Zostań mistrzem – szkolenie od podstaw
Część opisowa

Autorzy: Piotr Cieśla, Bartłomiej Majchrowski, Michał Sujewicz, Wiktor Śliwak, Michał Maćkowiak, Jarosław Woźniak

Redakcja: Kajetan Szczurowski

Redakcja merytoryczna: Karol Jagiełło, Miłosz Kanas

Spis treści

1. Czym są PLC? 3
2. Typy danych i typy zmiennych 4
3. Adresowanie, bramki logiczne, zbocza 6
4. Liczniki 11
5. Timery 14
6. Operacje matematyczne 18
Dodatek – pełna lista typów zmiennych 24
Źródła 26

1. Czym są PLC?

PLC – programowalne sterowniki logiczne. Są to nieduże komputery wyspecjalizowane w obsłudze i automatyzacji procesów przemysłowych, posiadające cykliczny obieg pamięci, co oznacza że programy są wykonywane po sobie, w takiej kolejności, w jakiej poszczególne bloki zostały napisane.

Pół wieku temu (i dawniej) do obsługi i automatyzacji procesów przemysłowych w fabrykach wykorzystywano wielkie i skomplikowane układy przekaźnikowe. Trzeba było stworzyć algorytm z poszczególnych fizycznych elementów, co było trudne, żmudne, czasochłonne i spowalniało procesy modernizacyjne (nawet najdrobniejsza zmiana w działaniu układu mogła wymagać generalnego remontu całej szafy elektrycznej). Tę sytuację poprawiło wprowadzenie na rynek sterowników PLC.

W roku 1968 w General Motors przyjęto założenia projektowe dotyczące cech sterownika PLC: łatwość tworzenia i edycji programów, prostota utrzymania ruchu w produkcji z możliwością napraw poprzez zastosowanie modułów plug in, niezawodność w warunkach przemysłowych i koszty porównywalne z szafami sterowniczymi oraz panelami przekaźnikowymi. Pierwszym projektantem PLC został Dick Morley, a jego dziełem był Modicon 084, ważący 64kg.

Sterowniki PLC zaczęły być popularne na początku lat 70, z początku głównie w przemyśle motoryzacyjnym, jednak niedługo potem reszta przemysłu przyjęła te nowe rozwiązania.

Sterownik składa się z CPU (procesora), wejść i wyjść cyfrowych oraz analogowych, pamięci i bloków komunikacyjnych. Cyfrowe i analogowe wejścia/wyjścia różnią się tym, że pierwsze umożliwiają tylko zadawanie sygnałów 0 i 1, a drugie wielkości fizyczne, takie jak wartość prądu czy napięcia.

Pierwsze sterowniki były programowane przez ludzi, którzy w przeszłości zajmowali się układami przekaźnikowymi, więc początkowe języki programowania PLC były graficzne (LAD i FBD). Do języków graficznych dołączyły również odpowiedniki tekstowe (ST oraz IL).

2. Typy danych i typy zmiennych

Podczas tworzenia programów sterujących w sterowniku PLC wymaganym jest, tak jak w przypadku każdego innego programu napisanego w jakimkolwiek języku, aby zadeklarować konkretne zmienne, które następnie są wykorzystywane do prawidłowego działania programu. W tym rozdziale wymieniliśmy tylko najczęściej używane typy, jest ich jednak więcej, dokładniejszą listę zamieściliśmy w formie dodatku na koniec naszej instrukcji.

Typy danych

- **Bool** – najbardziej podstawowa zmienna, w przypadku której wartości mogą przyjmować wyłącznie dwie logiczne wartości: 0 lub 1. W zależności od sterownika zajmuje 1 lub 8 bitów pamięci.
- **Byte** (bajt) – na zapis zmiennej przeznacza się ciąg 8 bitów (8 bitów jest równe 1 bajtowi).
- **Word** (słowo) – zmienne z tym typem są zapisywane za pomocą ciągu 16 bitów.
- **DWord** (podwójne słowo) – stanowi pewnego rodzaju rozwinięcie typu Word, ponieważ zmienne tego typu są zapisywane za pomocą 32 bitów.
- **LWord** (poczwórne słowo) – zmienne tego typu są zapisywane za pomocą ciągu bitów, których liczba wynosi 64.

Typy zmiennych arytmetycznych

- **Int** (integer) – najbardziej znany typ zmiennych, pojawia się również w językach takich jak C, C++, Java czy Python. Służy do działań na liczbach całkowitych. Do zapisu tego typu zmiennej wykorzystuje się 16 bitów (a więc 1 słowo), a zakres wartości przyjmowanych przez zmienne typu Int wynosi od -32 768 do 32 767.
- **UInt** (unsigned integer) – tak jak w poprzednim przypadku, wartości zmiennych zajmują 16 bitów w pamięci sterownika. Zmienna tego typu nie może przyjmować wartości ujemnych, co przekłada się na zakres, który wynosi od 0 do 65 535.
- **Real** – typ zmiennoprzecinkowy wykorzystywany do zapisu wartości rzeczywistych z zakresu: od -3.402E37 do -1.176E-35 (zakres liczb ujemnych), +1.176E-35 do +3.402E37 (zakres liczb dodatnich).

Zakres wartości przyjmowanych przez typ rzeczywisty zależy od parametrów sprzętowych sterowników.

Dodatkowe typy zmiennych

- **Char** – typ przeznaczony do zapisu znaków (np. ‘P’, ‘L’, ‘C’ czy ‘+’). Zajmuje on 8 bitów w pamięci.
- **String** – typ, który służy do zapisu ciągu znaków w standardzie Unicode (np. ‘PLC’, ‘Automatyk’). W zależności od wybranego przez nas standardu, znaki są kodowane

w sposób 7 bitowy (ASCII), 16 bitowy (UTF-16) lub 32 bitowy (UTF-32). Innymi słowy ilość bitów przeznaczona na zapis wartości jest zmienna.

- **Time** – zmienna, która z kolei służy do zapisu wartości czasu. Przeznacza się na ten typ aż 32 bity, w związku z czym w sterownikach S7-1200 oraz S7-1500 wartości mogą być zapisane z dokładnością do milisekund. Dlatego też zakres wartości może wynosić od T#–24d_20h_31m_23s_648ms do T#24d_20h_31m_23s_647ms.

