UKŁADY KOMBINACYJNE

Wprowadzenie. Zadanie przykładowe I. Metoda Karnaugha. Schemat sprzętowy. Program w C. Program w ST. Program w LD. Program ST w środowisku TwinCAT PLC Control. Program LD – PLC Control. Niepoprawne pomiary. Zadanie przykładowe II.

WPROWADZENIE

1. Układy przełączające

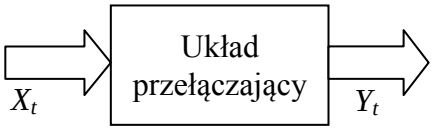
- Podział

Układy przełączające → Układy kombinacyjne (bez pamięci)
Układy przełączające → Układy sekwencyjne (z pamięcią)

- Realizacje sprzętowe

Układy kombinacyjne – bramki bez sprzężeń zwrotnych
Układy sekwencyjne – przerzutniki, bramki ze sprzężeniami zwrotnymi

- Opis matematyczny



Układy kombinacyjne – $Y_t = \lambda(X_t)$ – aktualny stan wyjść zależy wyłącznie od aktualnego stanu wejść.

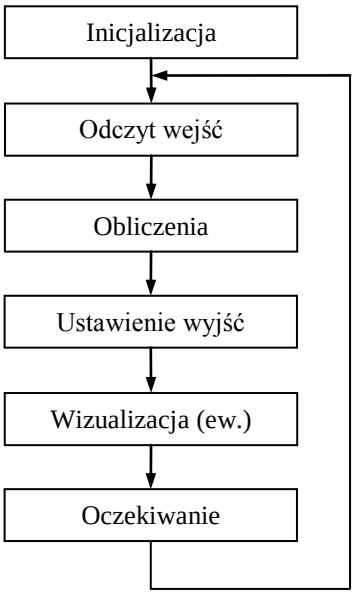
Układy sekwencyjne – $\begin{cases} Q_{t+1} = \delta(Q_t, X_t) \\ Y_t \quad = \mu(Q_t, X_t) \end{cases}$ – Q_t stan wewnętrzny

Stan wyjść zależy od wejść i stanu wewnętrznego (zależnego od poprzednich wejść → pamięć: $Y_t = \mu(\delta(Q_{t-1}, X_{t-1}), X_t)$).

2. Metodologia projektowania układów kombinacyjnych

- Sformułowanie tablicy wejść/wyjść
- Utworzenie funkcji przełączającej metodą Karnaugha
- Realizacja sprzętowa – bramki
- Realizacja programowa (języki C, ST, ew. inne)
  - funkcja przełączająca → wzór (podstawowa realizacja)
  - schemat bramkowy → funkcje AND, OR, NOT
  - tablica wejść/wyjść → indeksem do tablicy jest kombinacja we/wy
  - zestaw instrukcji *if ... then ... else* odpowiadający tablicy we/wy
- Niepoprawne pomiary

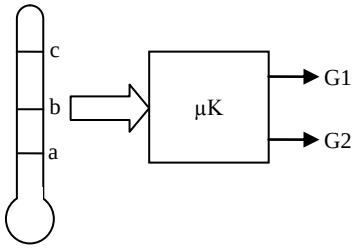
3. Struktura programu w prostym sterowniku



Cykl wykonywania programu – np. 10 ms, 0.1 s lub ponowne rozpoczęcie zaraz po poprzednim wykonaniu.
Wizualizacja – LEDy, bargrafy, wyświetlacz LCD.

ZADANIE PRZYKŁADOWE I

1. Sterowanie nagrzewaniem



a, b, c: 0 – temperatura poniżej poziomu
1 – temperatura powyżej poziomu lub mu równa

G1, G2 – grzejniki

Zadanie:

		G1	G2	
	t < a			
a ≤	t < b	G1	-	
b ≤	t < c	-	G2	
c ≤	t	-	-	

– obydwa grzejniki włączone

2. Tablica wejść/wyjść

c	b	a	G1	G2
0	0	0	1	1
0	0	1	1	0
0	1	1	0	1
1	1	1	0	0

Inaczej – tablica zero–jedynkowa, tablica prawdy

Uwaga. Stany nie ujęte w tablicy reprezentują awarie czujników pomiarowych (typowa reakcja na uszkodzenie → wyłączyć zasilanie).

METODA KARNAUGHA

1. Reguły tworzenia tablic Karnaugha

Chodzi o sformułowanie możliwie prostego wzoru, aby ułatwić realizację sprzętową (w realizacji programowej znaczenie minimalizacji jest mniejsze).

- Wyjścia rozpatruje się oddzielnie.
- Współrzędne pól elementarnych opisane są refleksyjnym kodem Graya.
- Puste pola uzupełnia się znakami nieokreśloności (-).

2. Tablice zadania

G1

c \ ba	00	01	11	10
0	1	1	0	-
1	-	-	0	-

G2

c \ ba	00	01	11	10
0	1	0	1	-
1	-	-	0	-

3. Reguły upraszczania – łączenie pól elementarnych

- Wszystkie pola „1” muszą być objęte
- Liczba pól łączonych ze sobą musi być potęgą 2 (1, 2, 4, 8, …).
- Połączone pola muszą tworzyć prostokąt lub kwadrat, im większy tym lepiej.
- Pola „-” dołącza się do pól „1”.
- Pola mogą zachodzić na siebie.
- Górny i dolny wiersz uważa się za sąsiednie, jak również lewą i prawą kolumnę (ze względu na kod Graya).

G1

c \ ba	00	01	11	10
0	1	1	0	-
1	-	-	0	-

G2

c \ ba	00	01	11	10
0	1	0	1	-
1	-	-	0	-

4. Tworzenie wynikowych wzorów – kolumny

- Wypisać kolumny wejść odpowiadające zakreślonym obszarom.
- Utworzyć iloczyn z tych wejść, które w wypisanych kolumnach mają niezmienione wartości (stale 0 lub 1), przy czym 1 odpowiada sygnałowi prostemu, a 0 – zanegowanemu.
- Wynikowy wzór jest sumą wzorów powstałych z zaznaczonych obszarów.

