wejście":P	%I0.0:P	Bool		FALSE
"Tag_1":P	%I0.1:P	Bool		FALSE
"wyjście"	%Q0.0	Bool		FALSE



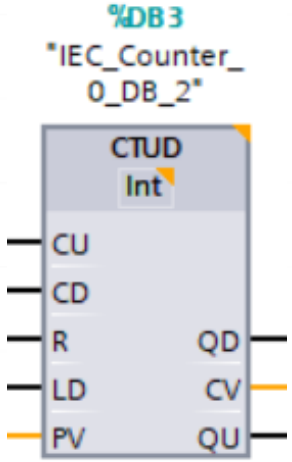
"IEC_Counter_0_DB".PV		DEC+/-		5
"IEC_Counter_0_DB".CV		DEC+/-		0
"wejście":P	%I0.0:P	Bool		FALSE
"Tag_1":P	%I0.1:P	Bool		FALSE
"wyjście"	%Q0.0	Bool		TRUE

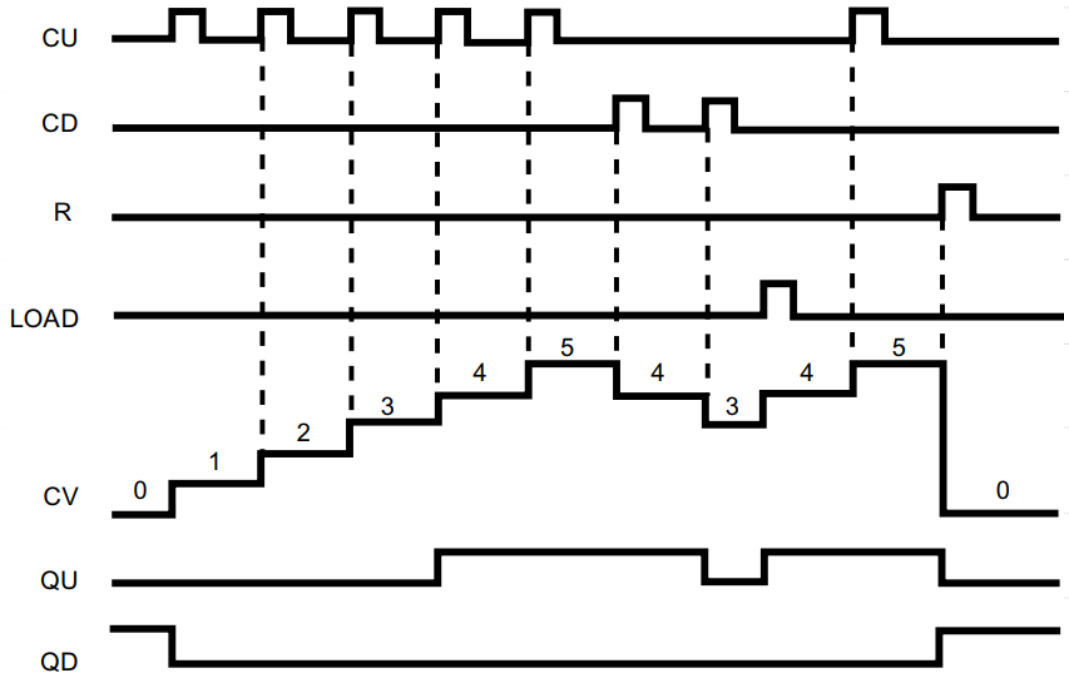


LAD – Licznik CTUD

CTUD jest to licznik zliczający w górę lub w dół o wartość 1, gdy wartość parametru wejściowego CU lub CD zmienia się z 0 na 1.

- **CU** – wejście licznika – zliczanie w górę
- **CD** – wejście licznika – zliczanie w dół
- **R** – kasowanie liczby zliczeń (na poziom)
- **LOAD** – wpis wartość **PV** do **CV** (na poziom)
- **QU** – wyjście licznika równe 1, gdy **CV** = **PV**
- **QD** – wyjście licznika równe 1, gdy **CV** = 0
- **PV** – ustalona wartość zliczeń (liczba całkowita)
- **CV** – bieżąca wartość zliczeń (liczba całkowita)



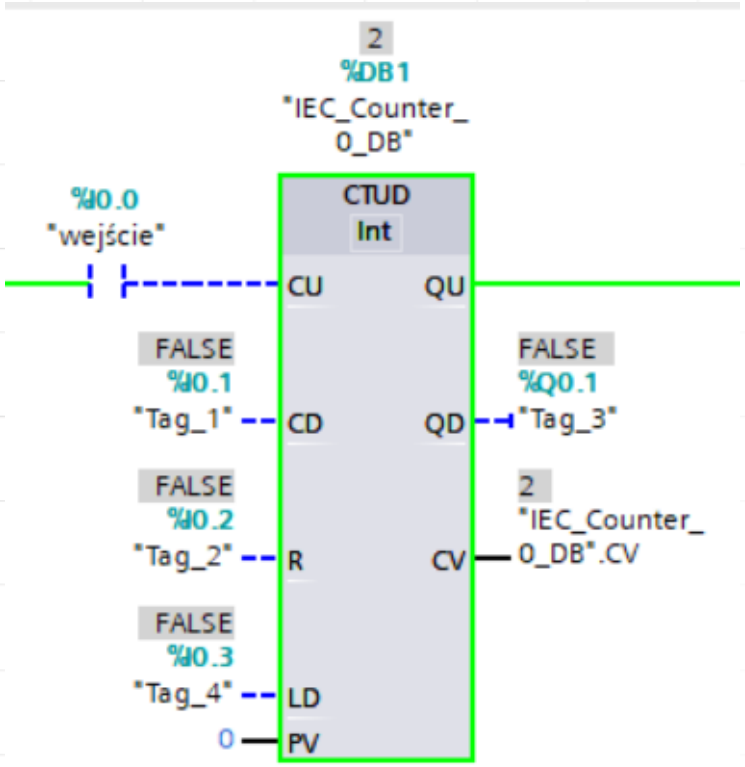


LAD – Licznik CTUD

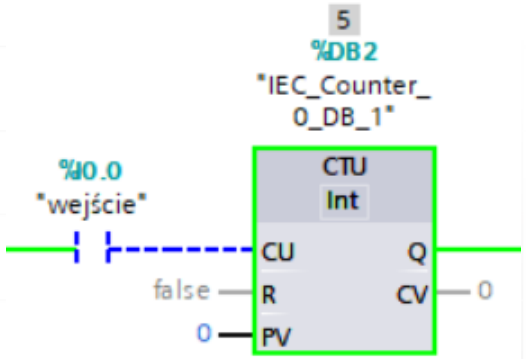
Na poprzednim slajdzie przedstawiono przebieg czasowy w przypadku licznika CTUD zliczającego liczby całkowite bez znaku (dla PV = 4).

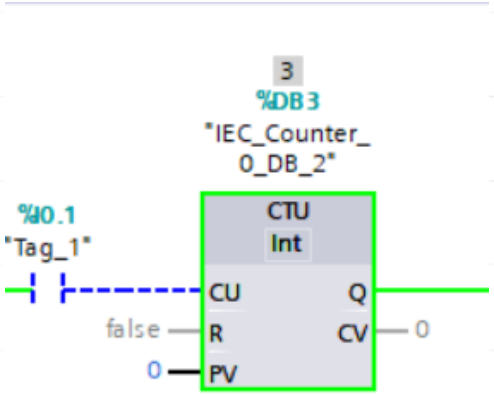
- Jeżeli wartość parametru CV (*current count value* – bieżąca wartość zliczeń) jest równa lub większa od wartości parametru PV (*preset value* – ustalona wartość), to parametr wyjściowy licznika QU = 1.
- Jeżeli wartość parametru CV jest mniejsza lub równa 0, to parametr wyjściowy licznika QD = 1.
- Jeżeli wartość parametru LOAD zmienia się z 0 na 1, to wartość parametru PV jest wpisywana do licznika jako nowa wartość CV.
- Jeżeli wartość parametru kasującego R zmienia się z 0 na 1, to bieżąca wartość zliczeń zostaje skasowana do 0.

LAD – Licznik CTUD

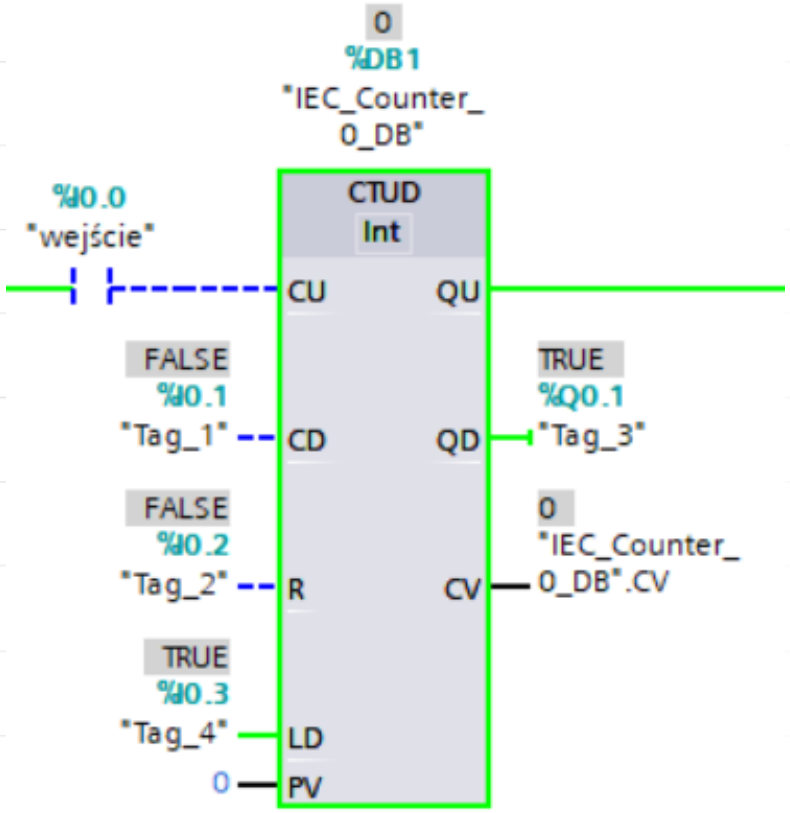


Poniżej przedstawiono działanie licznika CTUD oraz przedstawiono liczniki CTU dla wejścia zliczającego w górę (I0.0) i wejścia zliczającego w dół (I0.1).

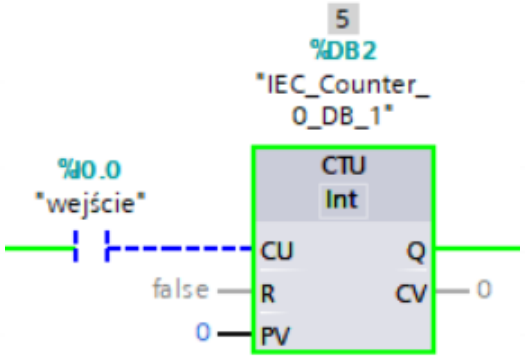


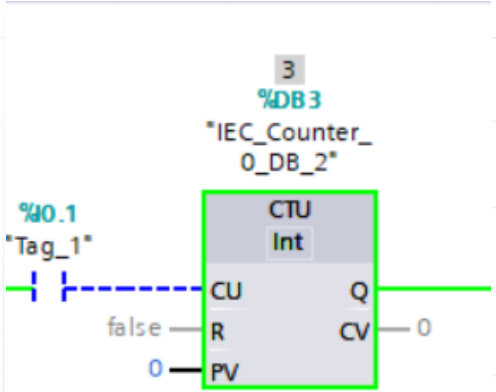


LAD – Licznik CTUD



Poniżej przedstawiono działanie licznika CTUD po aktywacji wejścia LD (I0.3, przepisuje wartość PV do CV) oraz przedstawiono liczniki CTU dla wejścia zliczającego w górę (I0.0) i wejścia zliczającego w dół (I0.1).





LAD – Comparator operations

Comparator operations	
CMP ==	Equal
CMP <>	Not equal
CMP >=	Greater or equal
CMP <=	Less or equal
CMP >	Greater than
CMP <	Less than
IN_Range	Value within range
OUT_Range	Value outside range
–|OK|–	Check validity
–|NOT_OK|–	Check invalidity

Typy danych liczbowych

Binarne		
Bool (bit)	False (0)	True (1)

Rejestry bitowe		
Byte	8 bitów	0…255
Word	16 bitów	0…65 535
DWord	32 bity	0…4 294 967 295
LWord*	64 bity	0…18 446 744 073 709 551 615

* Dostępne tylko w sterownikach Siemens serii S7-1500

Typy danych liczbowych

Całkowite		
Sint	8 bitów	-128…+127
INT	16 bitów	-32 768 …+32 767
DINT	32 bity	-2 147 483 648…+2 147 483 647
Usint	8 bitów	0…255
UINT	16 bitów	0…65 535
UDINT	32 bity	0…4 294 967 295
LINT*	64 bity	-9 223 372 036 854 775 808… +9 223 372 036 854 775 807
ULINT*	64 bity	0…18 446 744 073 709 551 615

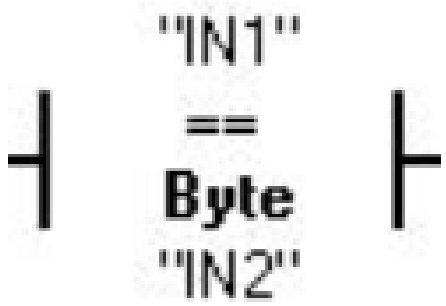
* Dostępne tylko w sterownikach Siemens serii S7-1500

Typy danych liczbowych

Rzeczywiste, zmiennoprzecinkowe		
Real	32 bity	$-3.402823e^{+38}$ … $-1.175495e^{-38}$ $+1.175495e^{-38}$ … $+3.402823e^{+38}$
Lreal*	64 bity	$-1.7976931348623158e^{+308}$… $-2.2250738585072014e^{-308}$ $+2.2250738585072014e^{-308}$… $+1.7976931348623158e^{+308}$

LAD – instrukcja porównania

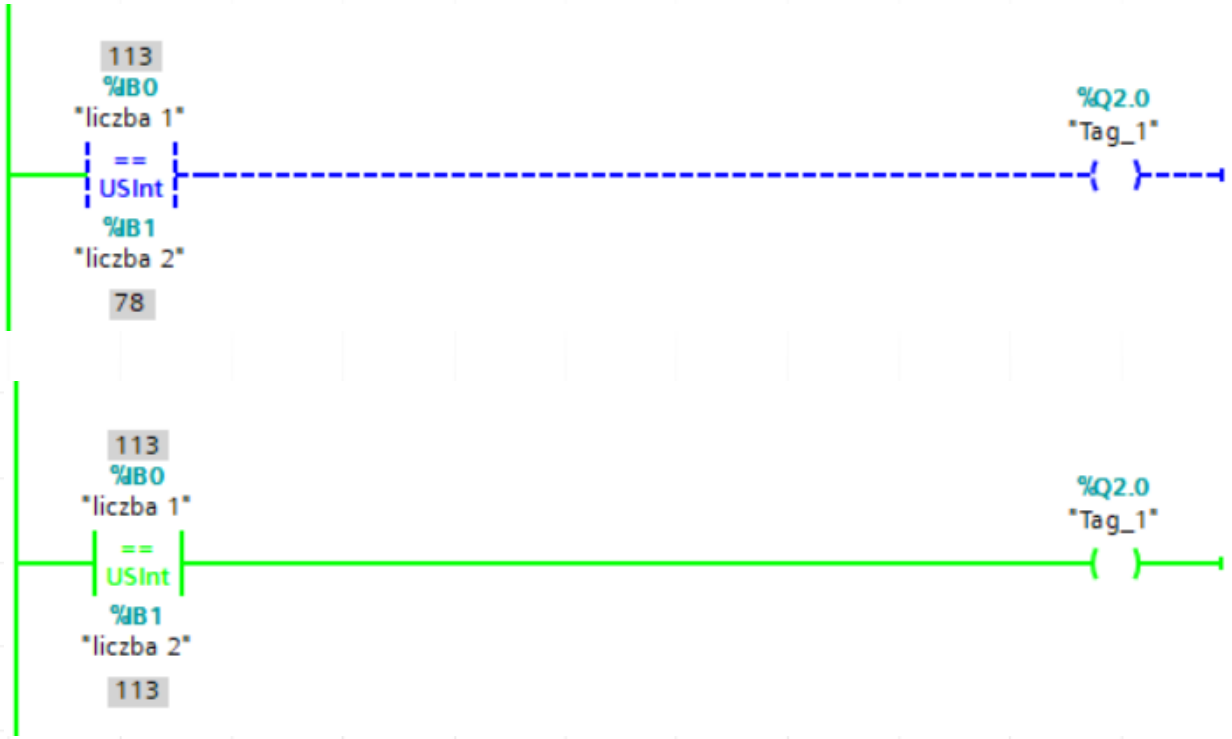
- Equals (==): Porównanie jest prawdą, jeśli IN1 jest równe IN2.
- Not equal (<>): Porównanie jest prawdą, jeśli IN1 nie jest równe IN2.
- Greater than or equal to (>=): Porównanie jest prawdą, jeśli IN1 jest większe od lub równe IN2.
- Less than or equal to (<=): Porównanie jest prawdą, jeśli IN1 jest mniejsze od lub równe IN2.
- Greater than (>): Porównanie jest prawdą, jeśli IN1 jest większe od IN2.
- Less than (<): Porównanie jest prawdą, jeśli IN1 jest mniejsze od IN2.



Wynik porównania LAD ma wartość TRUE, gdy porównanie jest prawdą.

LAD – instrukcja równe

- Equals (==): Porównanie jest prawdą, jeśli IN1 jest równe IN2.



Wynik porównania LAD ma wartość TRUE, gdy porównanie jest prawdą.

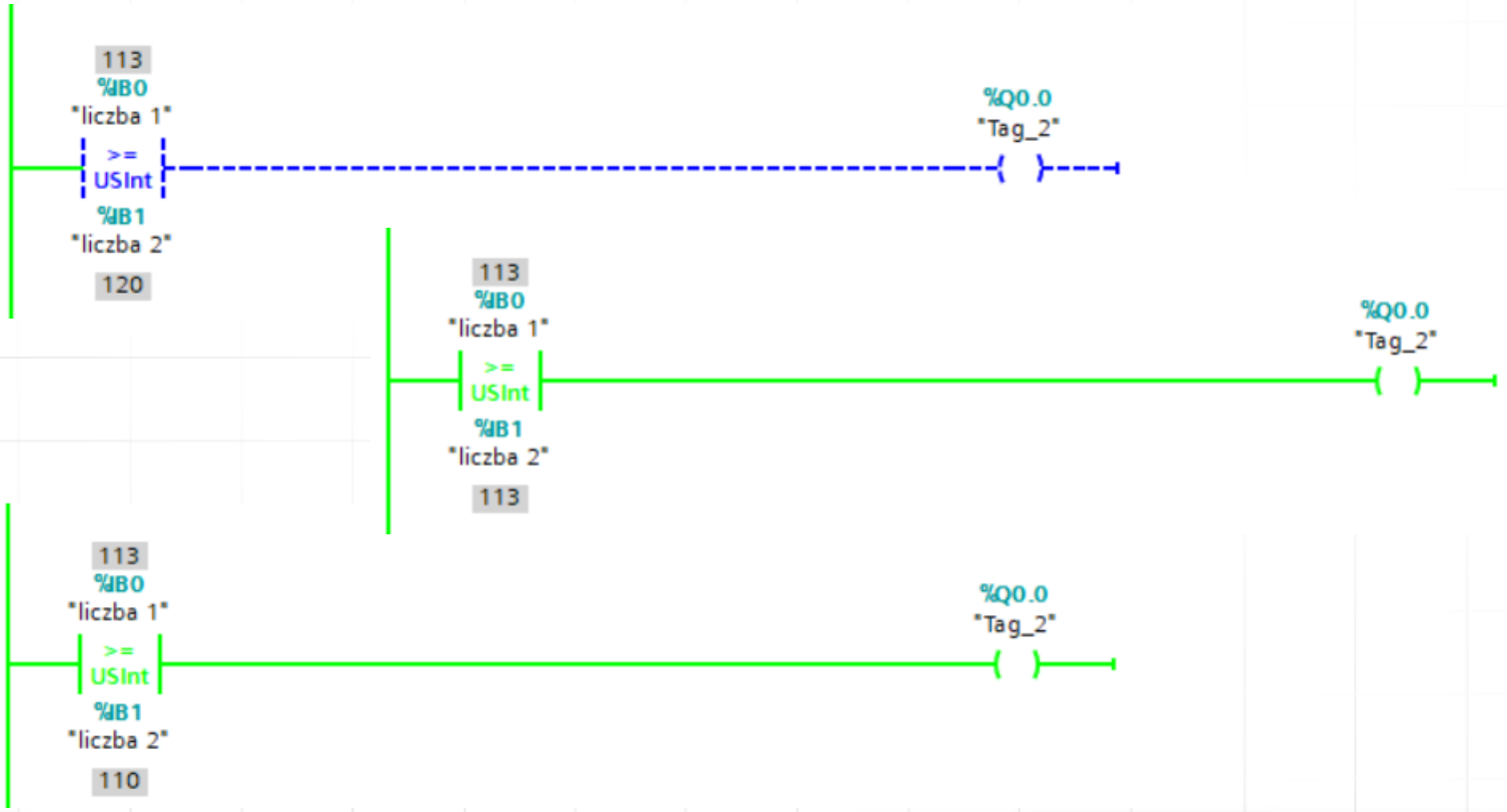
LAD – instrukcja nierówne

- Not equal (<>): Porównanie jest prawdą, jeśli IN1 nie jest równe IN2.

Wynik porównania LAD ma wartość TRUE, gdy porównanie jest prawdą.

LAD – instrukcja większe lub równe

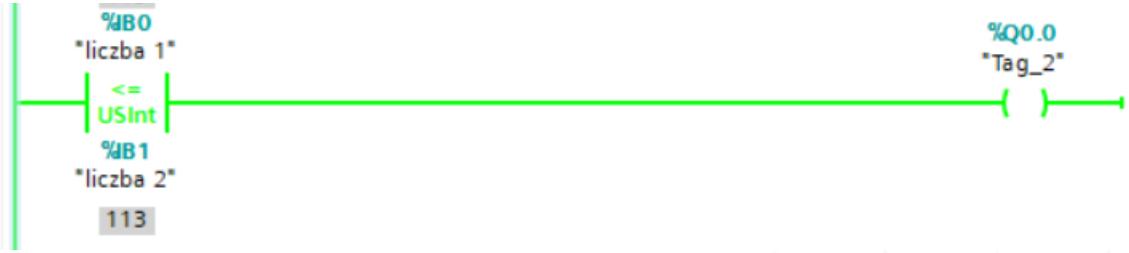
- Greater than or equal to (>=): Porównanie jest prawdą, jeśli IN1 jest większe od lub równe IN2.

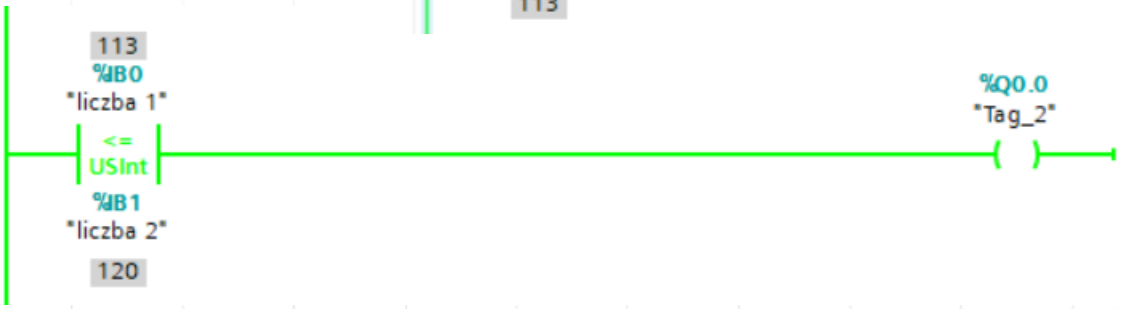


Wynik porównania LAD ma wartość TRUE, gdy porównanie jest prawdą.

LAD – instrukcja mniejsze lub równe

- Less than or equal to (<=): Porównanie jest prawdą, jeśli IN1 jest mniejsze od lub równe IN2.