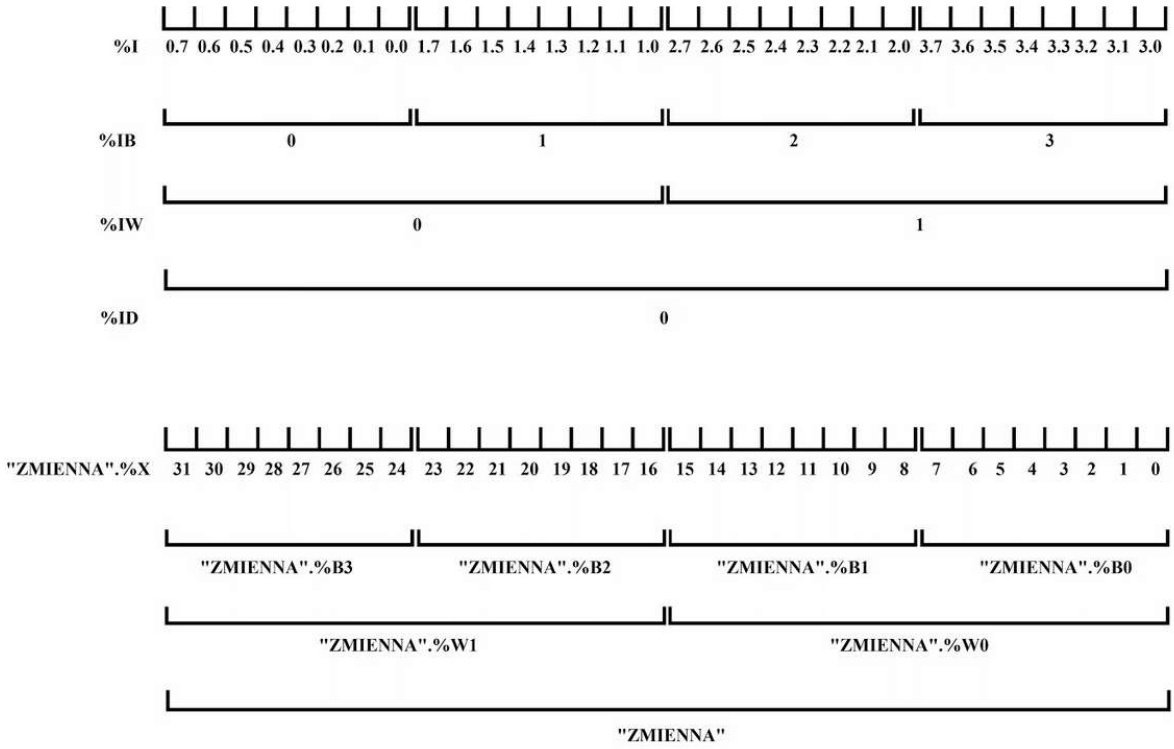
Zakres wartości typów „czasowych” mogą się różnić od parametrów sprzętowych konkretnych sterowników.

3. Adresowanie, bramki logiczne, zbocza

Adresowanie ogólnie

Z rozdziału drugiego wiemy, że w programowaniu PLC stosuje się zmienne różnych typów. Każda zmienna ma swój adres w pamięci sterownika. Adresowanie to metoda wskazania sterownikowi, którą zmienną programista wykorzystuje.

Bardzo ważną kwestią związaną z adresowaniem jest **nakładanie danych na siebie.**



W powyższym przykładzie adresy zachodzą na siebie. ID0 odwołuje się do 4 bajtów (0,1,2,3). IW2 odwołuje się do 2 bajtów (2 oraz 3). Zmiana wartości IW2 zawsze wpłynie na wartość ID0, ale nie każda zmiana wartości ID0 wpłynie na wartość IW2. Czasami jest to działanie pożądane, dlatego kompilator nie powiadomi nas o błędzie.

Adresowanie w oprogramowaniu e!COCKPIT:

Adresowanie wejść i wyjść odbywa się za pomocą struktury: **%<typ adresu><rozmiar adresu><położenie w pamięci>.**

Jako typ adresu przyjmuje się:

- **Wejście- I**
- **Wyjście- Q**

Jako rozmiar adresu:

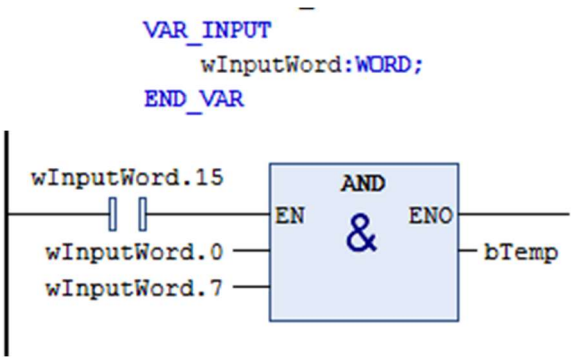
- **Bit – X np.: %IX2.1**
- **Bajt – B np.: %QB1**
- **Słowo – W np.: %IW1**

Położenie w pamięci:

- **Dla bitów <nr bajtu>. <nr bitu>**
- **Dla bajtów i słów <nr bajtu>**

Zmienne adresowane są od 0! A więc pierwszy bajt ma numer zerowy, drugi pierwszy itd.

Możliwe jest odwoływanie się do składowych danych większych typów zmiennych:



Na powyższym rysunku odnosimy się do **worda** (słowa, a więc typu danych, który ma 16 bitów, numery tych bitów to 0..15). Wykorzystujemy 3 bity słowa, a więc:

- wInputWord.15 – ostatni bit w słowie,
- wInputWord.0 – pierwszy bit w słowie,
- wInputWord.7 – 8 bit w słowie.

Adresowanie w oprogramowaniu TIA Portal:

Adresowanie dzielimy na trzy rodzaje:

- **Wejścia- I**
- **Wyjścia- Q**
- **Markery- M**

Sposób adresowania bitów jest następujący: **%<rodzaj adresu><nr bajtu>. <numer bitu> %I2.0.**

Każdy wykorzystywany adres ma przyporządkowany domyślny Tag, który także może być wykorzystywany przy adresowaniu. Nazwę tagu można zmienić w dowolnym momencie.

Adresowanie bajtu (BYTE): **%<rodzaj adresu> B <nr bajtu> %MB20.**

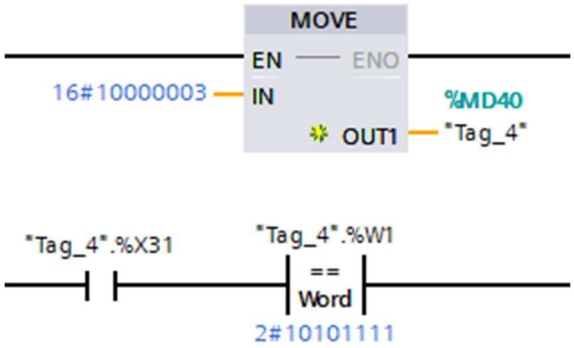
Adresowanie słowa (WORD): **%<rodzaj adresu>W<nr bajtu rozpoczynającego słowo> %MW22.**

Adresowanie podwójnego słowa (DWORD): **%<rodzaj adresu>D<nr bajtu rozpoczynającego podwójne słowo> %MD40**

Odwoływanie się do części danych w większym formacie: **<Tag>. %<typ danej><początek odczytywanych danych>**

Typ danej : **bit-X; bajt-B, słowo-W**

- **”TAG”.%X17** (odwołujemy się do 18 bitu w TAGu),
- **”TAG”.%W0** (odwołujemy się do pierwszego słowa w TAGu).



