

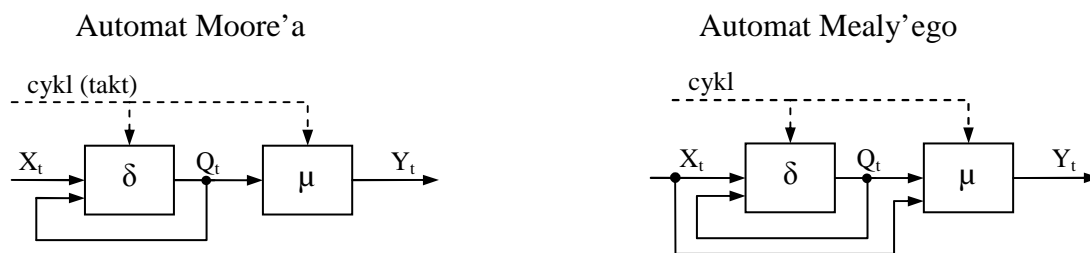
UKŁADY SEKWENCYJNE

Wprowadzenie. Napełnianie i opróżnianie. Programowanie. Przerzutnik RS. Układ Start–Stop. Automaty w języku LD. Zbiornik z trzema zaworami. Sterowanie symulowanym zbiornikiem. Urządzenia automatyki i sterowania.

WPROWADZENIE

1. Automaty Moore'a i Mealy'ego

Układy sekwencyjne nazywa się zwykle automatami.



Funkcja przejścia:	$\begin{cases} Q_{t+1} = \delta(Q_t, X_t) \\ Y_t = \mu(Q_t) \end{cases}$
Funkcja wyjścia:	$\begin{cases} Q_{t+1} = \delta(Q_t, X_t) \\ Y_t = \mu(Q_t, X_t) \end{cases}$

W automacie Moore'a wyjście zależy wyłącznie od stanu wewnętrznego Q (głównie takie automaty będą rozpatrywane). Stan wewnętrzny jest reprezentowany przez pamięć. Zmienna *stan* występuje w typowych realizacjach programowych układów sekwencyjnych.

2. Automaty synchroniczne i asynchroniczne

Automat synchroniczny – zmiany stanu następują w ściśle określonych momentach czasu (w automatach asynchronicznych tak nie jest).

Realizacje programowe układów sekwencyjnych, zwłaszcza złożonych, są zwykle automatami synchronicznymi. Automaty asynchroniczne są prostsze i dopuszczają zmiany cyklu (tzn. czasu wykonania programu).

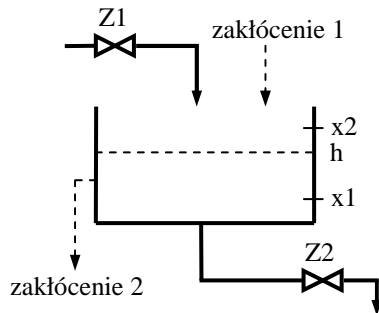
3. Rozpoznanie układu kombinacyjnego i sekwencyjnego

Jeżeli na podstawie obserwacji aktualnych wejść nie da się wprost stwierdzić, co należy uczynić, to mamy do czynienia z układem sekwencyjnym, tzn. układem z pamięcią (nie można podjąć decyzji znając tylko aktualne wejścia; trzeba rozważyć, co zdarzyło się wcześniej).

NAPEŁNIANIE I OPRÓŻNIANIE

1. Problem

Chodzi o cykliczne naprzemienne napełnianie i opróżnianie zbiornika za pomocą zaworów Z1, Z2.



$h, x1, x2$ – poziomy: aktualny, dolny, górny

Z1, Z2 – zawory: wlewowy, spustowy

Specyfikacja (algorytm):

- $h < x1 \rightarrow$ otwórz Z1, zamknij Z2
- $h > x2 \rightarrow$ zamknij Z1, otwórz Z2

Pytanie. Co uczynić przy poziomie pośrednim?

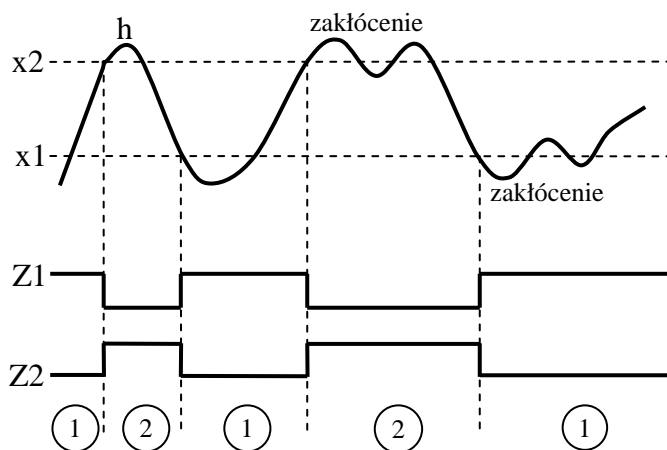
Ponieważ odpowiedź *nie jest* natychmiastowa, bo potrzebne jest rozważenie stanu poprzedniego, więc mamy do czynienia z układem sekwencyjnym.

2. Projektowanie systematyczne

1. Przebiegi czasowe
2. Definiowane stanów automatu
3. Graf stanów
4. Program: C, ST, LD

3. Przebiegi czasowe

Kreśli się je w uzgodnieniu z formułującym zadanie.



4. Definiowane stanów

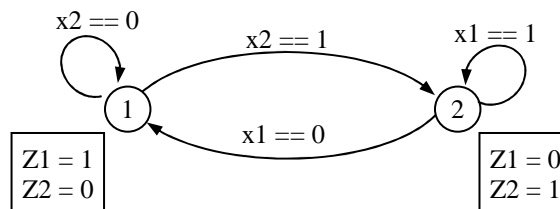
Należy przede wszystkim zwracać uwagę na te momenty, gdy automat ma zareagować, tzn. zmienić wyjście (sterowanie). Pomiedzy tymi momentami mamy zwykle do czynienia ze stanami stabilnymi, które należy odpowiednio nazwać. Są one podstawą do dalszej analizy.

- ① – napełnianie, ② – opróżnianie

Stany stabilne zaznacza się na przebiegu (zob. przebiegi).

5. Graf stanów

- Konstruowanie:
- 1) stany – małe kółka z nazwami
 - 2) przejścia – zorientowane strzałki z opisem warunku przejścia (wynika z treści zadania lub z przebiegu)
 - 3) wyjścia – prostokąty z wartościami odpowiednio do stanu
 - 4) pętle pozostawania w stanie – opis warunku (opcjonalnie).



Uwaga. Opisy grafu odpowiadają instrukcjom języka C.

