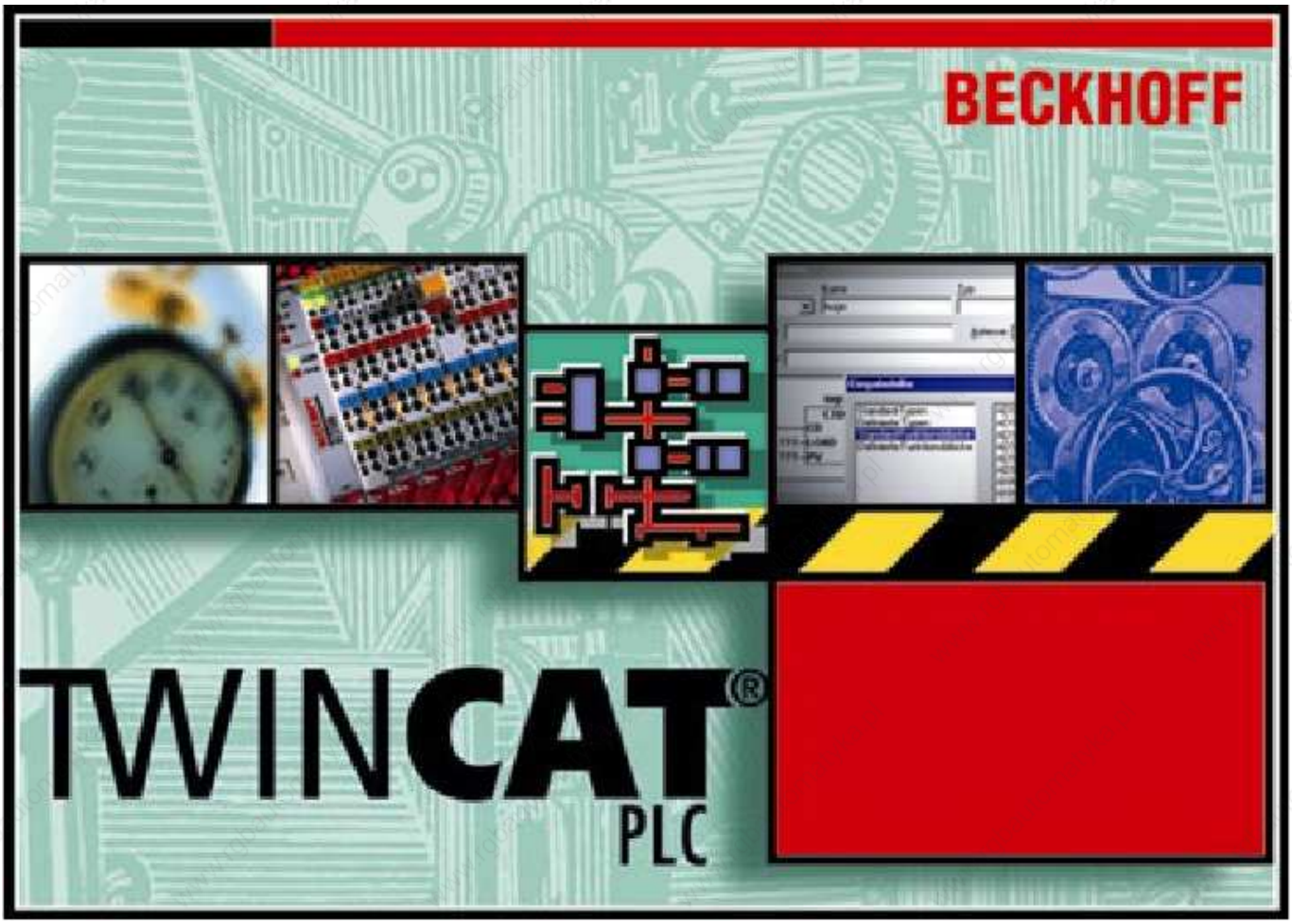


Podstawy obsługi programu TwinCAT PLC Control

Część 2

Wersja dokumentacji 1.1

Warszawa 2011

Wstęp

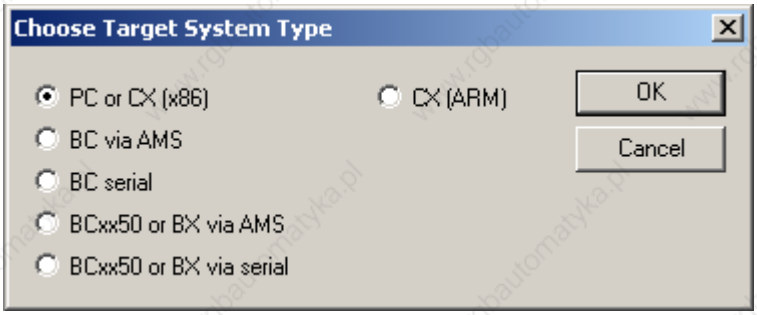
Niniejsza dokumentacja to jedynie krótki opis najważniejszych funkcji programu TwinCAT PLC Control. Niektóre zagadnienia zostały wyjaśnione skrótowo, a niektóre problemy znacznie uproszczone. Pełna dokumentacja dostępna jest w Beckhoff Information System oraz na infosys.beckhoff.com.

TwinCAT PLC Control jest narzędziem programistycznym pracującym w środowisku Windows NT/2000/XP/Vista/7 umożliwiającym tworzenie, zarządzanie oraz edytowanie programów PLC w językach normy IEC 61131-3: IL, LD, FBD, CFC, SFC, ST. Pozwala on na tworzenie funkcji, bloków funkcyjnych i programów, które można zapisać w formie projektu lub biblioteki. Poprawność działania algorytmów można sprawdzić dzięki wbudowanej opcji symulacji i wielu funkcjom diagnostycznym (debugger). TwinCAT PLC Control pozwala również na utworzenie wizualizacji wyświetlanej lokalnie lub poprzez www. Konfigurację sprzętową realizuje się za pomocą oprogramowania TwinCAT System Manager, które zostało opisane w oddzielnej dokumentacji.

1 Tworzenie nowego projektu

1.1 Typ projektu

Nowy projekt tworzymy wybierając z menu ***File → New***. W oknie ***Choose Target System Type***, wskazujemy typ sterownika, na który będzie wysłany program.

