Programowanie Sterowników Przemysłowych

Programowanie sterowników S7-1200 w języku SCL

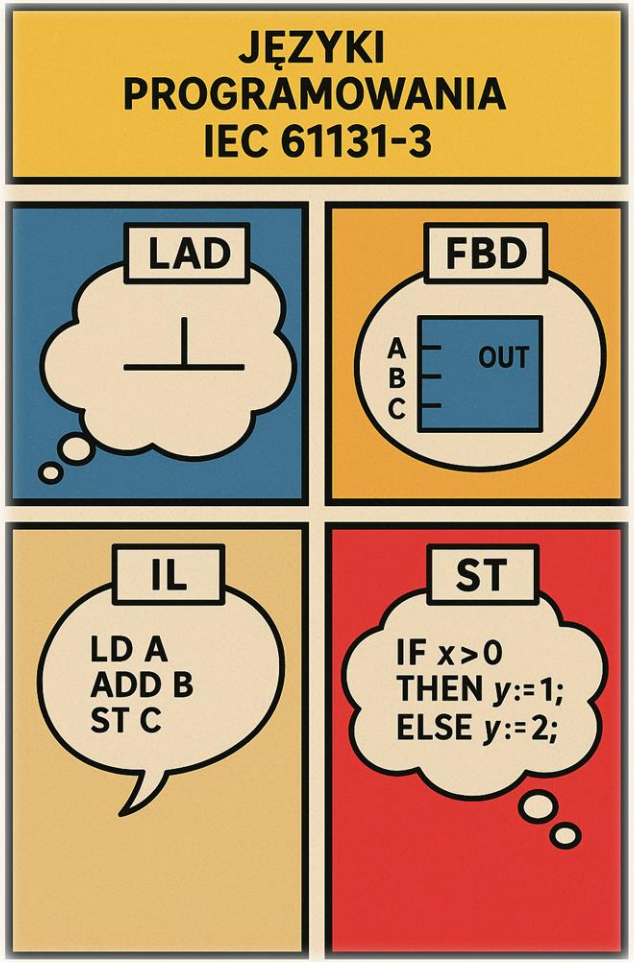
Paweł Strączyński

p.straczynski@tu.kielce.pl

Katedra Urządzeń Elektrycznych i Automatyki

Politechnika Świętokrzyska

Języki programowania sterowników PLC według normy IEC-61131-3

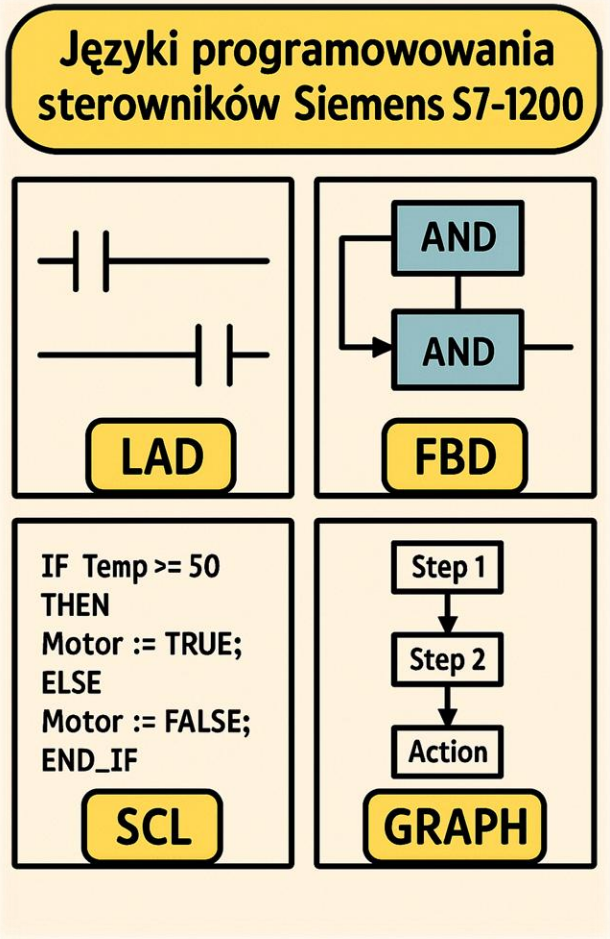


Języki zdefiniowane w normie IEC 61131-3:

- LAD (*Ladder Diagram*) – język drabinkowy
- FBD (*Function Block Diagram*) – schemat blokowy
- ST (*Structured Text*) – tekst strukturalny
- IL (*Instruction List*) – lista rozkazów (wycofany w nowszych wersjach normy)
- SFC (*Sequential Function Chart*) – diagram sekwencyjny