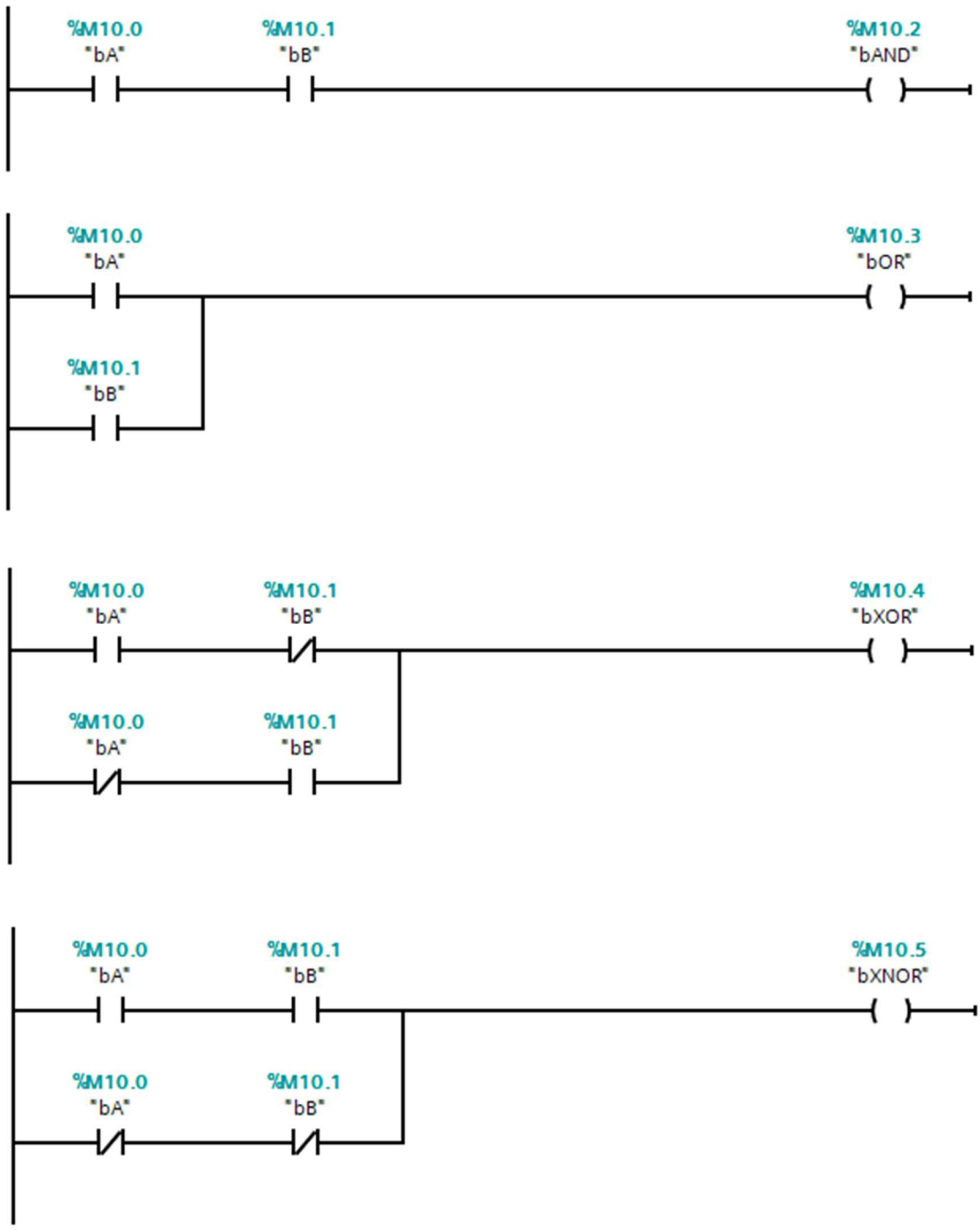
Na powyższym rysunku widać przykład adresowania w TIA Portal. „Tag_4” to nazwa zmiennej typu Double Word (podwójne słowo – 32 bity). Za pomocą struktury „Tag_4”.%X31 odwołujemy się do 32 bita, a za pomocą „Tag_4”.%W1 do drugiego słowa.

Bramki logiczne

Bramka logiczne wykorzystuje się w sytuacja wymagających określonego działania przy jakiejś relacji między zmiennymi boolowskimi, np. uruchomienie maszyny wymaga spełnienia dwóch warunków (np. dwóch jednocześnie wciśniętych przycisków).

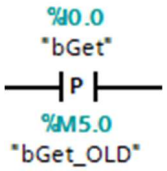
A	B	A AND B	A OR B	A XOR B	A XNOR B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	1

Bramki w PLC można zrealizować za pomocą następujących konstrukcji:

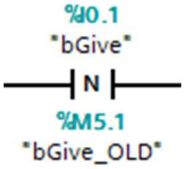


Zbocza wznoszące i opadające

Często, gdy na wejściu sygnał zmienia się na przeciwny chcemy wykonać coś tylko jednokrotnie. W takich sytuacjach bardzo przydatne są zbocza.



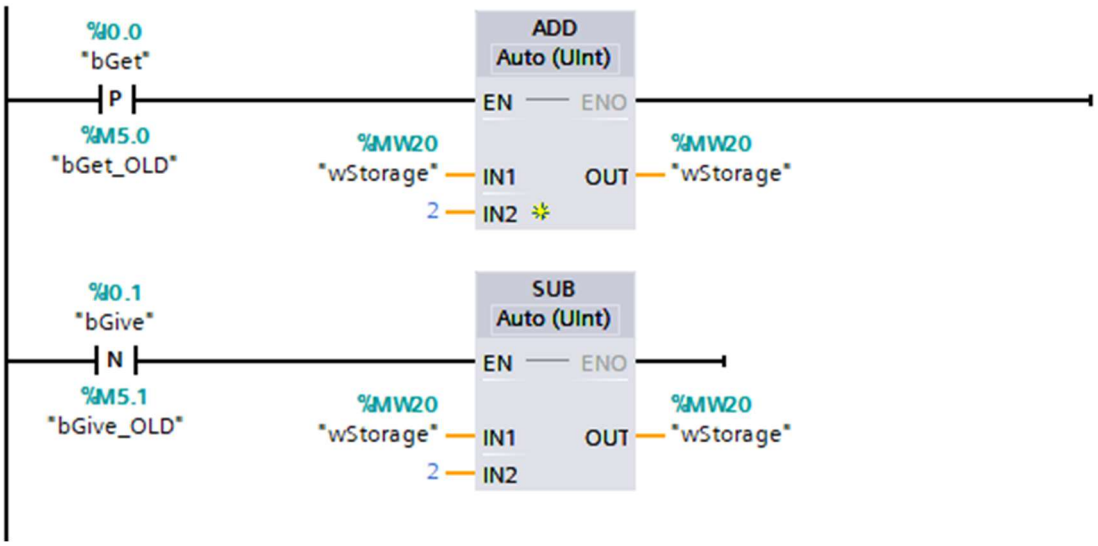
Zbocze wznoszące wykrywa przejście wejścia ze stanu niskiego do stanu wysokiego.



Zbocze opadające wykrywa przejście wejścia ze stanu wysokiego do niskiego.

Podczas adresowania zboczy u góry należy podać adres wejścia, na którym badamy zbocze. Na dole natomiast adres bitu, w którym będzie pamiętany stan z wejścia w cyklu poprzednim. Częstym błędem jest użycie w wielu zboczach tego samego adresu bitu (dolny adres). Taki błąd nie jest oznaczany przez kompilator, a powoduje złe działanie całej logiki systemu, więc należy na niego uważać.

W podanym przykładzie bez użycia zbocz w przypadku załączenia się wejścia operacja dodawania/odejmowania wykonywana byłaby w każdym cyklu.

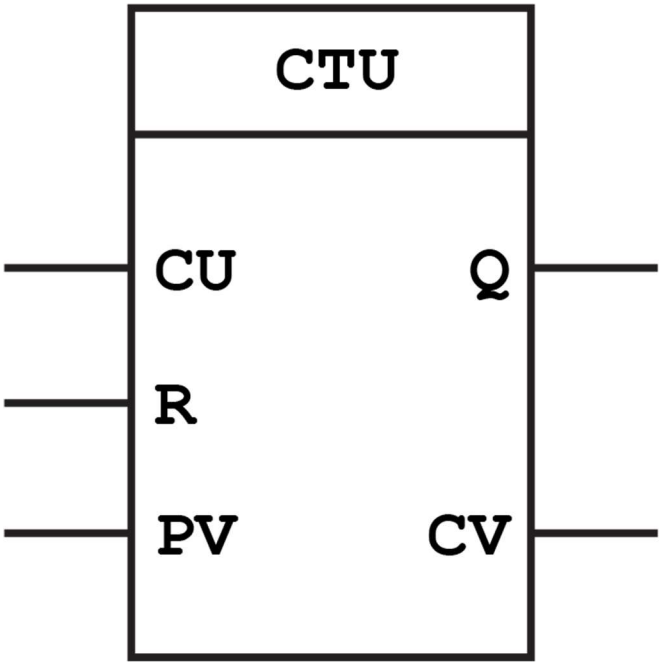


4. Liczniki

Głównym zadaniem liczników jest zliczanie ile razy pojawił się stan wysoki na wejściu takiego licznika i w momencie kiedy zostanie osiągnięta wymagana wartość, na wyjściu licznika zostaje ustawiony stan wysoki. W licznikach porównywane są ze sobą dwie dane: Preset Value (PV) i Current Value (CV). PV jest wartością ustawianą w programie, a CV obecną wartością, którą zwraca licznik (np. jeżeli licznik naliczy 3 impulsy wartość CV, będzie wynosiła 3).