G1

c	b	a
0	0	0
0	0	1
1	0	0
1	0	1
	↓	
	$\bar{b}$	

$G1 = \bar{b}$

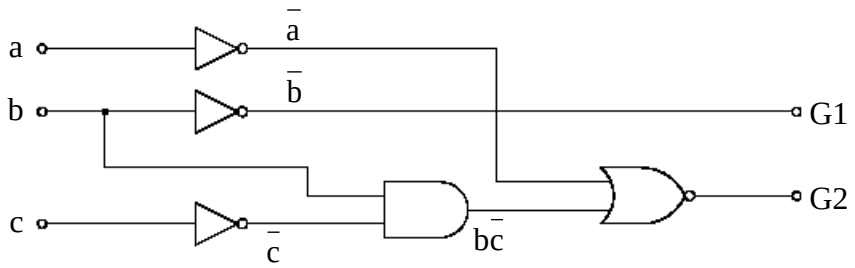
G2

c	b	a	c	b	a	
0	0	0	0	1	1	
1	0	0	0	1	0	
0	1	0	↓	↓		
1	1	0	$\bar{c} \bullet b$			– iloczyn
		↓				
		$\bar{a}$				

$G2 = \bar{a} + b\bar{c}$ – suma

Uwagi. Zaznaczenie zbyt małych obszarów w tablicy Karnaugha nie jest błędem, ale wynikowy wzór się rozrasta (nie jest minimalny), bo iloczynów w końcowej sumie przybywa. Zjawisko hazardu obecne w realizacjach sprzętowych nie występuje w realizacjach programowych, ponieważ obliczenia są wykonywane w tym samym cyklu (nie ma różnicy czasów propagacji bramek).

SCHEMAT SPRZĘTOWY



Programowym odpowiednikiem schematów sprzętowych jest graficzny język FBD.

PROGRAM W C

1. Operatory logiczne

char – typ zmiennych logicznych

! negacja, && iloczyn logiczny, || suma logiczna

```
char a,b,c,G1,G2;
...
G1=!b;
G2=!a||b&&!c;
```

2. if … else – tablica wejść/wyjść

```
if(!c&&!b&&!a) {G1=1; G2=1;}
    else
        if(!c&&!b&&a) {G1=1; G2=0;}
        else
            if(!c&&b&&a) {G1=0; G2=1;}
            else
                {G1=0; G2=0;}
```

Wersja powyższa ujmuje także przypadek z niepoprawnymi pomiarami (G1=0, G2=0 – zob. dalej).

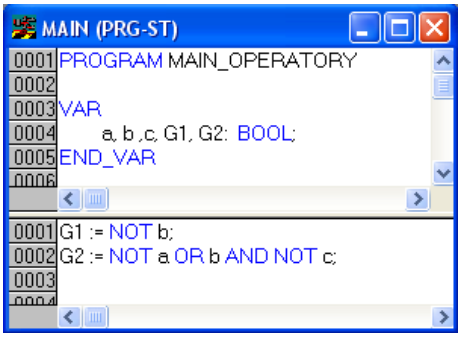
PROGRAM W ST

Norma PN–EN 61131–3: 2004(4). *Sterowniki programowalne. Część 3: Języki programowania* definiuje pięć języków:

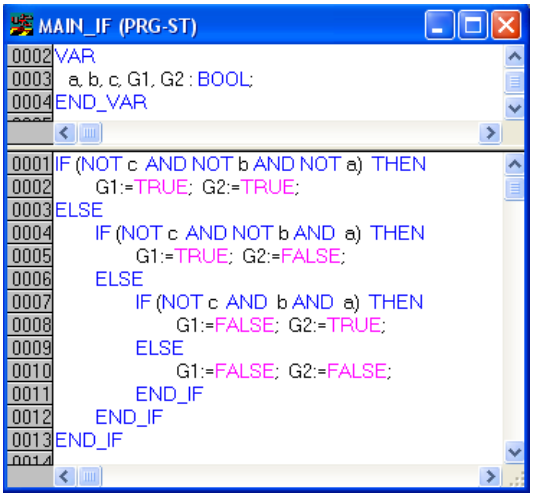
ST – tekst strukturalny (*Structured Text*)
IL – lista rozkazów (*Instruction List*)
LD – schemat drabinkowy (*Ladder Diagram*)
FBD – funkcjonalny schemat blokowy (*Function Block Diagram*)
SFC – sekwencyjny schemat funkcjonalny (*Sequential Function Chart*)

Będziemy stosować pakiet TwinCAT sterowników PLC/PAC firmy Beckhoff.