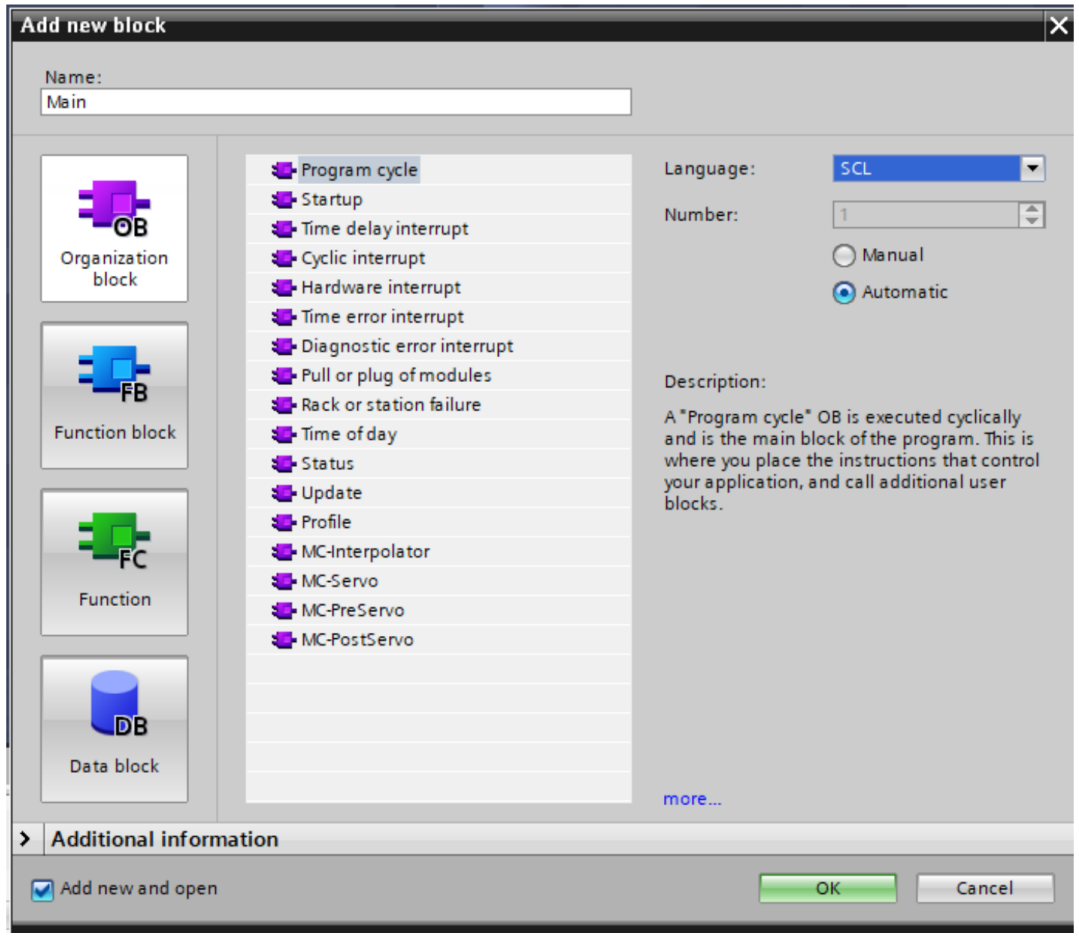
Języki programowania w Siemens S7-1200 (TIA Portal)

Język	Zgodność z IEC 61131-3	Obsługa w S7-1200	Uwagi
LAD	✅ Tak	✅ Tak	Graficzny język popularny w przemyśle
FBD	✅ Tak	✅ Tak	Schemat bloków funkcyjnych
ST / SCL	✅ Tak (ST)	✅ Tak (jako SCL)	Structured Control Language – odpowiednik ST w Siemens
IL	✅ Tak (już przestarzały)	❌ Nie	Nieobsługiwany przez TIA Portal ani S7-1200
SFC	✅ Tak	🔶 Opcjonalnie (GRAPH)	Możliwy jako rozszerzenie w TIA Portal (programowanie sekwencyjne)