Są trzy rodzaje liczników i wszystkie 3 występują w **TIAPortal** i **e!COCKPIT.**

CTU (Counter Up) – Licznik inkrementujący (zwiększający swój stan). Kiedy CV będzie równe lub większe PV, na wyjście Q zostanie puszczony stan wysoki.



CTU

CU – Inkrementacja. Doprowadzenie stanu wysokiego do tego wejścia zwiększa wartość CV o 1.

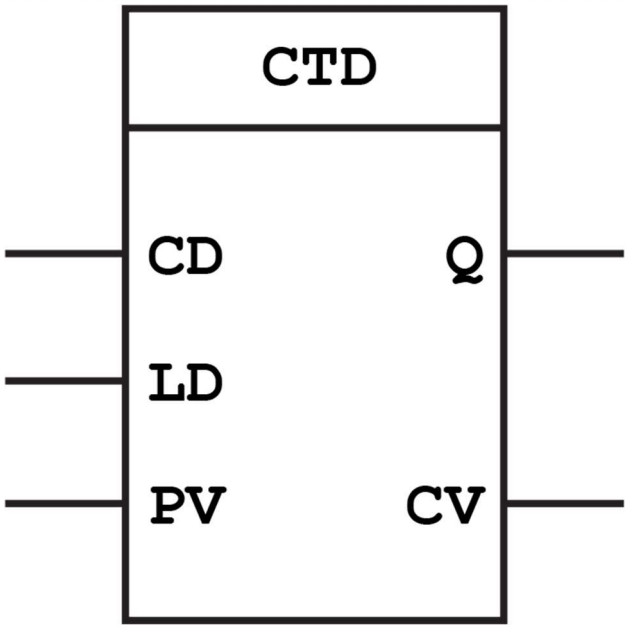
R – Reset. Po doprowadzeniu stanu wysokiego, CV zostanie wyzerowane.

PV – Preset value. Wartość ustalana w programie, gdy CV osiągnie wartość PV, to wtedy licznik ustawia wyjście Q.

Q – Wyjście.

CV – Current Value, wartość obecna.

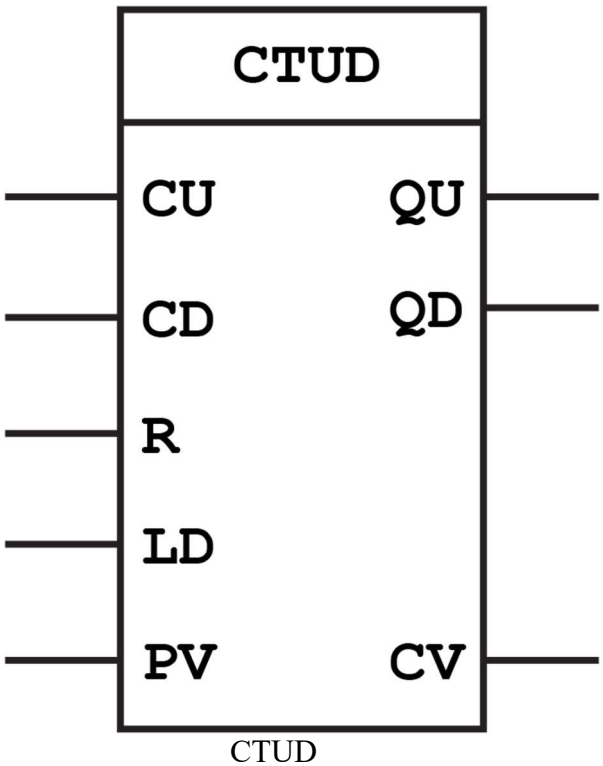
CTD (Counter Down) – Licznik dekrementujący (zmniejszający swój stan). Kiedy CV będzie równe lub mniejsze zero, na wyjściu Q zostanie puszczony stan wysoki.



CTD

CD – Dekrementacja, zmniejszająca wartość CV.
LD – Load. Jest to ustawienie wartości CV na PV (Czyli reset do wartości początkowej).
PV – Preset value. Wartość początkowa licznika.
Q – Wyjście.
CV – Current Value, wartość obecna

Istnieje jeszcze licznik CTUD, który jest kombinacją obu wcześniej wymienionych.



CTUD

Jak widać w liczniku CTUD wejścia są praktycznie takie same, jak w poprzednich dwóch licznikach. W przypadku doprowadzenie stanu wysokiego do CU licznik dodaje 1 do wartości CV, a przy CD dekrementuje te wartość. Wartością początkową w liczniku jest zero. W tym liczniku znajdują się też dwa wyjścia. QD wysyła stan wysoki, kiedy CV jest mniejsze lub równe 0 (jak w CTD), a QU, wysyła stan wysoki, kiedy CV jest większe lub równe PV (jak w CTU). Analogicznie jak w przypadku poprzednich liczników R zeruje wartość CV, a LD doprowadza do wartości równej PV.

W **LOGO! Soft Comfort** występuje za to Up/Down Counter, który przedstawiono na poniższym rysunku.

Q i R pełnią w tym przypadku takie same role jak w wyżej przedstawionych przypadkach.

Cnt jest zliczaniem. To, czy wartość CV jest inkrementowana, czy dekrementowana jest zależne od wartości zmiennej Dir. W przypadku, gdy Dir=0, licznik inkrementuje, a dla Dir=1 dekrementuje.

Przykład

Przykładowym programem wykorzystującym liczniki może być parking, zliczający liczbę wolnych miejsc. Na początku, kiedy parking jest pusty licznik wyświetla liczbę wszystkich miejsc parkingowych. Kiedy czujnik wykryje wjeżdżający samochód, wysyła informacje do sterownika, a liczba wolnych miejsc jest dekrementowana. Tak samo, kiedy samochód wyjeżdża zostaje wysłana informacja, a na jej podstawie liczba wolnych miejsc jest inkrementowana.

5. Timery

Przekaźniki czasowe. W programowaniu PLC rozróżniamy dwa podstawowe timery:
- opóźnienie włączania **TON** (Time On Delay),
- opóźnienie wyłączania **TOF** (Time Off Delay).

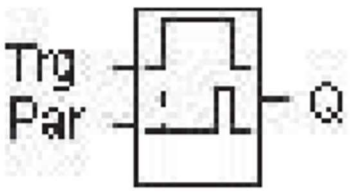
Występują również inne funkcje związane z czasem jak:
- Timer roczny,
- Timer tygodniowy,
- Opóźnienie włącz/wyłącz,
- Impuls czasowy TP (Time Pulse),
- Opóźnienie z podtrzymaniem.

Są również przekaźniki czasowe reagujące na zbocza. Działanie ich jest identyczne z wyjątkiem aktywacji, która opiera się na detekcji zbocza narastającego albo opadającego.

Reprezentacja w LOGO! przedstawia blok zdefiniowany w "user manual". Prezentuje złącza, które posiada blok. Reprezentacja graficzna przedstawia blok, który wykorzystywany będzie w waszych programach.