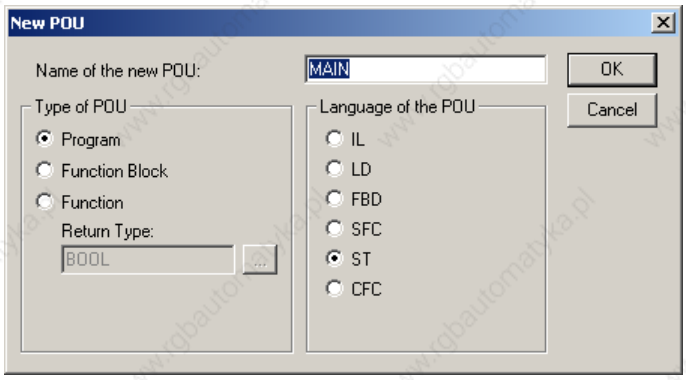


- PC or CX (x86) – opcja używana do połączenie wszystkich komputerów klasy PC oraz sterowników z architekturą procesora x86 np. sterowniki z serii CX10xx, panele z serii CP67xx, wszystkie komputery.
- CX (ARM) – opcja używana do połączenia sterowników z architekturą procesora ARM np. z serii CX90xx lub panele CP66xx.
- BCxx50 or BX via AMS – do połączenia wykorzystywany jest protokół komunikacyjny ADS. Pozwala on nawiązać połączenie ze sterownikiem wykorzystując interfejsy: Ethernet lub port szeregowy. Opcja może być używana do sterowników z serii BCxx50, BCxx20 oraz BX.
- BCxx50 or BX via serial – opcja używana do łączenia ze sterownikami przez port szeregowy. Sposób połączenia wykorzystywany dawniej, wymagał określenia adresów wejść/wyjść. Niezalecany w nowych aplikacjach. Opcja może być używana do sterowników z serii BCxx50, BCxx20 oraz BX.
- BC via serial i BC via AMS analogicznie jak wyżej dla sterowników BCxx00

Uwaga! Opcję wyboru typu sterownika można zmienić również po utworzeniu projektu w zakładce ***Resources → PLC Configuration***.

1.2 Typ obiektu

Po potwierdzeniu wyboru typu sterownika pojawia się okno **New POU** (Program Organisation Unit). Okno pozwala utworzyć program główny oraz wybrać dla niego język programowania.



Jednostką organizacyjną może być funkcja, blok funkcyjny lub program.

- **Program** może być wywoływany bezpośrednio przez task (opisany w rozdziale **4. Task**) lub w innym programie. Nie wymaga deklaracji w polu zmiennych – zawiera jedną instancję (każde wywołanie alokuje ten sam obszar pamięci), dlatego powinien być wywołany raz w cyklu i korzystać tylko ze zmiennych lokalnych bądź globalnych. W programie mogą być wywoływane funkcje, bloki funkcyjne i inne programy. Wartości zmiennych wewnętrznych widoczne są w trybie online.
- **Function block (blok funkcyjny)** może być wywołany w innym bloku lub programie. Sam może wywołać blok funkcyjny bądź funkcję. Każda instancja alokuje nowy obszar pamięci, w związku z tym wartości zmiennych przechowywane są do następnego wywołania bloku funkcyjnego i można podejrzeć je w trybie online. Blok funkcyjny może posiadać dowolną liczbę wejść i wyjść. Przykłady: R_Trig (wykrycie zbocza narastającego), FB_BasicPID (regulator PID).
- **Function (funkcja)** może wywoływać tylko inne funkcje, a sama może być wywołana wszędzie. Może mieć wiele wejść, a tylko jedno wyjście – zwracające jej wartość. Nie wymaga deklaracji, ponieważ posiada jedną instancję - alokuje zawsze ten sam obszar pamięci. W związku z tym nie przechowuje wartości zmiennych wewnętrznych i nie możliwe jest ich podejrzenie w trybie online. Przykłady: AND (iloczyn logiczny), REAL_TO_INT (funkcja konwersji typów), SIN (sinus).

1.3 Języki programowania

Języki programowania dostępne w TwinCAT są zgodne ze standardem IEC 61131-3, który jest międzynarodową normą dotyczącą programowania sterowników PLC. Zaletą tego standardu jest jego uniwersalność. Ta sama struktura tworzenia programów stosowana jest przez wielu producentów. Dostępne są dwa tekstowe oraz cztery graficzne języki programowania.

- Tekstowe:

- IL – Instruction List
- ST – Structured Text

- Graficzne

- LD – Ladder Diagram
- FBD – Function Block Diagram
- SFC – Sequential Function Chart
- CFC – Continous Function Chart

1.3.1 IL – Instruction List

Jest to język niskiego poziomu, strukturą języka podobny do asemblera. Kod składa się z szeregu instrukcji, z których każda rozpoczyna się nową linijką i w zależności od rodzaju operacji zawiera jeden lub więcej operatorów oddzielonych przecinkami. Przed instrukcją może znajdować się nazwa etykiety zakończona dwukropkiem. Linie mogą być oddzielone pustymi wierszami a komentarze mogą znajdować się w linii za instrukcją.

Przykład:

```
LD    17
ST    lint       (* komentarz *)
GE    5
JMPC  next
LD    idword
EQ    istruct.sdword
STN   test
next:
```

1.3.2 ST – Structured Text

Język ten jest podobny do języków wyższego rzędu (np. C, Pascal). Możliwe jest stosowanie instrukcji warunkowych oraz pętli (IF..THEN..ELSE; WHILE..DO; FOR itp.) Język ST pozwala na konstruowanie złożonych wyrażeń. Wyrażenia składają się z operatorów i argumentów. Wyrażenie zwraca wartość po zakończeniu obliczeń. Kolejność obliczania jest realizowana na podstawie ważności operatorów. Najpierw wykona się operacja najsilniejszego operatora a na końcu operatora najsłabszego. W przypadku wystąpienia kilku równoważnych operatorów, operacje zostaną wykonane od lewej do prawej. Blok funkcyjny wywoływany jest poprzez napisanie nazwy zmiennej a w nawiasie wypisanie jego zmiennych wejściowych lub wyjściowych. Poniżej zostało przedstawiona przykładowa deklaracja oraz wywołanie bloku funkcyjnego.

```
0002 VAR
0003     tonTimer: TON;
0004     bStart: BOOL;
0005     bOut AT %Q*: BOOL;

0001 tonTimer(IN:=bStart, PT:=t#10s, Q=>bOut, ET=>timET);
0002
0003
```

Kod wykonywany jest od góry do dołu, od lewej do prawej, zachowując ważność operatorów.

przypisanie

```
0001 rTemp:= rDane;
0002 iLicznik := iLicznik + 1;
0003 rWynik:=SIN(rSygnal);
0004
```

IF

```
0001 IF rSygnal > 100 THEN
0002     rWyjscie := rSygnal;      (*kod jest wykonywany *)
0003                               (* jeśli warunek rSygnal > 100 jest  spełniony *)
0004
0005 ELSIF rSygnal < 10 THEN      (* jeśli warunek rSygnal > 100 nie jest  spełniony *)
0006     rWyjscie := 100;          (* sprawdzany jest warunek rSygnal < 10 *)
0007                               (* jesli jest spełniony wykonywany jest kod po then *)
0008 ELSE
0009     rWyjscie := rWyjscie + 1;     (* w pozostałych przypadkach wykonywany *)
0010                                   (* jest kod po else *)
0011 END_IF
```