1. Operatory logiczne

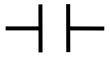


2. IF…THEN…ELSE

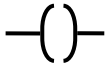


PROGRAM W LD

1. Elementy

 zmienna wejściowa prosta – argument

 zmienna zanegowana

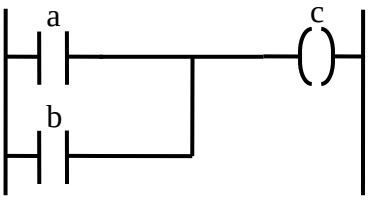


zmienna wyjściowa – rezultat

szczebel – „pusty” wzór

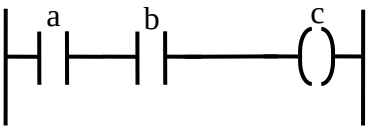
2. Operacje logiczne

Suma



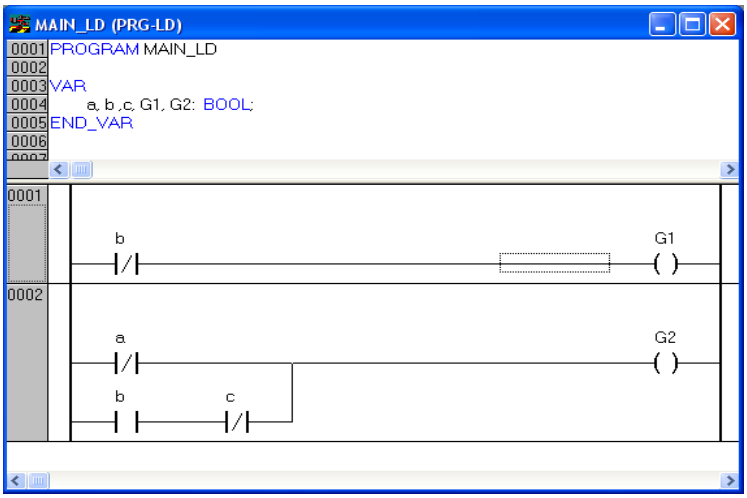
c = a + b

Iloczyn



c = a • b

3. Program – zadanie I



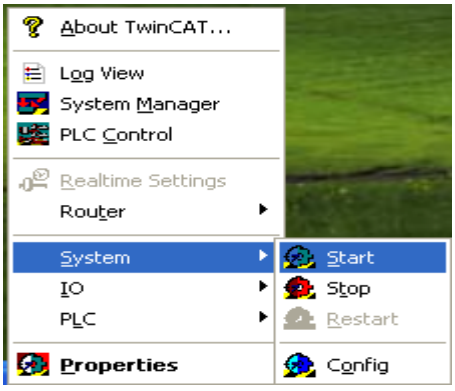
$G1 = \bar{b}$

$G2 = \bar{a} + b\bar{c}$

PROGRAM ST W ŚRODOWISKU TWINCAT PLC CONTROL

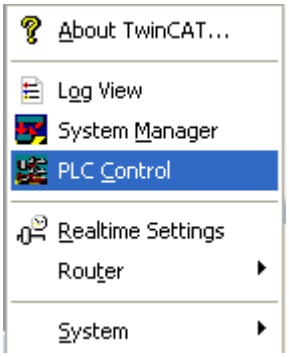
1. Uruchomienie systemu

- *System* > *Start* – jeśli system jest już uruchomiony, to *Start* wyszarzone.

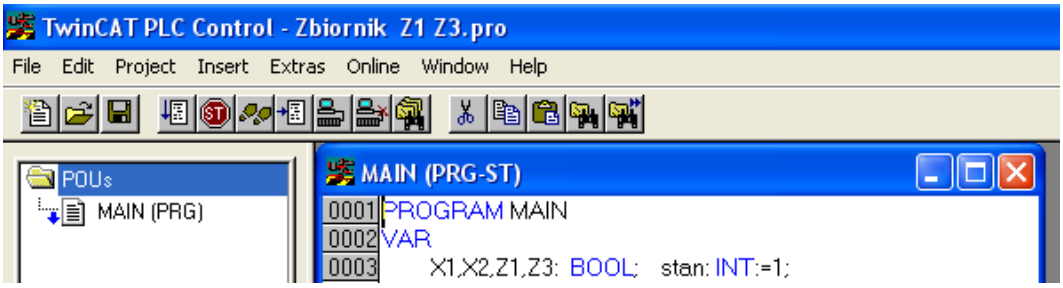


Odpowiedź *Cancel* na pytanie o rejestrację.

- *PLC Control*

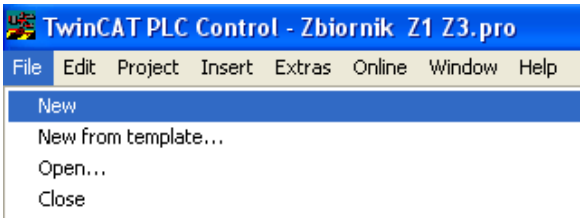


Pojawia się okno *TwinCAT PLC Control* z ostatnio uruchamianym projektem.

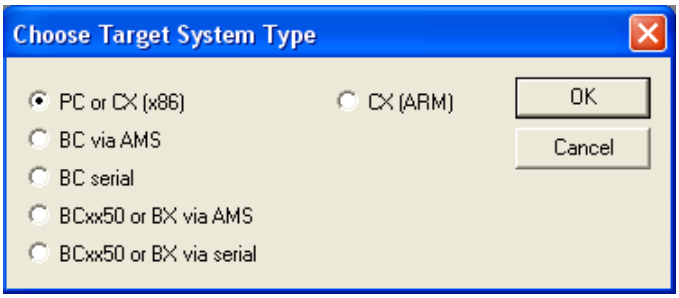


2. Nowy projekt

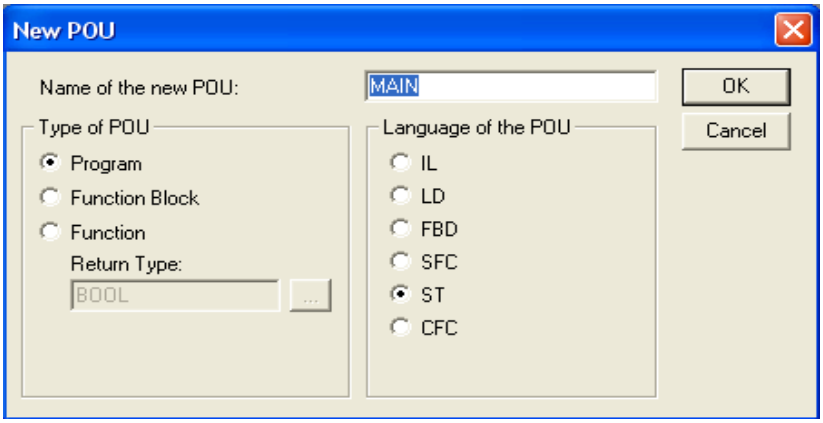
- *File > New*



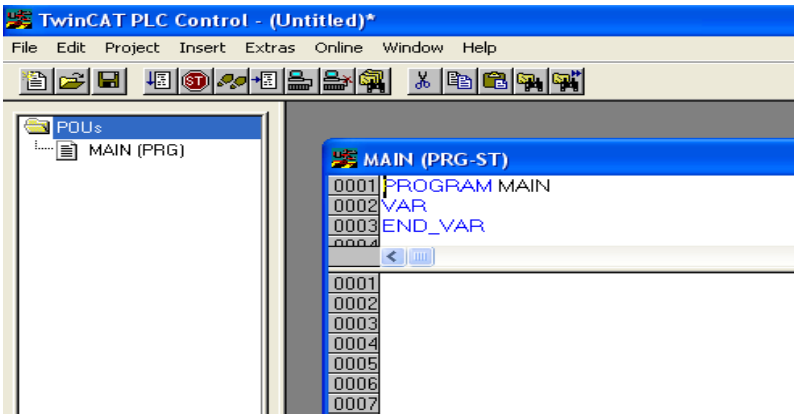
- Typ systemu docelowego
  Symulacja i prace domowe – *PC or CX (x86)*
  Laboratorium – *CX(ARM)*