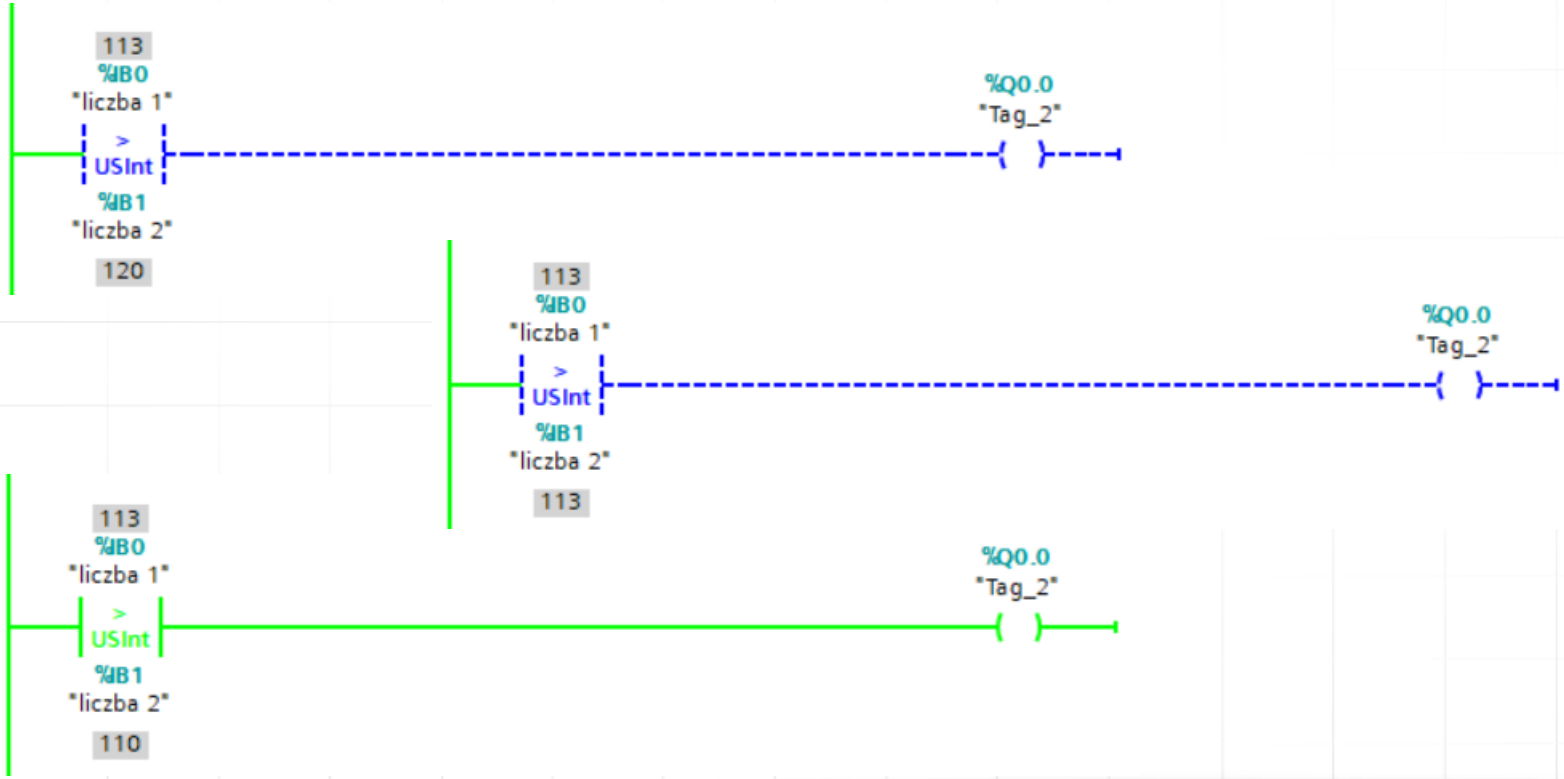




Wynik porównania LAD ma wartość TRUE, gdy porównanie jest prawdą.

LAD – instrukcja większe

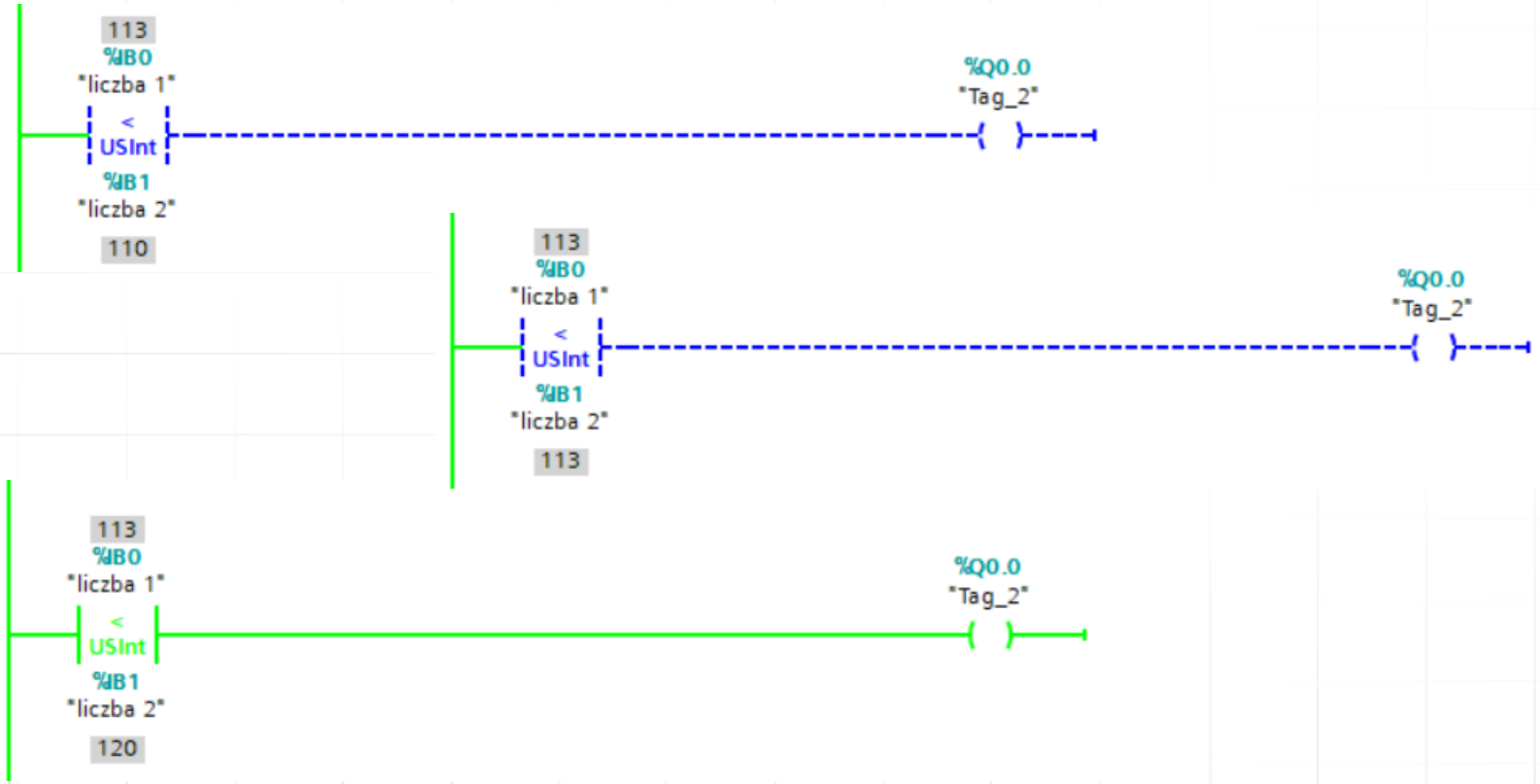
- Greater than (>): Porównanie jest prawdą, jeśli IN1 jest większe od IN2.



Wynik porównania LAD ma wartość TRUE, gdy porównanie jest prawdą.

LAD – instrukcja mniejsze

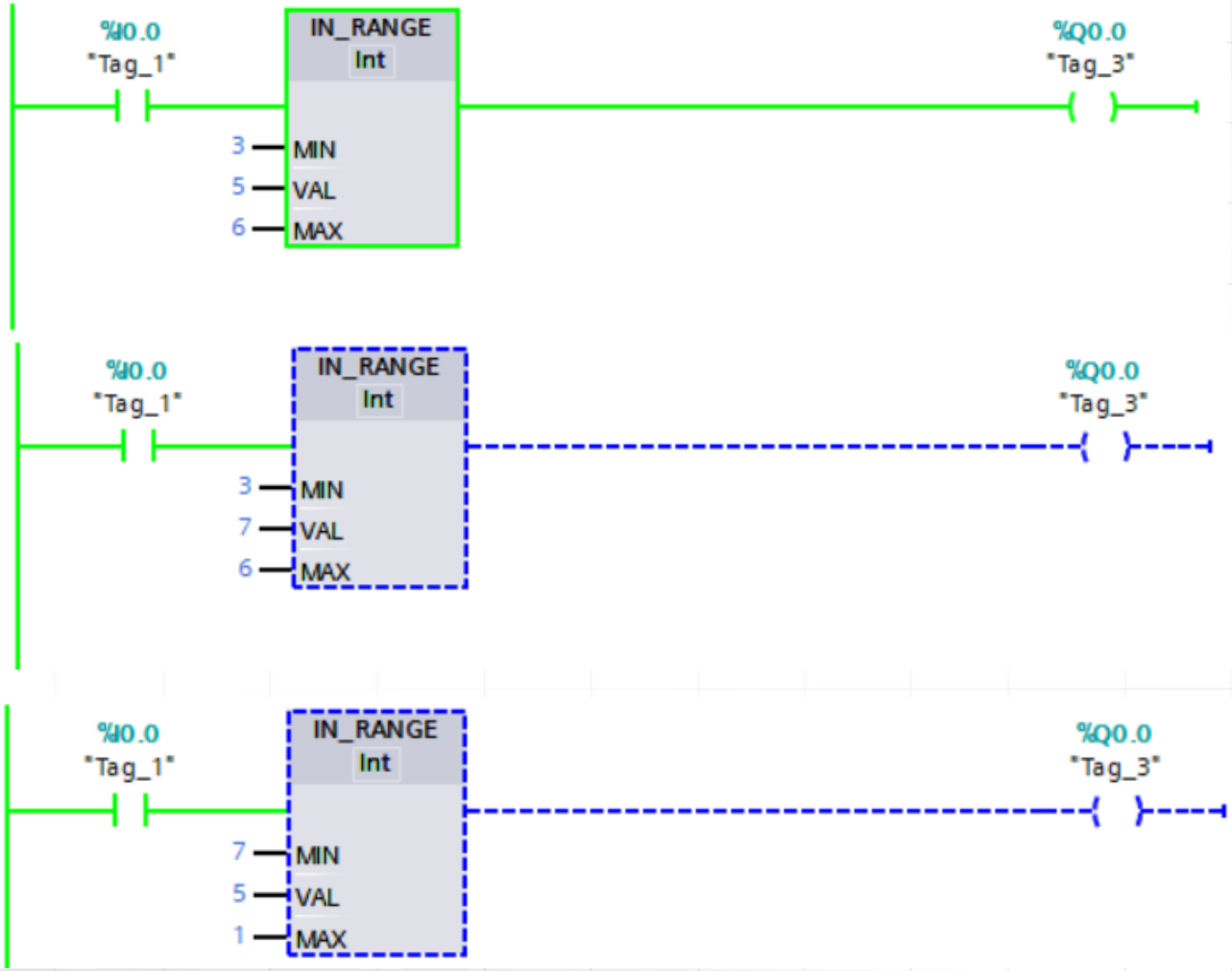
- Less than (<): Porównanie jest prawdą, jeśli IN1 jest mniejsze od IN2.



Wynik porównania LAD ma wartość TRUE, gdy porównanie jest prawdą.

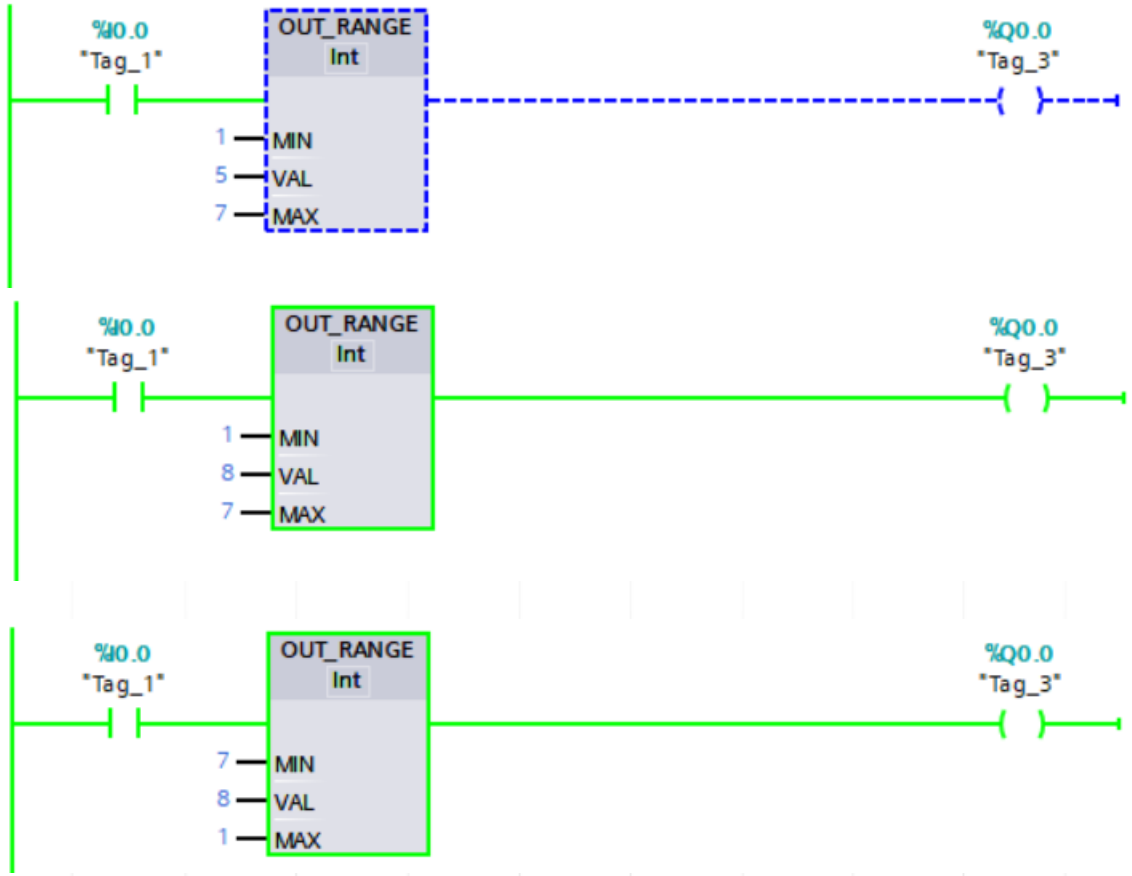
LAD – instrukcja IN_RANGE

Instrukcja IN_RANGE sprawdza czy zadana wartość VAL znajduje się w określonym przedziale (pomiędzy wartościami MIN i MAX, należy jednak pamiętać, że wartość MAX musi być większa od wartości MIN). Wartość 1 na wyjściu oznacza, że zadana wartość znajduje się w danym przedziale.



LAD – instrukcja OUT_RANGE

Instrukcja IN_RANGE sprawdza czy zadana wartość VAL znajduje się w określonym przedziale (pomiędzy wartościami MIN i MAX). Program porównuje czy zadana wartość jest większa od wartości MAX lub mniejsza od wartości MIN.



LAD – instrukcja |OK|

Instrukcja |OK| sprawdza czy wartość operandu (%ID0) jest prawidłową liczbą zmiennoprzecinkową. Wartość wyjściowa przyjmuje wartość 1, gdy podana liczba jest prawidłową liczbą zmiennoprzecinkową.

LAD – instrukcja |NOT_OK|

Instrukcja |OK| sprawdza czy wartość operandu (%ID0) nie jest prawidłową liczbą zmiennoprzecinkową. Wartość wyjściowa przyjmuje wartość 1, gdy podana liczba nie jest prawidłową liczbą zmiennoprzecinkową.

LAD – math functions

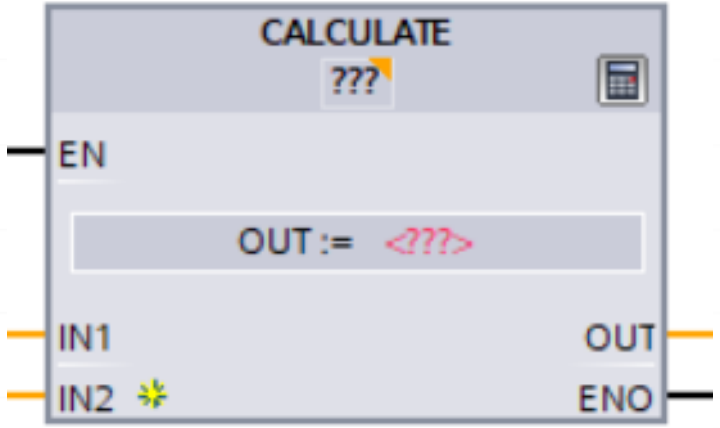
Math functions	
CALCULATE	Calculate
ADD	Add
SUB	Subtract
MUL	Multiply
DIV	Divide
MOD	Return remainder of di...
NEG	Create twos complemen
INC	Increment
DEC	Decrement
ABS	Form absolute value
MIN	Get minimum
MAX	Get maximum
LIMIT	Set limit value

SQR	Form square
SQRT	Form square root
LN	Form natural logarithm
EXP	Form exponential value
SIN	Form sine value
COS	Form cosine value
TAN	Form tangent value
ASIN	Form arcsine value
ACOS	Form arccosine value
ATAN	Form arctangent value
FRAC	Return fraction
EXPT	Exponentiate

LAD – instrukcja CALCULATE

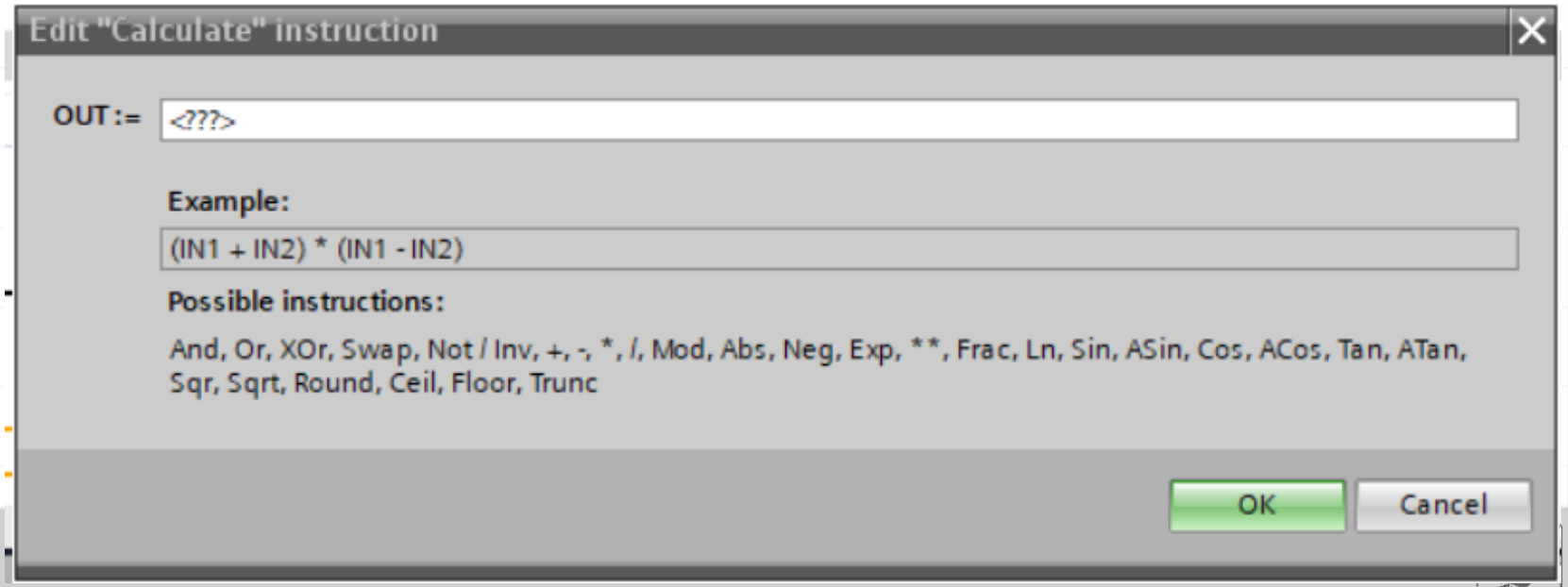
Instrukcja CALCULATE pozwala na utworzenie matematycznych funkcji, które operują na wejściach (IN1, IN2, INn) i tworzą wynik OUT zgodny ze zdefiniowanym przez użytkownika równaniem.

- Najpierw należy wybrać typ obsługiwanych danych. Wszystkie wejścia i wyjście muszą być tego samego typu.
- Aby dodać kolejne wejście należy nacisnąć ikonę przy ostatnim wejściu.

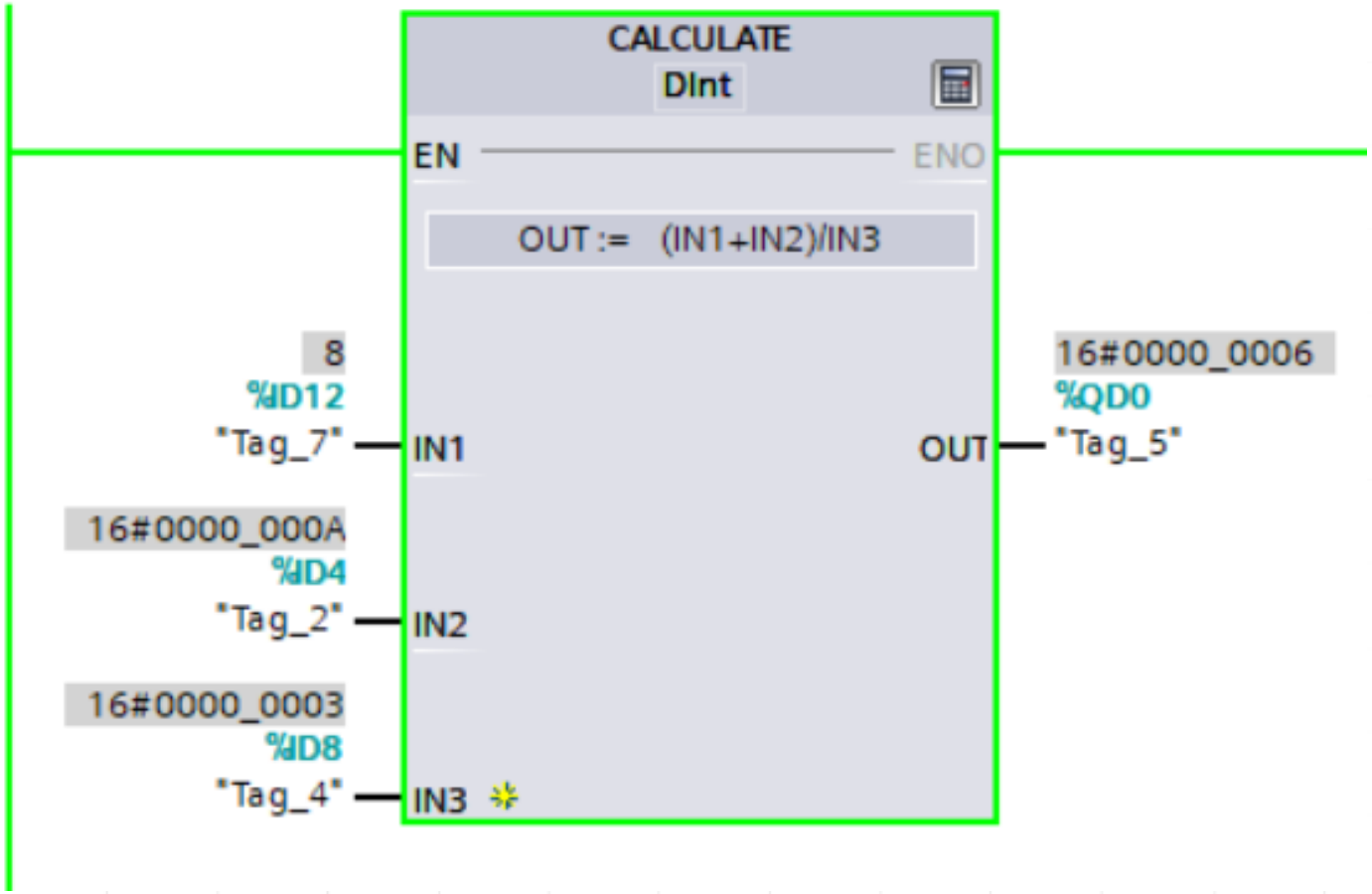


LAD – instrukcja CALCULATE

Należy nacisnąć ikonę kalkulatora dla otwarcia okna dialogowego i zdefiniowania funkcji matematycznej. W równaniach należy podawać wejścia (jak IN1 i IN2) i operacje. Po kliknięciu *OK* następuje zapisanie funkcji i automatyczne stworzenie wymaganych przez instrukcję wejść.
Przykład i lista możliwych operacji matematycznych pokazana jest na dole edytora.