PROGRAMOWANIE

1. Język C

• Instrukcja *switch*

Jest to podstawowa instrukcja do programowania sekwencji. Kod programu jest bezpośrednim odzwierciedleniem grafu automatu. Warunki zapisane przy pętlach nie występują w programie (chodzi o pozostawanie w stanie).

```
char x1,x2,Z1,Z2;
char stan=1;
...
switch(stan)
{
    case 1: Z1=1; Z2=0;
            if(x2==1) stan=2;
            break;
    case 2: Z1=0; Z2=1;
            if(x1==0) stan=1;
}
```

- **if ... else**

```

if(stan==1)
  { Z1=1; Z2=0; if(x2) stan=2; }
else
if(stan==2)
  { Z1=0; Z2=1; if(!x1) stan=1; }

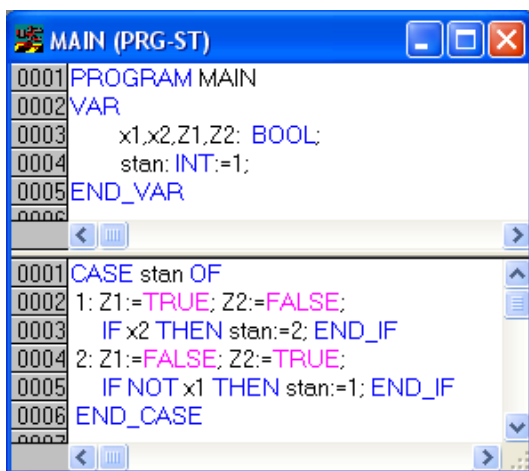
```

Uwaga. else jest potrzebne, aby w danym cyklu obliczeń wykonać tylko jedną instrukcję if.

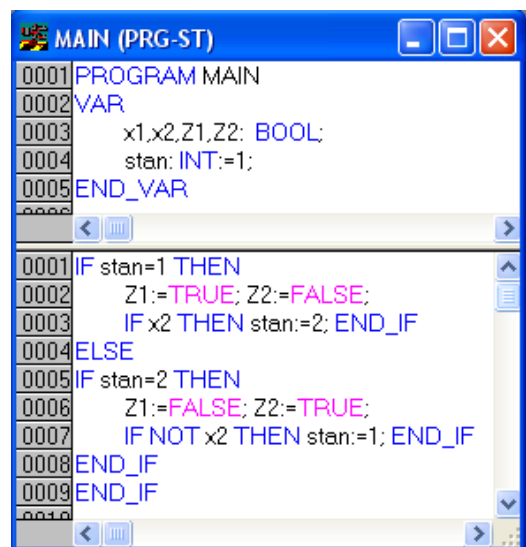
Rozwiązanie *if ... else* jest stosowane w prostych językach skryptowych, w których brak odpowiednika instrukcji *switch*.

2. Język ST

Odpowiednikiem instrukcji *switch* jest **CASE ... OF**.



Zbiornik – 2 zawory CASE, IF



3. Niepoprawne pomiary

Wymaganie technologiczne (przykładowe). W przypadku niepoprawnych pomiarów obydwie zawory należy zamknąć.

x1	0	1	niepoprawność (zero) → $x2 \cdot \overline{x1}$
x2	0	1	
	1	1	

```

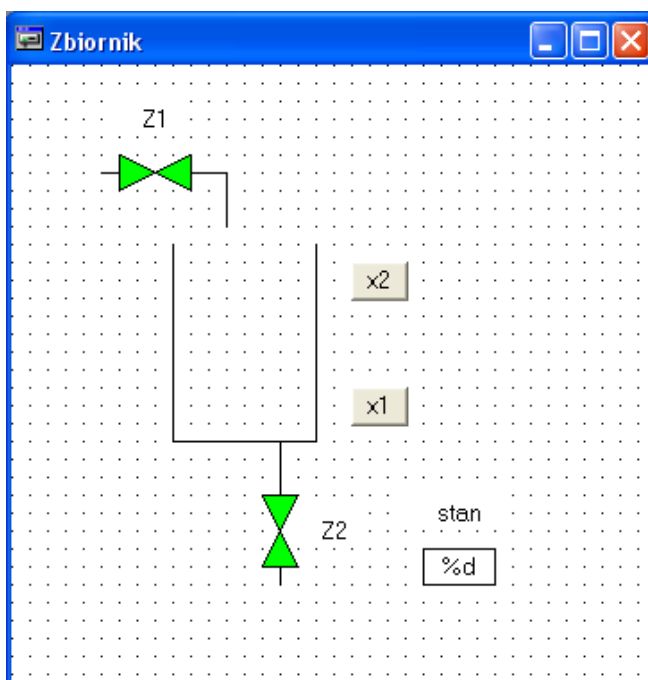
MAIN (PRG-ST)
0001 PROGRAM MAIN
0002 VAR
0003   x1,x2,Z1,Z2: BOOL;   stan: INT:=1;
0004 END_VAR
0005
0001 IF x2 AND NOT x1 THEN
0002   Z1:=FALSE; Z2:=FALSE;
0003 ELSE
0004   CASE stan OF
0005     1: Z1:=TRUE; Z2:=FALSE;
0006       IF x2 THEN stan:=2; END_IF
0007     2: Z1:=FALSE; Z2:=TRUE;
0008       IF NOT x1 THEN stan:=1; END_IF
0009   END_CASE
0010 END_IF

```

Zbiornik – 2 zawory POPRAWNOŚĆ

4. Prosta wizualizacja

- Ikona *Visualization* (na dole) > eksplorator *Visualizations*, menu *Add Object > Name ...* Zbiornik
- Docelowy obraz


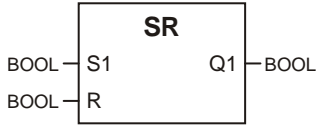
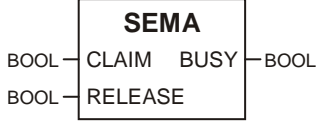


- Elementy
 Zawory: *Polygon, Variables – Change color, MAIN.Z1/Z2, Colors – zielony/czerwony*
 Przyciski: *Text, Input – Toggle variable, MAIN.x1/x2*
 Wyświetlacz stanu: *Rectangle, Text – %d (format), Variables – Text display, MAIN.stan*
 Napisy: *Rectangle, Text – Z1, Z2, stan, Colors – No frame color.*
- Porównać wizualizację układów:
 - nie uwzględniającego możliwych niepoprawnych pomiarów
 - zamykającego zawory przy niepoprawnych pomiarach.

PRZERZUTNIK RS

1. Bloki dwustanowe normy PN-EN 61131-3

- Oznaczenia wejść/wyjść na poniższych schematach i w opisach są następujące:
 Q – wyjście typu BOOL,
 R – wejście zerowania logicznego (*reset*),
 S – wejście ustawiające (*set*),
 Wartości początkowe wszystkich wejść są zerowe.