- Język programowania – *ST*

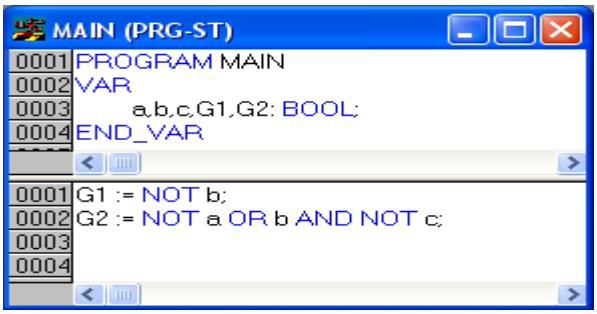


Pojawia się puste okno edytora programu z *Untitled* jako nazwą projektu. Górna część jest przeznaczona na deklaracje zmiennych i bloków funkcjonalnych, a dolna na właściwy kod.

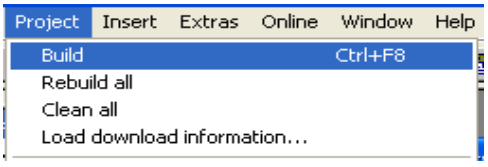


3. Kodowanie, kompilacja

- Deklaracje i kod



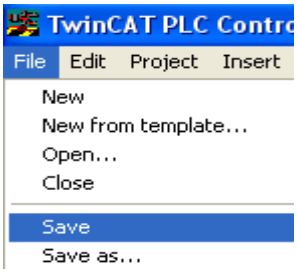
- *Project* > *Build* lub *Rebuild all*



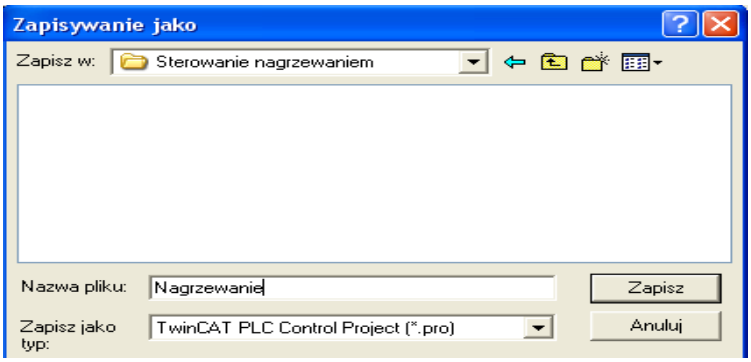
Informacja o wyniku kompilacji w dolnej części okna

Na *warningi* nie należy zwracać uwagi.

- Zapis pliku
  File > Save

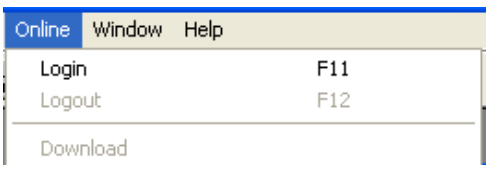


Należy najpierw utworzyć katalog na pliki projektu (których może być nawet kilkanaście), tutaj katalog *Sterowanie nagrzewaniem*, i w nim zapisać plik z kodem źródłowym – tutaj *Nagrzewanie.pro* (rozszerzenie dodawane automatycznie).

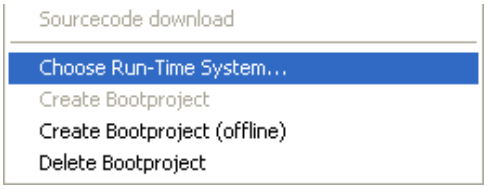


4. Symulacja

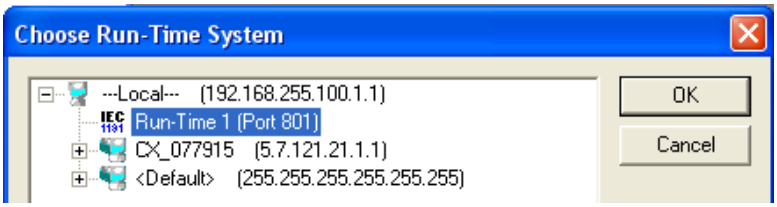
- *Online > Choose Run–Time System…*



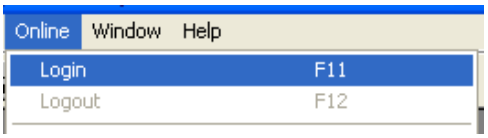
…



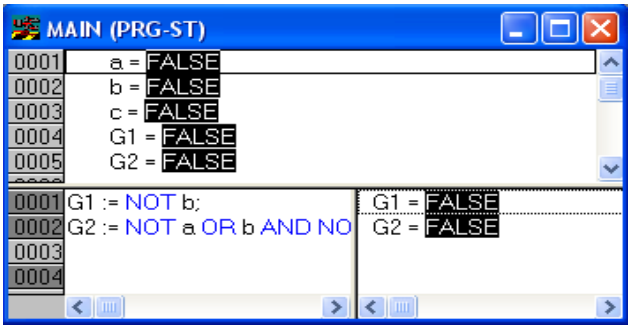
- *Local* > *RunTime 1* (*Port 801*) – lokalny komputer PC



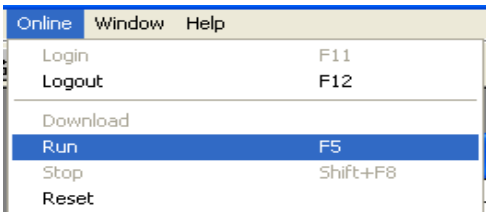
- Ładowanie programu
  Online > *Login*

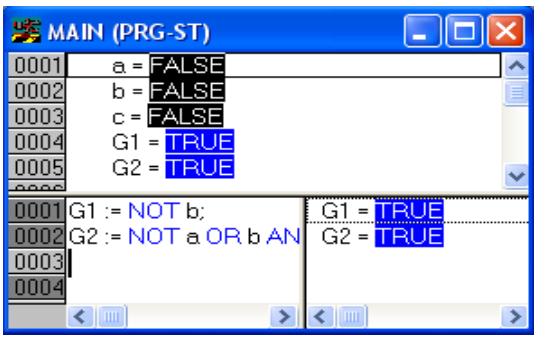


Odpowiedź *Tak* na pytanie o załadowanie programu.
Początkowe wartości zmiennych.