CASE

```
0001 (* instrukcje będą wykonywane          *)
0002 (*w zależności od stanu zmiennej iStep *)
0003 CASE iStep OF
0004 1:    bStart := TRUE;
0005 2:    bStart := FALSE;
0006       bEnd := TRUE;
0007 3:    iTemp := 30;
0008       bStart := FALSE;
0009 ELSE
0010       iTemp := 0;
0011       bStart := FALSE;
0012       bEnd := FALSE;
0013 END_IF
```

FOR

```
0001 FOR iIndex:=0 TO 10 DO
0002   IF aiTablica[iIndex] = 70 THEN
0003       aiTablica[iIndex] := aiTablica[iIndex] - 20;
0004   END_IF;
0005 END_FOR;
0006
```

WHILE

```
0001 (* jesli warunek iCouner <= 100 jest spełniony      *)
0002 (* będzie ciągle wykonywany kod znajdujący się *)
0003 (*  w pętli while *)
0004 WHILE iCounter <= 100 DO
0005       iCounter := iCounter+2;
0006 END_WHILE;
```

REPEAT

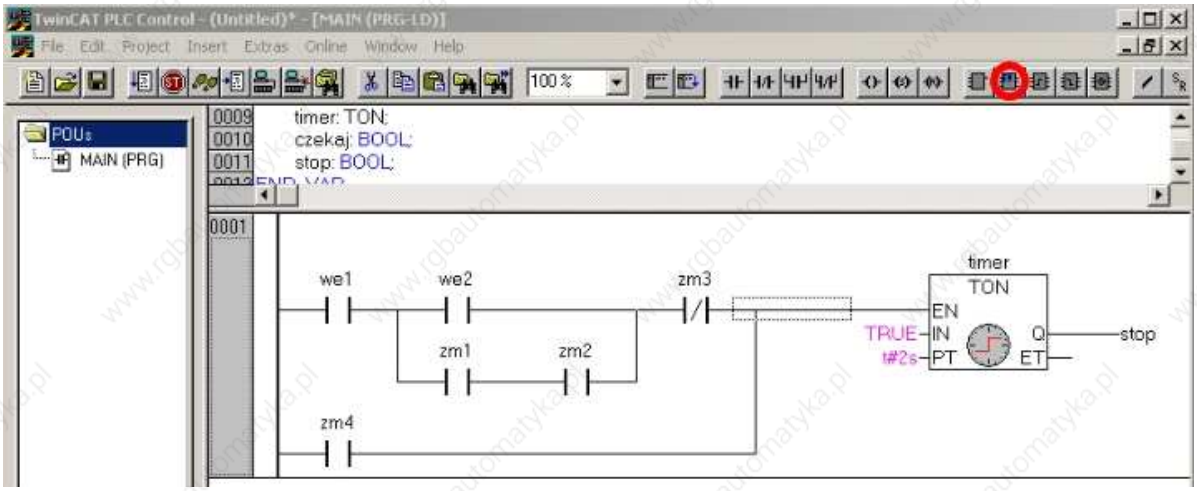
```
0001 (* jesli warunek iCouner <= 100 OR NOT bEnd   *)
0002 (* jest spełniony,  kod znajdujący się        *)
0003 (*w pętli REPEAT będzie wykonywany             *)
0004 REPEAT
0005       iCounter:=iCounter+2;
0006 UNTIL iCounter <= 100 OR NOT bEnd
0007 END_REPEAT;
```

Język ST bardzo często stosowany jest do tworzenia programów, bloków funkcyjnych wymagających zastosowania obliczeń matematycznych.

1.3.3 LD – Ladder Diagram

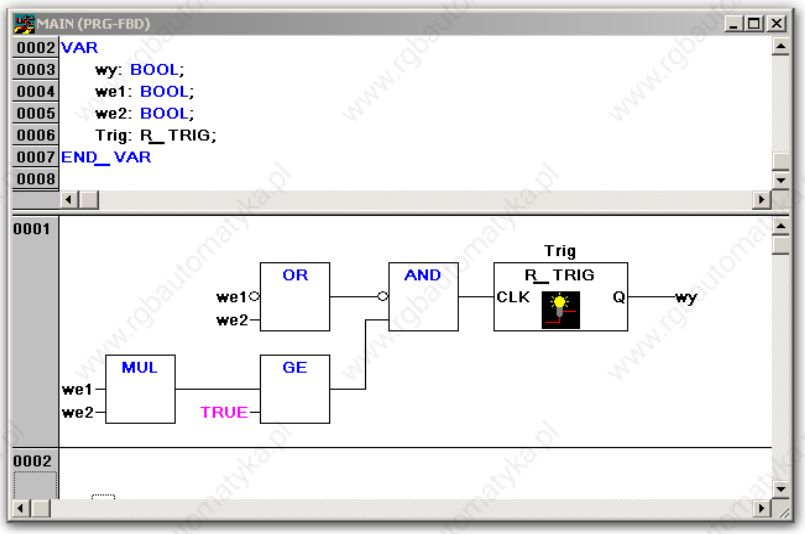
Graficzny język programowania. Kod napisany w języku LD zbliżony jest do struktury obwodu elektrycznego zawierającego połączone szeregowo lub równolegle elementy takie jak: styki, cewki, liczniki, przekaźniki czasowe itp.. Kod podzielony jest na linie (tzw. Network). Linie wykonywane są od góry do dołu. W języku drabinkowym najczęściej używa się zmiennych typu BOOL, tworząc rozbudowane warunki logiczne. Jeżeli warunek jest prawdziwy to sygnał przekazywany jest do kolejnego elementu. Operacje na innych typach zmiennych, np. INT, REAL, wykonane mogą być przez specjalnie wywołane bloki funkcyjne i funkcje - „box with EN” (kliknięcie PPM na linii bądź wybranie z górnego menu). Wybrany blok pojawi się w oknie kodu

programu w zaznaczonej linii (networku). Wstawiony blok funkcyjny jest blokiem typu AND. Chcąc wybrać inny typ bloczka funkcyjnego należy kliknąć na jego nazwę, ręcznie wpisać typ bloku funkcyjnego lub wybrać klawisz F2. W oknie Input assistant należy wybrać określony typ bloku funkcyjnego. Zaletą programów napisanych w języku drabinkowym jest łatwość analizy i czytelność kodu.