Bloki dwustanowe	
	RS – przerzutnik typu RS (<i>RS flip-flop</i>) $Q1 = \text{NOT } R1 \text{ AND } (Q1_{n-1} \text{ OR } S)$
	SR – przerzutnik typu SR (<i>SR flip-flop</i>) $Q1 = S1 \text{ OR } (\text{NOT } R \text{ AND } Q1_{n-1})$
	SEMA – semafor (<i>semaphore</i>) $\text{BUSY} = \text{TRUE}$ dla $\text{CLAIM} = \text{TRUE}$ $\text{BUSY} = \text{FALSE}$ dla $\text{RELEASE} = \text{TRUE}$ i $\text{CLAIM} = \text{FALSE}$

2. Realizacja przerzutnika RS według wzoru z normy

- ST

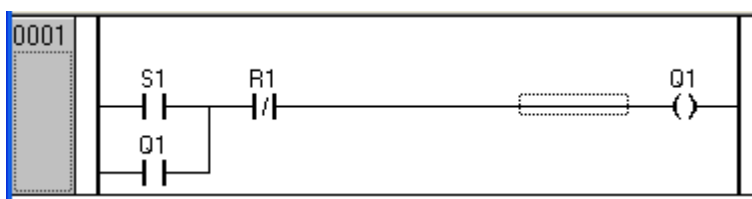
```

MAIN (PRG-ST)
0001 PROGRAM MAIN
0002 VAR
0003   R1,S1,Q1: BOOL;
0004 END_VAR
0001 Q1 := (S1 OR Q1) AND NOT R1;
0002
0003
  
```

RS – układ podtrzymujący (zapamiętujący),
 przełącznik załącz/wyłącz

Przerzutnik RS – ST wzór

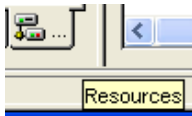
- LD



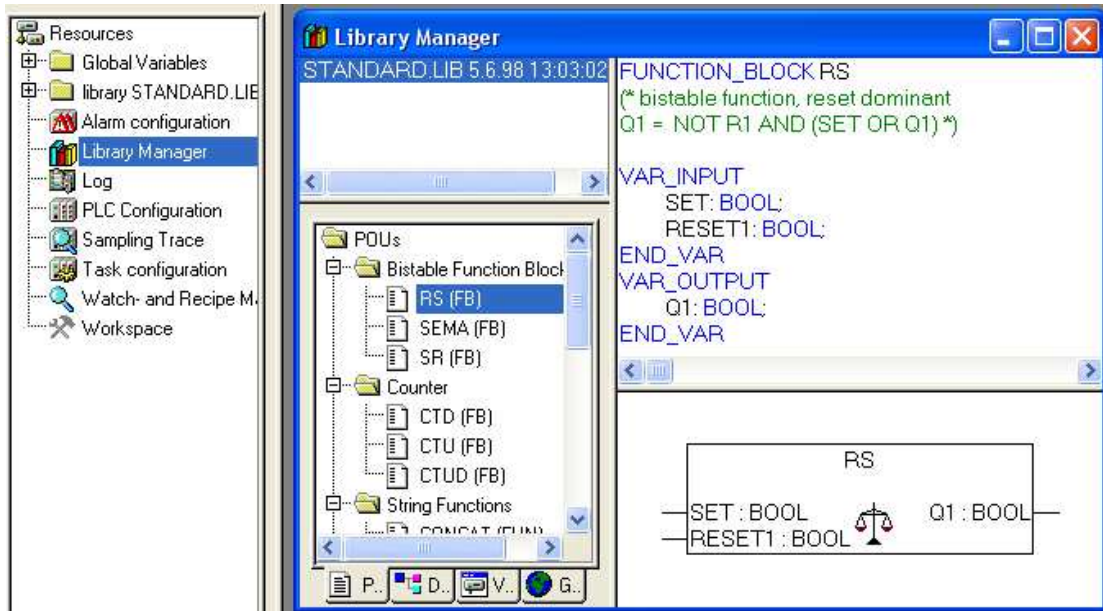
Przerzutnik RS – LD wzór

3. Biblioteczny blok RS (TwinCAT)

- Zasoby projektu – *Resources* (prawa dolna ikona pod eksploratorem)



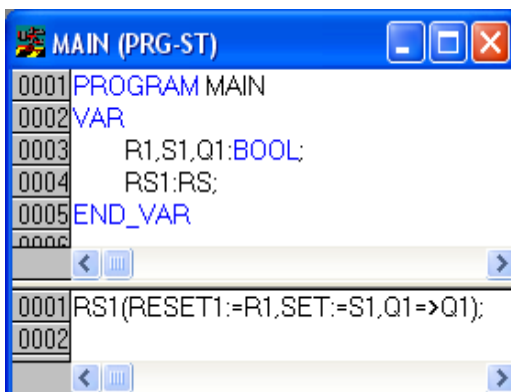
- *Library Manger* > *RS(FB)*



Wejścia/wyjścia: SET, RESET1, Q1

4. Blok RS w programie

- ST (F2 – *Input assistant*)

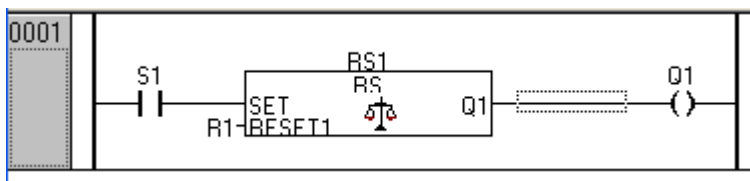
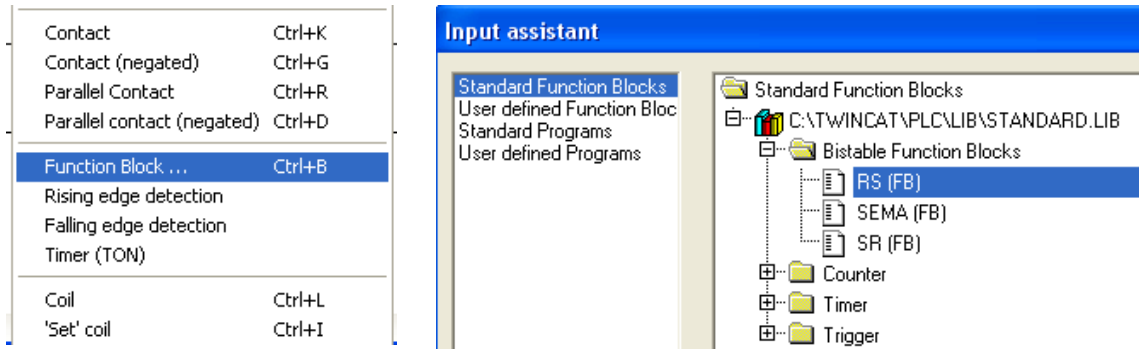


Deklaracje – zmienne wejściowe i wyjściowe
– instancja bloku (rezerwacja pamięci)

Przerzutnik RS – ST blok

- LD

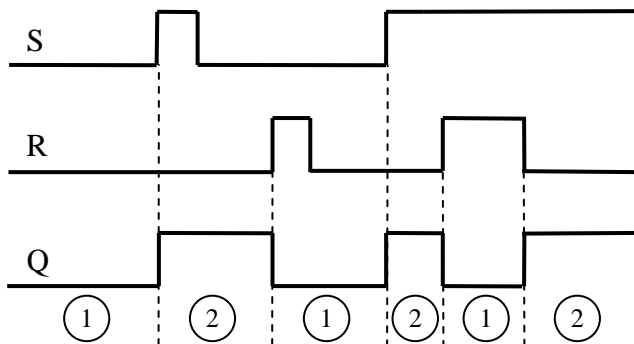
Menu *Function Block...* lub ikona (górny pasek) > Okno *Input assistant* > *RS(FB)*



Przerzutnik RS – LD blok

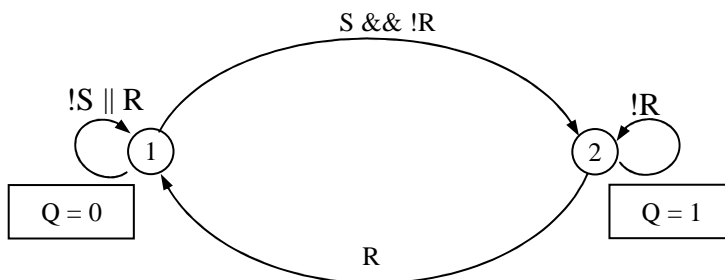
5. Program RS jako automat ST

- Przebiegi czasowe



- Stany
 (1) – wyzerowany (2) – ustawiony

- Automat



Oznaczenia jak w języku C.