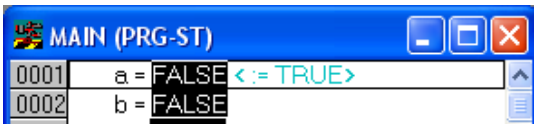
- *Online* > *Run*



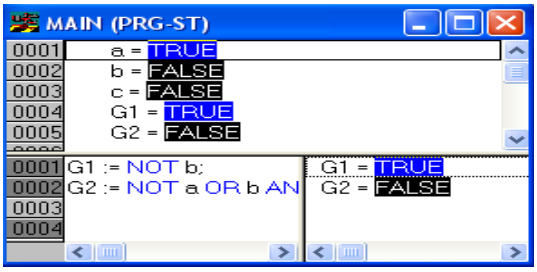


5. Zapisywanie nowej wartości

- 2 kl. zmienna
  Pojawia się proponowana nowa wartość.

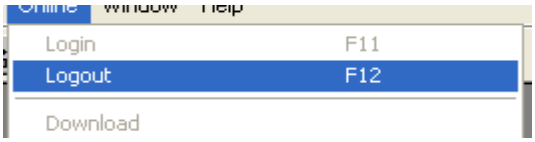


- *Online* > *Write Values* lub *Ctrl+F7*



6. Zakończenie

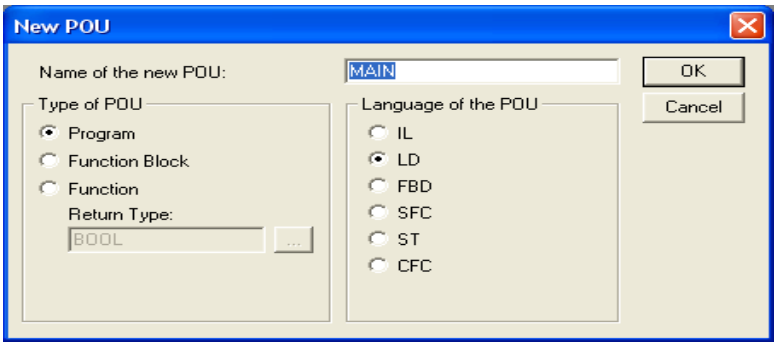
- *Online* > *Logout*



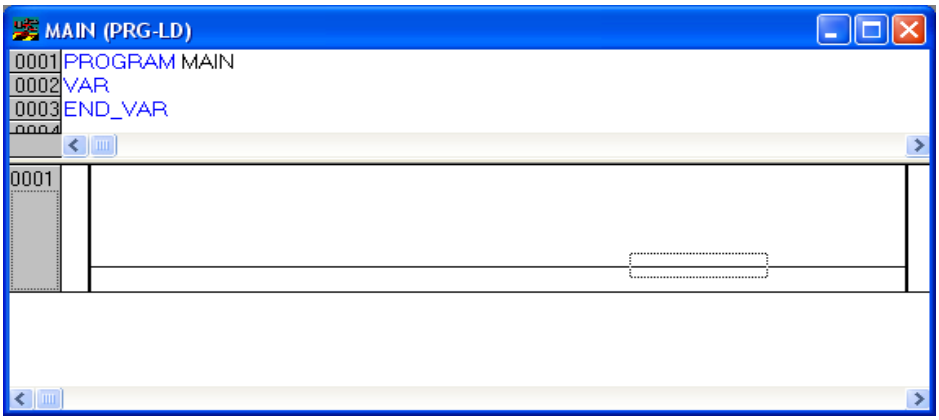
PROGRAM LD – PLC CONTROL

1. Nowy projekt, typ POU, język LD

- *File* > *New*
- *PC or CX (x86)*
- *Language – LD*

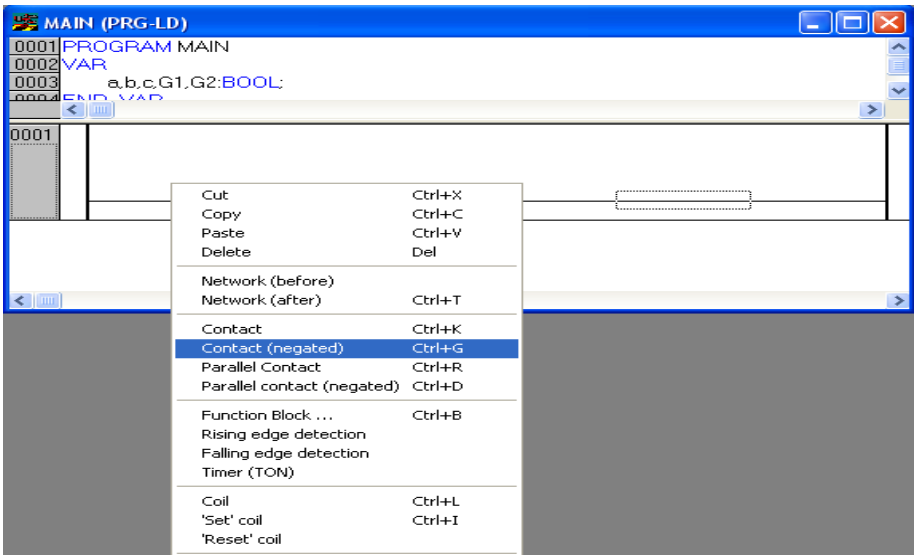


Okno edytora LD – deklaracje w górnej części (j.p.)
– dolna część przeznaczona na schemat; widoczny jeden szczebel drabinki.