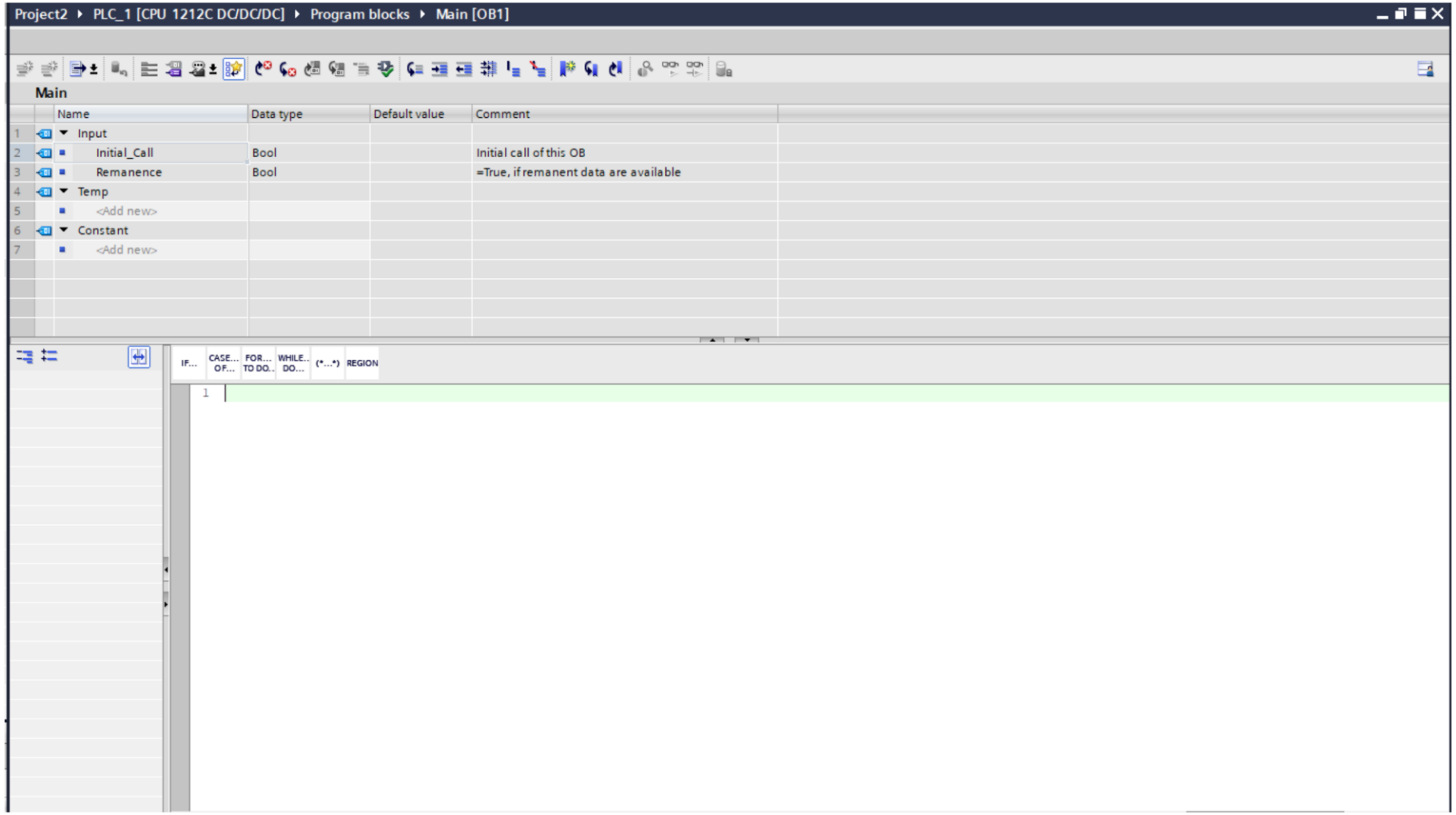


Język S7-SCL - konfiguracja języka w TIA Portal

Język SCL (ang. Structured Control Language) jest językiem tekstu strukturalnego (ST) wysokiego poziomu bazującym na Pascalu. SCL jest zgodny z normą IEC 61131-3 dotyczącą sposobów programowania sterowników PLC. Jako język wysokiego poziomu posiada typowe elementy języka takie jak możliwość definiowania zmiennych, instrukcje warunkowe, pętle, operacje porównania, operacje bitowe itd.