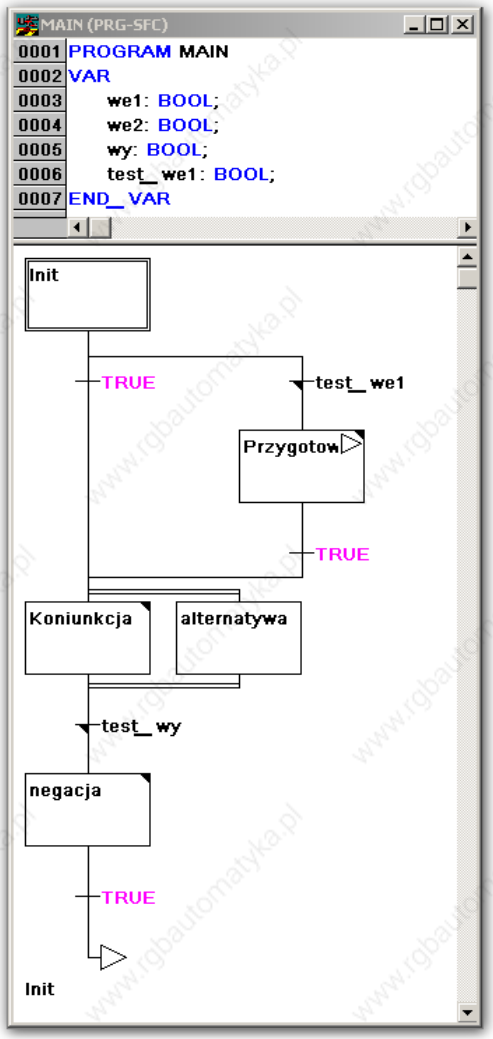


1.3.4 FBD - Function Block Diagram

FBD jest językiem graficznym. Kod tworzą linie wykonywane w określonej kolejności – linie wykonywane są od góry do dołu a ich zawartość od lewej do prawej. Linia zawierać może połączenie różnego typu elementów i dowolnych typów zmiennych – to daje większą swobodę niż język LD.

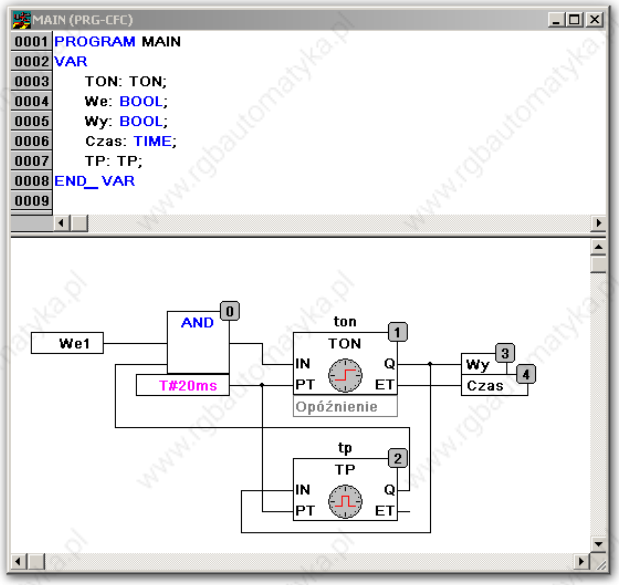


1.3.5 SFC – Sequential Function Chart



Język programowania, który graficznie przedstawia kolejność poszczególnych działań. Kod ujęty jest w blokach (**krokach**), które definiowane są przez użytkownika. Podczas tworzenia danego kroku programista może korzystać z dowolnego języka. Możliwe jest określanie warunku przejścia do następnego kroku (bloku), przy czym wartość wyjściowa warunku musi być TRUE lub FALSE. W języku SCF takie przejścia nazywane są tranzycjami. Kod wykonywany jest od góry do dołu, przy czym możliwe jest równoległe łączenie kroków. Wtedy dany poziom wykonywany jest kolejno od lewej do prawej. Język wykorzystywany jest, gdy należy zapewnić ścisłą kolejność wykonywania danego algorytmu. SFC stosowany jest np. do tworzenia programów sterujących sygnalizacją świetlną, procesów sekwencyjnej obróbki.

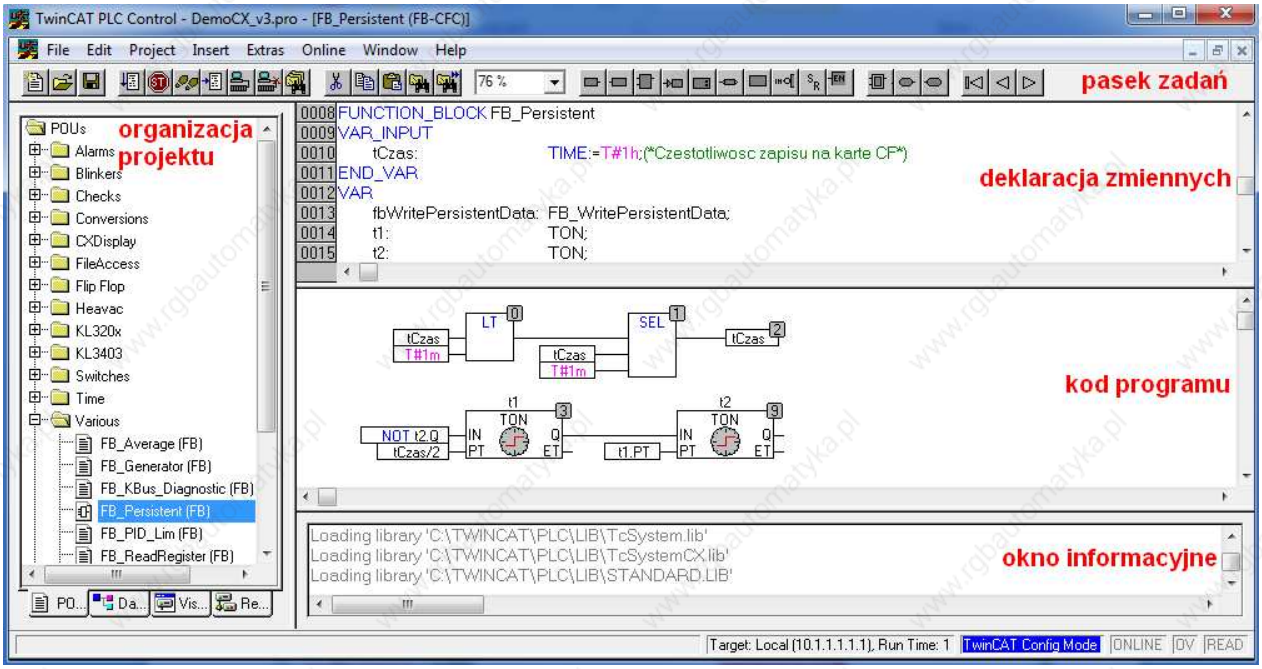
1.3.6 CFC – Continous Function Chart



Kod tworzony jest z elementów, które mogą być umieszczane w dowolnym miejscu na ekranie. Wejścia i wyjścia można dowolnie łączyć, a połączenia te są obrazowane liniami. Można wielokrotnie odwoływać się do tych samych zmiennych, wejść i wyjść. Bardzo łatwo tworzy się sprzężenia zwrotne. Kolejność wykonywania elementów kodu jest zgodna z przypisaną im numeracją niezależnie od ich umiejscowienia. Istnieje możliwość automatycznego i ręcznego numerowania bloków.

2 Główne okno programu

Edytor programu podzielony jest na 5 części: pasek zadań, organizację projektu, deklarację zmiennych, kod programu i okno informacyjne. Wyszczególnione części zostały opisane i pokazane na rysunku poniżej.