LAD – instrukcja CALCULATE

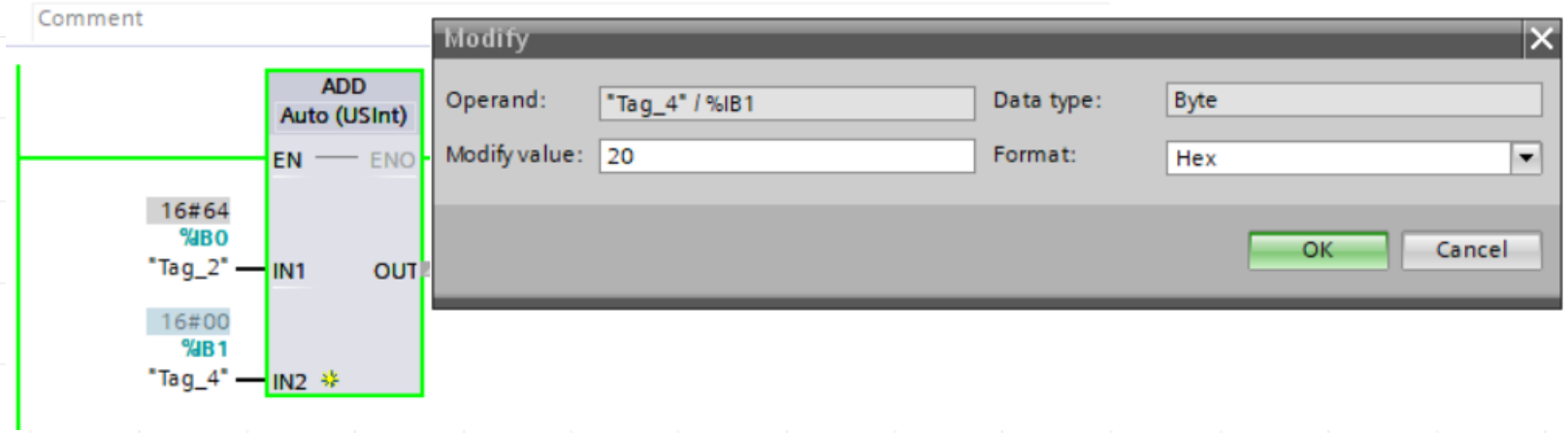


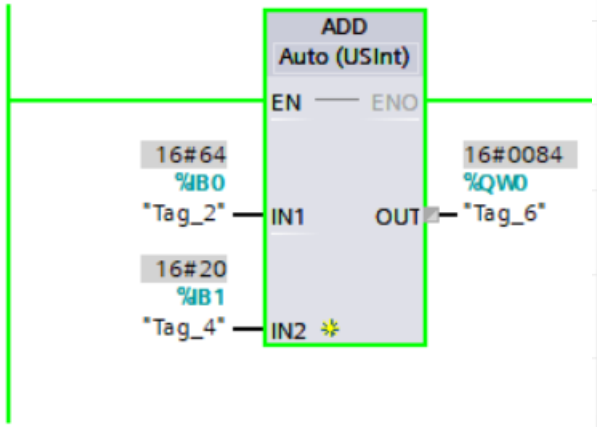
Możliwe typy zmiennych:

- Int,
- Dint,
- Real,
- LReal,
- USInt,
- Uint,
- Sint,
- UDInt,
- Byte,
- Word,
- Dword.

LAD – działania matematyczne

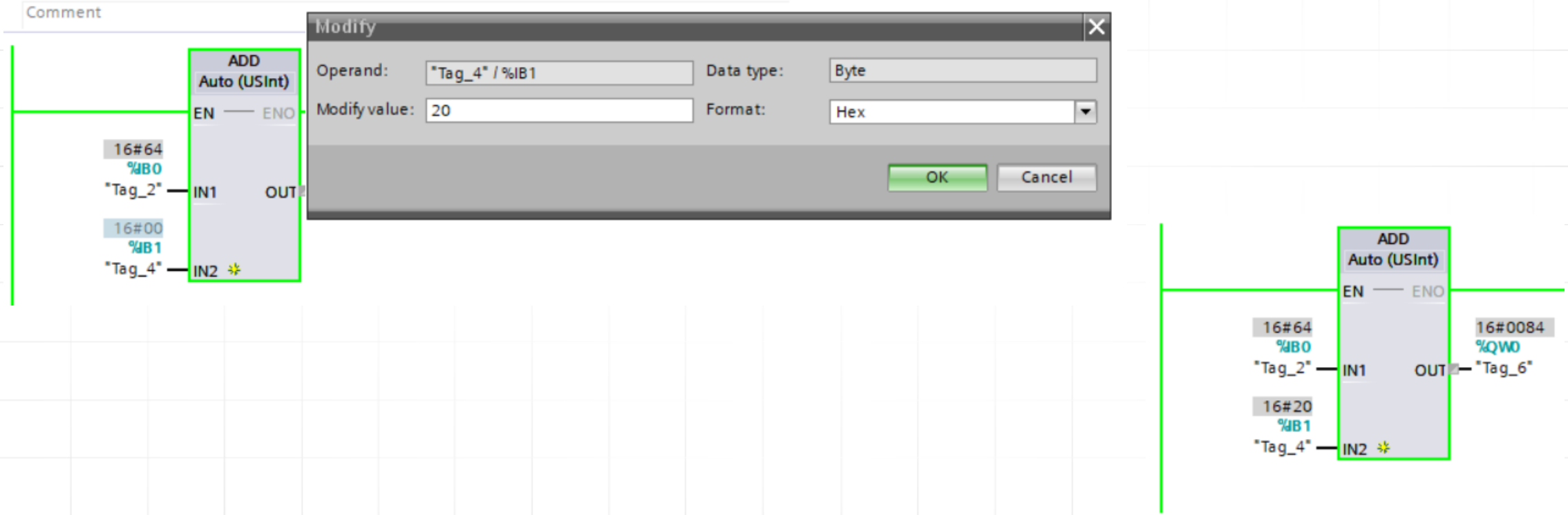
- w przypadku wcześniejszego zadeklarowania zmiennych automatycznie jest rozpoznawany i wpisany typ zmiennej, w innym przypadku typ trzeba wybrać z menu rozwijalnego (**???**)
- rozszerzenie liczby wejść poprzez kliknięcie w *
- skasowanie wejścia poprzez zaznaczenie i wykonanie polecenia delete





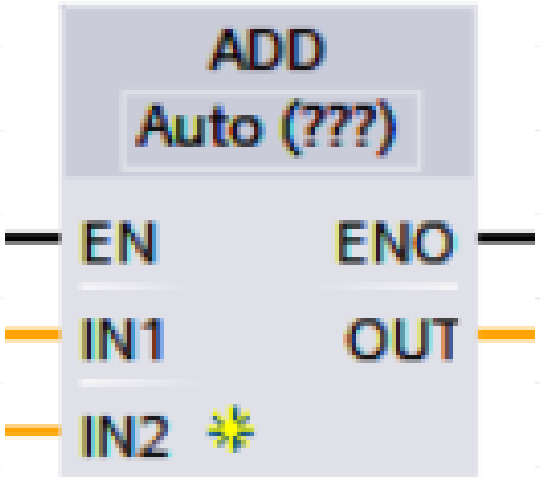
LAD – instrukcja dodawania

Poniżej przedstawiono przykład działania instrukcji ADD. Jeżeli do liczby 64(16) dodamy liczbę 20(16) to otrzymamy liczbę 84(16).



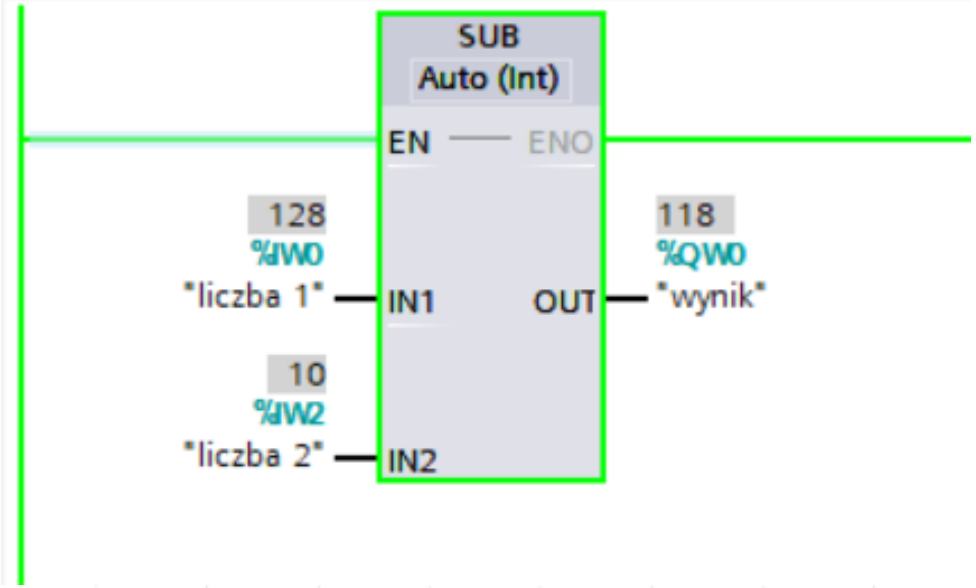
LAD – działania matematyczne

- kliknięcie w skrót operacji arytmetycznej powoduje pojawienie się menu rozwijalnego, co umożliwia zmianę typu operacji, bez zmiany parametrów we/wy
  - **SUB** – odejmowanie
  - **MUL** – mnożenie
  - **DIV** – dzielenie
  - **MOD** – reszta z dzielenia (tylko liczby typu INT)



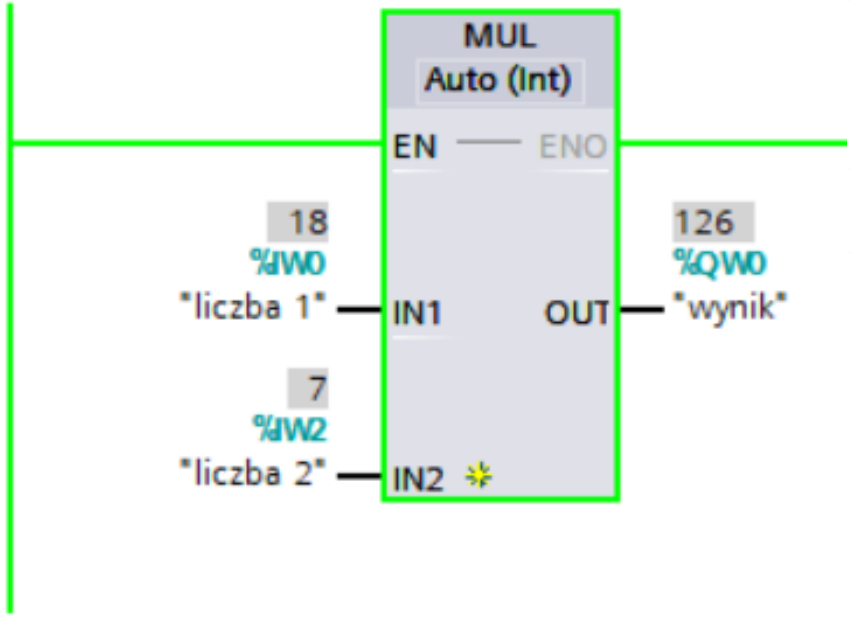
LAD – instrukcja odejmowania

Poniżej przedstawiono przykład działania instrukcji SUB. Jeżeli od liczby 128(10) odejmiemy liczbę 10(10) to otrzymamy liczbę 118(10).



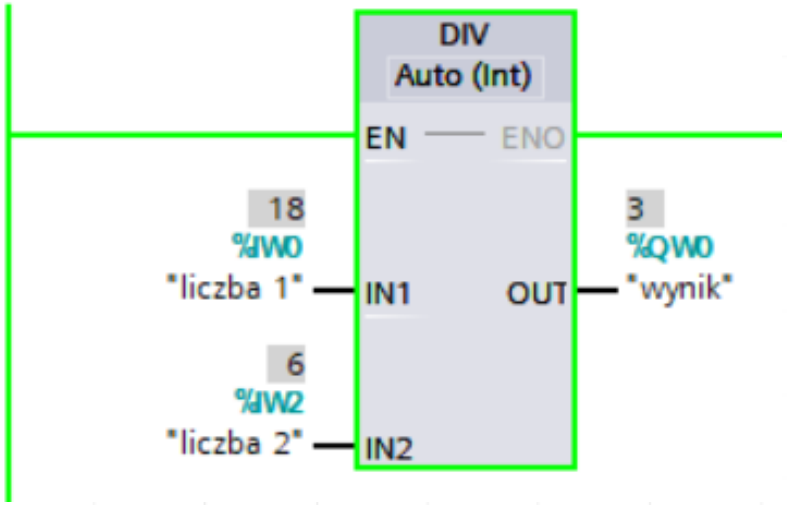
LAD – instrukcja mnożenia

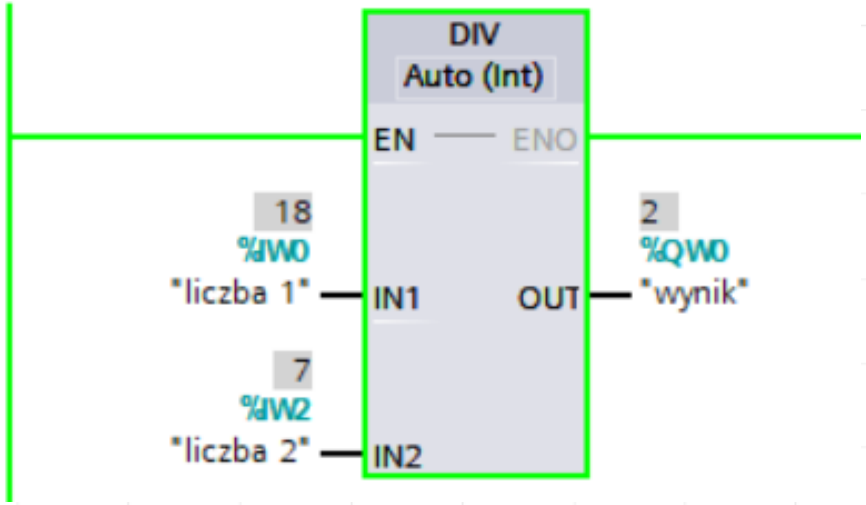
Poniżej przedstawiono przykład działania instrukcji MUL. Jeżeli liczbę 18(10) pomnożymy przez liczbę 7(10) to otrzymamy liczbę 126(10).



LAD – instrukcja dzielenia

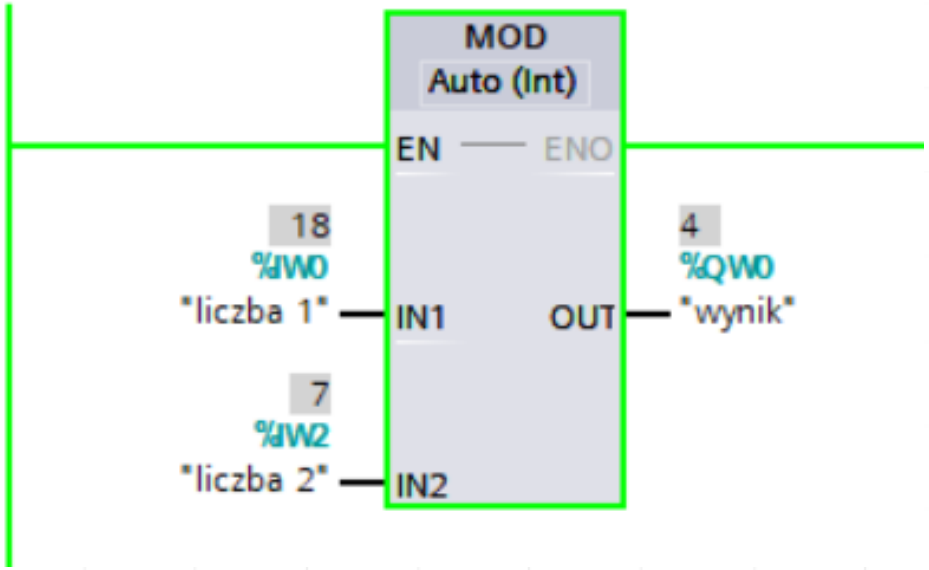
Poniżej przedstawiono przykład działania instrukcji DIV. Jeżeli liczbę 18(10) podzielimy przez liczbę 6(10) to otrzymamy liczbę 3(10). Jeżeli liczbę 18(10) podzielimy przez 7(10) to otrzymamy wynik 2(10) ponieważ liczby INT są liczbami całkowitymi.





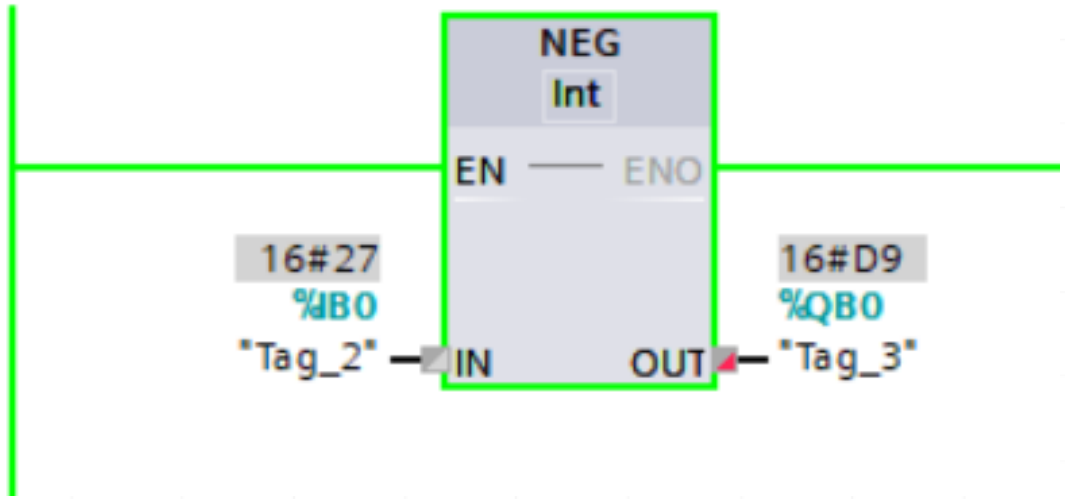
LAD – instrukcja reszta z dzielenia

Poniżej przedstawiono przykład działania instrukcji MOD. Jeżeli liczbę 18(10) podzielimy przez 7(10) to otrzymamy wynik 4(10) ponieważ 18 dzielone przez 7 daje 2 i 4 reszty.



LAD – instrukcja negacji

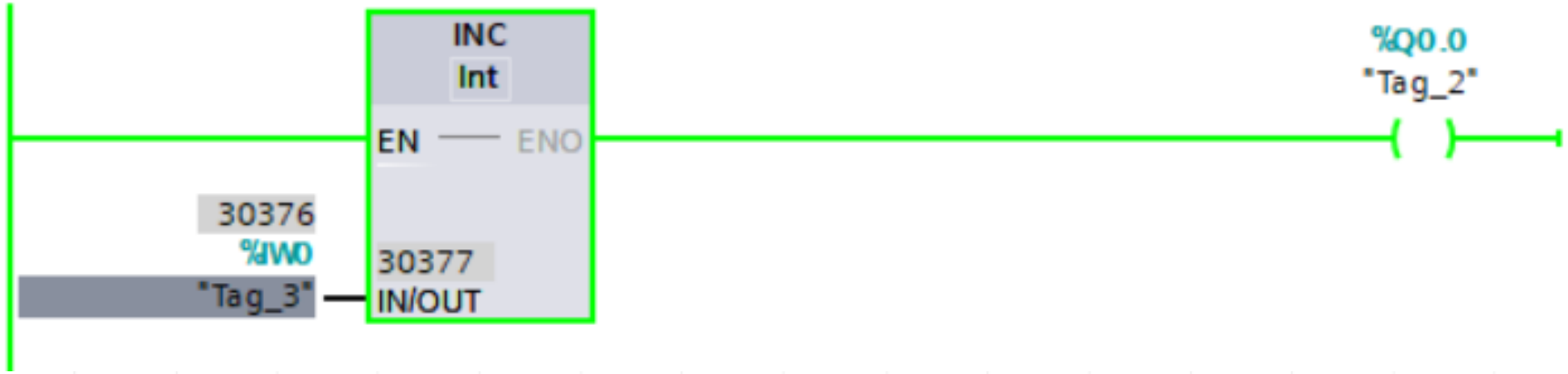
Operacja NEG zmienia wartość każdego bitu wejściowego na przeciwną wartość w bicie wyjściowym.



"Tag_2":P	%IB0:P	Hex	16#27	
"Tag_3"	%QB0	Hex	16#D9	

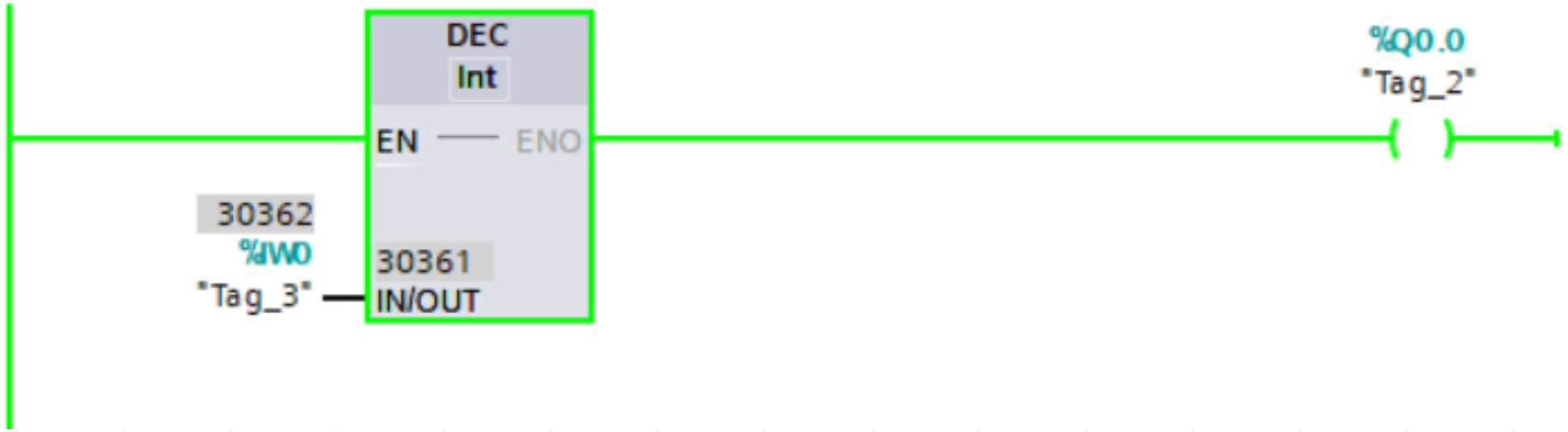
LAD – instrukcja dodawanie 1

Operacja inkrementacji INC – zwiększania wartości o 1 (tylko liczby całkowite – typu int).



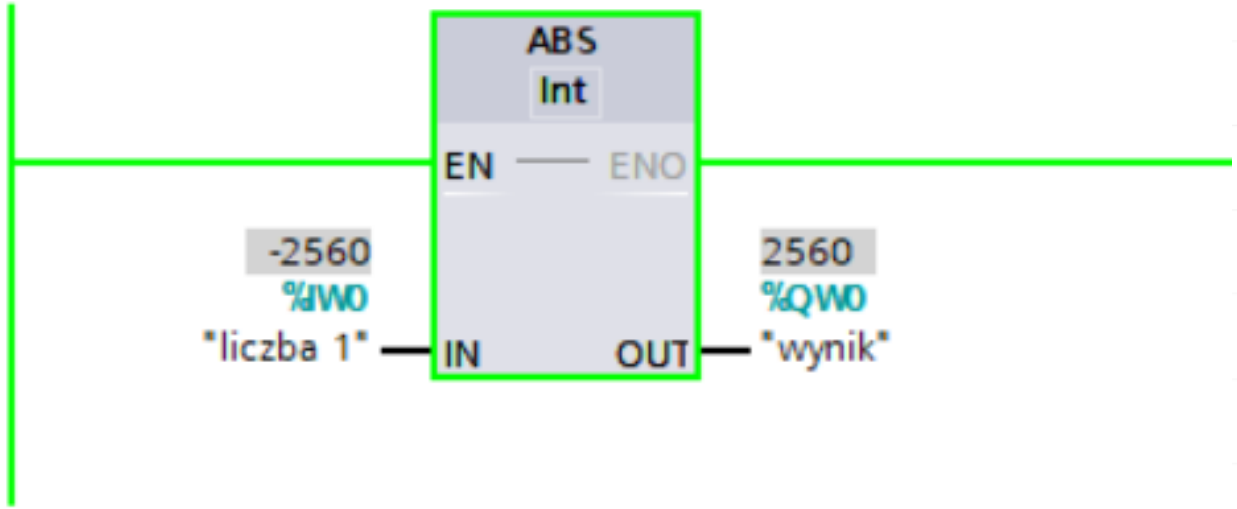
LAD – instrukcja odejmowanie 1

Operacja dekrementacji DEC – zmniejszania wartości o 1 (tylko liczby całkowite – typu int).



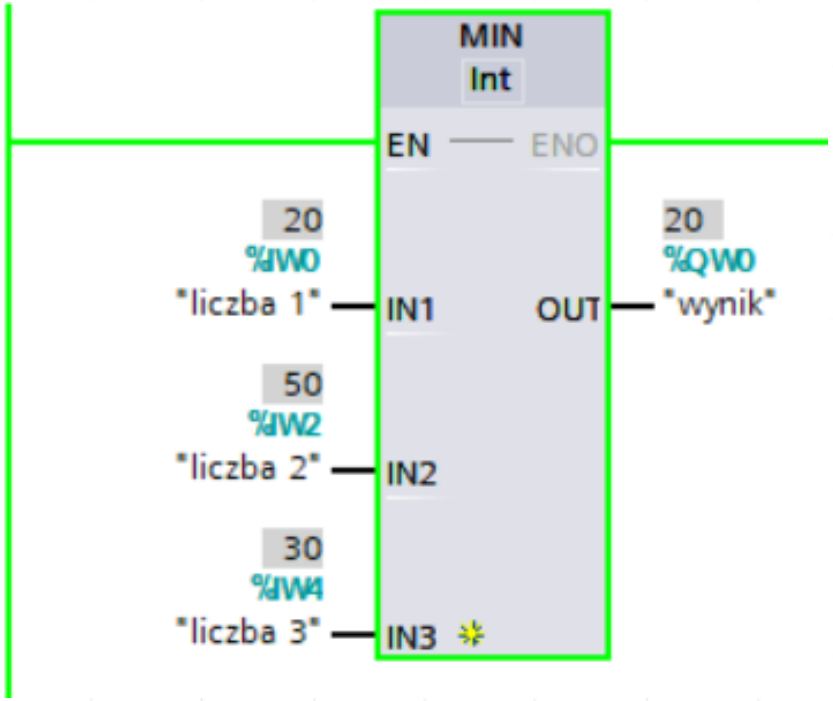
LAD – instrukcja wartość bezwzględna

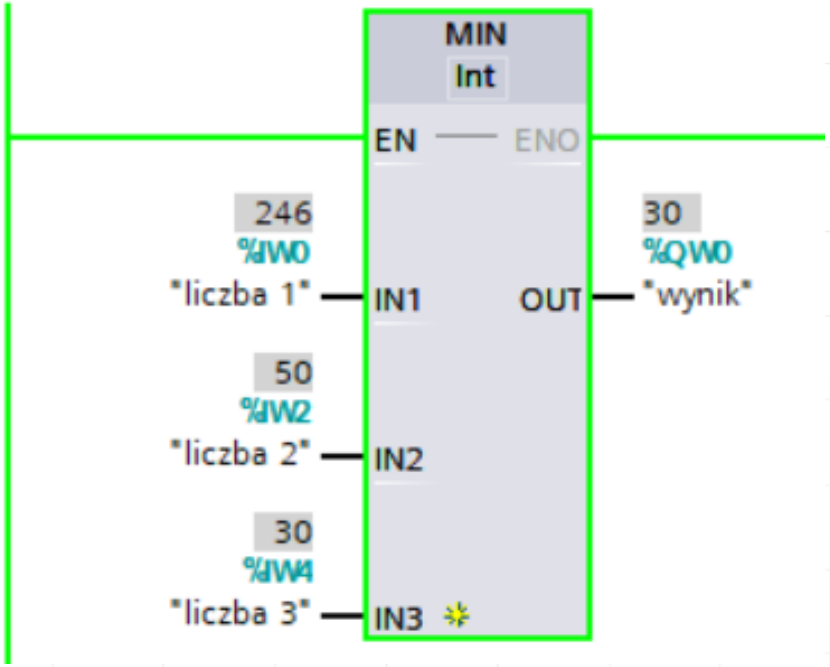
Operacja absolute ABS – podaje wartość bezwzględną z zadanej liczby.