LOGO! Soft Comfort

- **TON**



Reprezentacja w dokumentacji LOGO!

Opis złączy:

Wejście Trg - Zbocze narastające rozpoczyna odliczanie czasu (zmiana stanu z 0 na 1).

Parametr: T – określa czas, po którym wyjście zostanie wzbudzone (stan na wyjściu bloku zmieni się z 0 na 1).

Wyjście Q - Stan zmienia się z 0 na 1 po upływie zadanego czasu T pod warunkiem, że na wejściu Trg nadal jest 1.

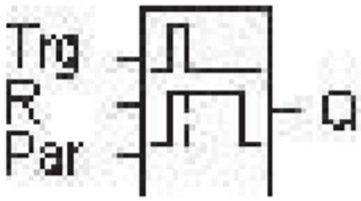
ON Delay reprezentacja graficzna

Działanie bloku wygląda następująco :

po podaniu sygnału „1" na wejście EN rozpoczyna się odmierzanie nastawionego czasu PT po jego odmierzeniu zostaje załączone wyjście Q . Wyjście zostaje wyłączone (deaktywowane) w przypadku zmiany stanu na wejściu EN na „0".

WAŻNE! Blok musi mieć stan wysoki przez cały czas trwania funkcji by działał prawidłowo.

- **TOF**



Reprezentacja w dokumentacji LOGO!

Opis złączy:

Wejście Trg - Zbocze opadające rozpoczyna odliczanie czasu (zmiana stanu z 1 na 0)

Wejście R - Zeruje licznik czasu i wymusza stan 0 na wyjściu Q.

Parametr:

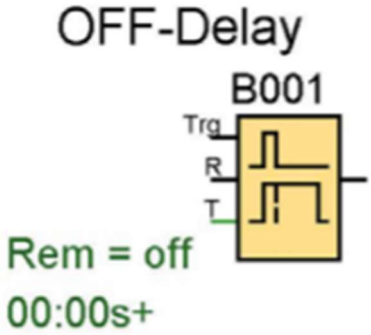
T – określa czas, po którym wyjście zostanie wyzerowane (stan na wyjściu bloku zmieni się z 1 na 0).

Podtrzymanie pamięci:

/ – brak podtrzymania

R – podtrzymanie aktywne

Wyjście Q - Wyjście zostaje wzbudzone pod wpływem sygnału na wejściu Trg i wraca do stanu 0 po upływie zadanego czasu T



OFF Delay reprezentacja graficzna

Działanie bloku wygląda następująco:

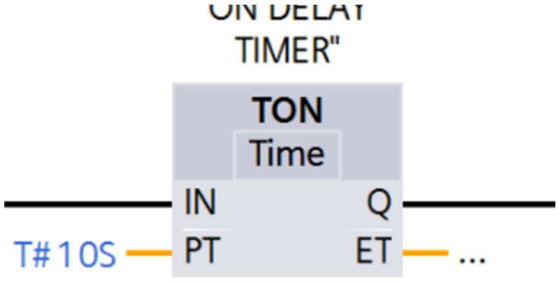
po podaniu sygnału „1” na wejście EN zostaje załączone wyście Q po zmianie sygnału z „1” na „0” rozpoczyna się odmierzanie nastawionego czasu PT . Wyjście Q zostaje wyłączone (deaktywowane) dopiero po odliczeniu nastawionego czasu.

Inaczej: służy do podtrzymania stanu wysokiego przez czas **T**.

Można użyć tego bloku być trzymać **impuls** przez zadany czas.

Blok ten posiada **RESET** o najwyższym priorytecie, który służy do zatrzymania odliczania i natychmiastowego wyłączenia wyjścia.

TIA Portal i e!COCKPIT



- **TON**

Opis złączy:IN – wejście (typu BOOL) uaktywniające timer,

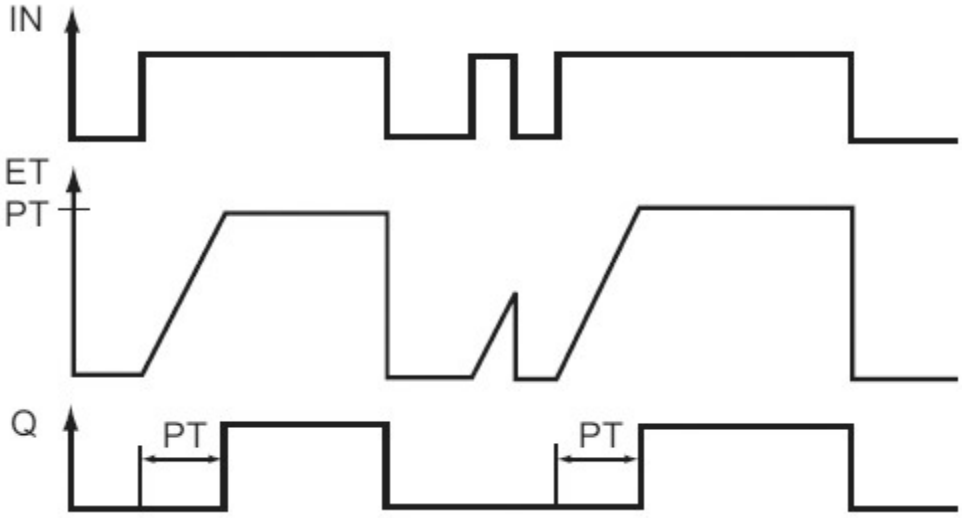
PT – wejście (typu TIME) służące do nastawiania czasu,

Q – wyjście (typu BOOL) timera,

ET – wyjście (typu TIME) określające, ile czasu już upłynęło.

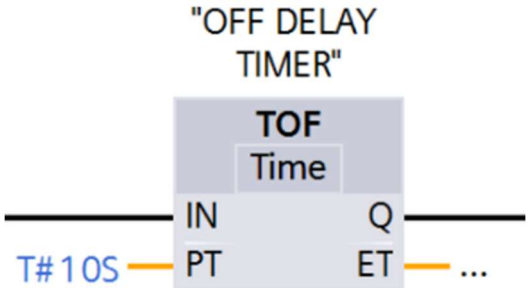
Działanie jest analogiczne do działania w oprogramowaniu LOGO!.

Działanie bloku wygląda następująco: po podaniu sygnału „1” na wejście IN rozpoczyna się odmierzanie nastawionego czasu PT, po jego odmierzeniu zostaje załączone wyjście Q. Wyjście zostaje wyłączone (deaktywowane) w przypadku zmiany stanu na wejściu IN na „0”. Blok musi mieć stan wysoki przez cały czas trwania funkcji, by działał prawidłowo.