- ST

```

MAIN (PRG-ST)
0001 PROGRAM MAIN
0002 VAR
0003   R1,S1,Q1:BOOL;
0004   stan: INT:=1;
0005 END_VAR
0006
0001 CASE stan OF
0002 1: Q1:=FALSE;
0003   IF S1 AND NOT R1 THEN stan:=2; END_IF
0004 2: Q1:=TRUE;
0005   IF R1 THEN stan:=1; END_IF
0006 END_CASE
0007

```

Przerzutnik RS – ST automat

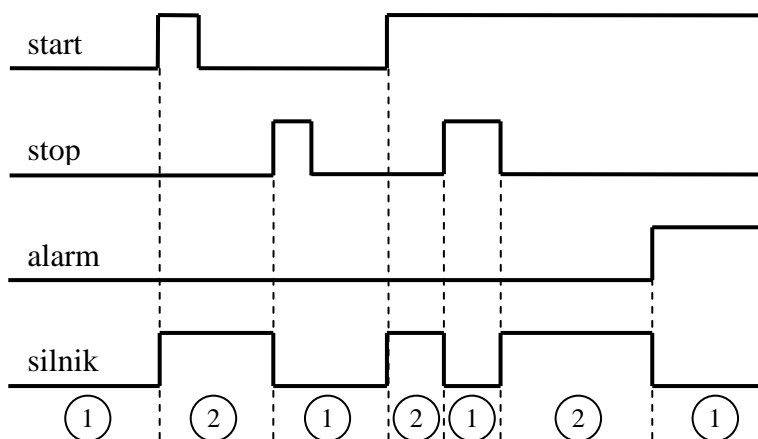
UKŁAD START–STOP

1. Problem

Silnik jest załączany przyciskiem *start*, a wyłączany przyciskiem *stop*. Raz włączony *silnik* pracuje, aż do przyciśnięcia *stop* (pomimo zwolnienia *start*). *Stop* ma priorytet nad *start*, tzn. jednoczesne naciśnięcie obydwu przycisków nie uruchamia *silnika*. Dostępny jest ponadto sygnał *alarm* działający tak jak *stop*, tzn. gdy jest on ustawiony, to *silnika* nie można uruchomić.

Wyjaśnienie. Zwykle *alarm* pochodzi z termicznego zabezpieczenia silnika.

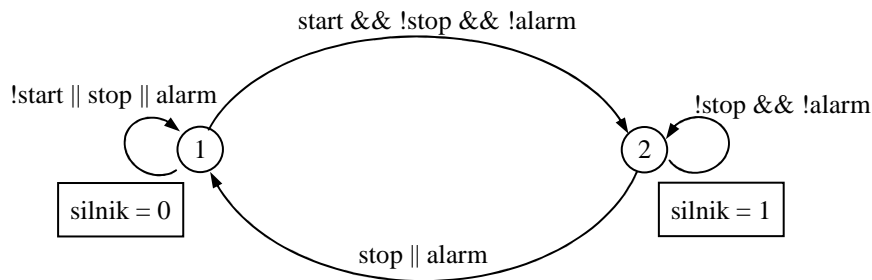
2. Przebiegi czasowe



3. Stany

- ① – zatrzymanie ② – praca

4. Automat



Oznaczenia jak w języku C.

5. Programy

- C

```

char start, stop, alarm, silnik;
char stan=1;
...
switch(stan)
{
    case 1: silnik=0;
            if(start&&!stop&&!alarm) stan=2;
            break;
    case 2: silnik=1;
            if(stop||alarm) stan=1;
}
    
```

- ST

```

MAIN (PRG-ST)
0001 PROGRAM MAIN
0002 VAR
0003     START, STOP, ALARM, SILNIK: BOOL;
0004     stan: INT:=1;
0005 END_VAR
0006
0007 CASE stan OF
0008 1: SILNIK:=FALSE;
0009 IF START AND NOT STOP AND NOT ALARM
0010 THEN stan:=2; END_IF
0011 2: SILNIK:=TRUE;
0012 IF STOP OR ALARM THEN stan:=1; END_IF
0013 END_CASE
    
```

Start Stop – CASE

6. Wzór bezpośredni

- Wzór zastępuje automat

$$silnik_i = (start_i + silnik_{i-1}) \cdot stop_i \cdot alarm_i$$

gdzie $silnik_{i-1}$ jest wartością zmiennej z poprzedniego cyklu obliczeniowego (i-1). Pozostałe oznaczenia dotyczą cyklu aktualnego (i).

- C

```
silnik = (silnik||start) &&!stop&&!alarm;
```

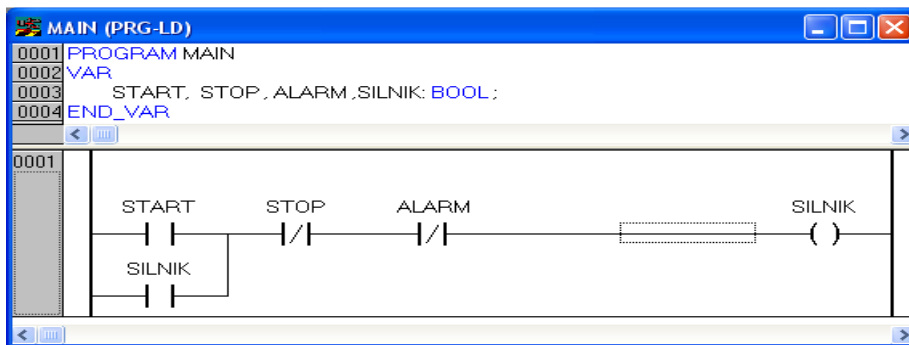
- ST

```

MAIN (PRG-ST)
0001 PROGRAM MAIN
0002 VAR
0003     START, STOP, ALARM,
0004     SILNIK: BOOL;
0005 END_VAR
0006
0007 SILNIK := (START OR SILNIK) AND NOT STOP
0008           AND NOT ALARM;
    
```