LAD – instrukcja minimum

Operacja minimum MIN – znajduje wartość najmniejszą spośród zadanych liczb.



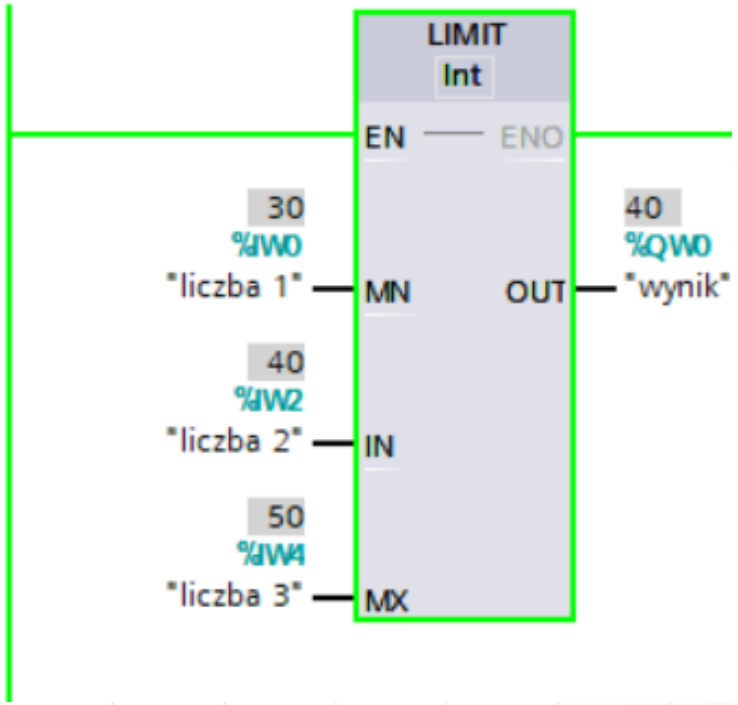


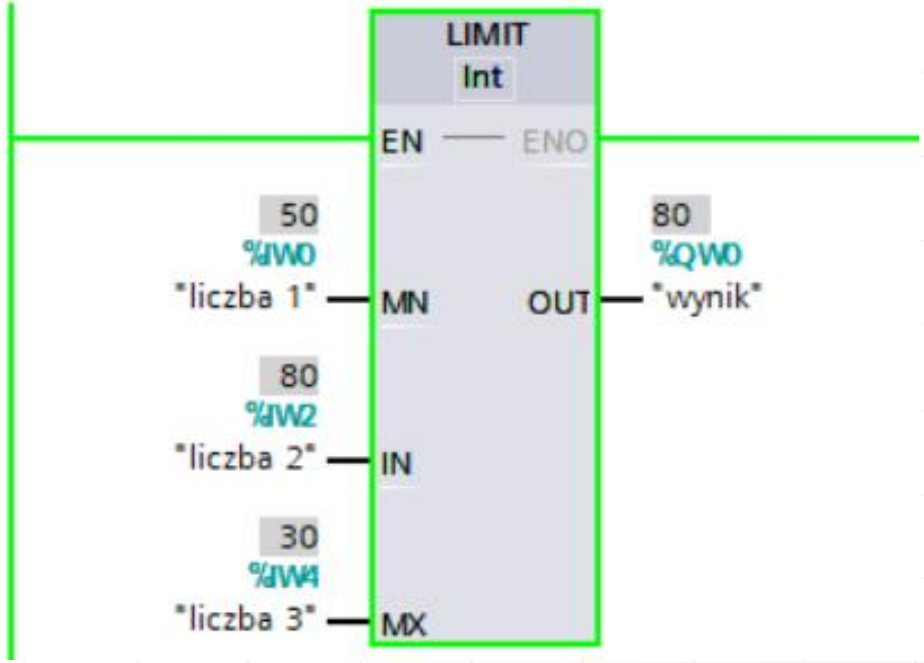
LAD – instrukcja maksimum

Operacja maksimum MAX – znajduje wartość największą spośród zadanych liczb.

LAD – instrukcja limit

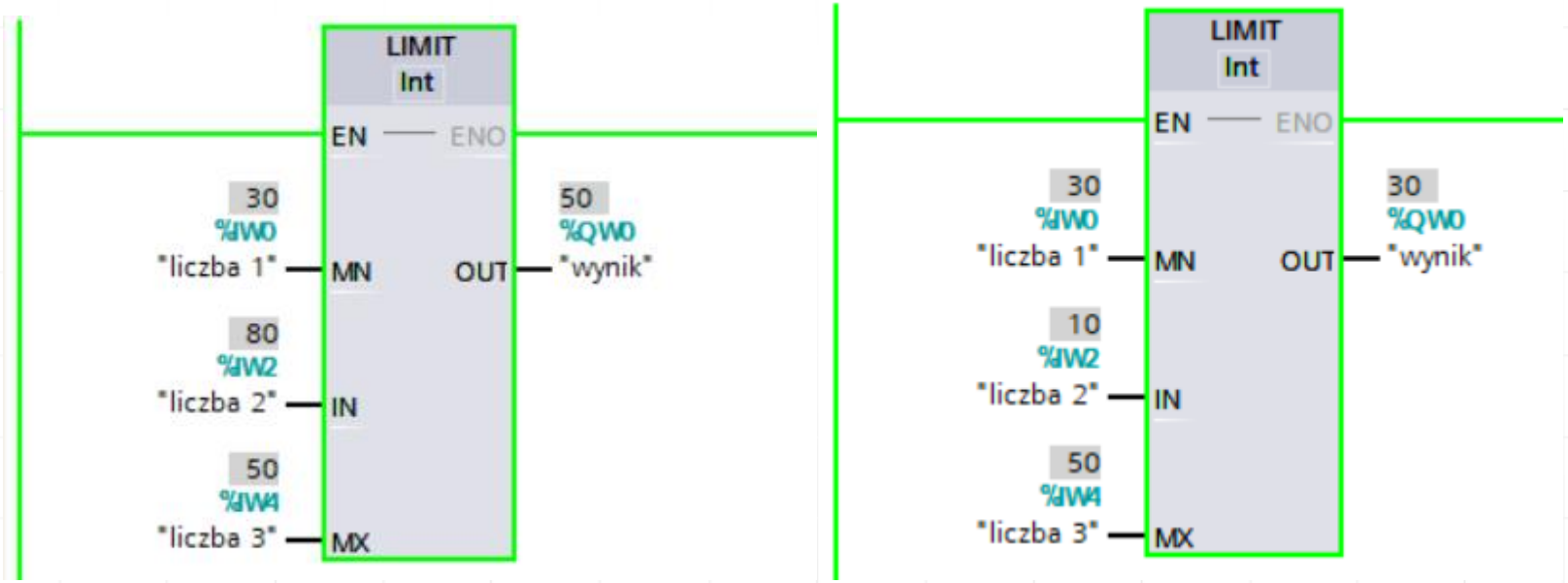
Operacja LIMIT – sprawdza, czy liczba zadana jako IN znajduje się w przedziale MIN<=IN<=MAX, jeżeli tak, to liczba IN zostaje przypisana do wartości wyjściowej OUT. Program działa tylko wtedy, kiedy MIN<=MAX.





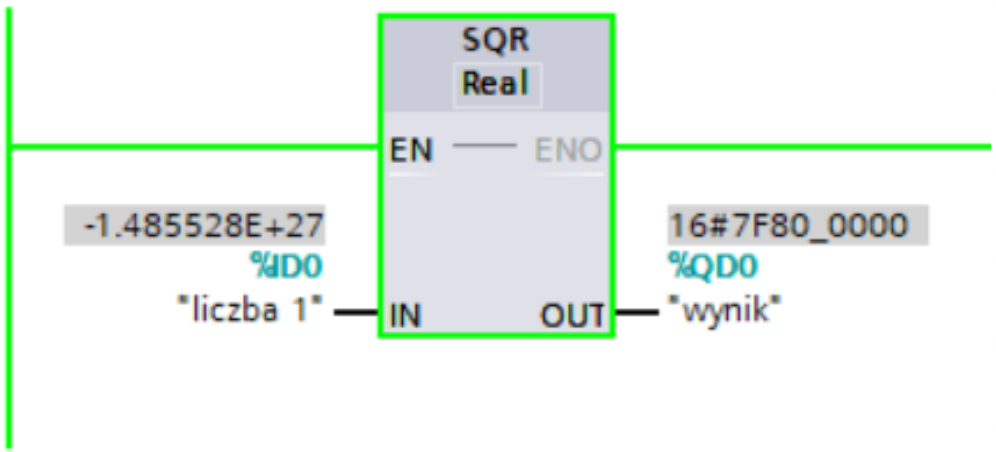
LAD – instrukcja limit

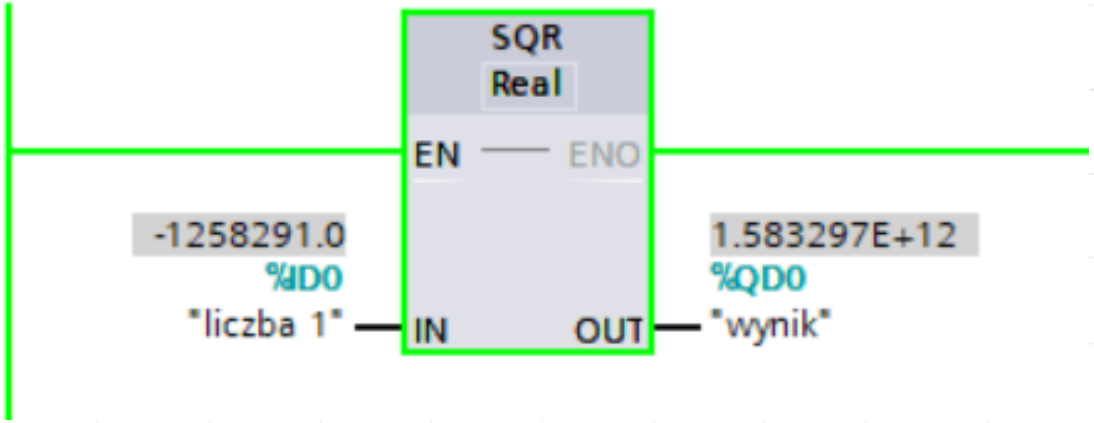
Operacja LIMIT – sprawdza, czy liczba zadana jako IN znajduje się w przedziale MIN<=IN<=MAX, jeżeli IN>MAX to MAX zastaje przypisana do wartości wyjściowej OUT, a jeżeli IN<MIN to MIN zostaje przypisana do wartości wyjściowej OUT.



LAD – instrukcja potęga kwadratowa

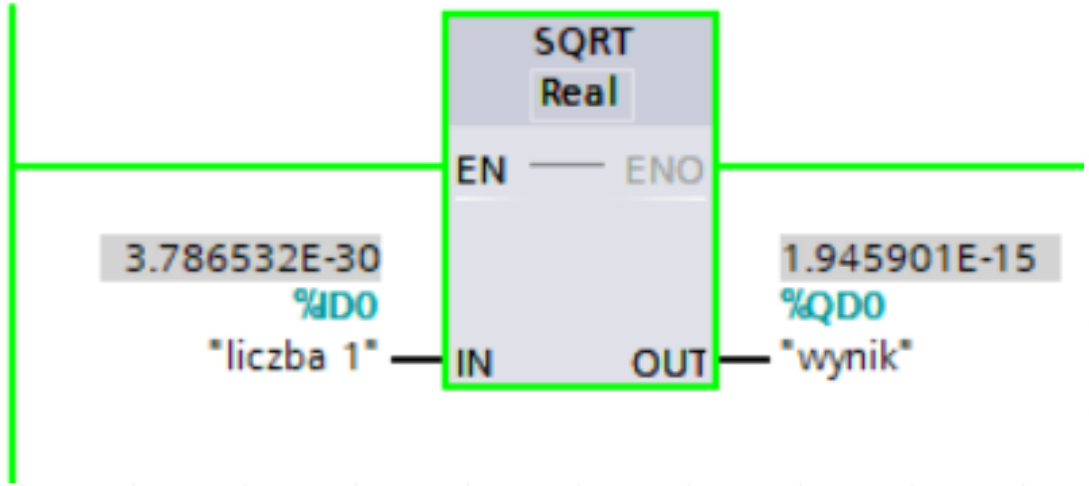
Operacja potęga kwadratowa SQR (tylko liczby rzeczywiste), podnosi zadaną liczbę do potęgi kwadratowej.

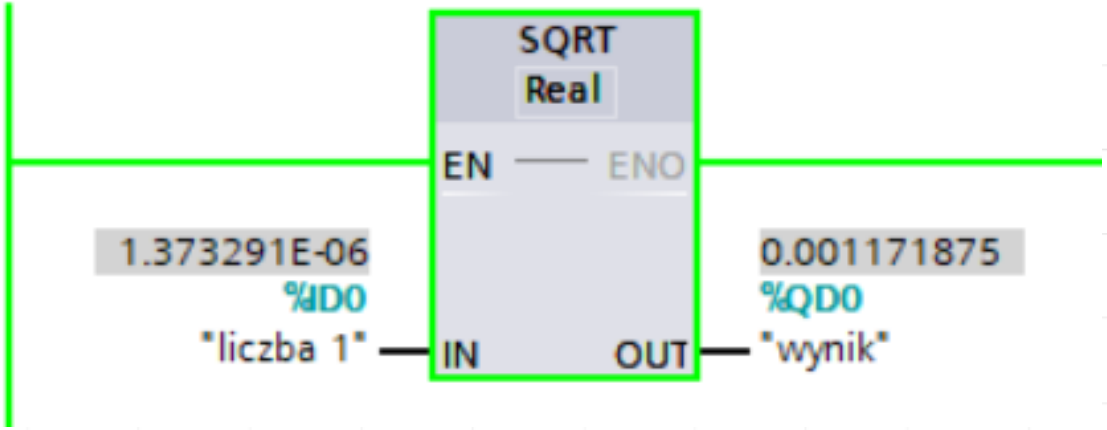




LAD – instrukcja pierwiastek kwadratowy

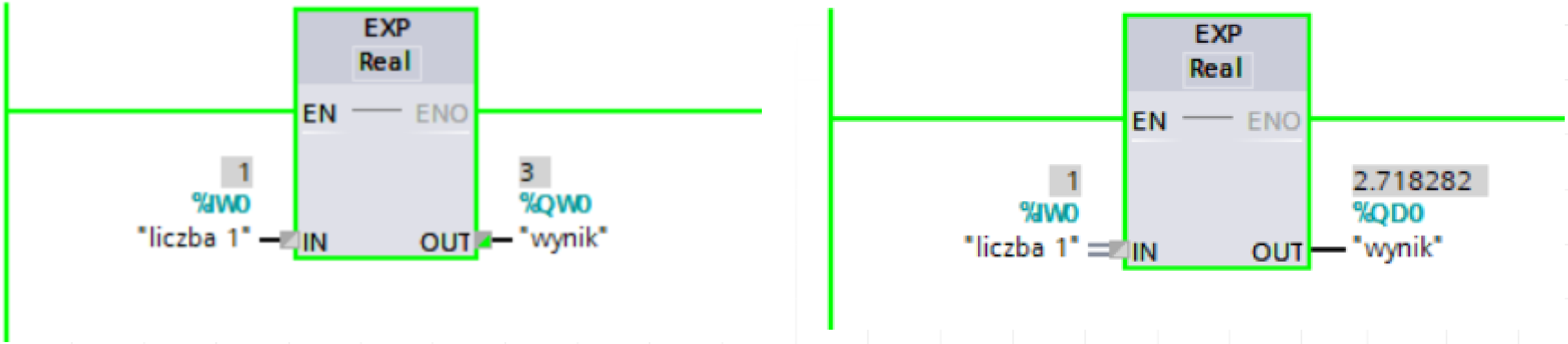
Operacja pierwiastek kwadratowy SQRT (tylko liczby rzeczywiste), oblicza pierwiastek kwadratowy z zadanej liczby.





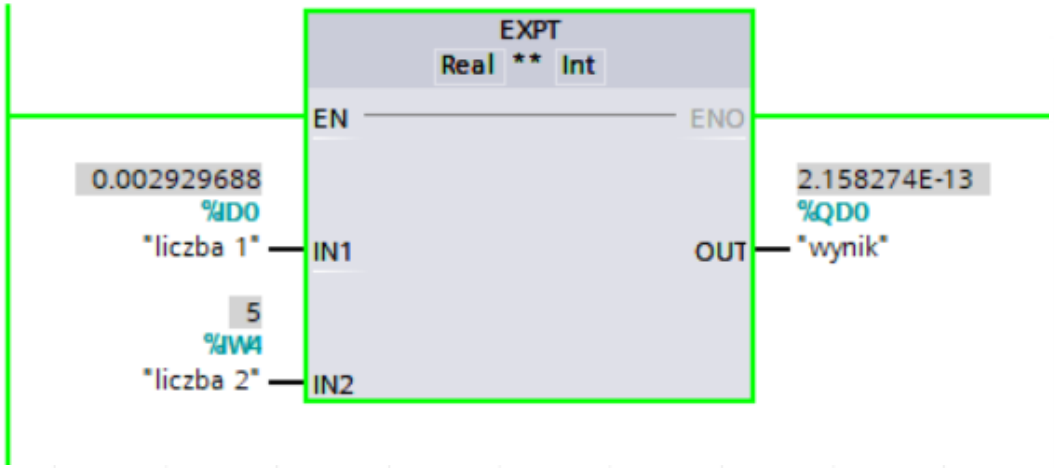
LAD – instrukcja potęgowania liczby e

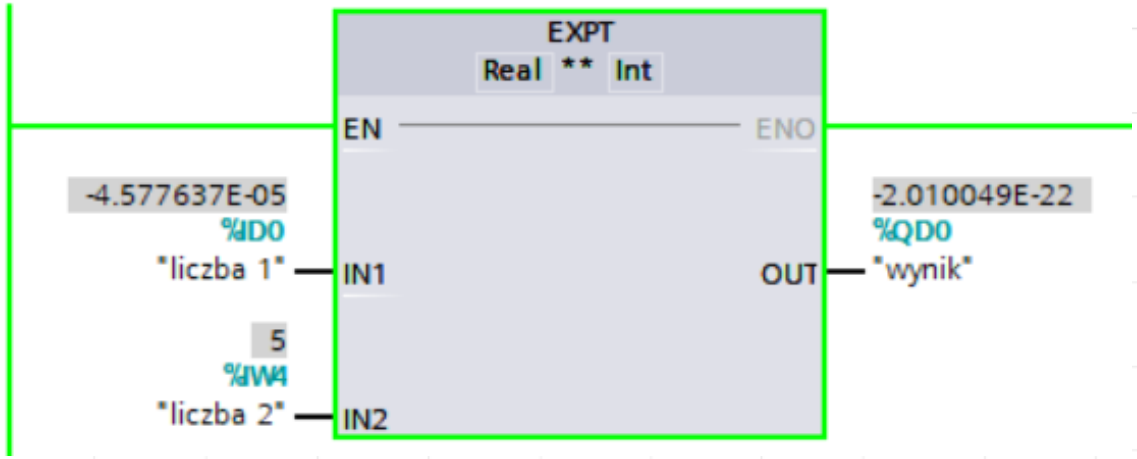
Operacja EXP potęguje liczby e do zadanej potęgi. $OUT=e^{IN}$, gdzie IN – wykładnik potęgi (liczby całkowite i rzeczywiste). Po lewej stronie pokazano wynik działania na liczbach całkowitych, po prawej na rzeczywistych.



LAD – instrukcja podnoszenia do dowolnej potęgi

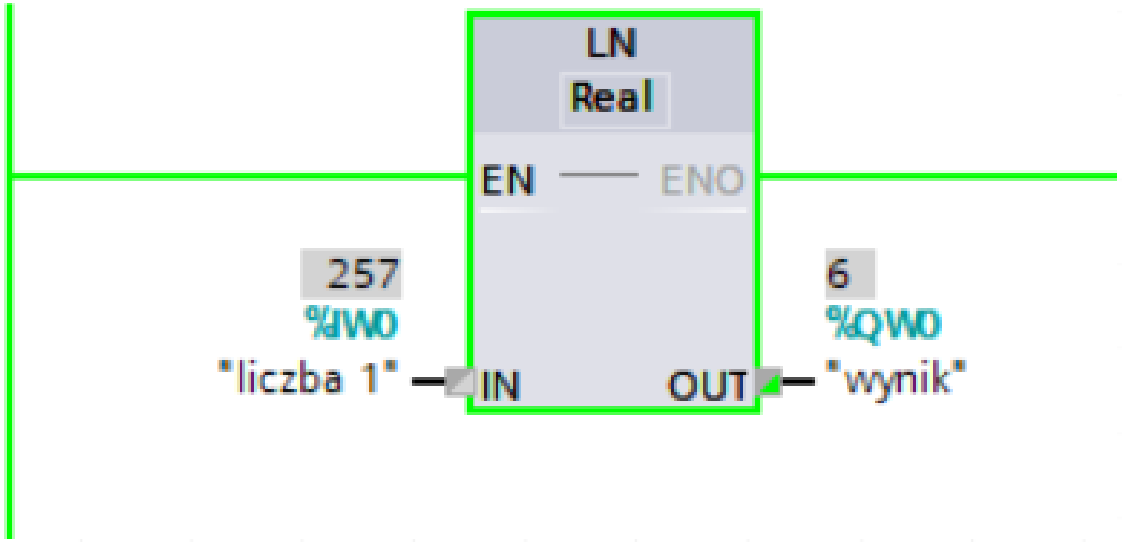
Operacja EXPT podnosi liczbę do dowolnej potęgi $OUT=IN1^{IN2}$, gdzie IN1 – podstawa potęgi (tylko liczby rzeczywiste), IN2 – wykładnik potęgi (liczby całkowite i rzeczywiste).





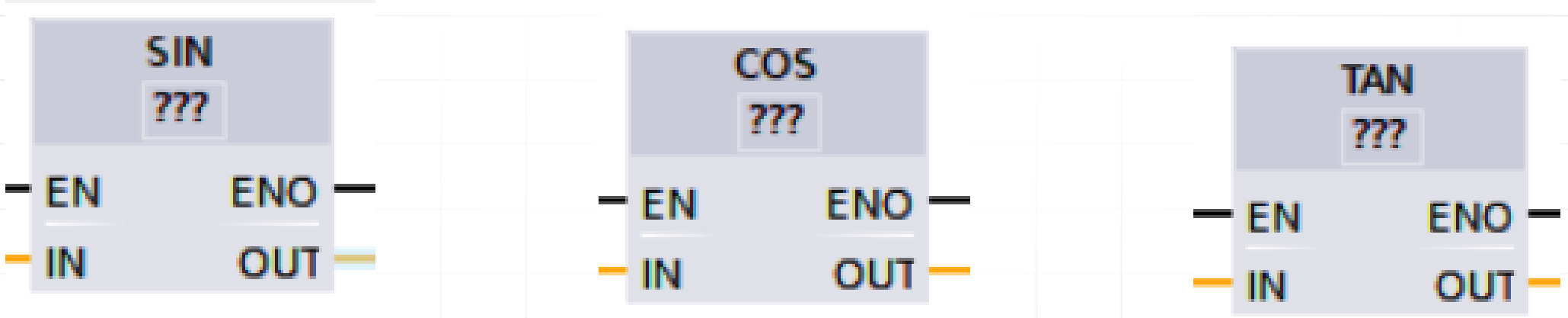
LAD – instrukcja logarytm naturalny

Operacja potęguje oblicza logarytm naturalny z zadanej liczby ln(IN) (liczby całkowite i rzeczywiste).