Przebiegi czasowe timera TON [1]

- **TOF**



Opis złączy:IN – wejście (typu BOOL) uaktywniające timer,

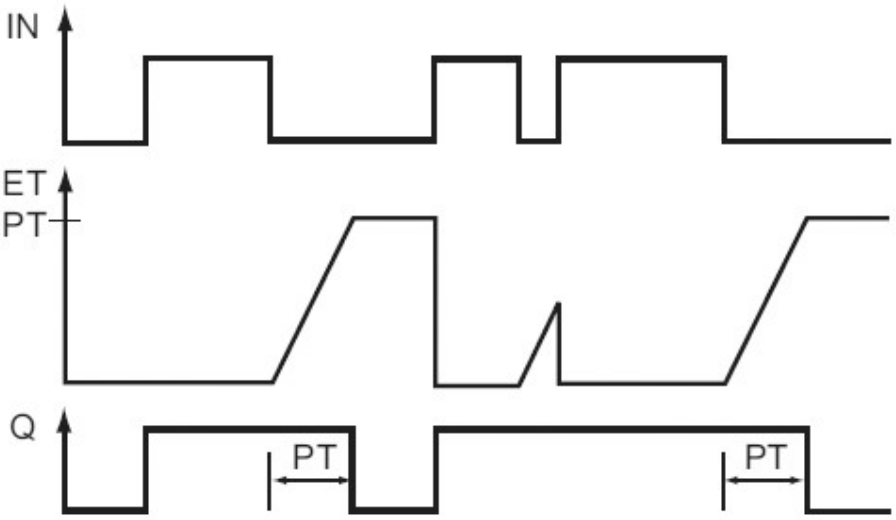
PT – wejście (typu TIME) służące do nastawiania czasu,

Q – wyjście (typu BOOL) timera,

ET – wyjście (typu TIME) określające, ile czasu

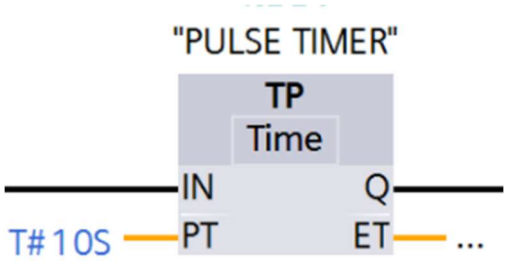
już upłynęło.

Działanie bloku wygląda następująco: po podaniu sygnału „1” na wejście IN zostaje załączone wyście Q po zmianie sygnału z „1” na „0” rozpoczyna się odmierzanie nastawionego czasu PT . Wyjście Q zostaje wyłączone (deaktywowane) dopiero po odliczeniu nastawionego czasu.



Przebiegi czasowe timera TOF [1]

- **TP**



Opis złączy:

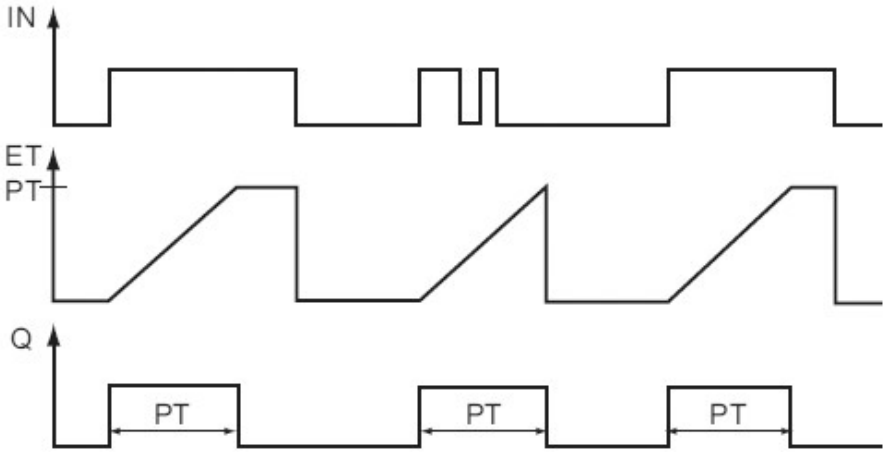
IN – wejście (typu BOOL) uaktywniające timer,

PT – wejście (typu TIME) służące do nastawiania czasu,

Q – wyjście (typu BOOL) timera,

ET – wyjście (typu TIME) określające, ile czasu już upłynęło.

Działanie bloku wygląda następująco : po podaniu sygnału „1” na wejście IN zostaje załączone wyjście Q oraz rozpoczyna się odmierzanie nastawionego czasu PT. Wyjście zostaje wyłączone (deaktywowane) po odliczeniu nastawionego czasu.



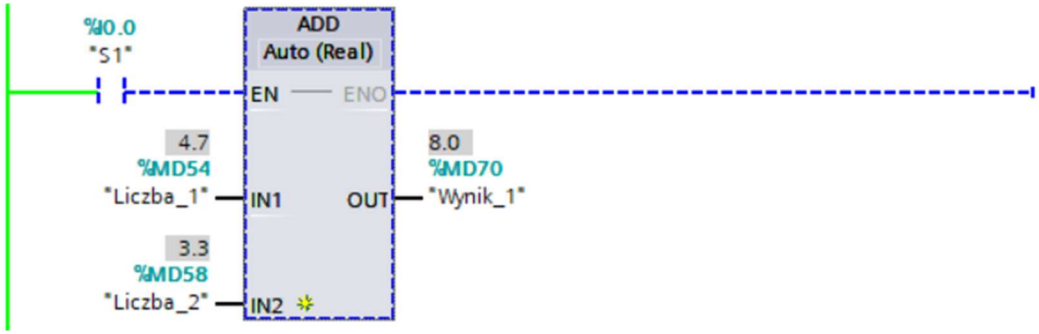
Przebiegi czasowe timera TP [1]

6. Operacje matematyczne

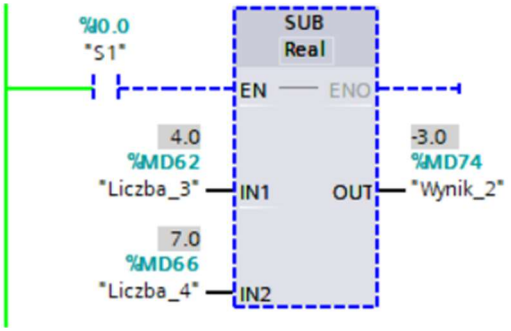
A. Dodawanie/odejmowanie:

Funkcja ADD wykonuje działanie sumy liczb otrzymanych z wejść IN1 i IN2. Aby działanie zostało wykonane, należy podać stan wysoki na wejście EN, wynik działania odczytujemy z wyjścia OUT.

.

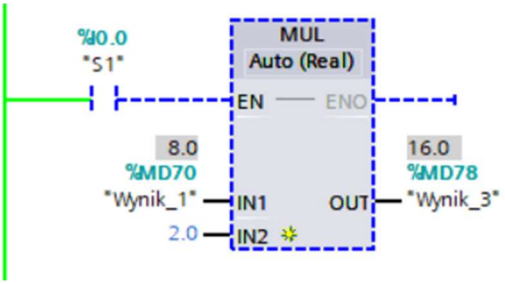


Funkcja SUB wykonuje działanie różnicy liczb otrzymanych z wejść IN1 i IN2. Aby działanie zostało wykonane, należy podać stan wysoki na wejście EN, wynik działania odczytujemy z wyjścia OUT.

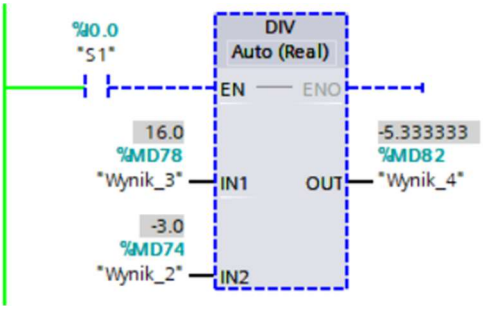


B. **Mnożenie/dzielnie**:

Funkcja MUL oblicza iloczyn liczb otrzymanych z wejść IN1 i IN2. Aby działanie zostało wykonane, należy podać stan wysoki na wejście EN, wynik działania odczytujemy z wyjścia OUT.