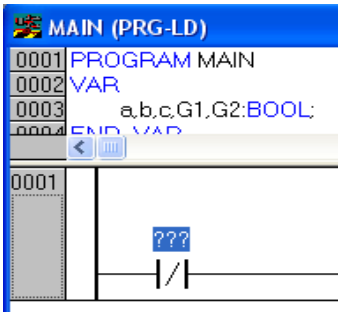
2. Tworzenie schematu LD

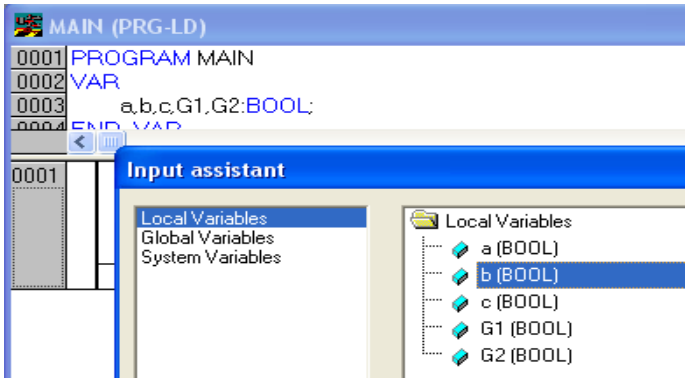
- Wybór miejsca, menu kontekstowe (prawy klawisz myszy) > wybór elementu schematu, np. *Contact* (*negated*).



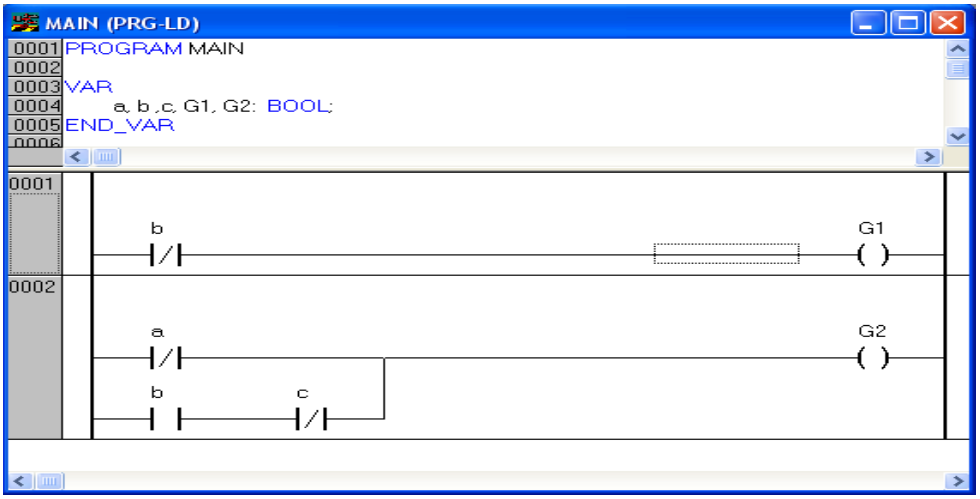
Pojawia się styk zanegowany, bez nazwy zmiennej – pytajniki ???

- Nazwa zmiennej
  Zaznaczyć ??? > przycisk F2 > okno *Input assistant* > wybór zmiennej, np. *b.*

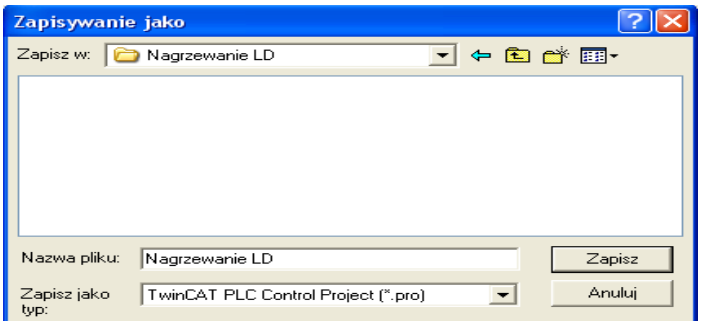




- Pełny schemat

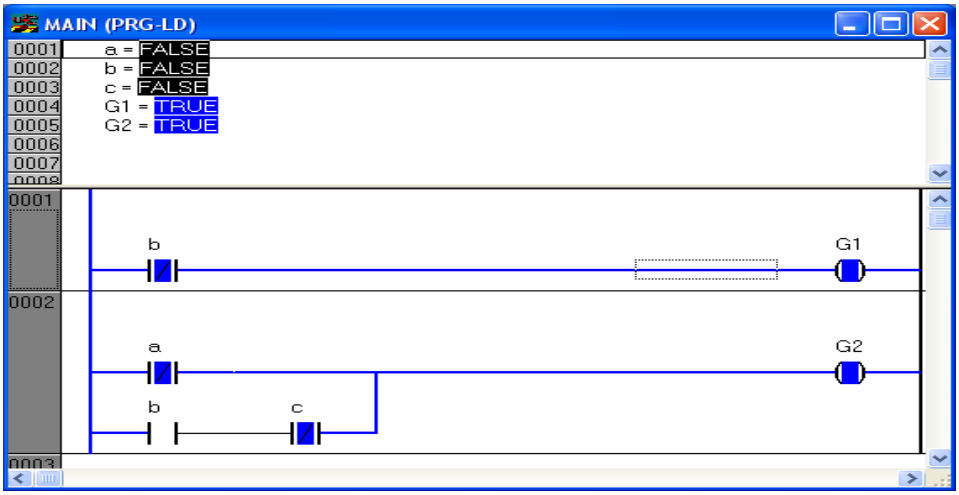


- Kompilacja *Project > Build*
- Zapis pliku *File > Save*
  Najpierw utworzono katalog *Nagrzewanie LD* na pliki projektu.