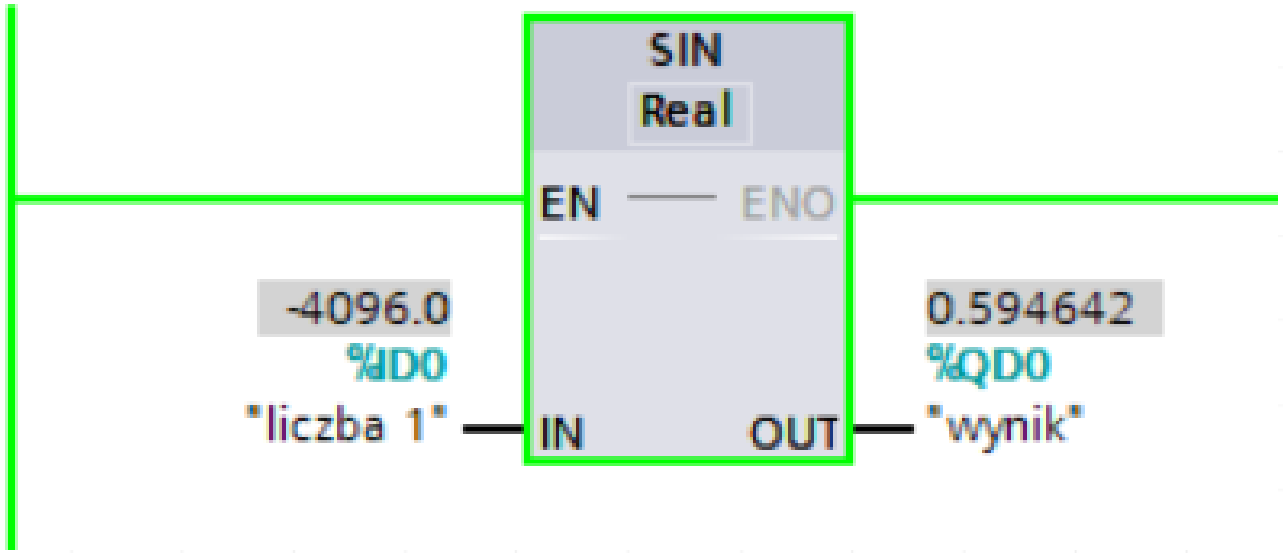
LAD – funkcje trygonometryczne

- operacja **SIN** (tylko liczby rzeczywiste)
- argument funkcji IN podawany w radianach
- dostępne także: **COS, TAN, ASIN, ACOS, ATAN**



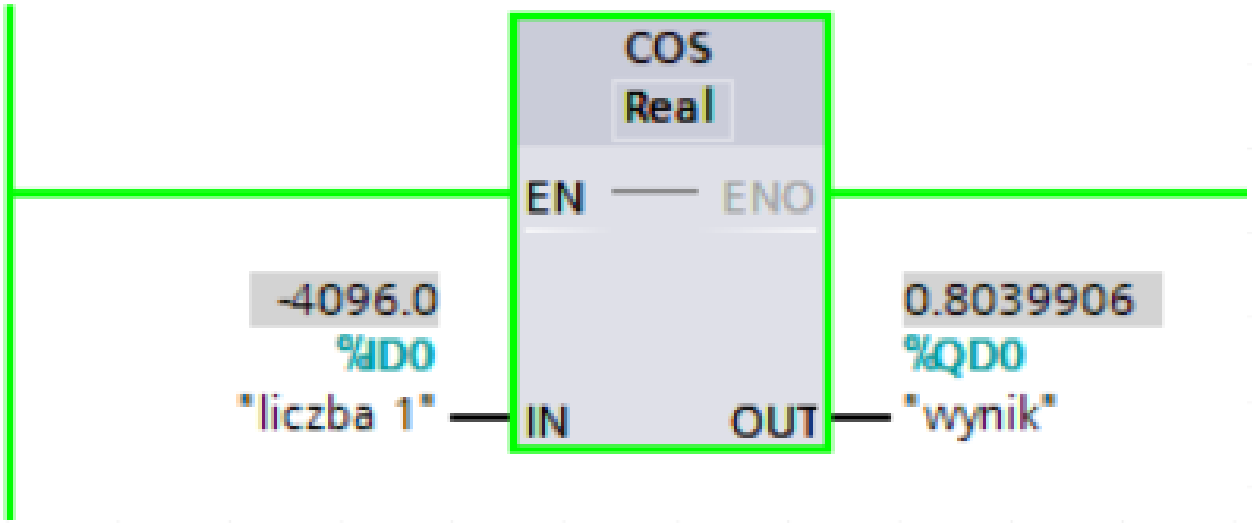
LAD – funkcje sinus

Operacja SIN (tylko liczby rzeczywiste) oddaje wartość sinus zadanej liczby, gdzie argument funkcji IN podawany jest w radianach.



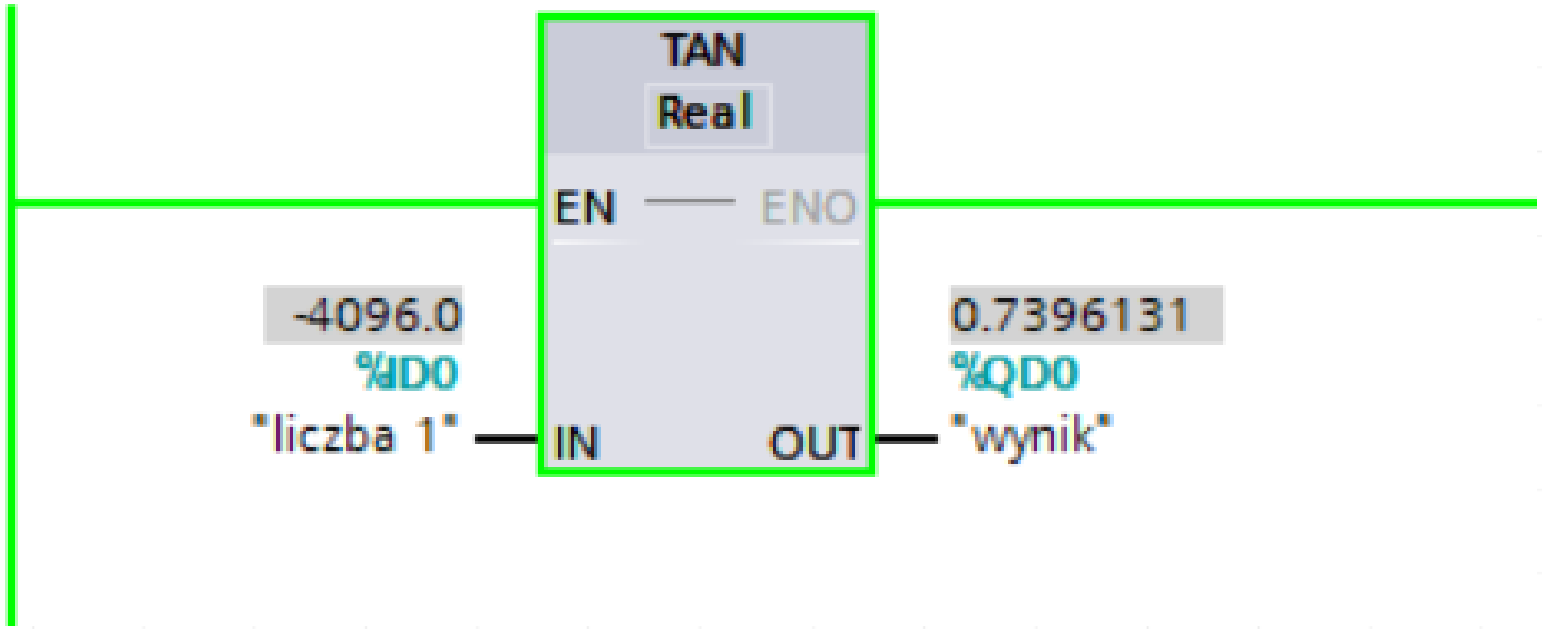
LAD – funkcje cosinus

Operacja cos (tylko liczby rzeczywiste) oddaje wartość cosinus zadanej liczby, gdzie argument funkcji IN podawany jest w radianach.



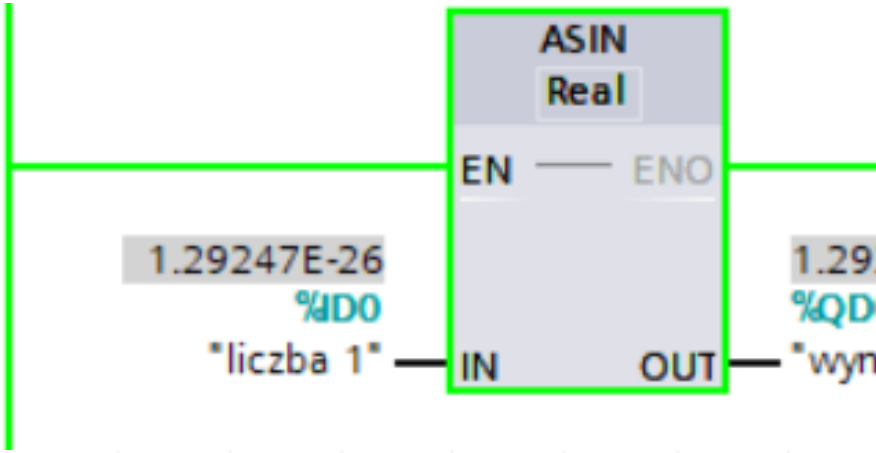
LAD – funkcje tangens

Operacja tan (tylko liczby rzeczywiste) oddaje wartość tangens zadanej liczby, gdzie argument funkcji IN podawany jest w radianach.



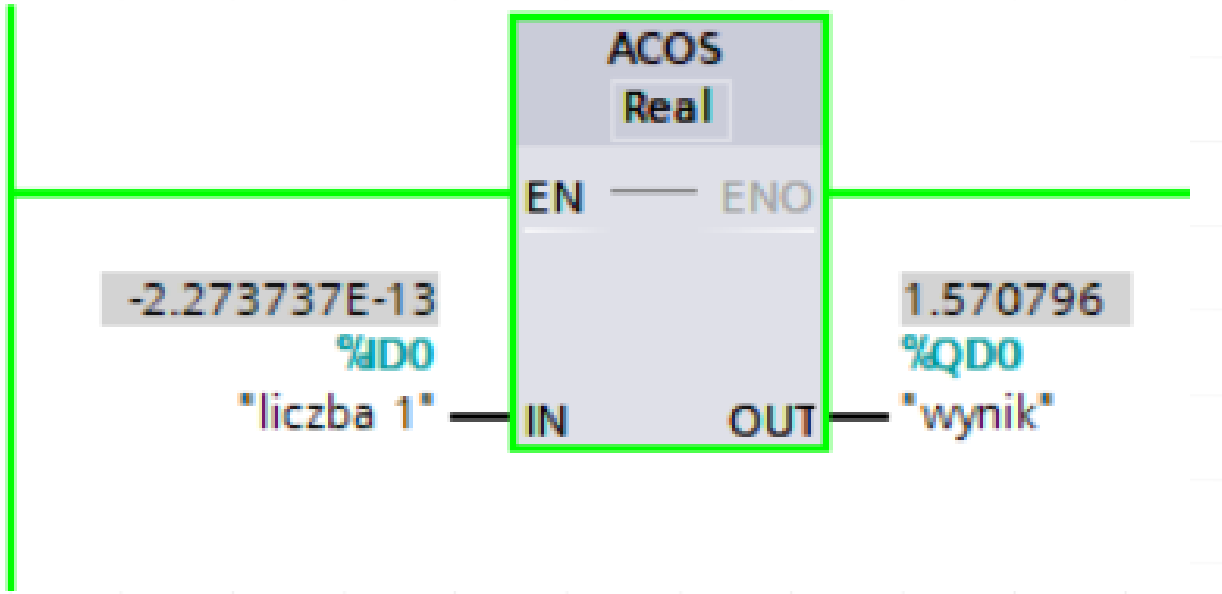
LAD – funkcje asinus

Operacja asin (tylko liczby rzeczywiste) oddaje wartość asinus zadanej liczby, gdzie argument funkcji IN podawany jest w radianach.



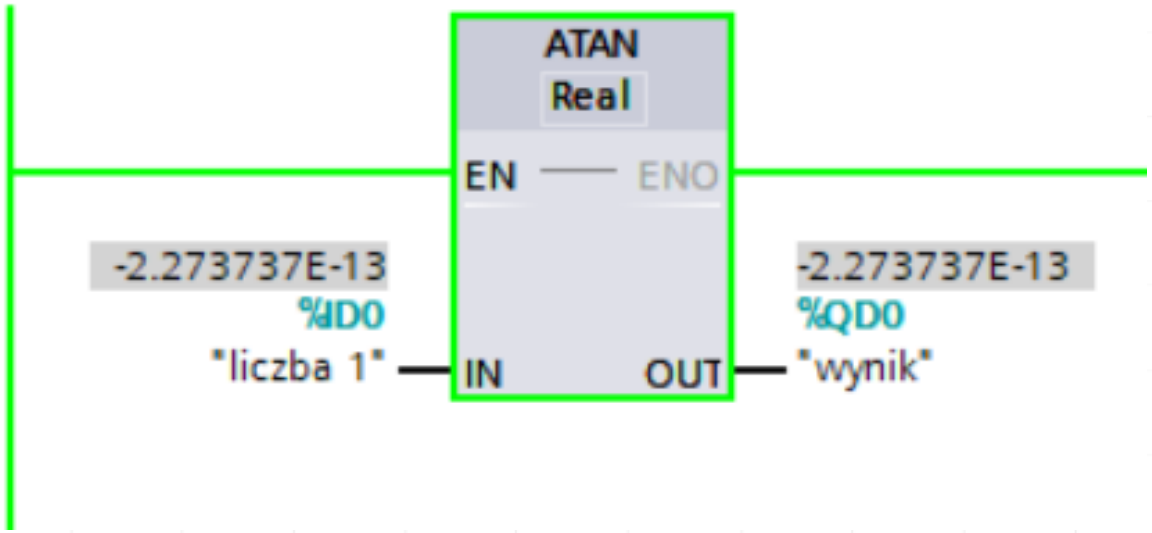
LAD – funkcje acosinus

Operacja acos (tylko liczby rzeczywiste) oddaje wartość acosinus zadanej liczby, gdzie argument funkcji IN podawany jest w radianach.



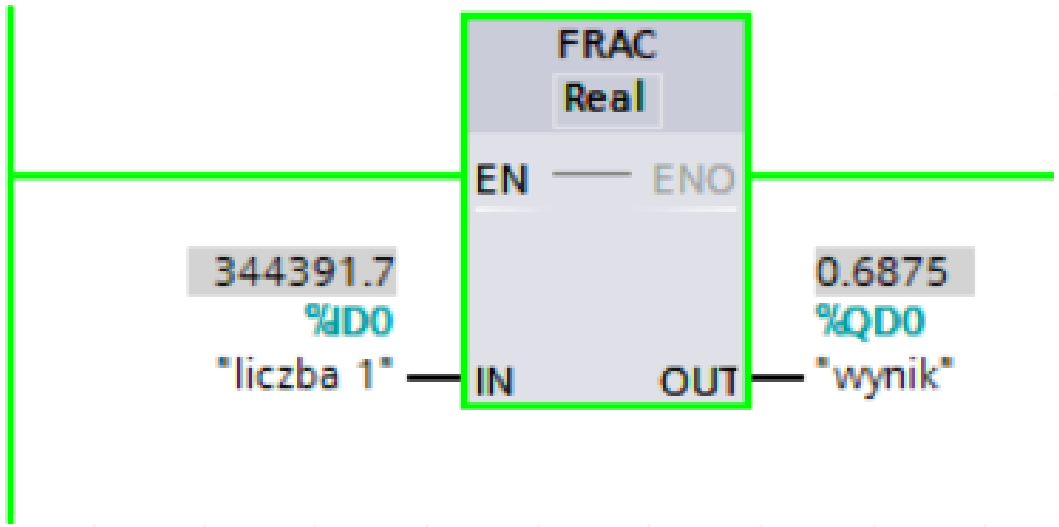
LAD – funkcje atangens

Operacja atan (tylko liczby rzeczywiste) oddaje wartość atangens zadanej liczby, gdzie argument funkcji IN podawany jest w radianach.



LAD – funkcje fractions

Operacja fractions (tylko liczby rzeczywiste) oddaje wartość po przecinku zadanej liczby.



LAD – wybrane funkcje

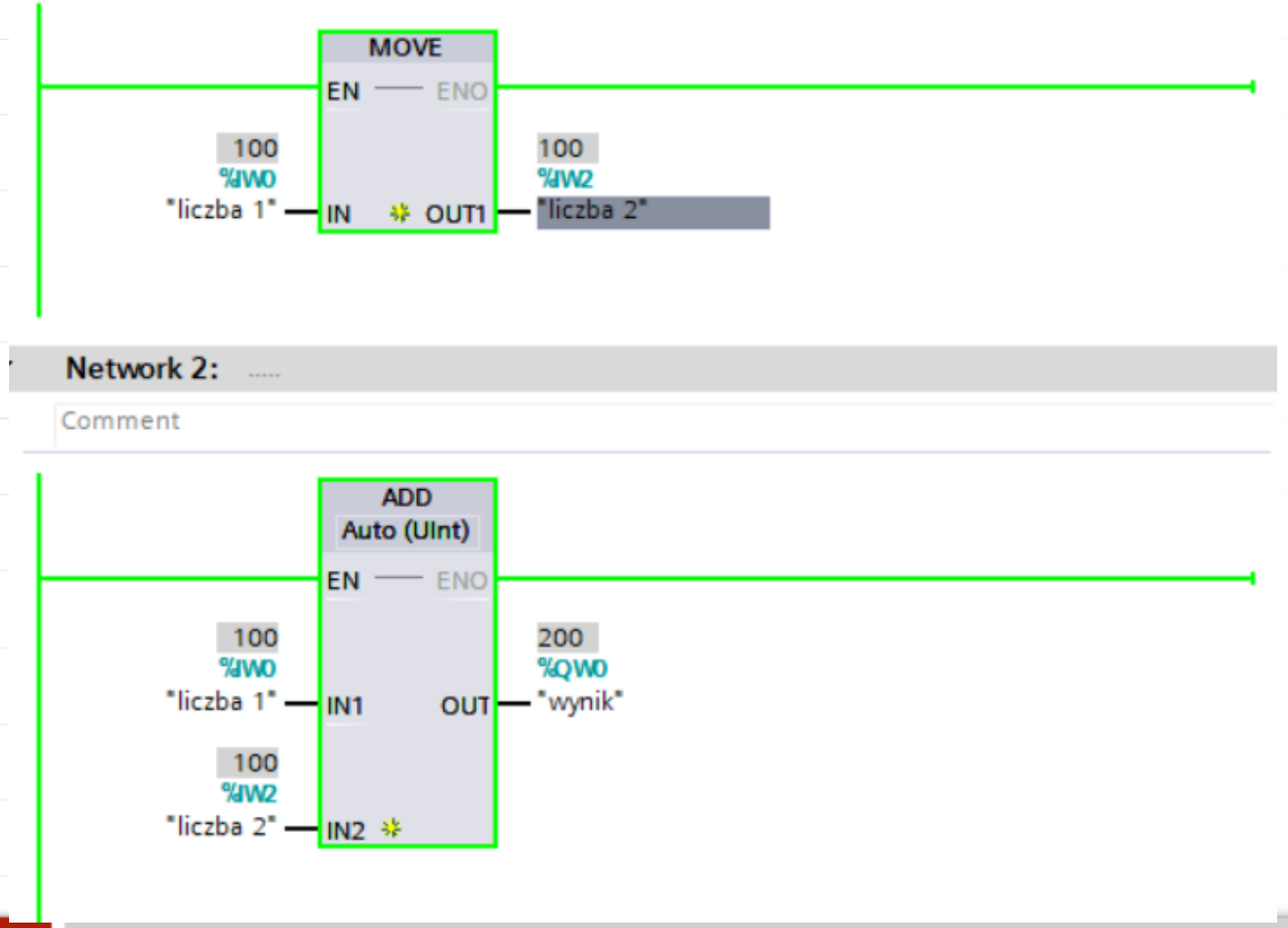
LAD – instrukcja MOVE

Instrukcja MOVE jest stosowana do kopiowania elementów danych do miejsca pamięci o nowym adresie oraz konwersji jednego typu danych na inny. Dane wejściowe instrukcji MOVE nie ulegają zmianie.

LAD – instrukcja MOVE

MOVE: kopiuje elementy danych pamiętane pod określonym adresem w miejsce pamięci o nowym adresie. Aby dodać kolejne wyjście należy nacisnąć ikonę obok parametru OUT1.

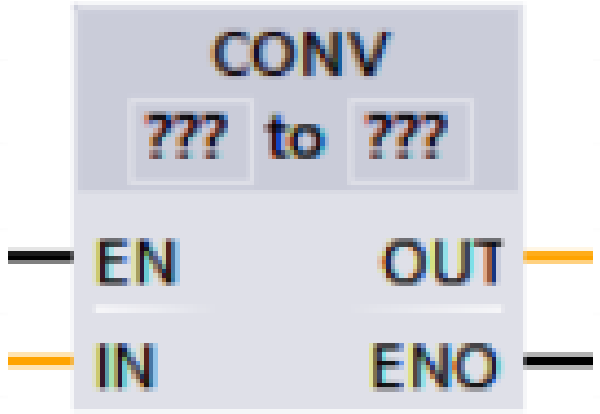
LAD – instrukcja MOVE



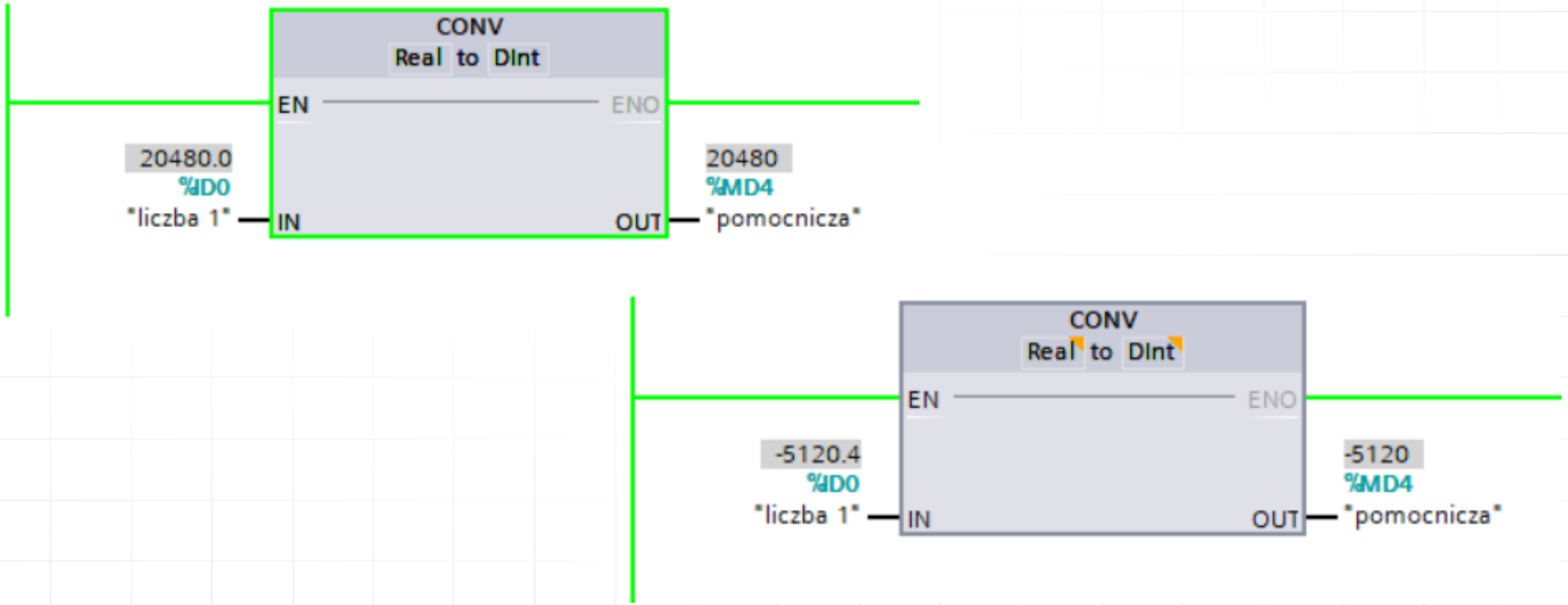
Po lewej stronie pokazano działanie programu w którym wartość „liczba 1” jest przepisywana na „liczba 2”. W funkcji Add „liczba 1” i „liczba 2” są do siebie dodawane i muszą mieć ten sam typ zmiennej.

LAD – instrukcja konwersji

Konwertuje elementy danych z jednego typu na inny.
Kliknięcie poniżej nazwy bloku pozwala wybrać z rozwijanego menu typ danych.
Po dokonaniu wyboru typu danych źródłowych (*convert from*) jest rozwijana lista możliwych konwersji (*convert to*).