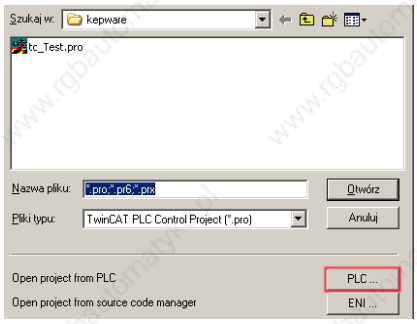


2.1 Pasek zadań, ikony skrótów.

Pierwsza grupa zawiera przyciski odpowiadające za tworzenie, otwieranie i zapisywanie programu:

"New" – tworzenie nowego projektu

"Open" (Ctrl+O) – Otwieranie projektu, po wybraniu ikony należy podać ścieżkę do pliku zapisanego na dysku lub wybrać opcję PLC odczytującą program ze strownika PLC



PLC.. – otwórz projekt z PLC, projekt zostanie otwarty tylko wtedy, gdy na sterownik został wysłany cały projekt wraz z kodem. Standardowo podczas wgrywania programu na PLC przesyłany jest tylko kod binarny - efekt pracy kompilatora.


"Save" (Ctrl+S) - zapisywanie projektu

Grupa przycisków odpowiadająca za start/stop programu oraz za debugger


„Run" (F5) - uruchomienie aplikacji w sterowniku


„Stop" (Shift+F8) - zatrzymanie aplikacji w sterowniku


"Toggle Breakpoint" (F9) - opcja debuggera, ustawia punkt w którym działanie punktu zostanie zatrzymane. W tym punkcie można podejrzeć wartości zmiennych oraz rozpocząć pracę krokową programu


"Step Over" (F10) - krokowe wykonanie programu, powoduje wykonanie program linijka po linijce, nie wchodząc do bloków funkcyjnych

Opcje logowania się na wybrany sterownik


"Login" (F11) – logowanie się na wybrany sterownik, w momencie logowania istnieje możliwość wgrania lub aktualizowania programu na sterowniku. TwinCAT pyta użytkownika jaką operację wykonać


"Logout" (F12) – wylogowanie ze sterownika

Grupa przycisków służąca do edycji programu


"Global Search" – wyszukiwanie w całym projekcie wpisanego ciągu znaków


"Cut" (Ctrl+X) – wycinanie zaznaczonego tekstu


"Copy" (Ctrl+C) – kopiowanie zaznaczonego tekstu


"Paste" (Ctrl+V) – wklejanie skopiowanego tekstu

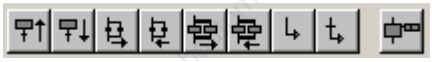

"Find" (Ctrl+F) – znajdowanie podanego ciągu znaków

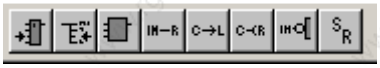

"Find Next" (F3) – znalezienie następnego dopoasowania

Pasek bloków funkcyjnych, wyświetlany tylko dla języków graficznych (inny dla każdego z nich)


dla języka LD


dla języka CFC

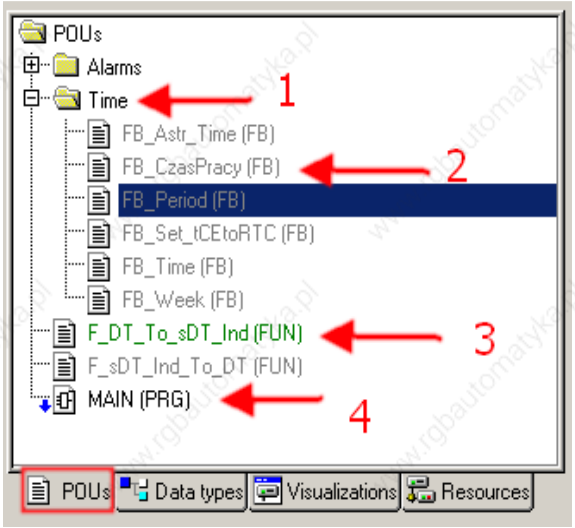

dla języka SFC


dla języka FBD

2.2 Menager organizacji projektu

Manager organizacji projektu jest narzędziem, które pomaga w zarządzaniu projektem. Menager wyświetla definiowane przez użytkownika programy, funkcje, bloki funkcyjne, wizualizacje, definiowane typy danych. Poniżej przedstawione są najważniejsze opcje menagera.

Zakładka POUs (Program Organization Unit)

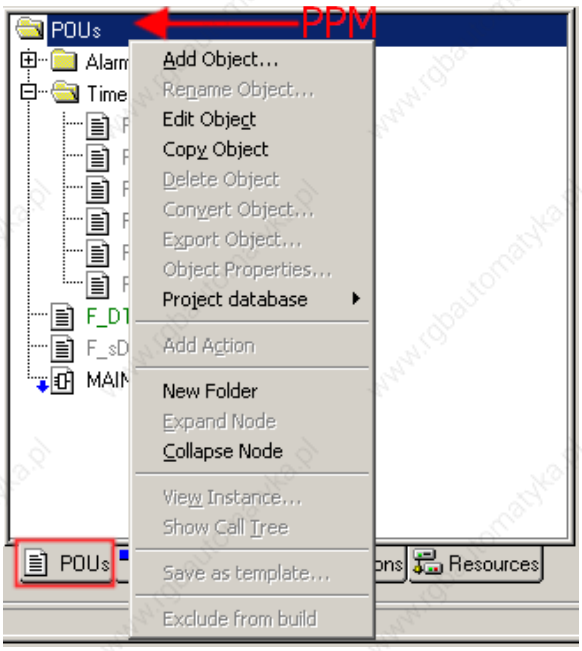


Zakładka przechowuje programy, funkcje i bloki funkcyjne. Elementy POUs można grupować w katalogi w celu lepszej organizacji pracy (nr 1).

Szara nazwa obiektu informuje programistę, że dany obiekt nie jest wywołany (wykorzystany) w programie (nr 2).

Nazwa POUs zaznaczona na zielono oznacza, że programista, wykluczył dany element kompilacji i jest sprawdzany pod kątem poprawności, wszystkie błędy w nim występujące są przez kompilator ignorowane (nr 3).

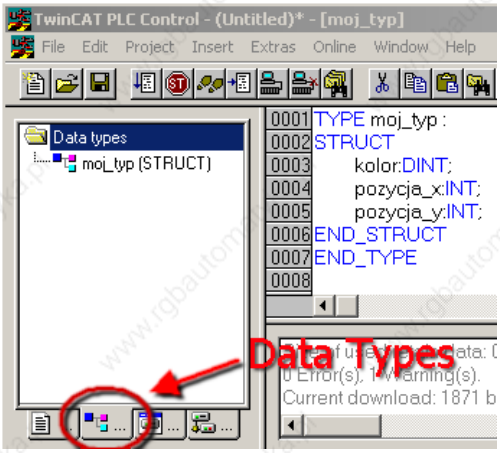
Kompilowane i używane POUs zaznaczane są kolorem czarnym.