Okno edytora programu w języku SCL w środowisku TIA Portal



Podstawowe instrukcje i operacje bitowe w języku SCL

1. Przepisania wartości We1 (I0.0) na wyjcie Q0.0 - Wy1,
2. Przepisania wyniku operacji iloczynu logicznego We1 · We1 na wyjcie Wy2,
3. Przepisania wyniku operacji sumy logicznej We1 + We1 na wyjście Wy3,
4. Przepisania na wyjście Wy4 negacji wejścia We1,
5. Przepisania wyniku operacji alternatywy wykluczającej We1 oraz We1 na Wy5.

```
1  "Wy1" := "We1";
2  "Wy2" := "We1" AND "We2";
3  "Wy3" := "We1" OR "We2";
4  "Wy4" := NOT "We1";
5  "Wy5" := "We1" XOR "We2";
```

	Name	Data type	Address	Retain	Accessible...	Writable...	Visible...
	We1	Bool	%I0.0		✓	✓	✓
	Wy1	Bool	%Q0.0		✓	✓	✓
	We2	Bool	%I0.1		✓	✓	✓
	Wy2	Bool	%Q0.1		✓	✓	✓
	Wy3	Bool	%Q0.2		✓	✓	✓
	Wy4	Bool	%Q0.3		✓	✓	✓
	Wy5	Bool	%Q0.4		✓	✓	✓

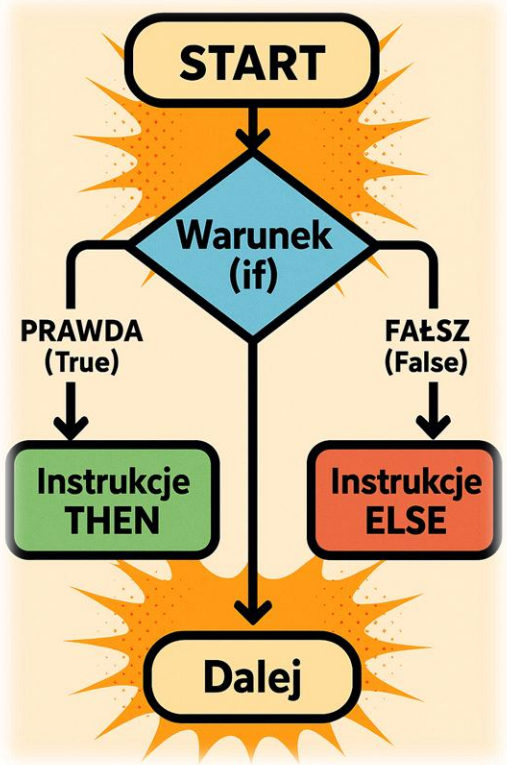
Detekcja zbocza sygnału

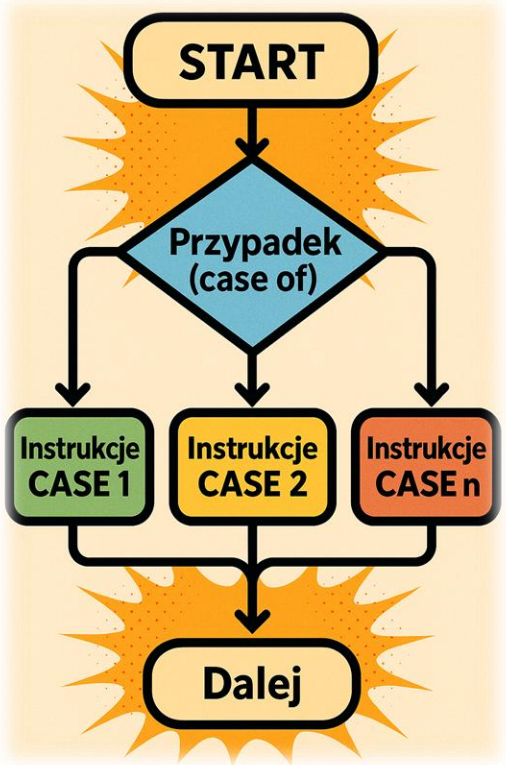
W języku SCL podobnie jak w języku drabinkowym istnieje możliwość detekcji zbocza sygnału. W bibliotece funkcji dostępne są funkcje `R_TRIG` oraz `F_TRIG` odpowiedzialne za detekcje zbocza narastającego (*rising*) oraz opadającego (*faling*). Funkcje te powiązane są z blokiem danych, gdyż aby detekcja zbocza była możliwa musi być pamiętany poprzedni stan sygnału.


```
1 "F_TRIG_DB"(CLK:="We1",
2             Q=>"Var1");
3 "IEC_Timer_0_DB_1".TP(IN := "Var1",
4                       PT := T#300ms,
5                       Q => "Wy2");
6
```