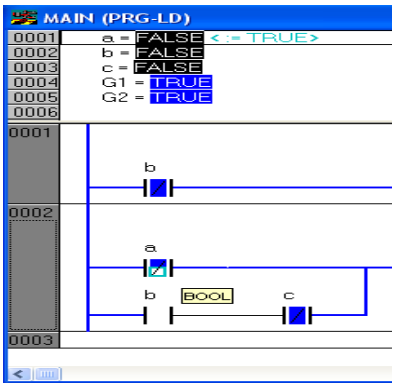
3. Symulacja

- *Online > Choose Run–Time System > Local > RunTime 1 (Port 801)*
- Ładowanie *Online > Login* (początkowe wartości zmiennych)
- Uruchomienie *Online > Run*

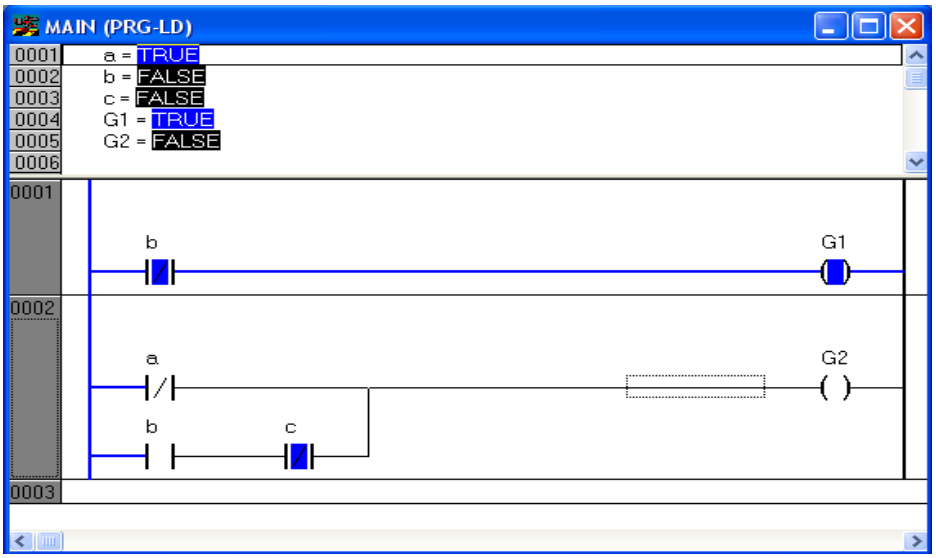


4. Nowa wartość zmiennej

- 2 kl. styk, np. *a*
- Połowa styku zmienia kolor; proponowana nowa wartość widoczna także w górnej części okna (deklaracje).



- Ctrl + F7 lub *Online* > *Write Values*



NIEPOPRAWNE POMIARY

1. Tablica poprawności pomiarów

c	b	a	P
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

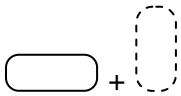
P = 1 – pomiary poprawne
P = 0 – pomiary niepoprawne

2. Tablica Karnaugha

c \ ba	00	01	11	10
0	1	1	1	0
1	0	0	1	0

Uwaga. Teraz w tablicy Karnaugha dotyczącej poprawności pomiarów nie ma pól „-” (nie ma nieokreśloności).

3. Wzór



$$P = \bar{c}\bar{b} + ab$$

4. Wymaganie technologiczne

W przypadku niepoprawnych pomiarów obydwie grzałki należy wyłączyć – G1=G2=0.

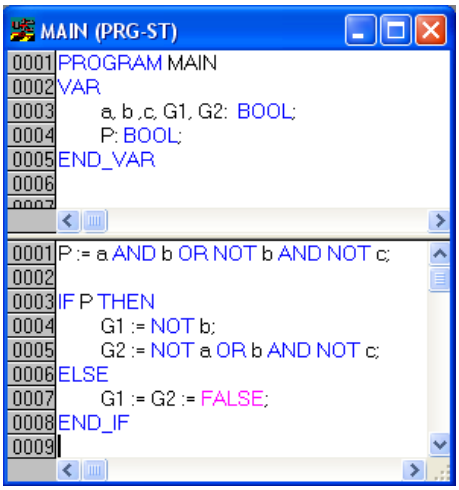
5. Programowanie

C

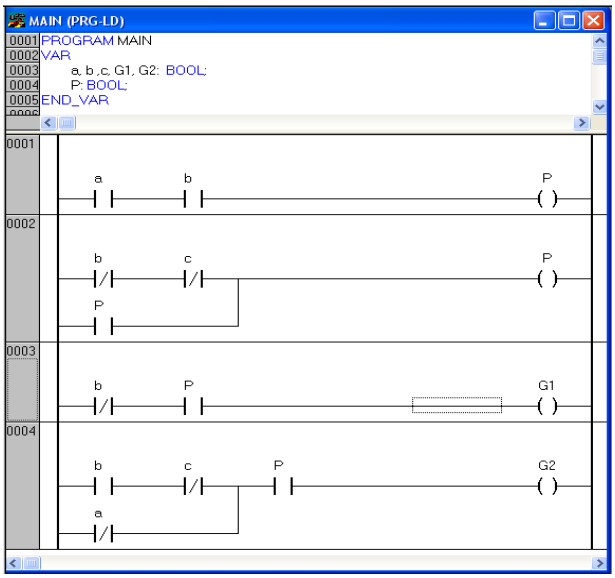
```
char P;              // poprawność pomiarów
...
P=a&&b||!b&&!c;

if(P)
     {G1=!b; G2=!a||b&&!c;}
else
     {G1=G2=0;}
```

ST



LD



$$P = a \cdot b$$

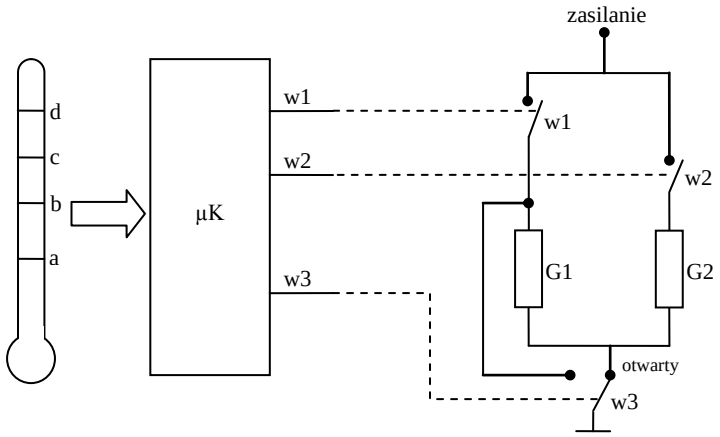
$$P = P + \bar{b} \cdot \bar{c}$$

ZADANIE PRZYKŁADOWE II

1. Sterowanie nagrzewaniem

Włączanie grzejników:

- równolegle G1, G2, gdy $t < t_a$
- tylko G1, gdy $t_a \leq t < t_b$
- tylko G2, gdy $t_b \leq t < t_c$
- szeregowo G1, G2, gdy $t_c \leq t < t_d$
- wyłączone G1, G2, gdy $t_d \leq t$



Połączenie powyższe pozwala uzyskać cztery stopnie mocy grzejnej przy dwóch grzejnikach.