Start Stop – ST wzór

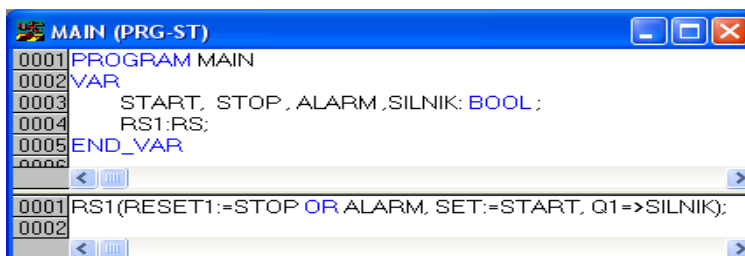
- LD



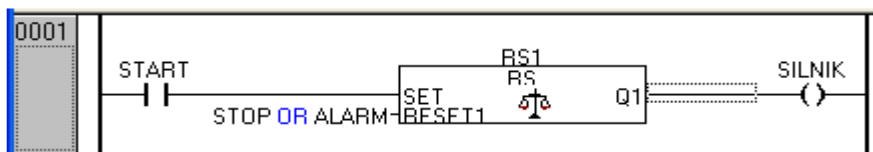
Start Stop – LD wzór

7. Start–Stop z przerzutnikiem RS

- Wzór opisujący układ Start–Stop (powyżej) jest prostym rozszerzeniem wzoru dla przerzutnika RS.
- ST i LD



Start Stop – ST i RS



LD i RS

AUTOMATY W JĘZYKU LD

1. Zasady programowania

- Stany automatu reprezentują zmienne boolowskie S1, S2 itd., których wartości TRUE odpowiadają znajduwaniu się w danych stanach. Wartością początkową S1 jest TRUE.
- Do realizacji układów sekwencyjnych wykorzystuje się dwie specjalne cewki (S), (R), które ustawiają/Set lub zerują/Reset przypisane im zmienne.
- Szczegół przejścia między stanami i, j zawiera wejściową zmienną S_i , warunek przejścia odpowiadający gałęzi grafu automatu oraz dwie równoległe cewki, ustawiającą (S) nowy stan S_j oraz zerującą (R) dotychczasowy stan S_i .



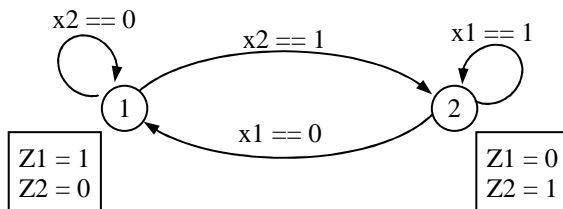
lub



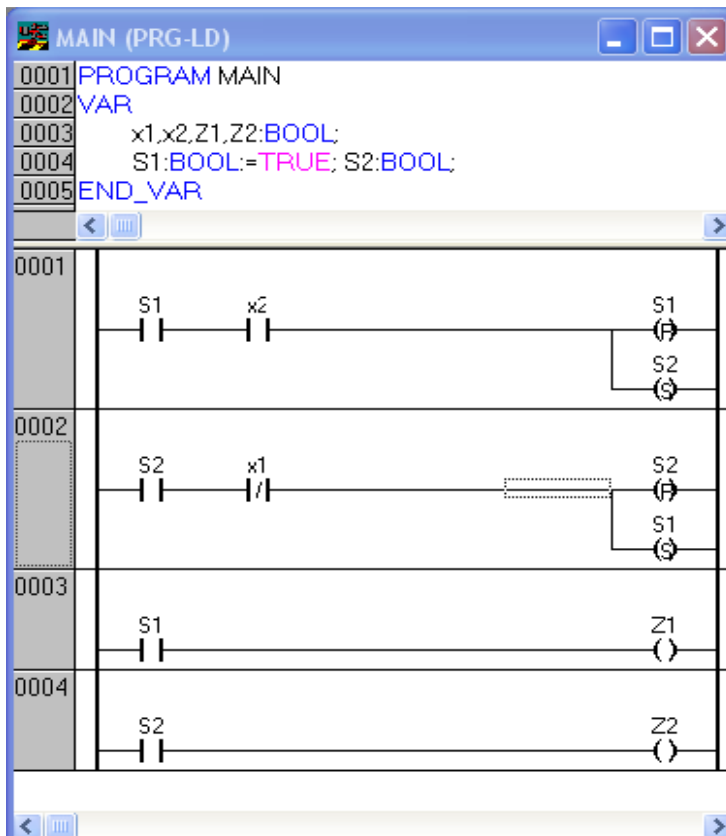
- Szczepel z wyjściem sterującym (cewka) ma jako wejścia (styki równoległe) te spośród zmiennych S1, S2, ... reprezentujące stany, w których wyjście przyjmuje wartość TRUE.