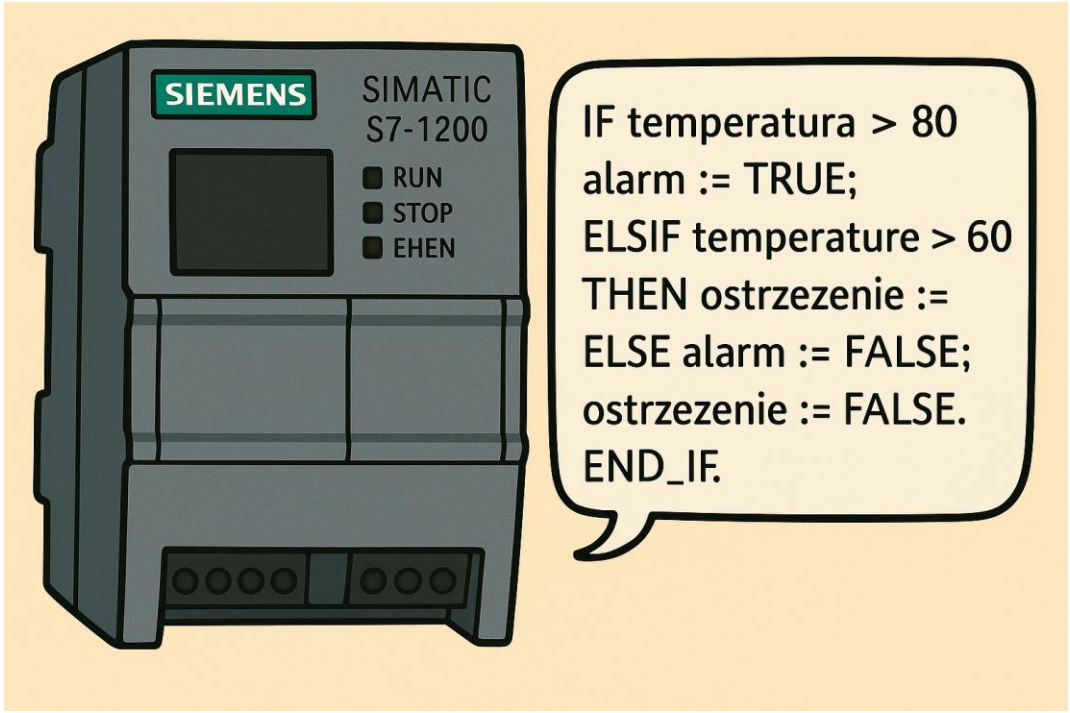
Instrukcje warunkowe

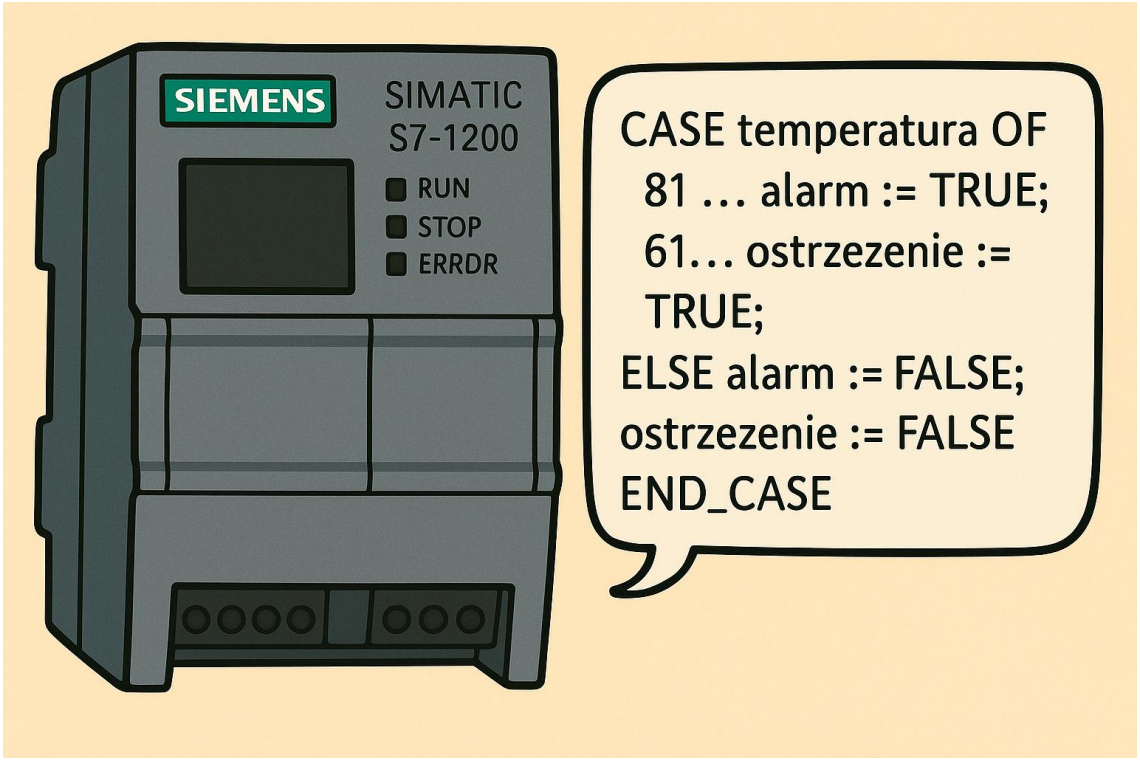
- Służą do wykonania różnych działań w zależności od warunku logicznego sygnału.
- Typowe konstrukcje to:
  - IF…THEN…ELSE
  - CASE…OF