LAD – instrukcja konwersji

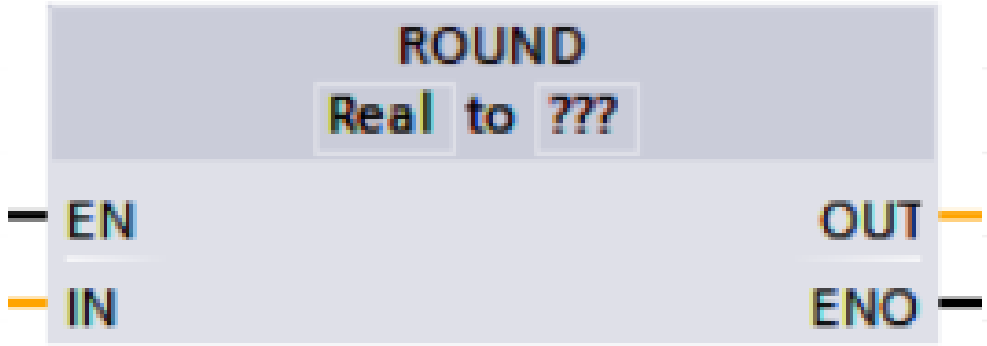


LAD – instrukcja zaokrąglania

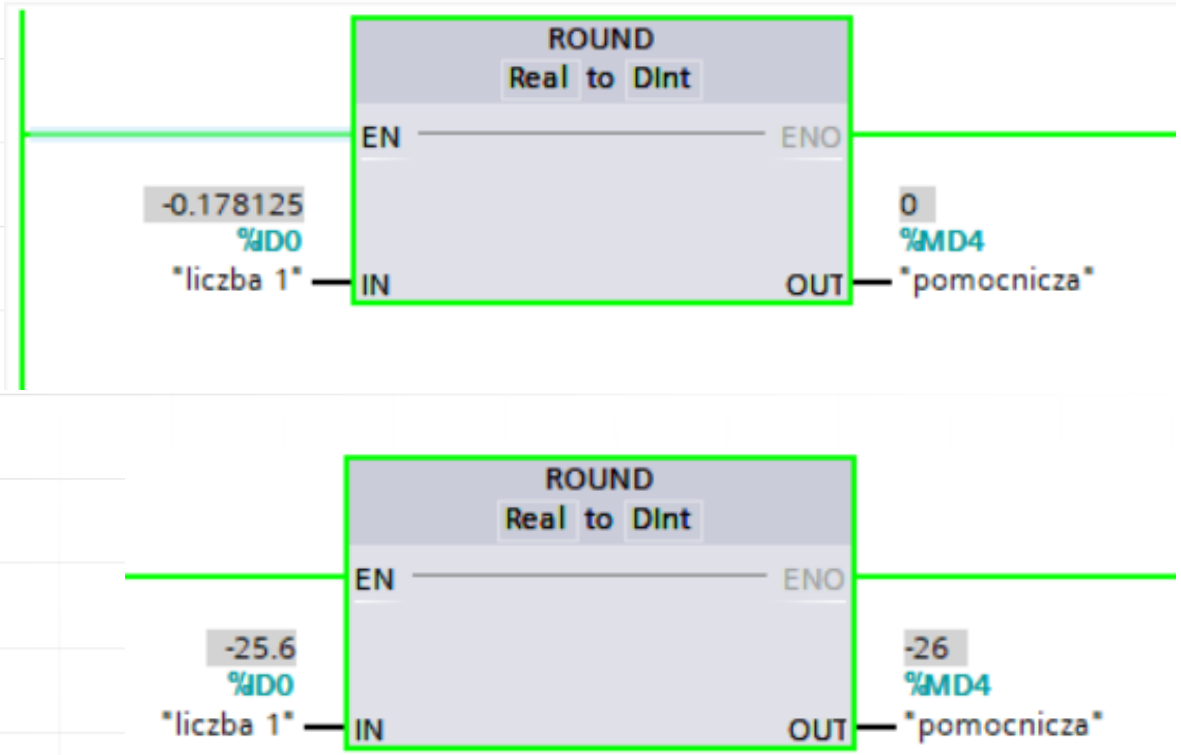
ROUND zamienia liczbę rzeczywistą (Real lub LReal) na całkowitą. Część ułamkowa liczby rzeczywistej jest zaokrąglana do najbliższej wartości całkowitej (IEEE – round to nearest). Jeśli wartość liczby jest dokładnie pomiędzy dwoma całkowitymi (np. 10,5), to liczba jest zaokrąglana do najbliższej całkowitej liczby parzystej. Przykładowo:

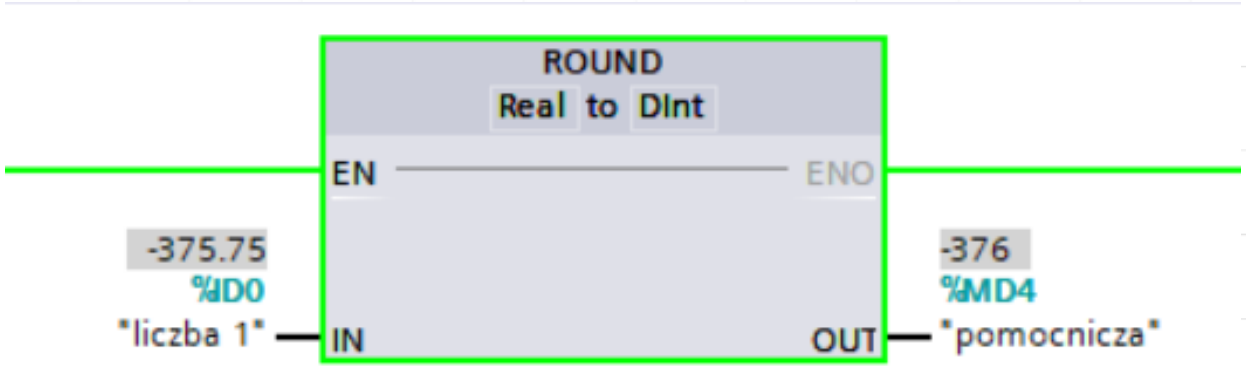
- ROUND (10.5) = 10
- ROUND (11.5) = 12

Dla LAD/FBD, w bloku instrukcji należy kliknąć „???”, aby wybrać typ danych wyjściowych, na przykład, „DInt”.



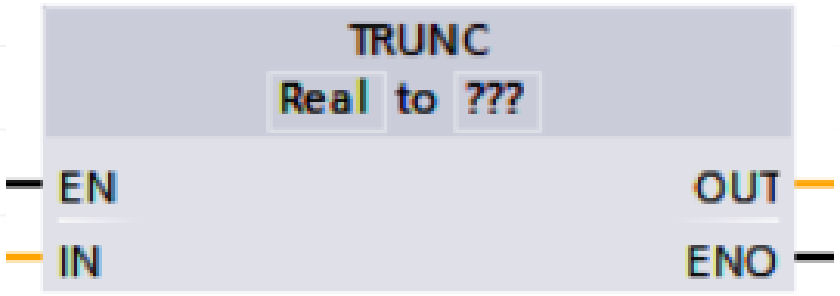
LAD – instrukcja zaokrąglania

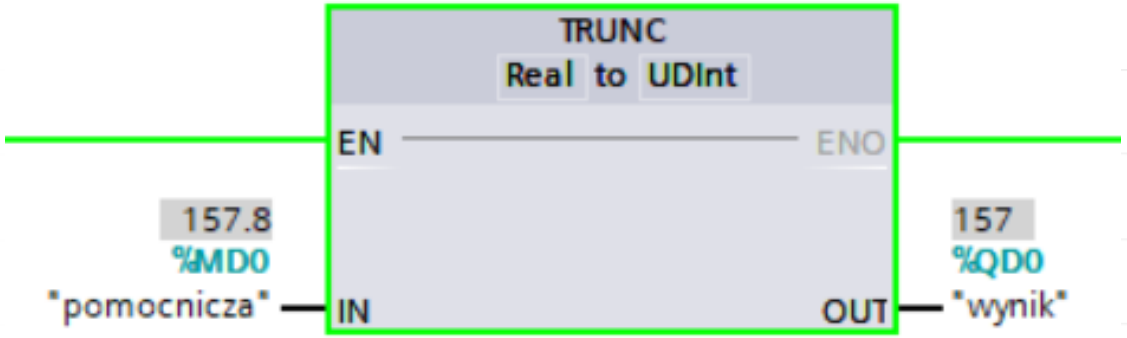




LAD – instrukcja skrócenia

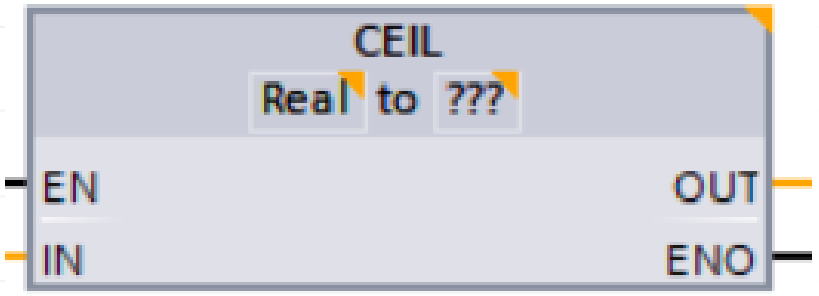
TRUNC zamienia liczbę rzeczywistą (Real lub LReal) na całkowitą. Część ułamkowa liczby rzeczywistej jest skrócona do zera.(IEEE – round to zero).

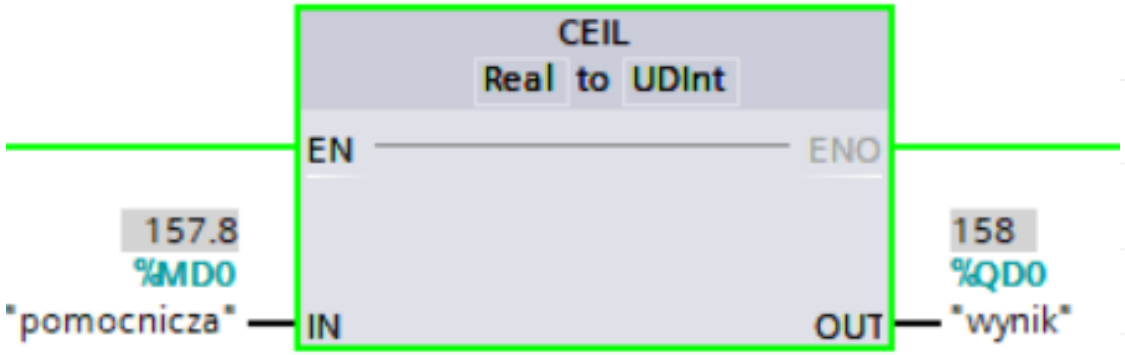




LAD – znajdowanie najbliższych liczb całkowitych

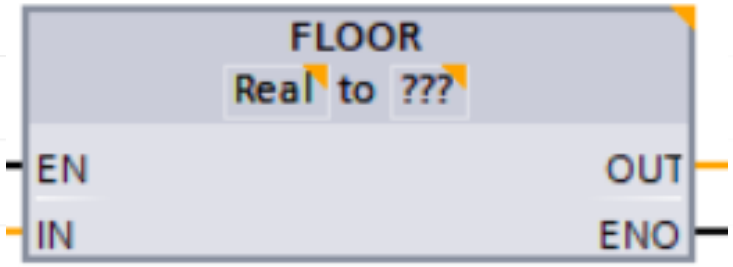
CEIL zamienia liczbę rzeczywistą (Real lub LReal) na najmniejszą liczbę całkowitą większą lub równą liczbie rzeczywistej (IEEE – *round to +infinity*).

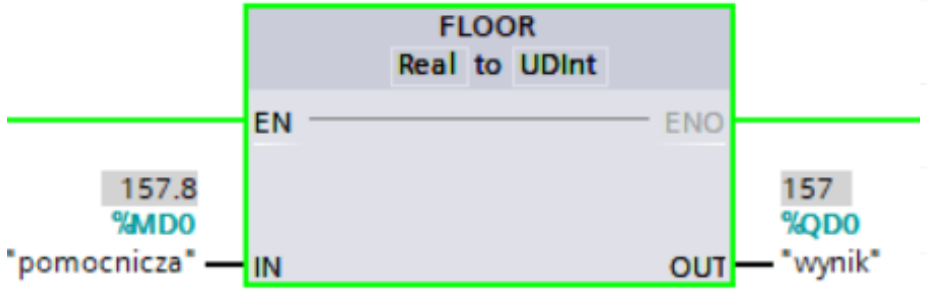




LAD – znajdowanie najbliższych liczb całkowitych

FLOOR zamienia liczbę rzeczywistą (Real lub LReal) na największą liczbę całkowitą mniejszą lub równą liczbie rzeczywistej (IEEE – *round to +infinity*).

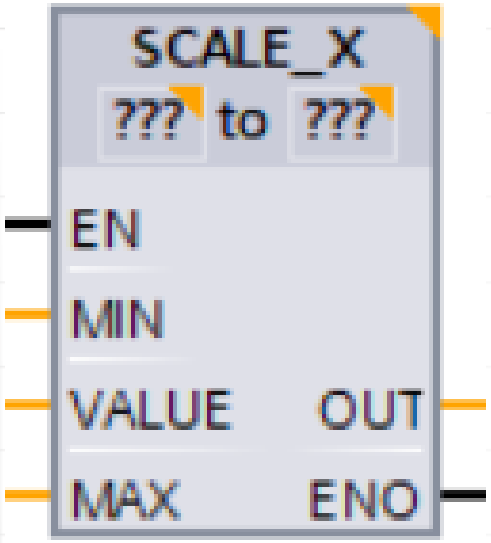


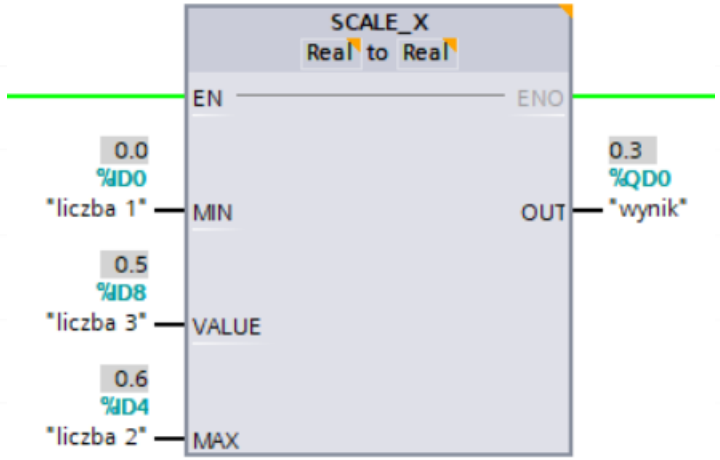


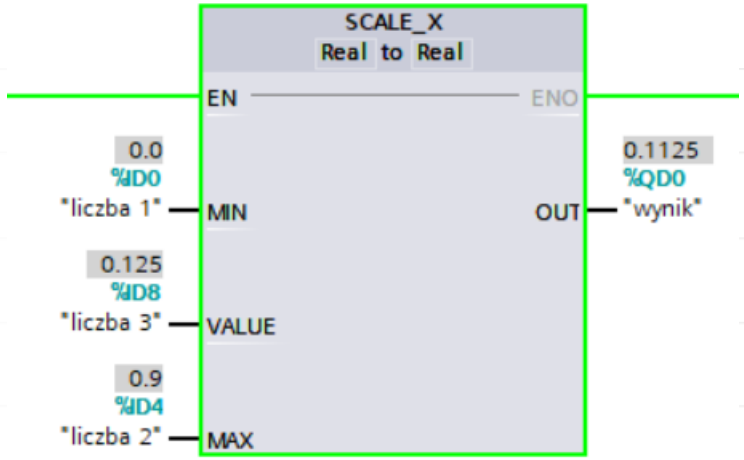
LAD – instrukcje skalowania i normalizacji

SCALE_X skaluje znormalizowany parametr VALUE (gdzie (0.0 <= VALUE <= 1.0)) do typu danej i zakresu wartości wyspecyfikowanych przez parametry MIN i MAX:

OUT = VALUE * (MAX – MIN) + MIN

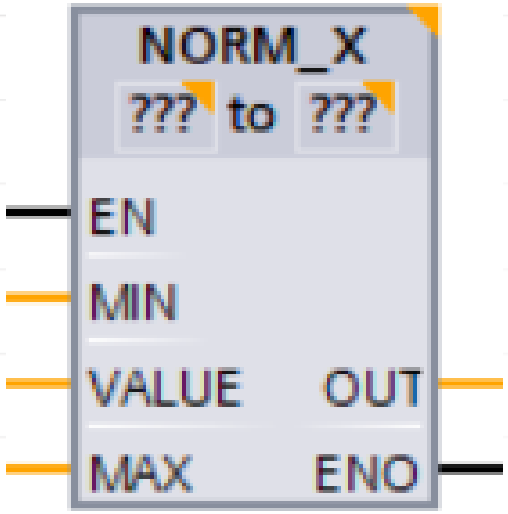


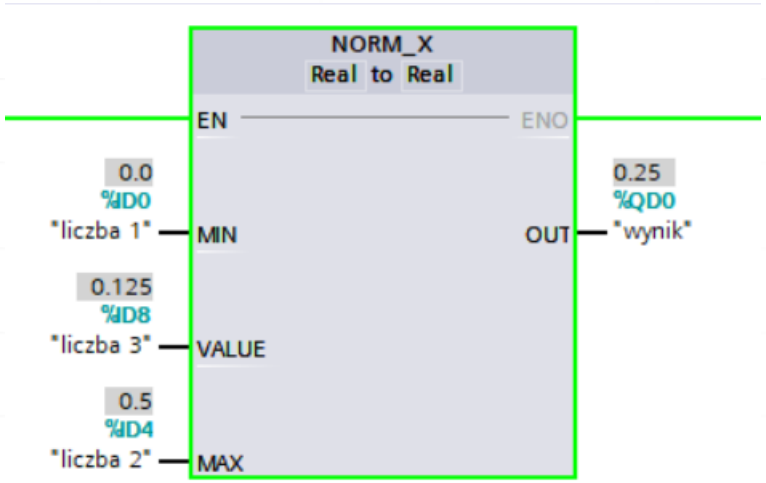


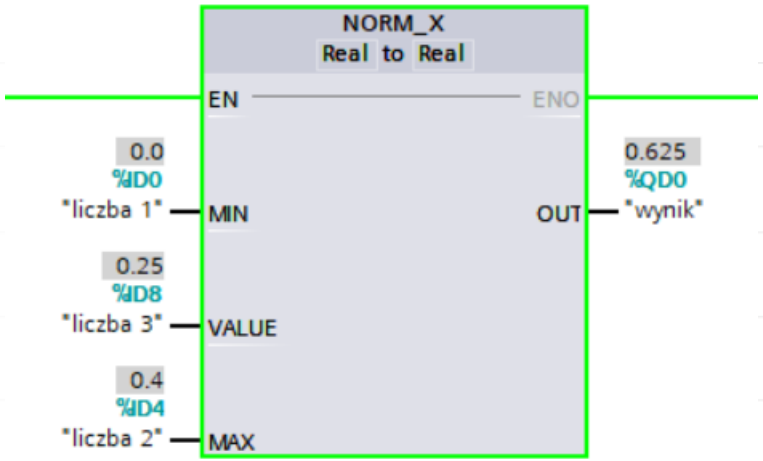


LAD – instrukcje skalowania i normalizacji

NORM_X normalizuje parametr VALUE wewnątrz zakresu wartości wyspecyfikowanych przez parametry MIN i MAX:
OUT = (VALUE – MIN)/(MAX – MIN),
gdzie (0.0 <= OUT <= 1.0).

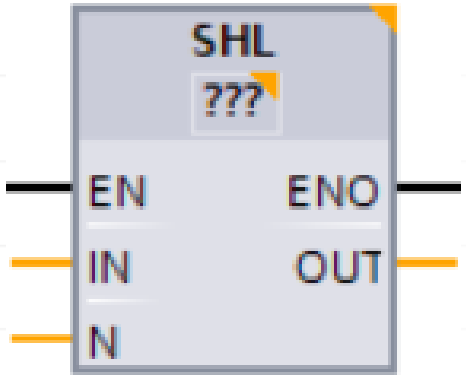


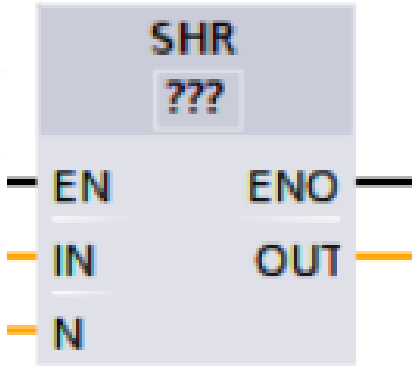


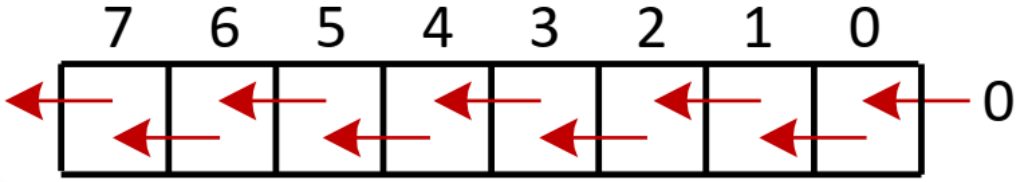


LAD – Operacje przesunięcia

- **SHL – przesunięcie w lewo**
- **SHR – przesunięcie w prawo**
- **IN** – rejestr podlegający przesunięciu (rejestry bitowe i liczby całkowite)
- **N** – liczba pozycji, o którą rejestr ulegnie przesunięciu
- **OUT** – rejestr po operacji przesunięcia

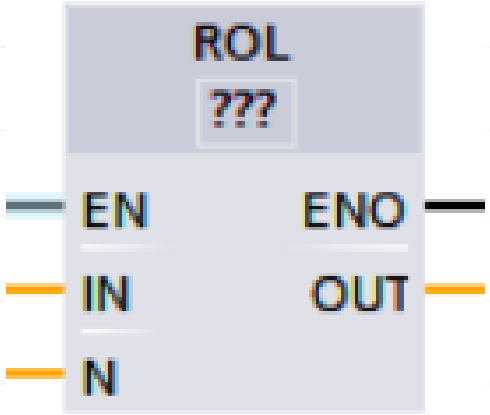


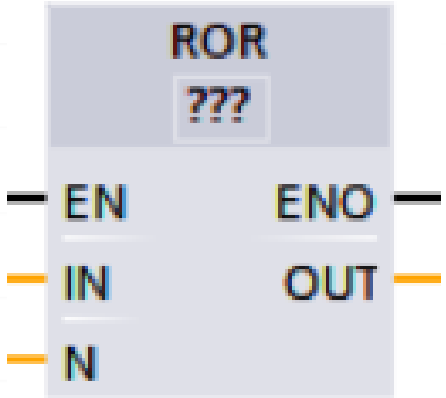


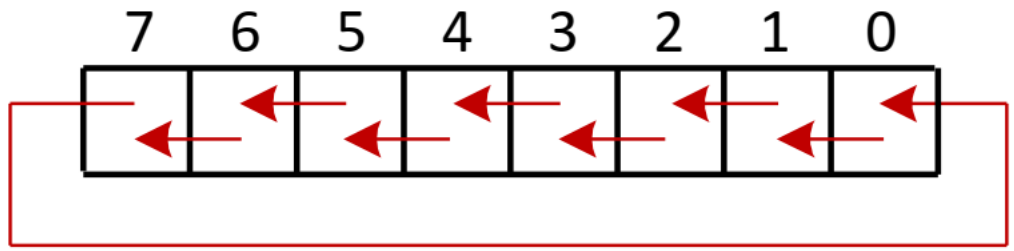


LAD – Operacje rotacji

- **ROL – rotacja w lewo**
- **ROR – rotacja w prawo**
- **IN** – rejestr podlegający rotacji (tylko rejestry bitowe)
- **N** – liczba pozycji, o którą rejestr ulegnie rotowaniu
- **OUT** – rejestr po operacji rotacji

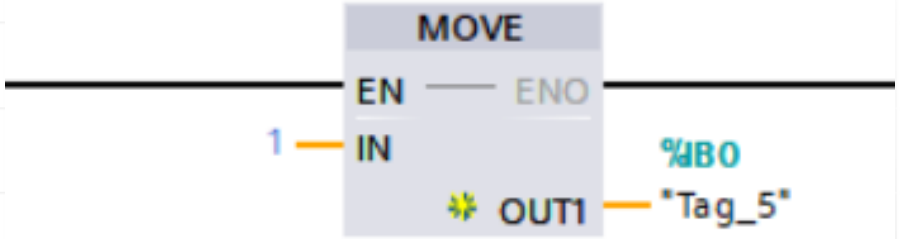


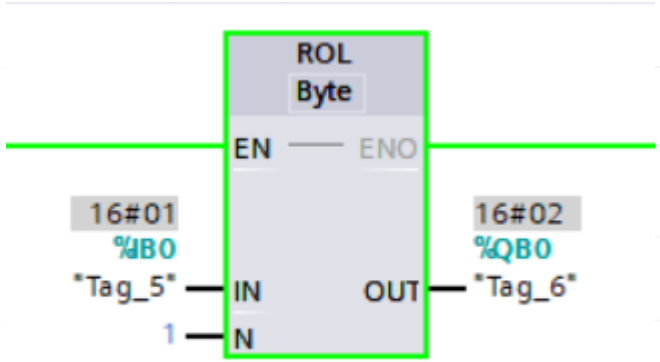




LAD – Operacje rotacji

- **ROL – rotacja w lewo**
- **ROR – rotacja w prawo**
- **IN** – rejestr podlegający rotacji (tylko rejestry bitowe)
- **N** – liczba pozycji, o którą rejestr ulegnie rotowaniu
- **OUT** – rejestr po operacji rotacji



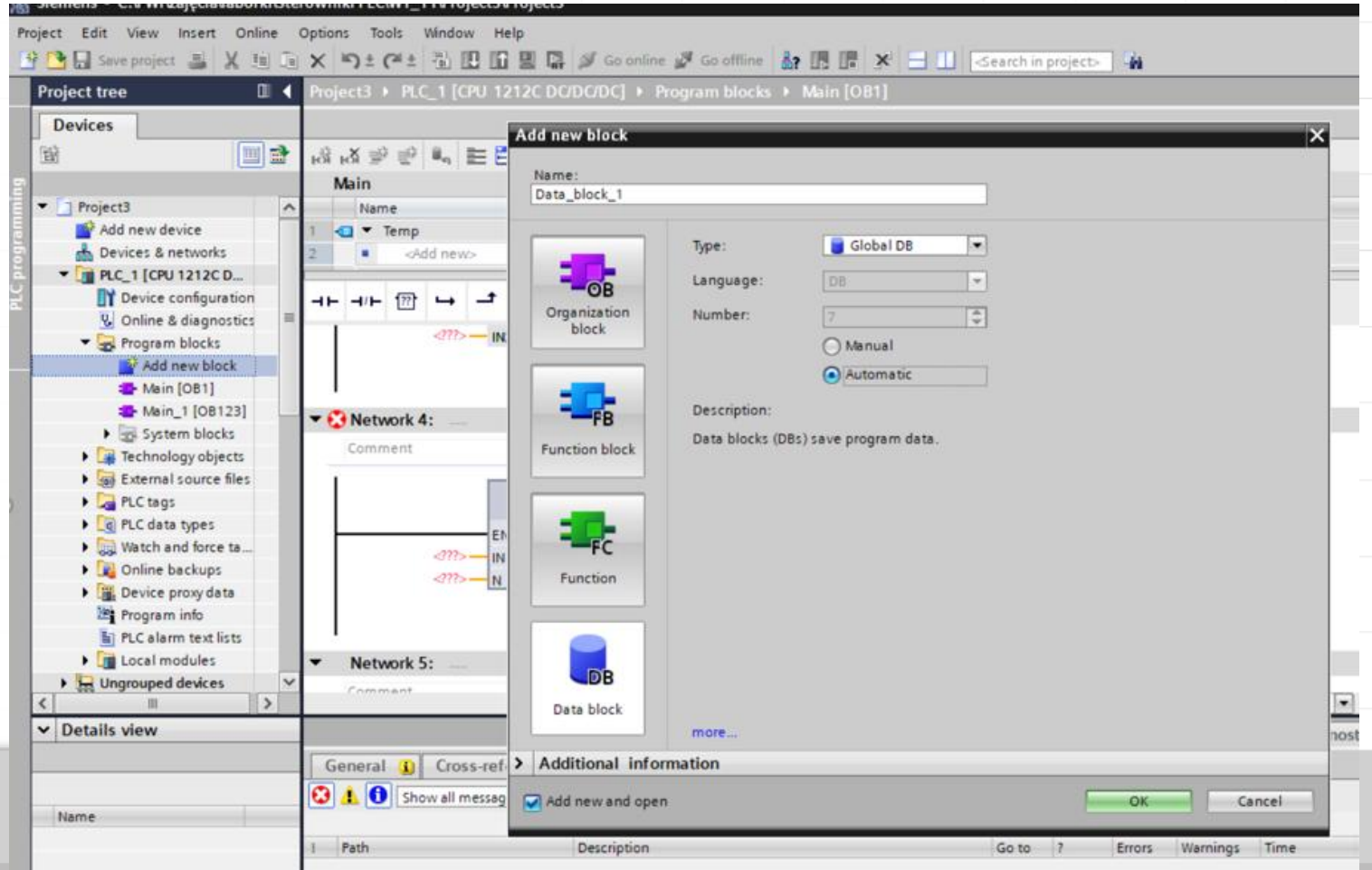


Deklaracja zmiennych – bloki danych

Deklaracja bloku danych:

Project tree → Program blocks → Add new block → Data block

- Klasyczny blok danych jest typu Global DB (zmienne mają zakres globalny).
- Proponowana nazwa bloku to: ***Data_block_****[kolejny_numer_bloku]*. Nazwa bloku powinna być zgodna z jego przeznaczeniem.