2. Napędzanie i opróżnianie

- Automat



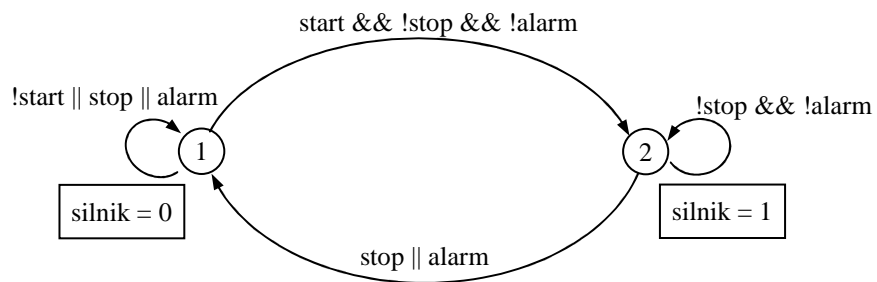
- LD



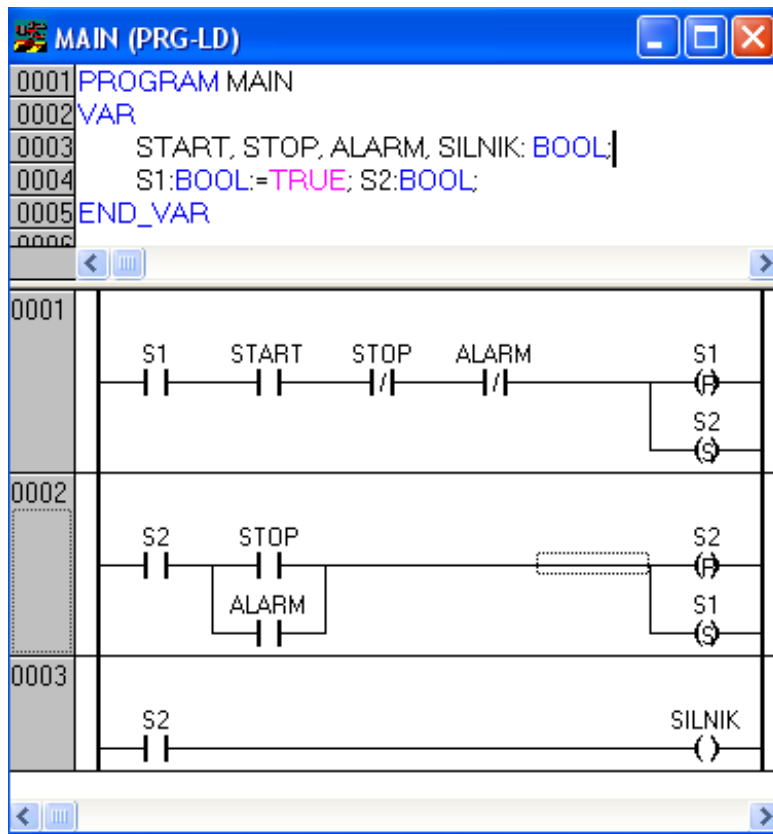
Zbiornik – 2 zawory LD

3. Układ Start-Stop

- Automat



- LD *StartStop* – LD

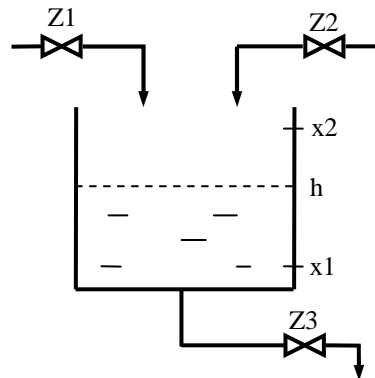


ZBIORNIK Z TRZEMA ZAWORAMI

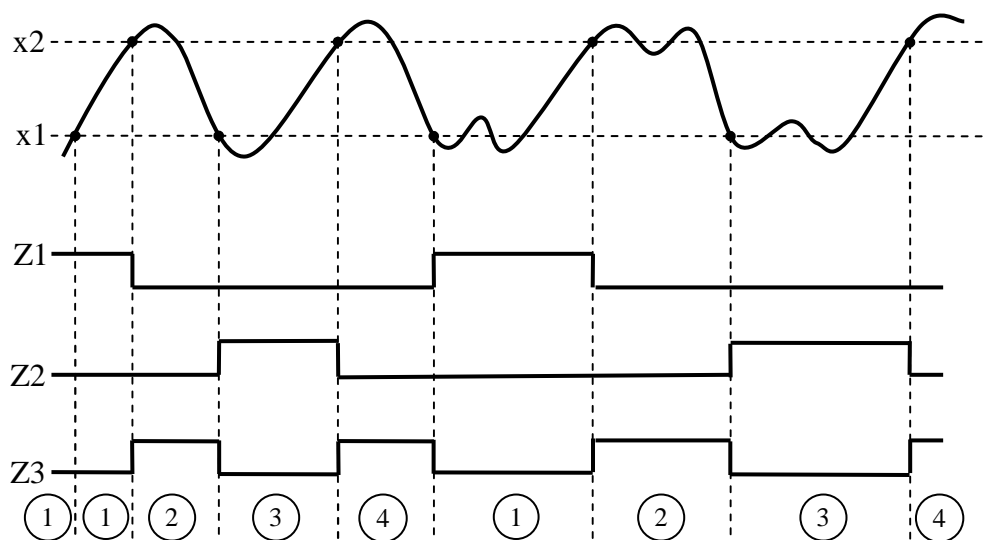
1. Problem

Zbiornik pokazany na rysunku jest napełniany na zmianę zaworami dopływowymi Z1, Z2, a po napełnieniu opróżniany zaworem Z3. Sekwencja pracy wygląda następująco:

- otwarcie Z1, aż poziom osiągnie x_2
- zamknięcie Z1, otwarcie Z3, aż poziom spadnie do x_1
- zamknięcie Z3, otwarcie Z2, aż poziom osiągnie x_2
- zamknięcie Z2, otwarcie Z3, aż poziom spadnie do x_1
- zamknięcie Z3, otwarcie Z1 itd.



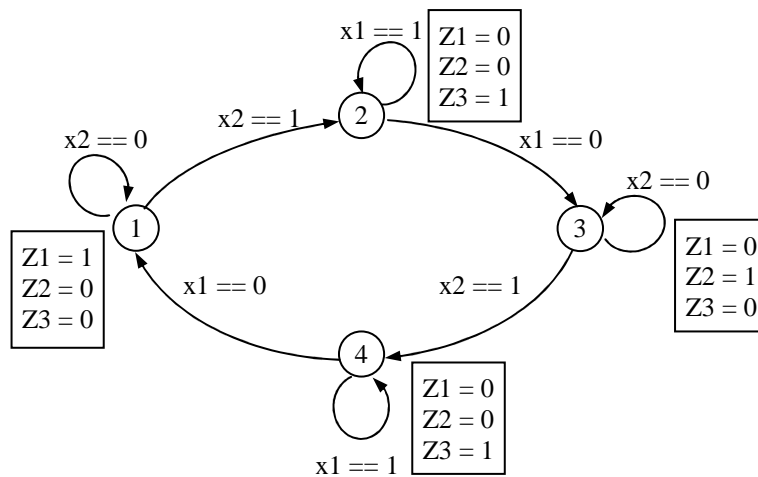
2. Przebiegi czasowe



3. Stany

- ① – napełnianie z zaworu Z1
- ② – opróżnianie po Z1
- ③ – napełnianie z zaworu Z2
- ④ – opróżnianie po Z2

4. Automat



Oznaczenia jak w języku C.

5. Programy

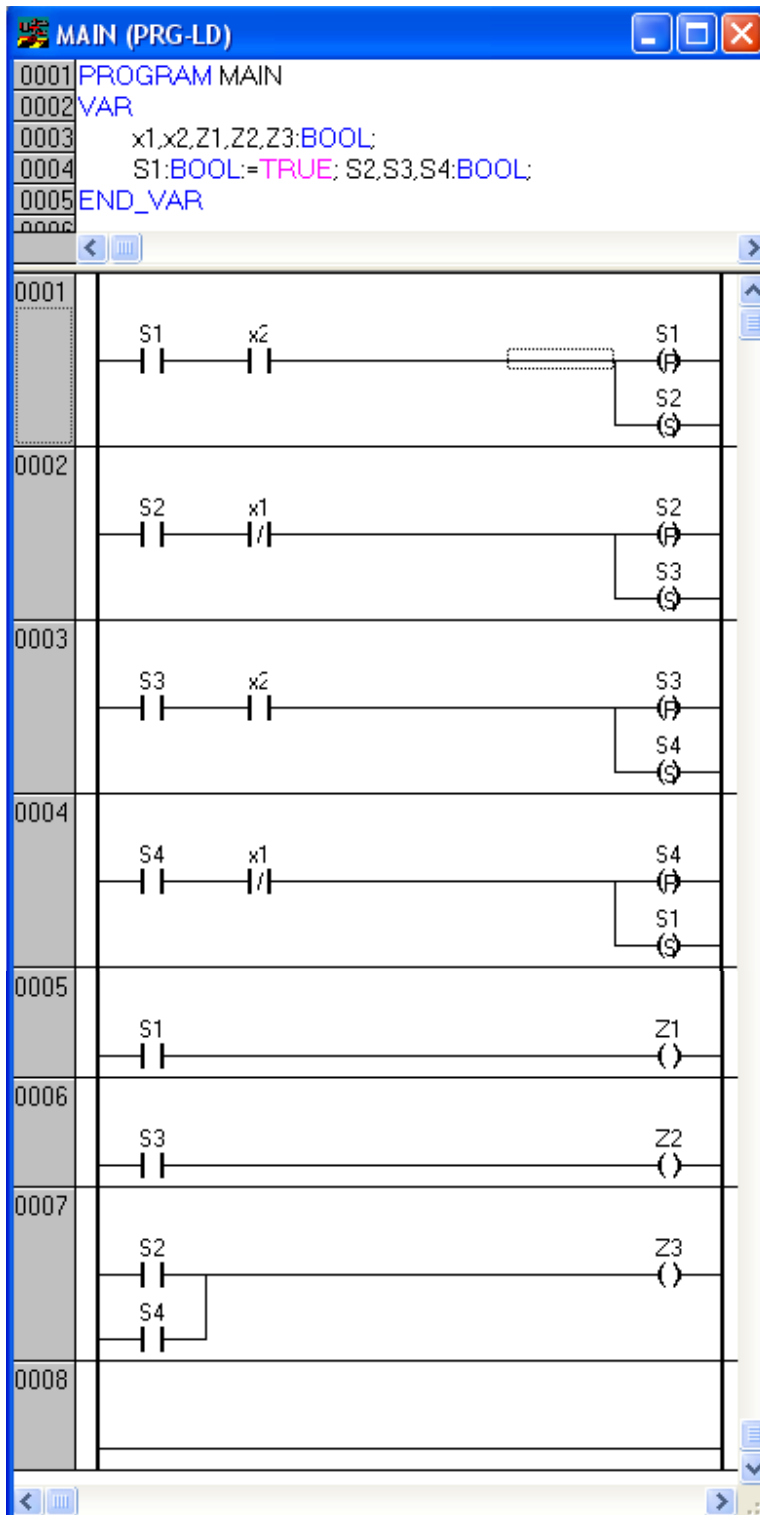
- C

```
switch(stan)
{
    case 1: Z1=1; Z2=Z3=0;
           if(x2) stan=2;
           break;
    case 2: Z1=Z2=0; Z3=1;
           if(!x1) stan=3;
           break;
    case 3: Z1=Z3=0; Z2=1;
           if(x2) stan=4;
           break;
    case 4: Z1=Z2=0; Z3=1;
           if(!x1) stan=1;
           break;
}
```