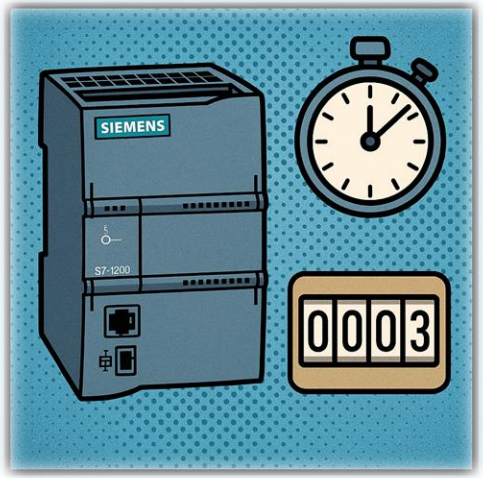
Instrukcje warunkowe





Timery i liczniki

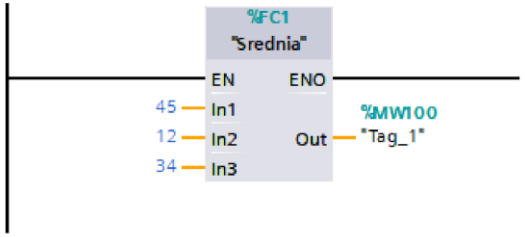
W języku SCL znaleźć można wszystkie rodzaje układów do odmierzania czasu i liczniki jakie dostępne są w języku drabinkowym (LAD). Instrukcje timerów dostępne są w zakładce `Instrukction - > Basic Instructions - > Timer operations` jak na rysunku. Instrukcje liczników w zakładce `Counter operations`. Timer oraz Licznik można dodać metodą przeciągnij i upuść - tak samo każdy inny blok programu.

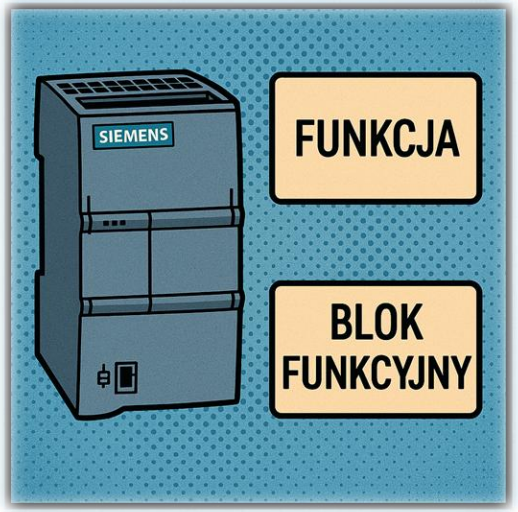


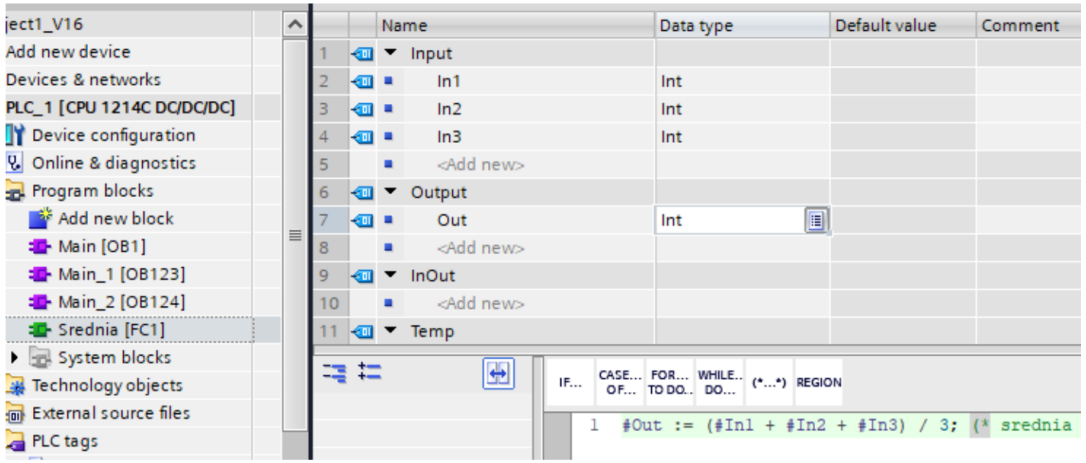
Timer operations		V1.0
TP	Generate pulse	V1.0
TON	Generate on-delay	V1.0
TOF	Generate off-delay	V1.0
TONR	Time accumulator	V1.0
RESET_TIMER	Reset timer	V1.0*
PRESET_TIMER	Load time duration	V1.0*