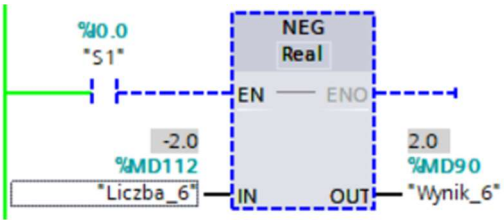


Funkcja DIV oblicza iloraz liczb otrzymanych z wejść IN1 i IN2. Aby działanie zostało wykonane, należy podać stan wysoki na wejście EN, wynik działania odczytujemy z wyjścia OUT.

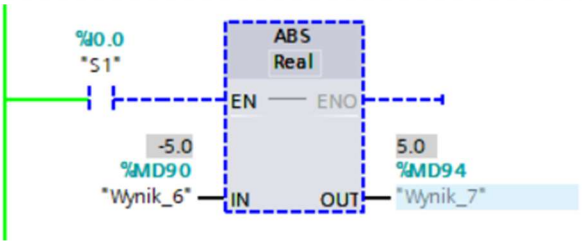


C. **NEG, ABS, MOD, INC, DEC:**

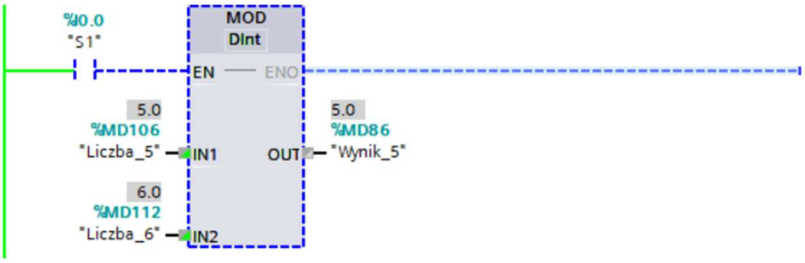
Funkcja NEG (wartość przeciwna) na wyjściu OUT podaje liczbę przeciwną do otrzymanej na IN.



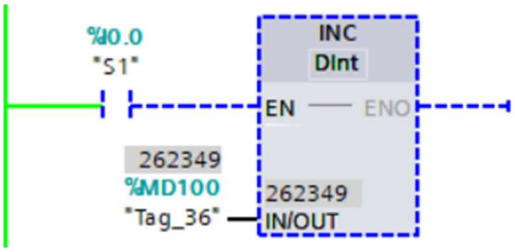
Funkcja ABS (wartość bezwzględna) służy do wyznaczania wartości bezwzględnej liczby całkowitej/rzeczywistej.



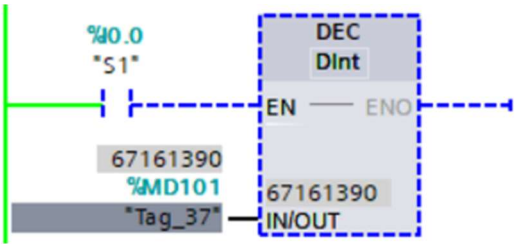
Funkcja MOD (modulo) wykonuje operację dzielenia z resztą. Aby działanie zostało wykonane, należy podać stan wysoki na wejście EN. Parametry IN1 oraz IN2 muszą być tego samego typu.



Funkcja INC (inkrementacja) wykonuje [$IN/OUT \coloneqq IN/OUT + 1$].

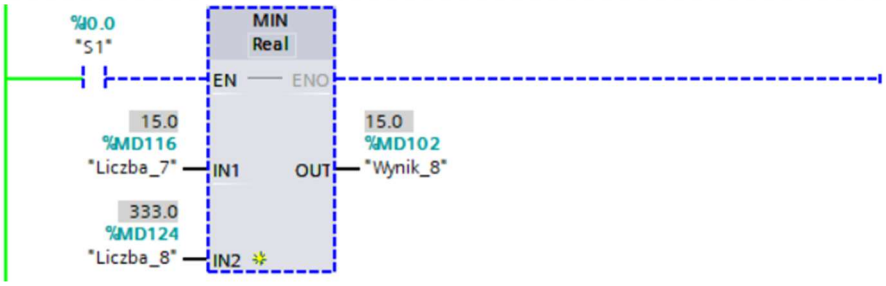


Funkcja DEC (dekrementacja) wykonuje [$IN/OUT \coloneqq IN/OUT - 1$].

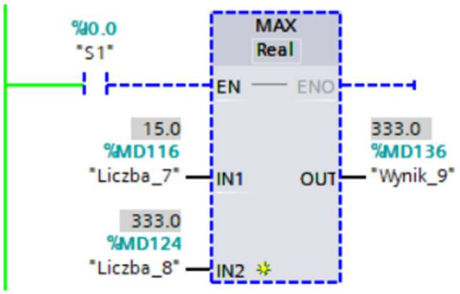


D. **MIN,MAX,LIMIT**:

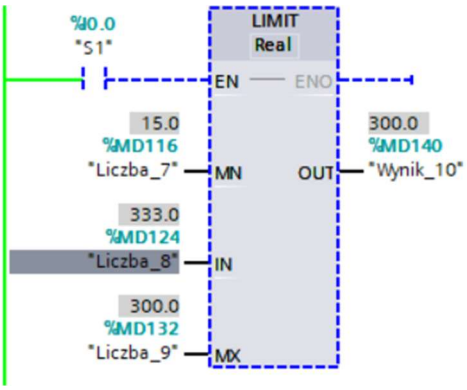
Funkcja MIN (wartość minimalna) spośród otrzymywanych wartości wybiera najmniejszą i oddaje ją na wyjściu OUT.



Funkcja MAX (wartość maksymalna) spośród otrzymywanych wartości wybiera największą i oddaje ją na wyjściu OUT.

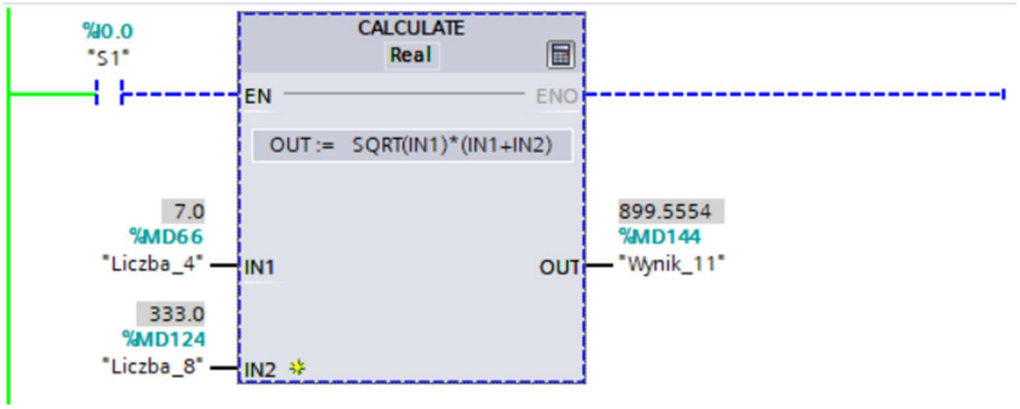


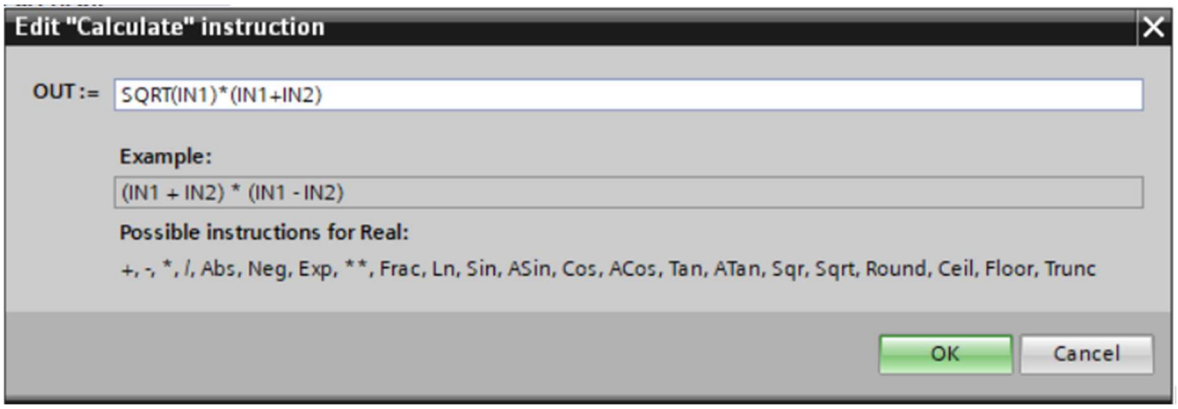
Funkcja LIMIT (wartość z przedziału) sprawdza, czy otrzymana wartość mieści się w przedziale, jeżeli tak, to zostanie ona wyświetlana, w przeciwna wypadku blok wyświetli jedną z granic przedziału (zależy czy liczba jest mniejsza/większa niż przedział).