- ST *Zbiornik – 3 zawory CASE*

```
0001 PROGRAM MAIN
0002 VAR
0003     x1,x2,Z1,Z2,Z3: BOOL;  stan: INT:=1;
0004 END_VAR
0005
0006 CASE stan OF
0007 1: Z1:=TRUE; Z2:=Z3:=FALSE;
0008   IF x2 THEN stan:=2; END_IF
0009 2: Z1:=Z2:=FALSE; Z3:=TRUE;
0010   IF NOT x1 THEN stan:=3; END_IF
0011 3: Z1:=Z3:=FALSE; Z2:=TRUE;
0012   IF x2 THEN stan:=4; END_IF
0013 4: Z1:=Z2:=FALSE; Z3:=TRUE;
0014   IF NOT x1 THEN stan:=1; END_IF
0015 END_CASE
```

- LD



Zbiornik – 3 zawory LD

STEROWANIE SYMULOWANYM ZBIORNIKIEM

1. Symulacja zmian poziomu

- Założenia
 - Poziom H może zmieniać się w przedziale 0...100%.
 - Otwarcie jednego z zaworów wlewowych Z1, Z2 powoduje całkowite napełnienie pustego zbiornika w ciągu 20 s. Otwarcie zaworu Z3 opróżnia zbiornik również w ciągu 20s.
 - Czujniki x1, x2 umieszczone są na wysokości odpowiednio 20 i 80%.
- Zmiana poziomu w ciągu jednego cyklu
Przyjmując, że program symulacyjny będzie wykonywany z domyślnym cyklem $T\#10ms (=0.01 s)$, zmiana poziomu w ciągu jednego cyklu spowodowana otwarciem zaworu wlewowego wyniesie

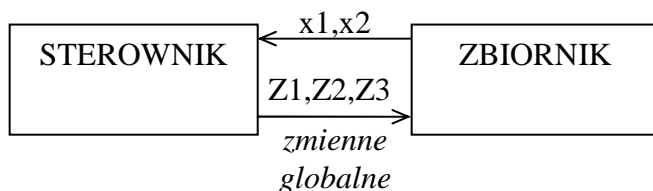
$$\Delta H = \frac{100\%}{20s} \cdot 0.01s = 0.05\%$$

Odpowiednia instrukcja w języku ST będzie mieć postać

$$H := H + 0.05;$$

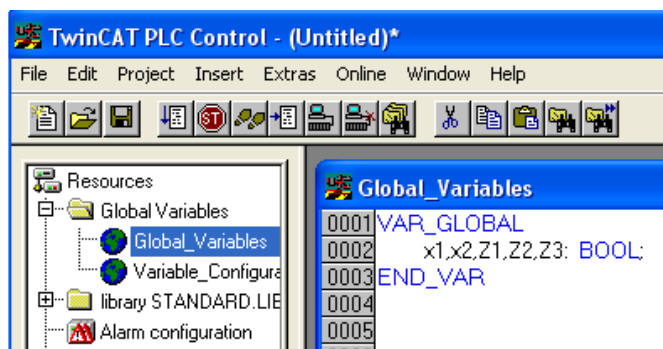
2. Współdziałanie programów

- Zakłada się, że sterowanie i symulacja poziomu będą dwoma oddzielnymi programami – STEROWNIK I ZBIORNIK, wymieniającymi dane za pośrednictwem zmiennych globalnych (wymaganie normy IEC 61131-3).



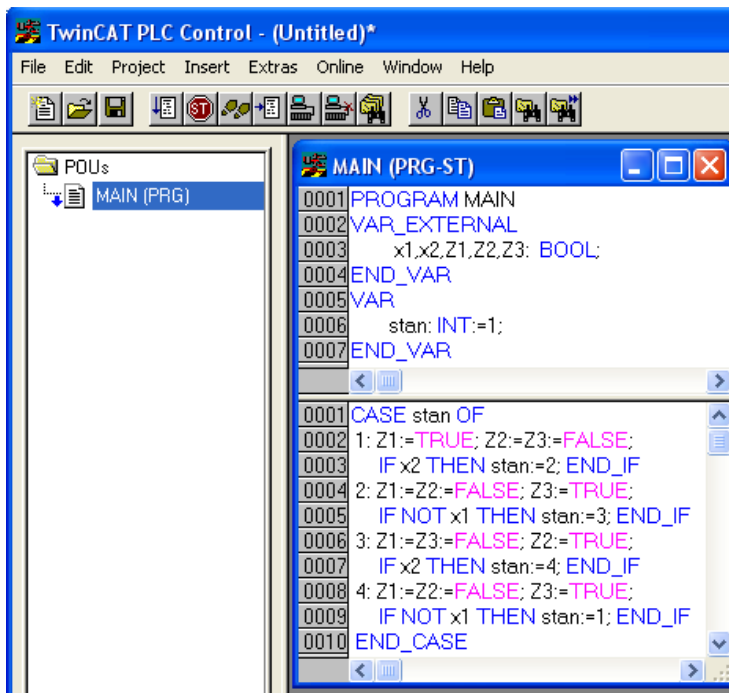
3. Deklaracja zmiennych

- Zakładka *Resources > Global variables > Global variables*



4. Program STEROWNIK (zmiana deklaracji zmiennych)

- ST



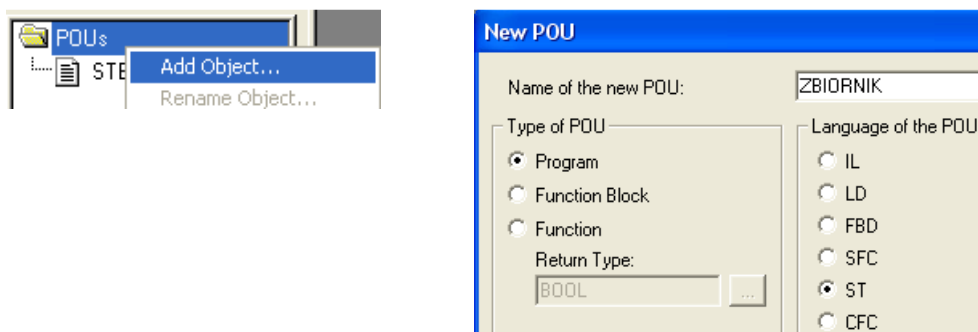
x1, x2, Z1, Z2, Z3 – zmienne zewnętrzne VAR_EXTERNAL
stan – zmienna lokalna