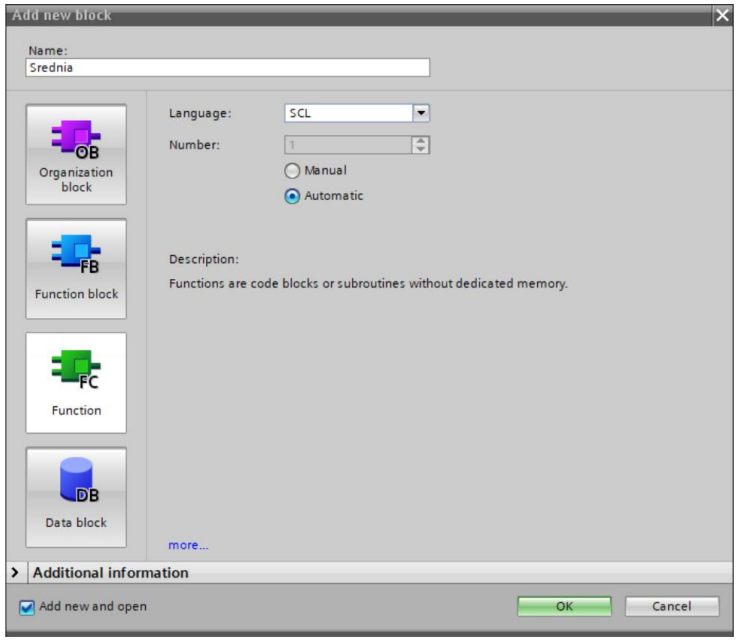
Funkcje i bloki funkcyjne

Język SCL podobnie jak LAD i FBD umożliwia tworzenie funkcji i bloków funkcyjnych (funkcji zawierających blok danych). Funkcje i bloki funkcji pisane w języku SCL mogą być wywoływane w programie głównym pisanym zarówno w SCL-u jak i języku drabinkowym czy języku bloków funkcyjnych.