2. Tablica wejść/wyjść

d	c	b	a	w_1	w_2	w_3
0	0	0	0	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	1	0
0	1	1	1	0	1	1
1	1	1	1	0	0	0

3. Tablice Karnaugha i kolumny

w_1

dc \ ba	00	01	11	10
00	1	1	0	-
01	-	-	0	-
11	-	-	0	-
10	-	-	-	-

d	c	b	a
0	0	0	0
0	0	0	1
0	1	0	0
0	1	0	1
1	1	0	0
1	1	0	1
1	0	0	0
1	0	0	1
		$\bar{b}$	

$w_1 = \bar{b}$

w_2

dc \ ba	00	01	11	10
00	1	0	1	-
01	-	-	1	-
11	-	-	0	-
10	-	-	-	-

d	c	b	a
0	0	0	0
0	0	1	0
0	1	1	0
0	1	1	0
1	1	0	0
1	1	1	0
1	0	0	0
1	0	1	0
			$\bar{a}$

d	c	b	a
0	0	1	1
0	0	1	0
0	1	1	1
0	1	1	0
$\bar{d}$		b	

$w_2 = \bar{a} + b\bar{d}$

w_3

dc \ ba	00	01	11	10
00	0	0	0	-
01	-	-	1	-
11	-	-	0	-
10	-	-	-	-

d	c	b	a
0	1	0	0
0	1	0	1
0	1	1	1
0	1	1	0
$\bar{d}$	c		

$w_3 = c\bar{d}$

4. Programowanie – stan normalny

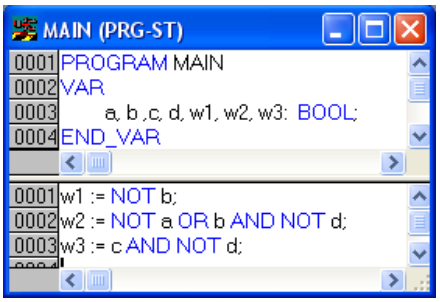
C

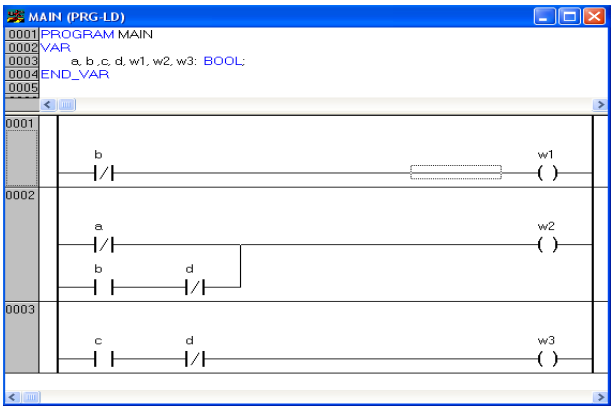
```
char d,w1,w2,w3;
```
– dodatkowe deklaracje

…

```
w1=!b; w2=!a||b&&!d; w3=c&&!d;
```

ST



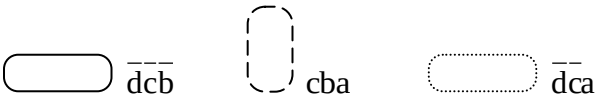


5. Niepoprawne pomiary

Tablica poprawności

dc \ ba	00	01	11	10
00	1	1	1	0
01	0	0	1	0
11	0	0	1	0
10	0	0	0	0

Zmienna P



$$P = \bar{d}\bar{c}\bar{b} + cba + \bar{d}\bar{c}a$$

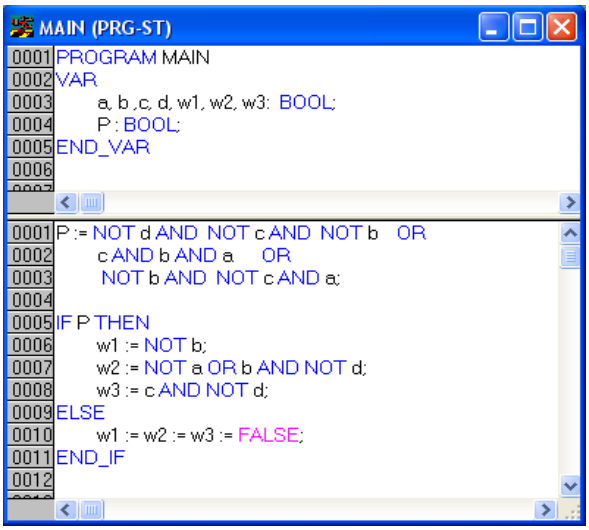
Programowanie

C

```
char P;
...
P=!d&&!c&&!b || c&&b&&a || !d&&!c&&a;

if(P)
     {w1=!b; w2=!a||b&&!d; w3=c&&!d;}
else
     w1=w2=w3=0;
```

ST


MAIN (PRG-ST)
0001 PROGRAM MAIN
0002 VAR
0003 a, b ,c, d, w1, w2, w3: BOOL;
0004 P : BOOL;
0005 END_VAR
0006
0001 P := NOT d AND NOT c AND NOT b OR
0002 c AND b AND a OR
0003 NOT b AND NOT c AND a;
0004
0005 IF P THEN
0006 w1 := NOT b;
0007 w2 := NOT a OR b AND NOT d;
0008 w3 := c AND NOT d;
0009 ELSE
0010 w1 := w2 := w3 := FALSE;
0011 END_IF
0012