Deklaracja zmiennych – bloki danych

- Deklarujemy tylko nazwę zmiennej i jej typ.
- Nie deklarujemy adresu zmiennej!
- Możemy przyporządkować zmiennej wartość początkową.
- Zmienna jest dostępna w programie jako:

Data_block_*[numer_bloku].nazwa_zmiennej*

np. gdy blok danych ma nazwę ***zbiornik***, a zmienna ma nazwę *zawor_wody*:

zbiornik*.zawor_wody*

- Zazwyczaj nazwę bloku lub zmiennych dobiera się tak, aby zbędny był komentarz, jednak w przypadku niejednoznacznych nazw zaleca się dodanie komentarza.

Deklaracja zmiennych – tablice

- Tablica jednowymiarowa deklarowana jako typ danych: ***Array****[lo..hi] of type*

gdzie:

lo – dolny zakres tablicy (minimalna wartość –32 768),

hi – górny zakres tablicy (maksymalna wartość +32 767),

type – typ danych elementów tablicy (poza typem *Array*),

- Przykład: tablica o nazwie ***pomiar*** *Array[0..3] of Real* zawiera 4 elementy typu Real: *pomiar[0]*, *pomiar[1]*, *pomiar[2]*, *pomiar[3]*,
- Dostęp do elementów tablicy poprzez *indeks*, np. dostęp do elementu nr 2: *pomiary_V[indeks]*, po uprzednim ustawieniu zmiennej *indeks* na wartość 2.

Deklaracja zmiennych – tablice

- Tablica wielowymiarowa (maksymalnie 6 rozmiarów), np. tablica dwuwymiarowa:

Array*[lo1..hi1, lo2..hi2,] of type*

gdzie:

lo1, *hi1* – dolny, górny zakres pierwszego rozmiaru tablicy (minimalna wartość –32 768),

lo2, *hi2* – dolny, górny zakres drugiego rozmiaru tablicy (maksymalna wartość +32 767),

type – typ danych elementów tablicy (poza typem *Array*),

- Przykład: tablica o nazwie ***wspolrzedne*** *Array[0..1, 0..2] of UInt* zawiera 6 elementów (2*3) typu UInt:

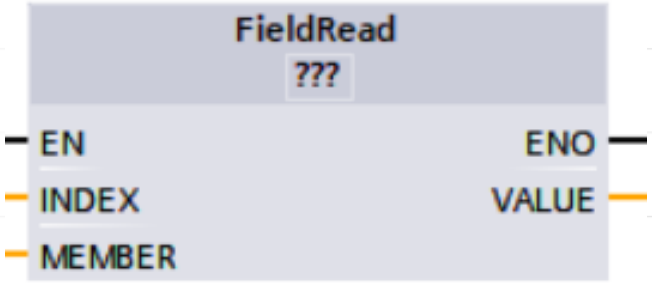
wspolrzedne[0,0], *wspolrzedne[0,1]*, *wspolrzedne[0,2]*,
wspolrzedne[1,0], *wspolrzedne[1,1]*, *wspolrzedne[1,2]*,

- Dostęp do elementów tablicy poprzez [*indeks1*, *indeks2*] np. dostęp do elementu nr 0,2:

wspolrzedne[indeks1, indeks2], po uprzednim ustawieniu *indeks1*=0, *indeks2*=2.

Deklaracja zmiennych – tablice

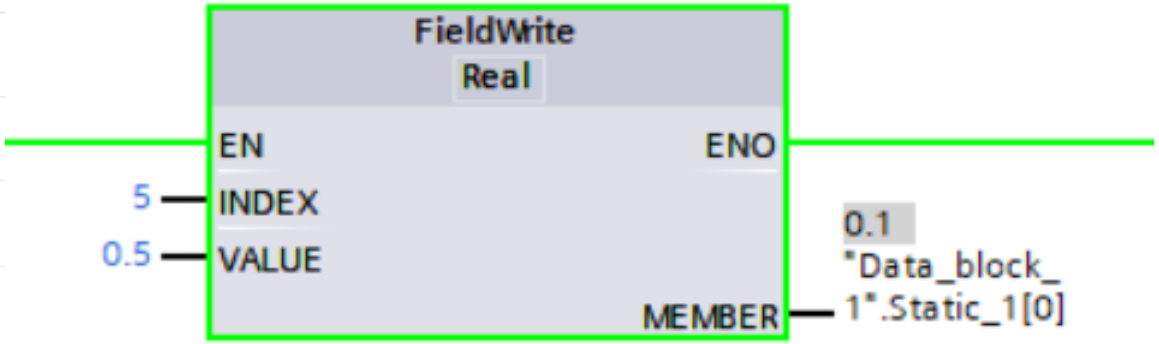
- **FieldRead** odczytuje i wpisuje na wyjście **VALUE** element tablicy o indeksie **INDEX** z tablicy, której pierwszy element jest określony przez parametr **MEMBER**.



- **FieldWrite** przesyła wartość z wejścia **VALUE** do tablicy, której pierwszy element jest określony przez **MEMBER** do komórki wskazanej przez **INDEX**.

Deklaracja zmiennych – tablice

- **FieldWrite** przesyła wartość z wejścia **VALUE** do tablicy, której pierwszy element jest określony przez **MEMBER** do komórki wskazanej przez **INDEX**.



Static_1	Array[0..6] o...		
Static_1[0]	Real	0.0	0.1
Static_1[1]	Real	0.0	0.0
Static_1[2]	Real	0.0	0.1
Static_1[3]	Real	0.0	0.0
Static_1[4]	Real	0.0	0.0
Static_1[5]	Real	0.0	0.5
Static_1[6]	Real	0.0	0.0

Przetworniki A/C

Wejścia analogowe:

- Dzięki nim sygnał analogowy ciągły jest zamieniany na sygnał cyfrowy, reprezentowany jako liczby.
- Przetwornik A/C (analogowo-cyfrowy) charakteryzują:
  - częstotliwość przetwarzania, np. 10 kHz, to znaczy pomiar co 100 µs,
  - zakres pomiaru napięcia, np. od -10 V do +10 V lub prądu, np. od 0 do 20 mA,
  - rozdzielczość określoną przez liczbę bitów, np. 10 bitów.

Przetworniki A/C

- Pochodną zakresu pomiarowego i rozdzielczości jest tzw. ziarno:

$$\text{ziarno} = \frac{\text{zakres pomiaru napięcia}}{2^{\text{rozdzielczość}}-1}$$

$$\text{ziarno} = \frac{+10\ \text{V} - (-10\ \text{V})}{2^{10}-1} = \frac{20\ \text{V}}{1023} = 0{,}01955034\ \text{V/bit}$$

- Co oznacza, że jeżeli wartość napięcia na wejściu przetwornika A/C zmieni się o 19,555034 mV, to odpowiadająca mu wartość cyfrowa zmieni się o 1.

Przetworniki A/C

Przykład doboru przetwornika A/C pomiar temperatury ciała ludzkiego z rozdzielczością 0,01 ºC:

- Temperatura jest wielkością fizyczną wolnozmienną, dlatego wystarczy bardzo wolny przetwornik A/C, bo jest zdecydowanie tańszy od szybkich.
- Zakładamy zakres zmian temperatury ciała od 32 ºC do 42 ºC, co przy rozdzielczości 0,01 ºC daje liczbę ziaren: (42 ºC - 32 ºC) / 0,01 ºC = 1000, zatem w zupełności wystarczy przetwornik 10 bitowy (bo $2^{10} = 1024$).
- Zakres pomiaru napięcia przez przetwornik A/C zależy od rodzaju użytego przetwornika temperatury na napięcie (lub na rezystancje – wtedy wymagany dodatkowy układ dopasowujący rezystancja → napięcie). Będzie to jednak zazwyczaj przetwornik A/C unipolarny.

Przetworniki A/C

Przykład doboru przetwornika A/C: pomiar sygnału napięciowego sinusoidalnego o częstotliwości 50 Hz i amplitudzie±25 V z rozdzielczością 0,001 V, przy założeniu 200 próbek na okres:

- Konieczne będzie zastosowanie przetwornika bipolarnego. W przypadku zakresu napięć, które mierzy przetwornik mniejszego niż ±25 V, wymagane będzie użycie dodatkowego dzielnika napięciowego.
- Ponieważ sygnał mierzony ma częstotliwość 50 Hz, to przy założeniu 200 próbek/okres należy dobrać przetwornik o częstotliwości konwersji minimum 50 Hz * 200 = 10 kHz.
- Zakres pomiarowy ±25 V z rozdzielczością 0,001 V daje liczbę ziaren:

(+25 V – (-25 V)) / 0,001 V = 50 000

Wymaga to użycia przetwornika 16 bitowego (bo 216 = 65 536).

Przetworniki A/C

- **Rozdzielczość a dokładność pomiaru**
  Nie należy utożsamiać rozdzielczości z dokładnością
- Np. 10 bitowy przetwornik A/C o zakresie pomiarowym od 0 do 10 V ma rozdzielczość 9,775171 mV/bit.
- O tym, jak jest jego dokładność decyduje cały tor pomiarowy. Ma na to wpływ wiele czynników, np. rezystancja połączeń przewodów pomiarowych z punktem pomiaru, czy też obecność zakłóceń. Obowiązuje tu zasada „najsłabszego ogniwa”.

Przetworniki A/C

Wyjścia analogowe:

- Dzięki nim sygnał cyfrowy, reprezentowany jako liczby, jest zamieniany na sygnał analogowy dyskretny, (wartość napięcia pojawia się co okres przetwarzania *T*p i pozostaje niezmienna do czasu *T*p+1).
- Przetwornik C/A (cyfrowo-analogowy) charakteryzują:
  - częstotliwość przetwarzania, np. 10 kHz, to znaczy pomiar co 100 µs,
  - zakres napięcia wyjściowego, np. od -10 V do +10 V, lub prądu wyjściowego, np. od 0 do 20 mA .
  - rozdzielczość określoną przez liczbę bitów, np. 10 bitów.

Przetworniki A/C

- Pochodną zakresu pomiarowego i rozdzielczości jest tzw. ziarno:

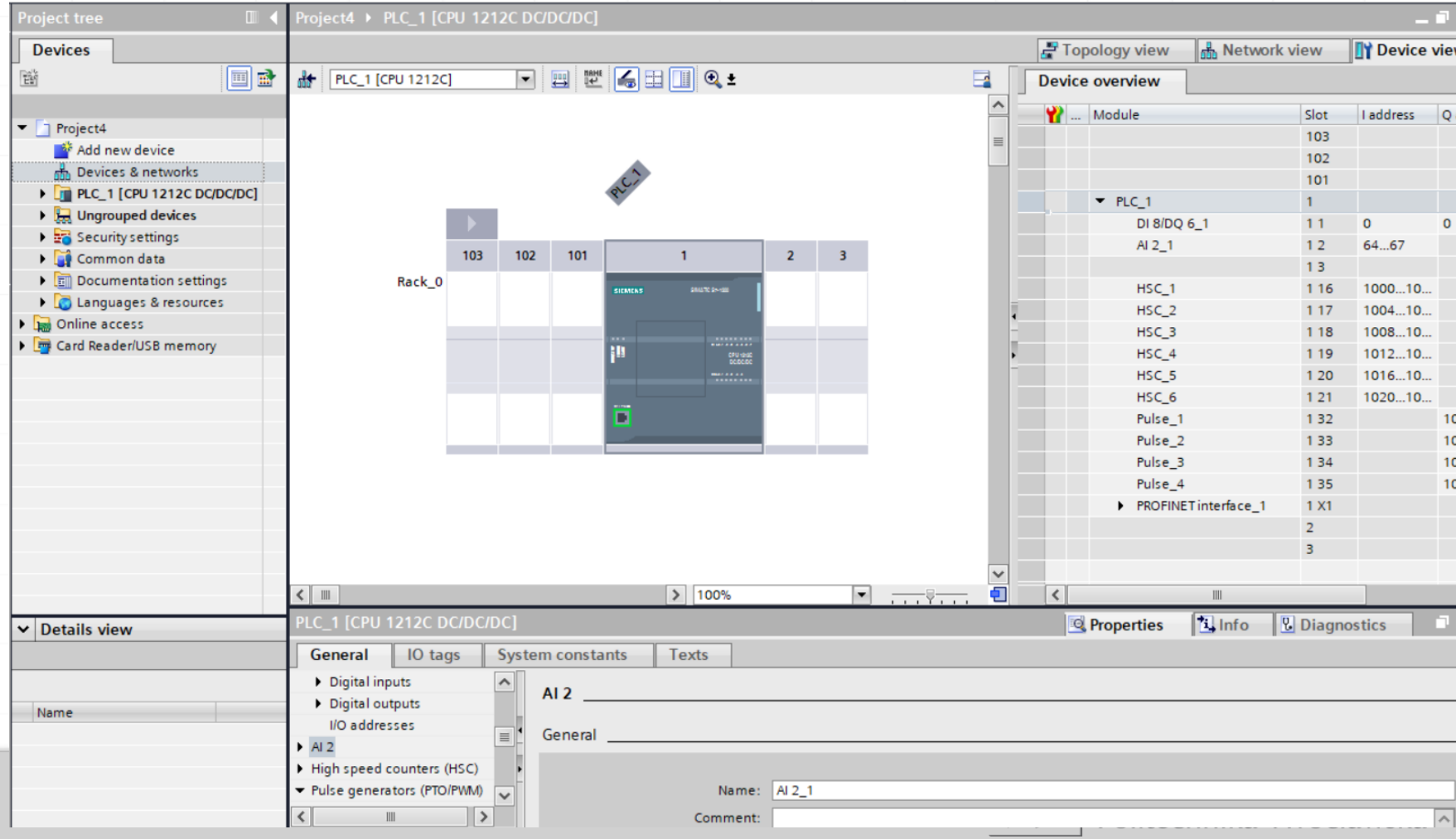
$$\text{ziarno} = \frac{\text{zakres zmian napięcia lub prądu wyjściowego}}{2^{\text{rozdzielczość}}-1}$$

$$\text{ziarno} = \frac{+10\ \text{V} - (-10\ \text{V})}{2^{10}-1} = \frac{20\ \text{V}}{1023} = 0{,}01955034\ \text{V/bit}$$

- Co oznacza, że jeżeli wartość cyfrowa na wejściu przetwornika C/A zmieni się o 1, to wartość napięcia na wyjściu przetwornika C/A zmieni się o 19,555034 mV.

Przetworniki A/C – wejścia analogowe

- Parametry wejść analogowych można w TiA Portal sprawdzić i niektóre zmienić na etapie konfiguracji sterownika: ***General → AI 2 / AQ2 → Analog inputs → Chanel0*** lub ***General → AI 2 / AQ2 → Analog inputs → Chanel1***



Przetworniki A/C – wejścia analogowe

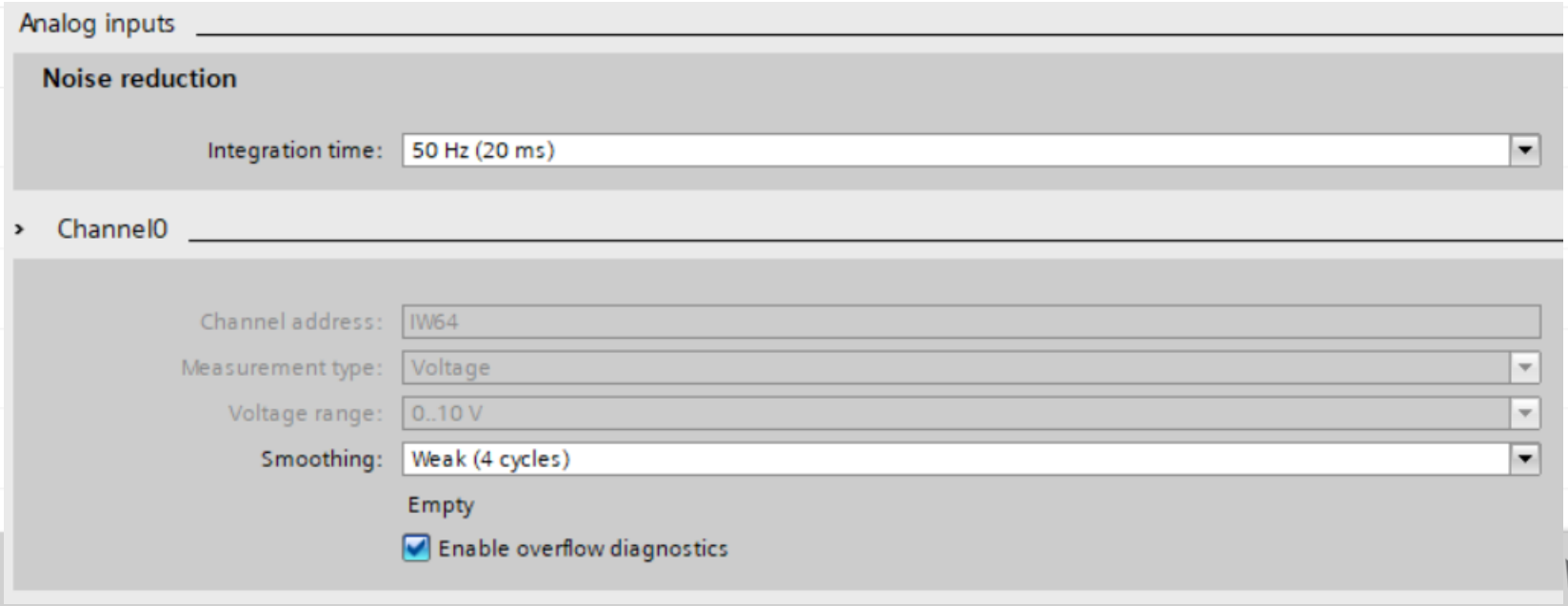
- Parametry niezmienne - adresy kanałów analogowych: Chanel0: IW64; Chanel1: IW66

Device overview

...	Module	Slot	I address	Q address	Type	Article no.	Firmware	Comment
		103						
		102						
		101						
	PLC_1	1			CPU 1212C DC/DC/DC	6ES7 212-1AE40-0XB0	V4.2	
	DI 8/DQ 6_1	1 1	0	0	DI 8/DQ 6			
	AI 2_1	1 2	64...67		AI 2			
		1 3						
	HSC_1	1 16	1000...10...		HSC			
	HSC_2	1 17	1004...10...		HSC			
	HSC_3	1 18	1008...10...		HSC			
	HSC_4	1 19	1012...10...		HSC			
	HSC_5	1 20	1016...10...		HSC			
	HSC_6	1 21	1020...10...		HSC			
	Pulse_1	1 32		1000...10...	Pulse generator (PTO/P...			
	Pulse_2	1 33		1002...10...	Pulse generator (PTO/P...			
	Pulse_3	1 34		1004...10...	Pulse generator (PTO/P...			
	Pulse_4	1 35		1006...10...	Pulse generator (PTO/P...			
	PROFINET interface_1	1 X1			PROFINET interface			
		2						
		3						

Przetworniki A/C – wejścia analogowe

- Parametry przetworników A/C możliwe do zmiany:
  - Integration time - czas całkowania (dotyczy wszystkich kanałów):
    - 60 Hz (16,6 ms); 50 Hz (20 ms) – domyślnie; 10 (100 ms).
  - Smoothing – czas wygładzania (dla każdego kanału oddzielnie):
    - None (1 cycle); Weak (4 cycles) – domyślnie; Medium (16 cycles); Strong (32 cycles).
  - Opcja „Enable overflow diagnostic” - powoduje wywołanie bloku diagnostycznego w przypadku przekroczenia zakresu pomiarowego.



Przetworniki A/C – wyjścia analogowe

- Parametry wyjść analogowych można w TiA Portal sprawdzić i niektóre zmienić na etapie konfiguracji sterownika:
  General → AI 2 / AQ2 → Analog outputs
- Standardowe adresy kanałów analogowych (niemożliwe do zmiany na etapie konfiguracji):
  Chanel0: QW80; Chanel1: QW82

Przetworniki A/C – wyjścia analogowe

- Parametry przetworników C/A możliwe do zmiany:
  - Reaction to CPU STOP – (wartość prądu na wyjściu C/A po zastopowaniu PLC - dotyczy obydwu kanałów):
    - Use substitute value,

    w polu *Substitute value for channel on a change from RUN to STOP* można ustawić wartość prądu z zakresu od 0,000 do 20,000 mA (dla każdego kanału oddzielnie); standardowo ustawiona wartość to 0,000 mA,
    - Keep last value – Zachowuje wartość prądu z ostatniego przetwarzania.

Przetworniki A/C

- W przypadku klasycznego, bitowego sposobu odczytu/zapisu przetworników A/C i C/A każda zmiana przetwornika na przetwornik o innej rozdzielczości wymaga zmian w programie.
- W przypadku przekroczenia zakresu napięcia podanego na wejście przetwornika A/C użytkownik może o tym nie wiedzieć, zatem istnieje duże prawdopodobieństwo uszkodzenia przetwornika:
  np. dla 10-bitowego przetwornika A/C o zakresie pomiarowym od 0 do 10 V, wartość bitowa 1023 będzie odczytana zarówno dla wejściowego napięcia równego 10 V, ale również np. 12 V, 20 V czy też 50 V.
- Aby tego uniknąć Siemens wprowadził specjalny format odczytu/zapisu przetworników A/C i C/A zwany **S7 analog format**.