- Zmiana nazwy – *Rename Object*



5. Program ZBIORNIK

- Dodanie nowego programu – *Add Object*
Domyślna nazwa MAIN zastąpiona nazwą ZBIORNIK.



- ST

```

ZBIORNIK (PRG-ST)
0001 PROGRAM ZBIORNIK
0002 VAR_EXTERNAL
0003   x1,x2,Z1,Z2,Z3: BOOL;
0004 END_VAR
0005 VAR
0006   H:REAL;      (* poziom *)
0007 END_VAR
0008
0009
0001 IF Z1 THEN H:=H+0.05; END_IF      (* wlewanie *)
0002 IF Z2 THEN H:=H+0.05; END_IF
0003 IF Z3 THEN H:=H-0.05; END_IF      (* spust *)
0004
0005 IF H<0.0 THEN H:=0.0; END_IF      (* ograniczenia *)
0006 IF H>100.0 THEN H:=100.0; END_IF
0007
0008 IF H>=80.0 THEN x2:=TRUE;
0009   ELSE x2:=FALSE; END_IF          (* czujnik *)
0010 IF H>=20.0 THEN x1:=TRUE;
0011   ELSE x1:=FALSE; END_IF
  
```

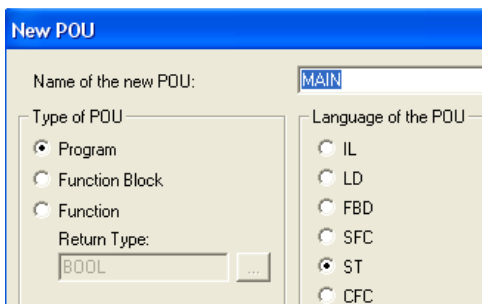
H – zmienna lokalna

6. Program MAIN

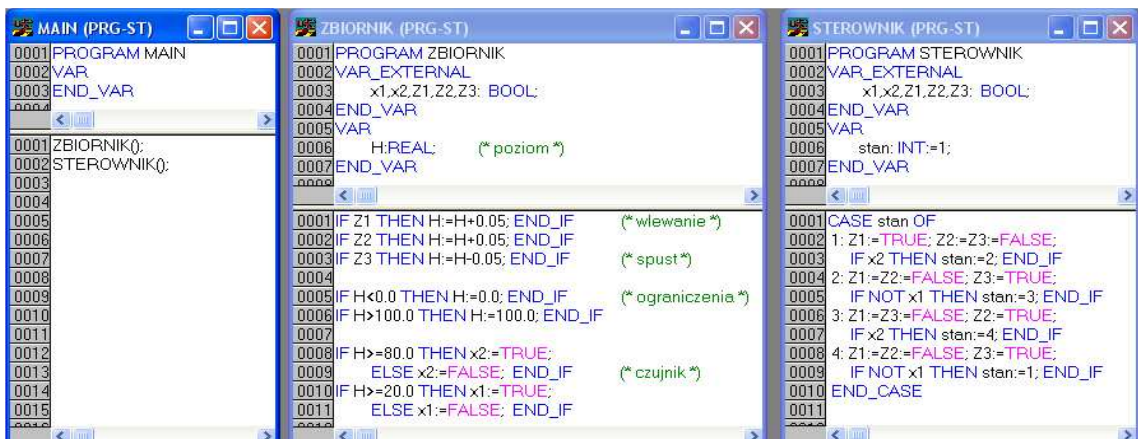
- TwinCAT wymaga, aby główny program MAIN wywoływał programy podrzędne STETROWNIK i ZBIORNIK tak jak funkcje bezparametrowe.

Dodanie programu MAIN – Add Object (j.w.)

- ST

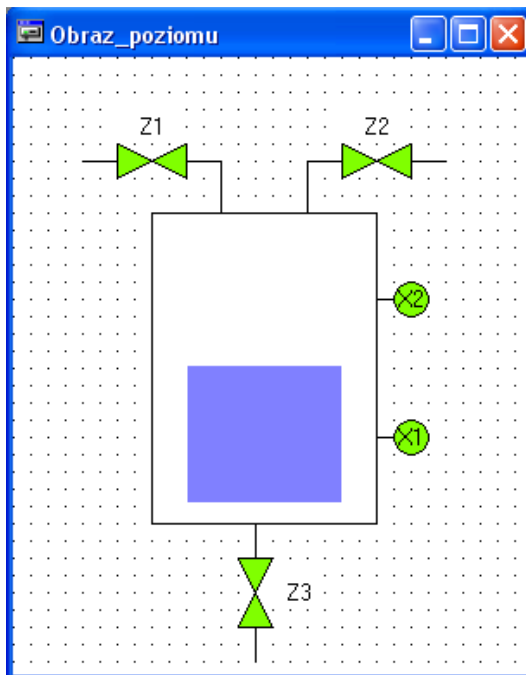


Zestaw programów aplikacji



7. Obraz poziomu

- Docelowy obraz



x1, x2, Z1, Z2, Z3 – zmiana koloru poziom – bargraf

- Poziom H – bargraf bez skali (*No scale*)

Ikona *Bar display*



Konfiguracja bargrafu

Configure bar display	Configure scale and variable
Diagram type: <input type="text" value="No Scale"/>	Scale start: <input type="text" value="0"/>
Orientation: <input type="radio"/> Horizontal <input checked="" type="radio"/> Vertical	Scale end: <input type="text" value="100"/>
Running direction: <input checked="" type="radio"/> Bottom - Up <input type="radio"/> Up - Bottom	Main scale: <input type="text" value="20"/>
Color area markers: <input type="text" value="No markers"/>	Sub scale: <input type="text" value="20"/>
<input type="text" value="Variable/Scale"/> <input type="text" value="Color areas"/>	Unit: <input type="text"/>
<input type="text" value="OK"/> <input type="text" value="Cancel"/>	Scale format (C-Syntax): <input type="text" value="%.0f"/>
	Variable: <input type="text" value="ZBIORNIK.H"/>
	Invisible: <input type="text"/>

Sygnalizatory i sondy poziomu

Emerson



Controlmatica



Przyciski i wyłączniki krańcowe

Rockwell