E. **CALCULATE**:

Funkcja CALCULATE oblicza działanie, które jej zadamy w oknie [OUT :=].



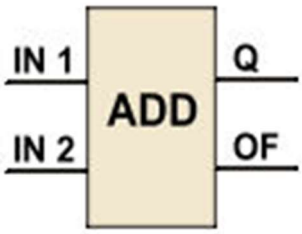


Jeżeli przy wejściach bloku występuje symbol żółtej gwiazdy, to oznacza, że można do niego dodać więcej wartości wejściowych.

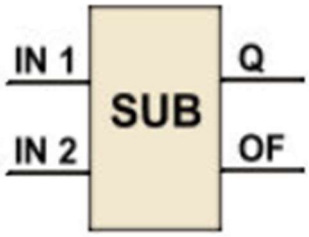
LOGO! Soft Comfort:

Podstawowe operacje matematyczne:

Funkcja ADD (dodawanie), sumuje IN1 i IN2, wynik podaje na Q albo na OF, kiedy wynik jest błędny (poza zakresem zmiennej).

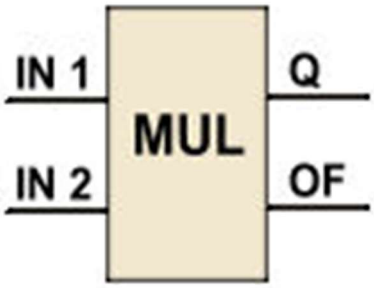


Funkcja SUB (odejmowanie), różnica IN1 i IN2 jest podawana na Q albo na OF, kiedy wynik jest błędny.

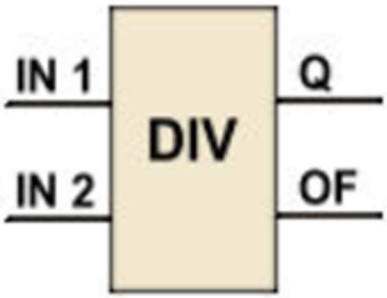


.

Funkcja MUL (mnożenie), iloczyn IN1 i IN2 jest podawany na Q albo na OF, kiedy wynik jest błędny.



Funkcja DIV (dzielenie), iloraz IN1 i IN2 jest podawany na Q albo na OF, kiedy wynik jest błędny.



Dodatek – pełna lista typów zmiennych

Typy zmiennych arytmetycznych

- **Sint** (short integer) – za pomocą tego typu określa się zmienne, które mogą przyjmować wartości całkowite z zakresu od -128 do 127. Do ich zapisu w pamięci sterownika wykorzystuje się 8 bitów.
- **USInt** (unsigned short integer) – kolejny typ przeznaczony do zapisu liczby całkowitej, z tymże zmienne nie mogą przyjmować wartości ujemnych. Również są zapisywane za pomocą 8 bitów, a zakres wartości wynosi od 0 do 255
- **Int** (integer) – najbardziej znany typ zmiennych, również znany w językach programowania takich jak C, C++, Java czy Python. Do zapisu tego typu zmiennej wykorzystuje się 16 bitów (a więc 1 słowo), a zakres wartości przyjmowanych przez zmienne typu Int wynosi od -32 768 do 32 767.
- **UInt** (unsigned integer) – tak jak w poprzednim przypadku, wartości zmiennych zajmują 16 bitów w pamięci sterownika. Zmienna tego typu nie może przyjmować wartości ujemnych, co przekłada się na zakres, który wynosi od 0 do 65 535.
- **DInt** (double integer) – kolejny typ umożliwiający zapis liczby całkowitej, na który przeznacza się aż 32 bity (2 słowa). Zakres przyjmowanych wartości wynosi od –2 147 483 648 do 2 147 483 647.
- **UDInt** (unsigned double integer) – typ zmiennych całkowitych, który również nie może przyjmować wartości ujemnych, w związku z czym wartości mogą wynosić od 0 do 4 294 967 295.
- **Real** – typ zmiennoprzecinkowy wykorzystywany do zapisu wartości rzeczywistych z zakresu: od -3.402E37 do -1.176E-35 (zakres liczb ujemnych), +1.176E-35 do +3.402E37 (zakres liczb dodatnich).
- **LReal** (long real) – typ, który również służy do zapisu liczb rzeczywistych o większej dokładności niż w przypadku typy Real, co przekłada się na większą ilość wykorzystywanych bitów (wartość wynosi 64).

Zakres wartości przyjmowanych przez typ rzeczywisty zależy od parametrów sprzętowych sterowników.

Dodatkowe typy zmiennych

- **Char** – typ przeznaczony do zapisu znaków (np. ‘P’, ‘L’, ‘C’ czy ‘+’). Zajmuje on 8 bitów w pamięci.
- **String** – typ, który służy do zapisu ciągu znaków w standardzie Unicode (np. ‘PLC’, ‘Automatyk’). W zależności od wybranego przez nas standardu, znaki są kodowane w sposób 7 bitowy (ASCII), 16 bitowy (UTF-16) lub 32 bitowy (UTF-32). Innymi słowy ilość bitów przeznaczona na zapis wartości jest zmienna.
- **Data** – specjalna zmienna przeznaczona do zapisu dat. Na zapis tego typu zmiennych przeznacza się wartość 16 bitów. Zakres dat w przypadku sterowników firmy Siemens wynosi: od D#1990–1–1 do D#2168–12–31.
- **Time** – zmienna, która z kolei służy do zapisu wartości czasu. Przeznacza się na ten typ aż 32 bity, w związku z czym w sterownikach S7-1200 oraz S7-1500 wartości

mogą być zapisane z dokładnością do milisekund. Dlatego też zakres wartości może wynosić od T#–24d_20h_31m_23s_648ms do T#24d_20h_31m_23s_647ms.

- **Time_of_day** – typ zmiennych przeznaczony do zapisu wartości godziny (czasu) dnia. Również zawiera 32 bity, a jej zakres wartości może wynosić od TOD#0:0:0.0 do TOD#23:59:59.999
- **DTL** – specjalny typ, który jest wykorzystywany do zapisu konkretnej daty oraz godziny w danym dniu. Zakres wartości przyjmowanych przez konkretne sterowniki firmy Siemens może wynosić od DTL#1970–01–01–00:00:00.0 do DTL#2554–12–31–23:59:59.999 999 999.

Zakres wartości typów „czasowych” mogą się różnić od parametrów sprzętowych konkretnych sterowników.

Źródła

[1] http://s7-scl.pl/funkcje-czasowe-odmierzanie/, dostęp 21.03.2021.