Niebieska strzałka przy nazwie POUs informuje, że dany obiekt będący w pamięci sterownika różni się od tego w projekcie (nr 4) i różnice zostaną wgrane podczas następnego logowania.

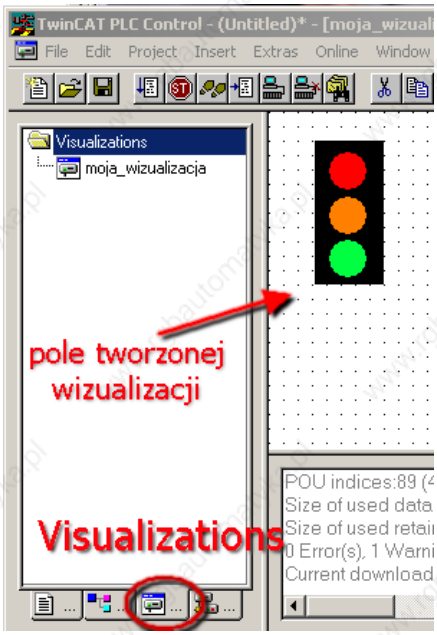
Klikając prawym przyciskiem myszy na folder POUs pojawia się menu kontekstowe, zawierające wiele opcji dotyczących obiektów. Najważniejsze z nich to tworzenie, edytowanie, usuwanie i eksportowanie obiektów.

Zakładka Data Types



Zakładka Data Types pozawala na tworzenie własnych typów zmiennych. Tworzone typy to najczęściej struktury i typy wyliczeniowe ENUM (Enumeration). Polami struktur mogą być zmienne typów standardowych lub stworzonych przez użytkownika.

Zakładka Visualizations

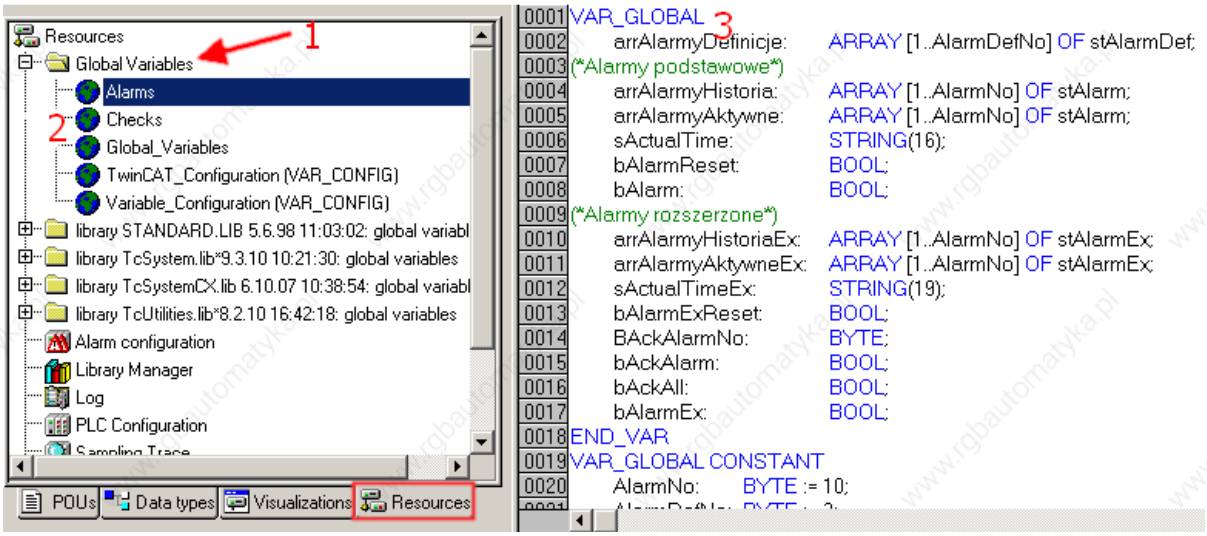


Zakładka Visualizations pozawala na tworzenie własnych wizualizacji. Wizualizacja może zawierać obiekty dynamiczne, których stan zależy od wartości zmiennych przypisanych w projekcie PLC.

Wizualizacje można przesłać do sterowników z serii CX, paneli operatorskich oraz komputerów przemysłowych, na którym zainstalowana jest biblioteka TwinCAT PLC HMI lub TwinCAT PLC HMI WEB. Nie ma ograniczeń typów oraz liczby przesyłanych zmiennych z PLC do wizualizacji. Biblioteka PLC HMI pozwala wyświetlać wizualizację na Panelu (PLC z serii CX przesyłają wizualizację poprzez złącze DVI). Biblioteka PLC HMI WEB pozwala wyświetlić wizualizację w przeglądarce internetowej.

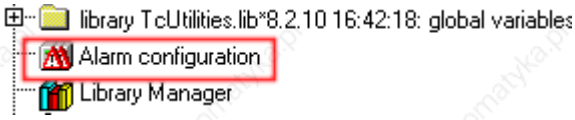
Zakładka Resources

Zmienne Globalne (Global Variables)



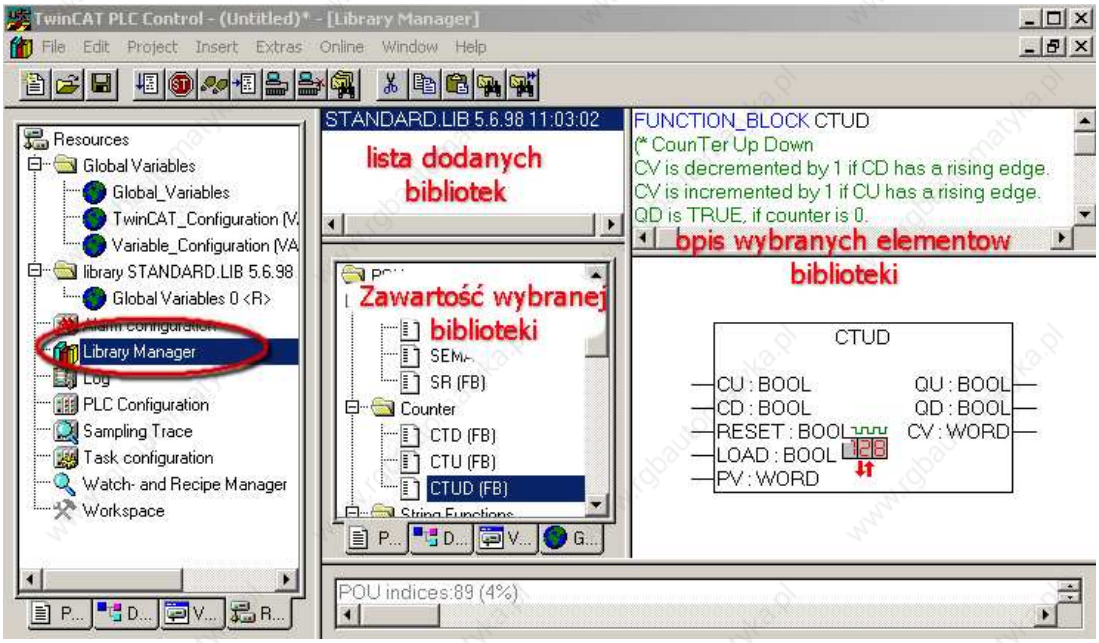
Pozycja Global Variables służy do deklaracji zmiennych globalnych. W folderze Global Variables oznaczonego nr 1 przechowywane są wszystkie zmienne globalne występujące w projekcie. Zmienne globalne można grupować tworząc nowe tzw. obiekty/foldery zmiennych (obiekty/foldery oznaczono na rysunku nr 2). Deklaracja zmiennych przebiega w oknie deklaracji oznaczonym nr 3. Składnia deklaracji jest taka sama jak zmiennych lokalnych.