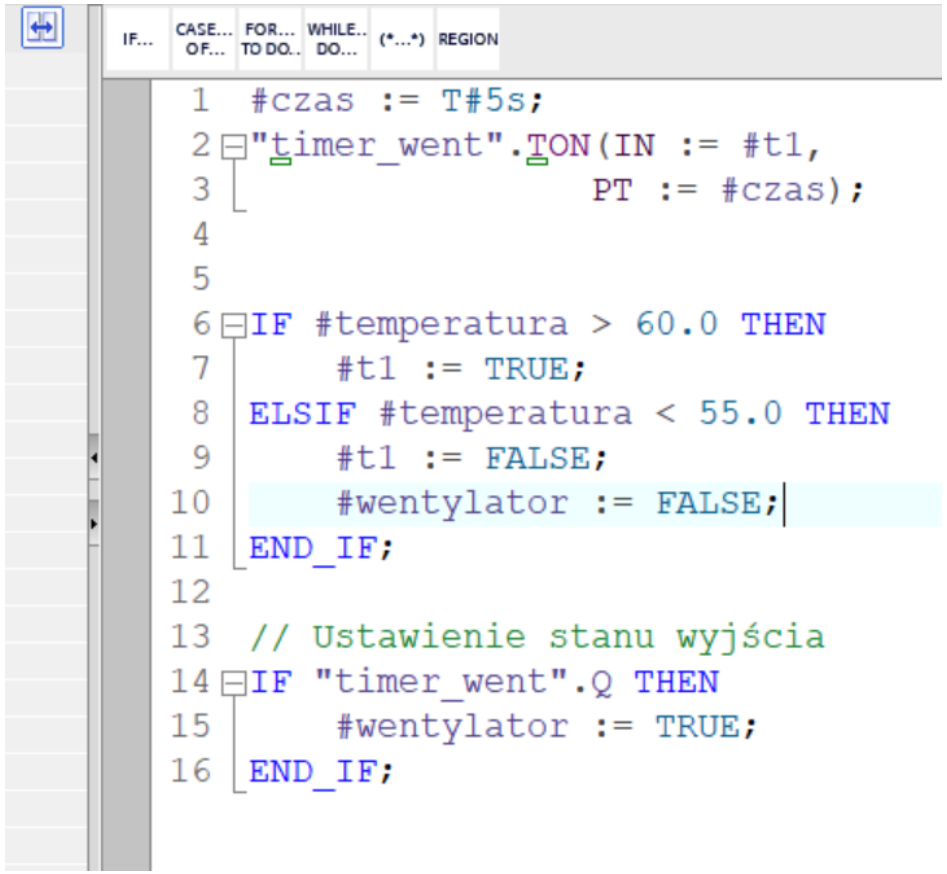




Przykład 1

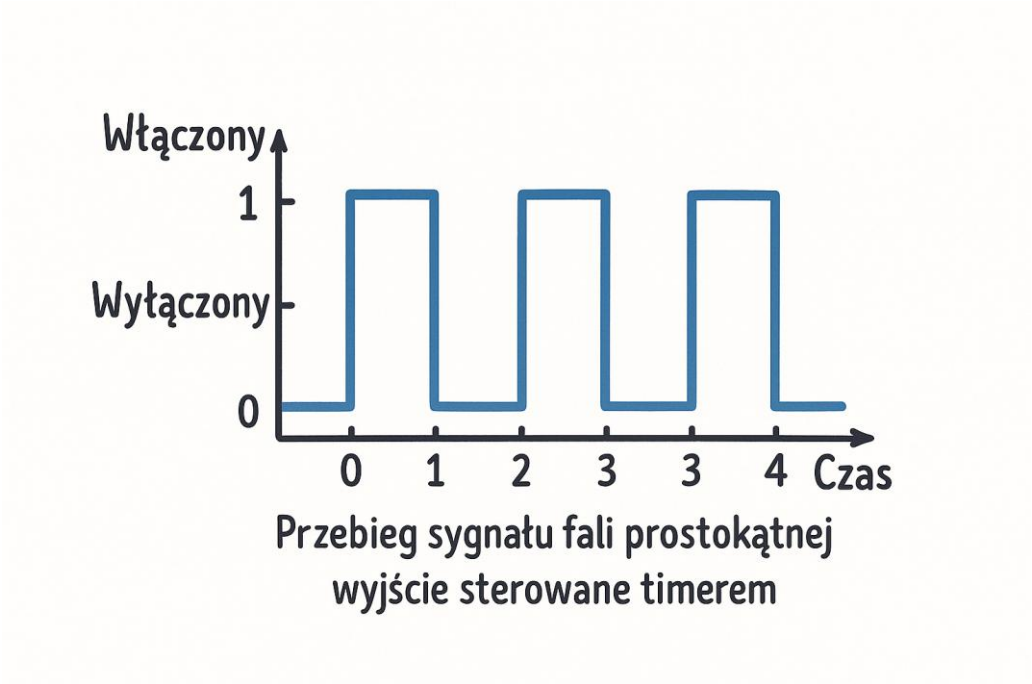
Sterowanie pracą wentylatora

- Gdy temperatura > 60.0, uruchamiamy timer. Po 5 sekundach Q timera przyjmuje wartość TRUE → załączenie wentylatora.
- Gdy temperatura < 55.0, timer jest resetowany, wentylator wyłączany natychmiast.
- Temperatura w zakresie 55–60°C nie zmienia stanu — histereza zabezpiecza przed „klapaniem”.



Przykład 2

Generator fali prostokątnej




```
IF...  CASE... OF...  FOR... TO DO..  WHILE.. DO...  (*...*)  REGION

 1  #czas := T#5s;
 2 "timer_went".TON(IN := TRUE,
 3                  PT := #czas);
 4
 5
 6 IF #temperatura > 60.0 THEN
 7     #t1 := TRUE;
 8 ELSIF #temperatura < 55.0 THEN
 9     #t1 := FALSE;
10     #wentylator := FALSE;
11 END_IF;
12
13 // Ustawienie stanu wyjścia
14 IF "timer_went".Q THEN
15     #wentylator := TRUE;
16 END_IF;
```