Przetworniki A/C

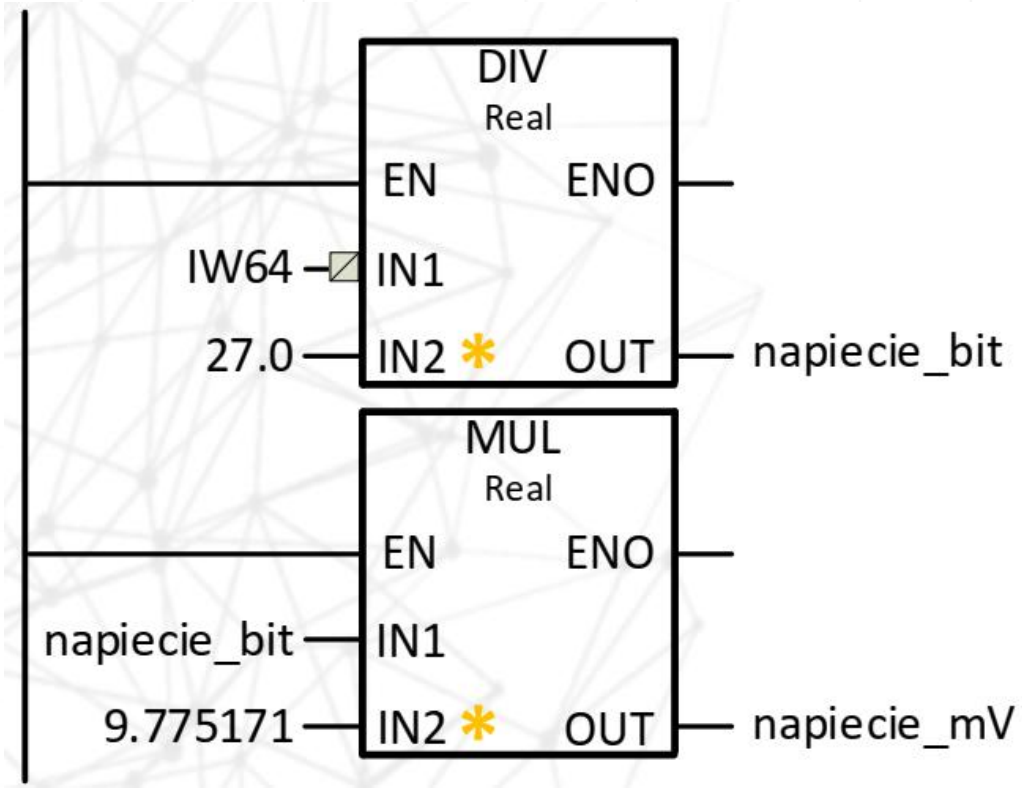
- S7 analog format zakłada stały zakres bitowy od 0 do 32767, niezależnie od rozdzielczości przetwornika.
- Zakres ten jest podzielony na 3 obszary:
  - od 0 do 27648 – zakres „normalnej” pracy przetwornika w standardowym, dopuszczalnym zakresie zmian napięcia/prądu,
  - od 27649 do 32511 – przekroczenie zakresu (overshoot),
  - od 32512 do 32767 – przepełnienie (overflow) - dodatkowa sygnalizacja pulsacją czerwonej diody *Error* na panelu sterownika.

Przetworniki A/C i C/A

- zapis *S7 analog format uniezależnia obliczenia od typu użytego przetwornika*,
- zapis *S7 analog format* jest inny w przypadku przetworników unipolarnych (<0; 27648>) oraz bipolarnych (<-27648; 27648>),
- bez zmiany typu przetwornika unipolarny -> bipolarny, w zasadzie wszystkie obliczenia można dokonywać wykorzystując zapis *S7 analog format*, jednak w wyniku obliczeń otrzymane wartości mogą znacząco różnić się od tego standardu,
- zaleca się przekonwertowanie standardu *S7 analog format* do przedziału <0; 1>, co umożliwia, w sposób prosty, otrzymanie wyniku obliczeń także w tym przedziale,
- w zasadzie nie korzysta się w ogóle z zapisu bitowego,
- zapis napięciowy stosuje się np. w przypadku zadawania napięcia, które chcemy wygenerować na wyjściu przetwornika C/A lub w przypadku, gdy np. chcemy podać wartość napięcia zmierzonego za pomocą przetwornika A/C.

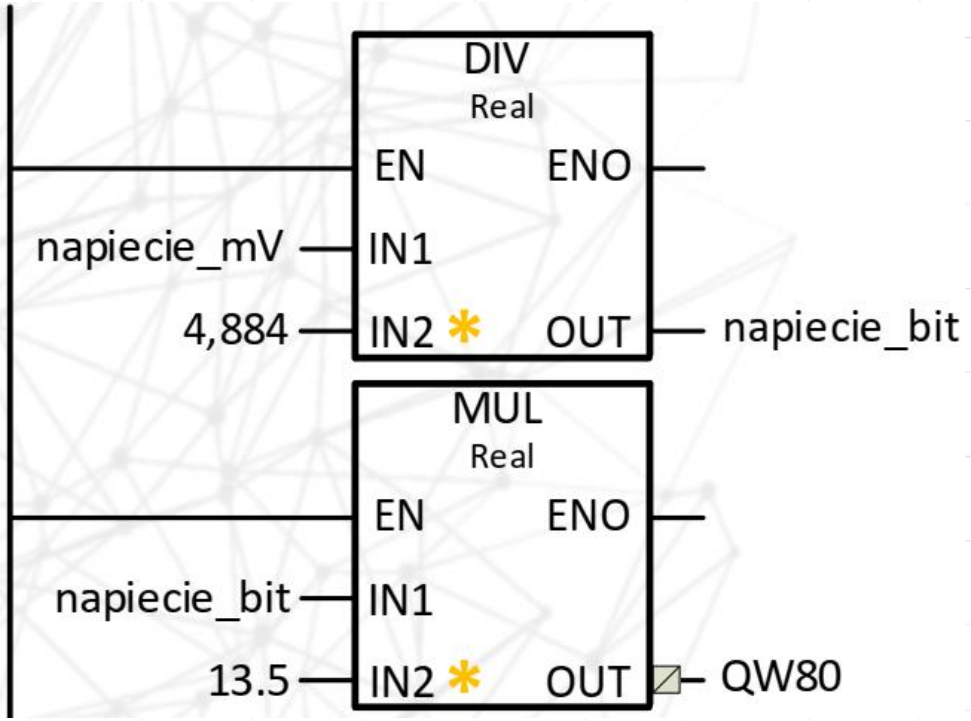
Przetworniki A/C – przykład programu

- napięcie podawane jest na przetwornik A/C kanał 0 (przetwornik 10-bitowy o zakresie pomiarowym od 0 do 10 V),
- w wyniku pomiaru w rejestrze IW64 znajduje się wartość zmierzona w formacie S7 analog format,
- po podzieleniu przez wartość 27, otrzymujemy wartość napięcia w bitach (napiecie_bit), 27648/(2^10)=27,
- po pomnożeniu przez 9,775171 (ziarno) otrzymujemy wartość napięcia w mV (napięcie_mV),
- przeliczenie na wartość bitową ma tylko znaczenie „dydaktyczne”. Zazwyczaj przelicza się wprost ze standardu *S7 analog format* na wartość napięcia: *napiecie_mV* = *IW64* * 10 000 / 27648.



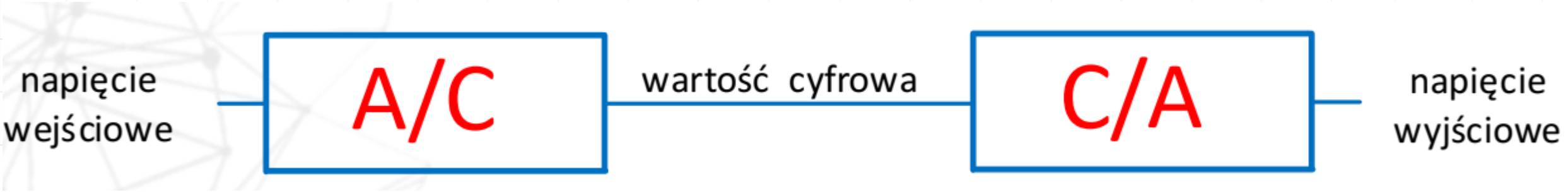
Przetworniki C/A

- Przetwornik C/A (Signal Board), 12- bitowy o zakresie wyjściowym od -10 do +10 V,
- napięcie, które chcemy otrzymać na wyjściu C/A wpisujemy w mV do zmiennej *napiecie_mV*,
- po podzieleniu przez wartość 4,884 (ziarno) otrzymujemy wartość napięcia w bitach (*napiecie_bit*),
  (10 000 mV – (-10 000mV)) / (2^12-1) = 4,884 mV/bit,
- po pomnożeniu przez 13,5 otrzymujemy wartość napięcia w formacie *S7 analog format*, która po auto konwersji na typ *Int* wpisywana jest do rejestru *QW80*, co jest równoważne pojawieniu się napięcia na fizycznym wyjściu analogowym. (27648 – (-27648)) / (2^12) = 13,5.



- Przeliczenie na wartość bitową ma tylko znaczenie „dydaktyczne”. Zazwyczaj przelicza się wprost wartość napięcia na standard *S7 analog format*: *QW80* = *napiecie_mV* * (2*27648) / (2*10 000).

Przetworniki A/C i C/A



Napięcie wejściowe ≠ napięcie wyjściowe

Im większa częstotliwość próbkowania tym napięcie na wyjściu tego układu będzie „bardziej zbliżone” kształtem do napięcia wejściowego, jednak praktycznie nigdy kształty obydwu napięć nie będą jednakowe.

Funkcje

Funkcja (podprogram) to wydzielona i nazwana część programu, która raz napisana może być wielokrotnie wywoływana w programie poprzez podanie nazwy funkcji.

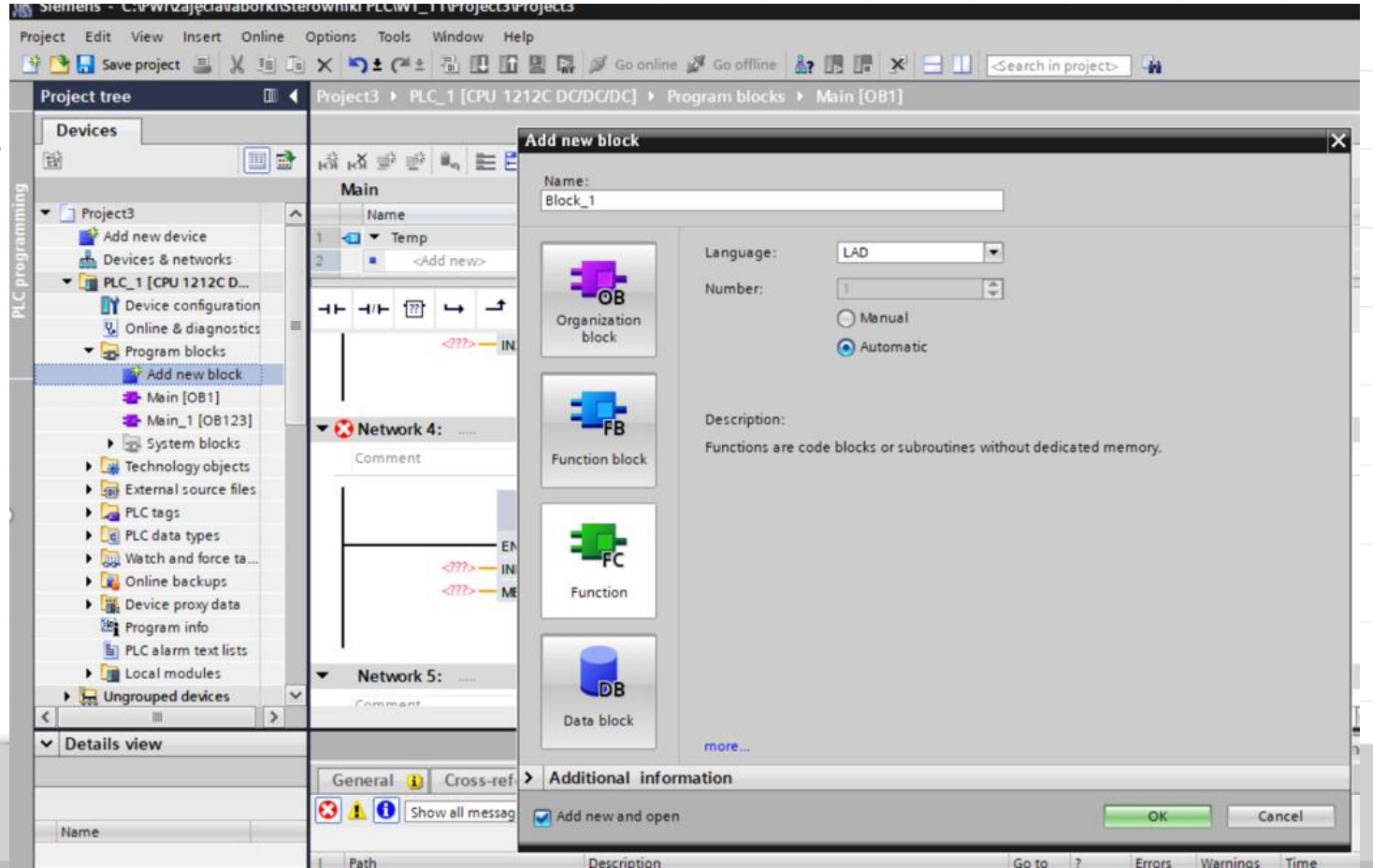
- Funkcję można wywołać z parametrami wejściowymi.
- Funkcja może zwracać parametry wyjściowe.
- Wewnątrz funkcji stosowane jest adresowanie symboliczne (podobnie jak w bloku danych).
- Poza parametrami wejściowymi i wyjściowymi pozostałe parametry używane wewnątrz funkcji są typu lokalnego.

Funkcje

- Aby w TiA Portal zadeklarować funkcję należy wykonać:

Project tree → Program blocks → Add new block → Function

- Można zmienić nazwę funkcji oraz wybrać język programowania.
- Standardowa nazwa funkcji to ***Block_**[kolejny_numer_funkcji]*.



Parametry funkcji

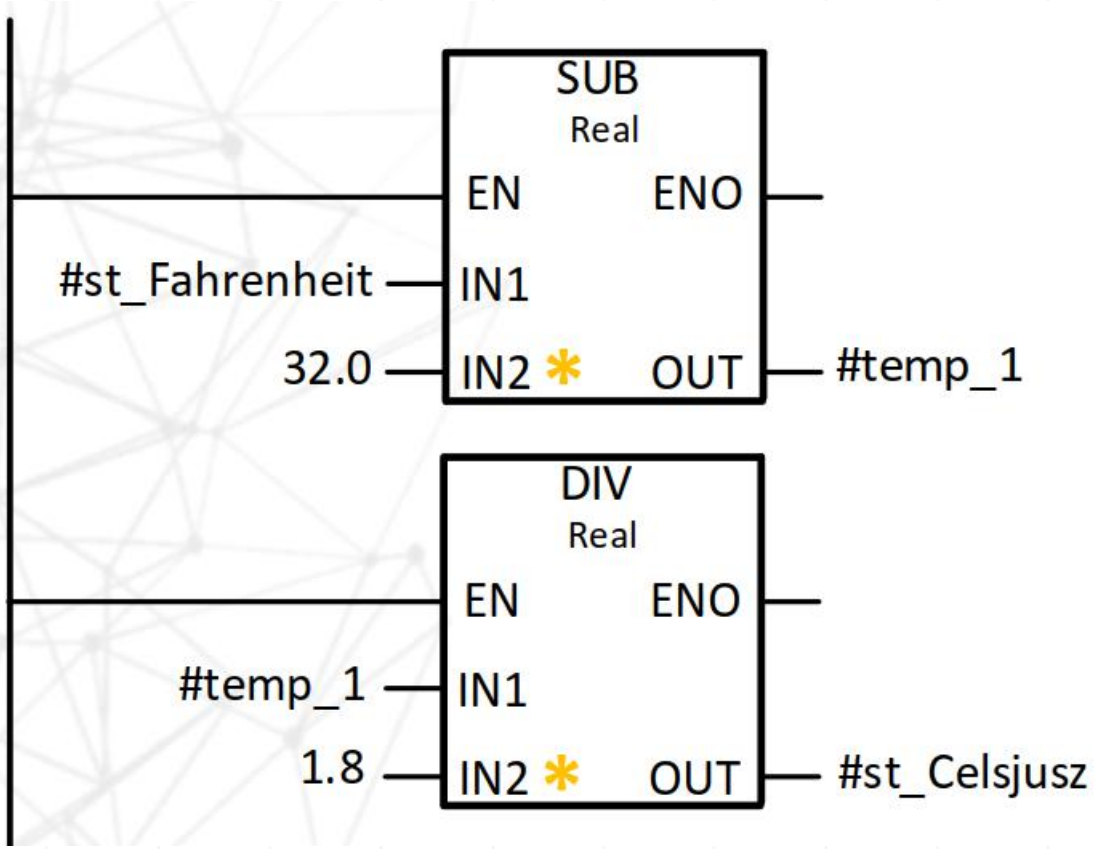
- Parametry funkcji określamy w bloku interfejsu funkcji, przed pisaniem kodu programu.
- Określamy tylko nazwę parametru oraz jego typ danych.
- W kodzie programu parametry funkcji są poprzedzone znakiem **#**.
- Rodzaje parametrów:
  - **Input** - parametry wejściowe, czyli argumenty funkcji,
  - **Output** - parametry wyjściowe, czyli te która funkcja zwraca,
  - **InOut** - parametry które są parametrami wejściowymi, wewnątrz funkcji mogą być modyfikowane i są zwracane na wyjściu funkcji,
  - **Temp** – zmienne typu lokalnego wykorzystywane tylko wewnątrz funkcji,
  - **Constant** – stałe, których wartość podawana jest na etapie deklaracji; nie można ich zmieniać.

Przykład funkcji

- **Funkcja, która przelicza temperaturę podaną w stopniach Fahrenheita na stopnie Celsjusza.**
- Deklaracja funkcji:
  - ***Project tree*** → ***Program blocks*** → ***Add new block*** → ***Function***

  Funkcji nadajemy nazwę ***Fahrenheit_Celsjusz***.
- Parametry funkcji:
  - Input st_Farenheit Real
  - Output st_Celsjusz Real
  - Temp temp_1 Real

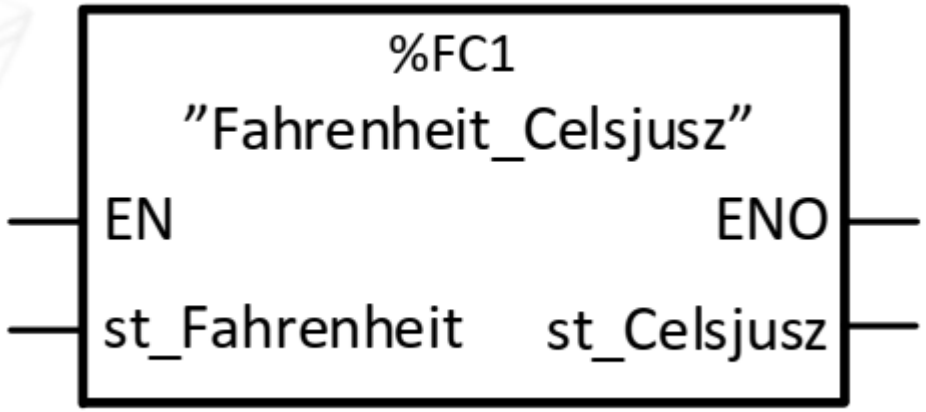
Przykład funkcji



$$st\_Celsjusz = \frac{st\_Fahrenheit - 32}{1,8}$$

Przykład funkcji

Funkcję wywołuje się, podając jej nazwę.



Blok funkcyjny

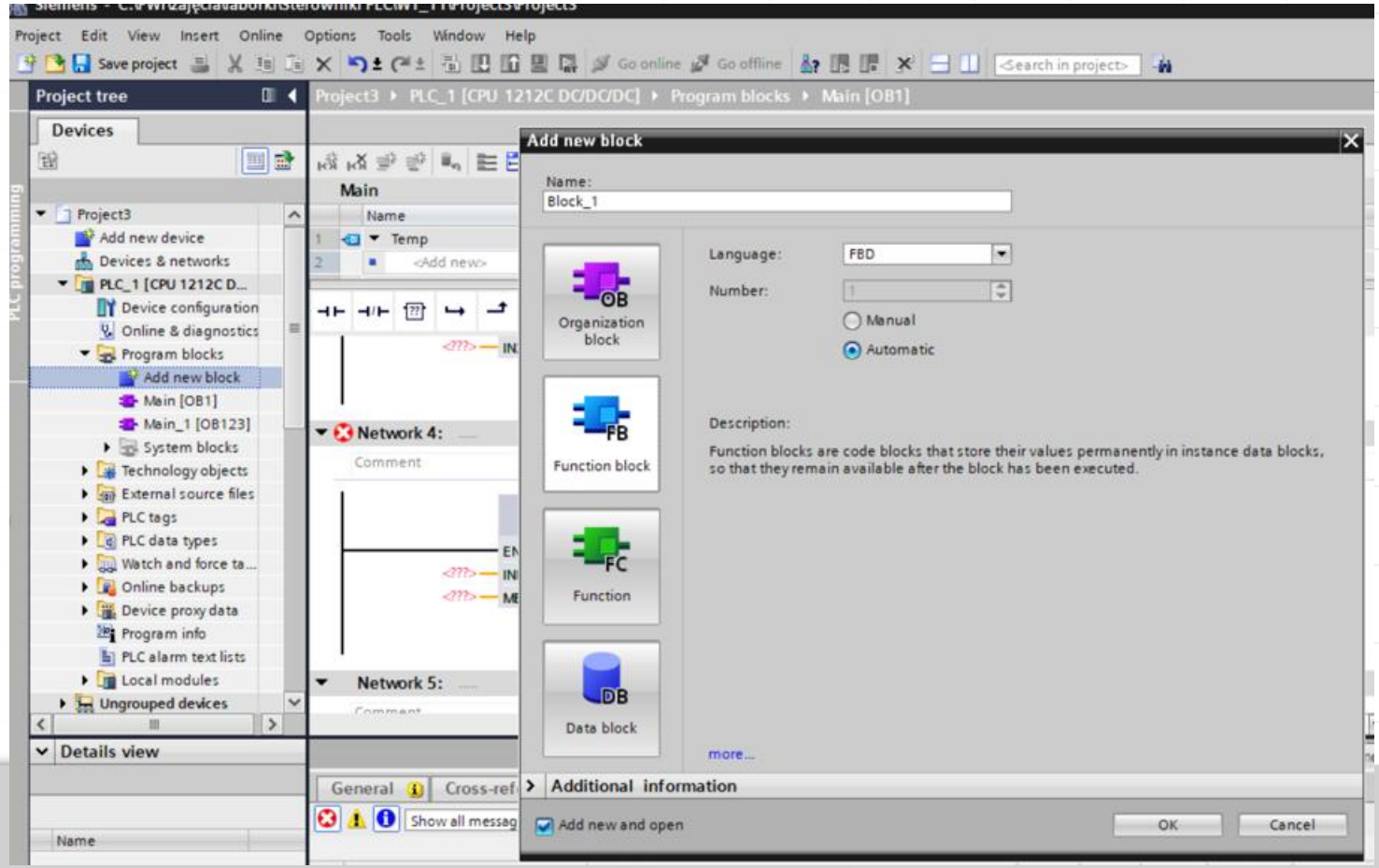
- **Blok funkcyjny** to wydzielona i nazwana część programu, która raz napisana może być wielokrotnie wywoływana w programie poprzez podanie nazwy bloku funkcyjnego.
- Blok funkcyjny można wywołać z parametrami wejściowymi, może także zwracać parametry wyjściowe.
- Wewnątrz bloku funkcyjnego stosowane jest adresowanie symboliczne (podobnie jak w blokach danych i funkcjach).
- Blok funkcyjny stosujemy, gdy operacje w nim zawarte potrzebują kilku cykli programowych, aby je prawidłowo wykonać.
- Blok funkcyjny tym się różni od funkcji, że zapamiętuje wartości wybranych parametrów przy każdym wywołaniu, zatem wymaga dodatkowego bloku danych, w którym te parametry są zapisane.
- Każde użycie bloku funkcyjnego, to nowy blok danych, zatem możliwe jest zadeklarowanie innych danych początkowych każdego wywołania.

Blok funkcyjny

- Aby w TiA Portal zadeklarować blok funkcyjny należy wykonać:

Project tree → Program blocks → Add new block → Function block

- Można zmienić nazwę bloku funkcyjnego oraz wybrać język programowania.
- Standardowa nazwa bloku funkcyjnego to ***Block_****[kolejny_numer_bloku_funkcyjnego]*.



Blok funkcyjny

- Parametry bloku funkcyjnego (nazwę oraz typ danych) określamy przed pisaniem kodu programu.
- W kodzie programu parametry bloku funkcyjnego są poprzedzone znakiem **#**.
- Rodzaje parametrów:
  - **Input** - parametry wejściowe, czyli argumenty bloku funkcyjnego,
  - **Output** - parametry wyjściowe, czyli te które blok funkcyjny zwraca,
  - **InOut** - parametry które są parametrami wejściowymi, wewnątrz bloku funkcyjnego mogą być modyfikowane i są zwracane na wyjściu funkcji,
  - **Static** - parametry które są które są dostępne tylko lokalnie, ale ich wartości są pamiętane pomiędzy kolejnymi wywołaniami/wykonaniami bloku funkcyjnego,
  - **Temp** – zmienne typu lokalnego wykorzystywane tylko wewnątrz bloku funkcyjnego,
  - **Constant** – stałe, których wartość podawana jest na etapie deklaracji; nie można ich zmieniać.

Bibliografia

1. Podręcznik pierwsze kroki z SIMATIC S7 – 1200
2. Materiały opublikowane w ramach Projektu „3P Projektowanie Przemysłu Przyszłości” dostępne na licencji Creative Commons Uznanie autorstwa 4.0 Międzynarodowa. Autorami materiałów są: Janusz Staszewski oraz Fundacja Manus.
3. Tadeusz Legierski i inni, „Programowanie sterowników PLC”. Wydawnictwo Pracowni Komputerowej Jacka SKALMIERSKIEGO 2008.
4. Tomasz Gilewski; „Szkoła programisty PLC Język LAD w programowaniu sterowników przemysłowych”. Helion 2018.
5. Tomasz Gilewski; „Szkoła programisty PLC Sterowniki przemysłowe”. Helion 2017.
6. Janusz Kwaśniewski; „Inteligentny dom i inne systemy sterowania w 100 przykładach”. Wydawnictwo BTC 2011.
7. Janusz Kwaśniewski; „Programowalny sterownik SIMATIC S7-300 w praktyce inżynierskiej”. Wydawnictwo BTC 2009.

Dziękuję za uwagę

Politechnika Wrocławska

Wydział Mechaniczny