Alarm Configuration



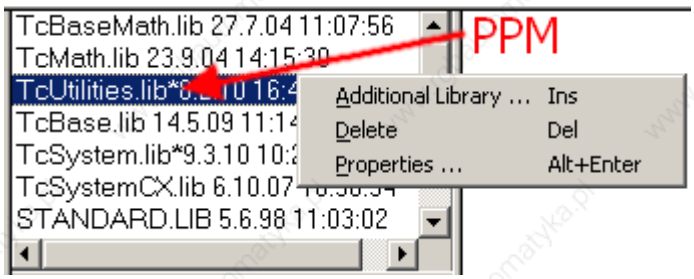
Alarmy są wspierane przez TwinCAT'a na sterownikach z Win XP

Library Manager



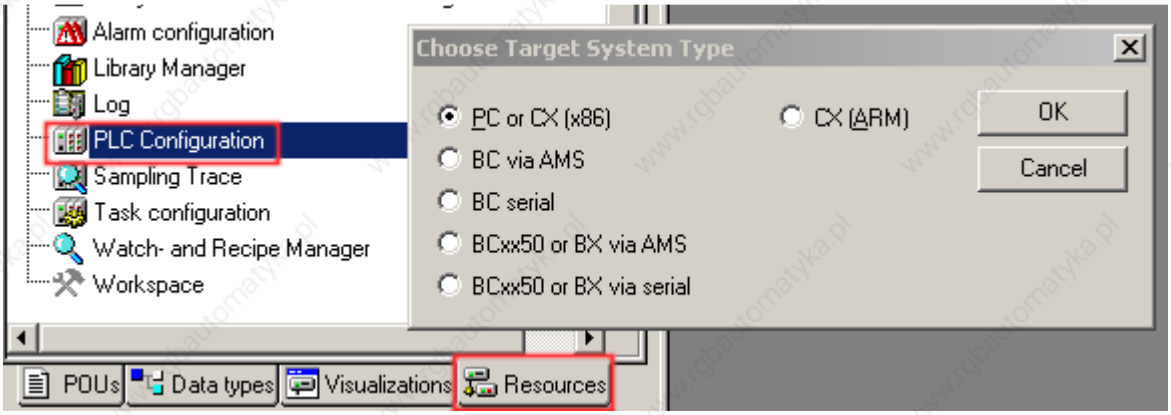
Library Manager pozwala efektywnie zarządzać bibliotekami dołączanymi do projektu. Zawiera informacje o aktualnie załączonych bibliotekach jak również pozwala na dołączanie kolejnych lub usuwanie tych nieużywanych.

Po utworzeniu programu dołączona jest do niego tylko standardowa biblioteka zawierająca podstawowe elementy. Dzięki temu program szybko się kompiluje i zajmuje mało miejsca w pamięci. Użytkownik może dodać do projektu inne biblioteki klikając prawym przyciskiem myszy na okno oznaczone jako „**lista dodanych bibliotek**". W managerze można przeglądać zawartość danej biblioteki oraz opis poszczególnych jej elementów.



Dodanie biblioteki jest możliwe po wybraniu funkcji ***Additional Library…*** z menu rozwiniętym po kliknięciu PPM w oknie z listą dodanych bibliotek.

PLC Configuration

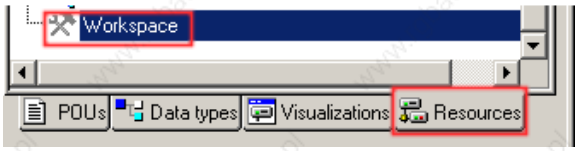


PLC Configuration pozwala wybrać typ sterownika, na który jest pisany program. Jest to to samo okno, które występuje podczas tworzenia projektu.

Sampling Trace

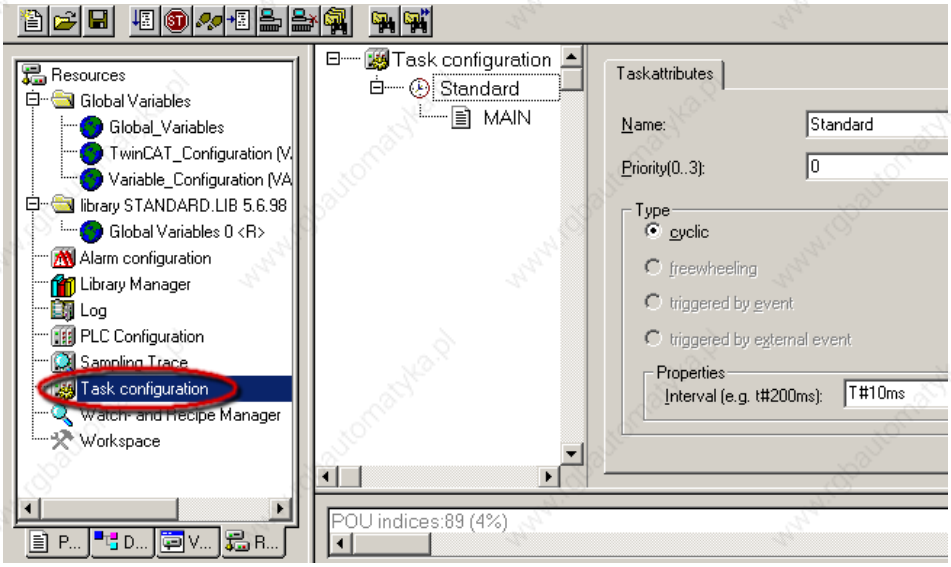
Narzędzie za pomocą którego można rysować wartości zmiennych na wykresie, obecnie zastąpione darmowym programem Scope View. Program ten jest instalowany razem z TwinCAT'em.

Workspace



Workspace służy do ustawiania opcji PLC Control – opcje zostały opisane w rozdziale **10.3 Menu Project**

Taks Configuration



Task configuration służy do konfiguracji tasków. Pozwala dodawanie, edytowanie, usuwanie, ustawianie priorytetu oraz ustawianie czasu cyklu tasku. Konfiguracja tasku jest opisana w rozdziale **4. Task(Wątek).**

Watch and Recipe Manager

Watch and Receipe Manager pozwala na podgląd wybranych zmiennych. Zmienne możemy podzielić na grupy. Grupy dodaje się w polu oznaczonym na rysunku nr 1 klikając **PPM**, wybierając menu kontekstowego ***New Watch List***. Zmienne dodajemy w polu oznaczonym nr 2 przez wybranie **F2** i uruchomienie **Input Assistant**.

Inną przydatną funkcją tego okna jest możliwość przechowywania aktualnych wartości zmiennych po wylogowaniu się. Pozwala to np. wpisać te wartości jako startowe do innego sterownika. W ty celu zmiennym należy nadać wartości startowe i będąc zalogowanym na sterowniku wybrać z menu **Extras → Read Recipe**. Po wylogowaniu wartości startowe będą zawierały odczytane wartości zmiennych (będą zapisane w projekcie, widoczne offline). W celu wpisania tych wartości do zmiennych na sterowniku należy w trybie online wybierać opcję **Extras → Write Recipe**. Wartości widziane w trybie offline będą teraz aktualnymi tych zmiennych.

2.3 Okno deklaracji zmiennych

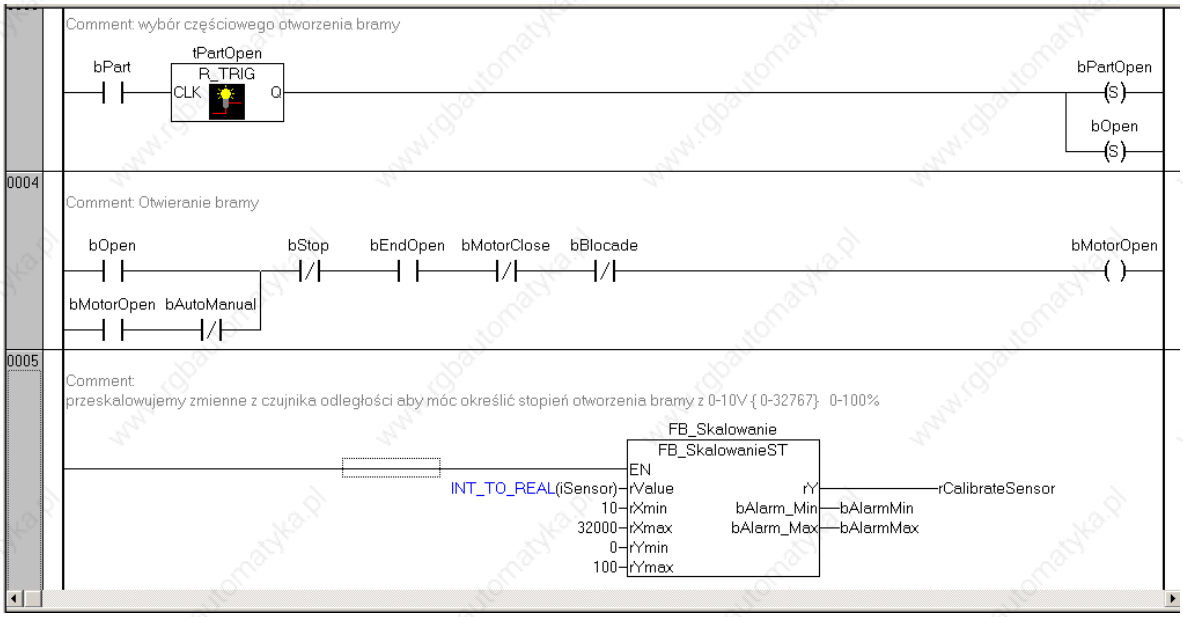
Zawiera deklaracje wszystkich zmiennych lokalnych danego elementu . Zmienne deklarowane są pomiędzy znacznikami np:. VAR END_VAR. Wszystkie znaczniki opisane zostały w rozdziale **5. Deklaracja zmiennych.**

Przykładowe okno deklaracji zmiennych przedstawiono na rysunku poniżej.

```
0001 FUNCTION_BLOCK FB_BlokFunkcyjny
0002 VAR_INPUT
0003     bInput : BOOL;                    (* zmienne wejściowe *)
0004     wStatus: WORD;
0005 END_VAR
0006 VAR_OUTPUT
0007     bOutput: BOOL;                    (*  zmienne wyjściowe *)
0008     byControl: BYTE;
0009 END_VAR
0010 VAR
0011     iZwyklaZmienna: INT;
0012     arrZawory: ARRAY [1..10] OF INT;  (* zmienna tablicowa *)
0013     byMemory AT      : BYTE;          (* zmienna odwołująca sie do     *)
0014                                       (* przestrzeni adresowej %MB0    *)
0015 END_VAR
```

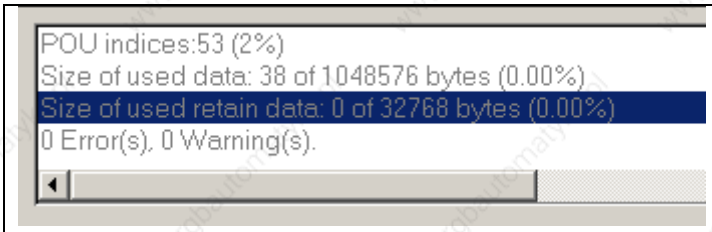
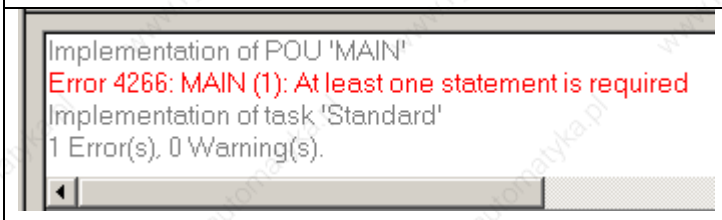
2.4 Okno kodu programu

Zawiera kod programu. Do pisania programu można wykorzystać 6 języków programowania opisanych w rozdziale **1.3 Języki programowania.** Poniżej pokazano przykładowy program napisany w języku drabinkowym LD.



2.5 Okno informacyjne

Okno można zamknąć i otworzyć wciskając kombinację klawiszy Shift+Esc. Pokazuje ono komunikaty, ostrzeżenia lub błędy wygenerowane przez kompilator podczas kompilacji programu oraz informacje dodatkowe np. o dodanych bibliotekach, zaimportowanych blokach - po ich dodaniu. Po zakończeniu kompilacji na samym dole okna dostępne jest podsumowanie zawierające dane na temat wielkości skompilowanego programu, wielkości zadeklarowanych zmiennych przechowywanych po zaniku zasilania oraz liczba znalezionych błędów i ostrzeżeń. W oknie informacji pokazują się również treść ostrzeżenia (warning:) oraz treść i numer błędu (error:). Dwukrotnie klikając na linijkę informującą o błędzie kursor zostanie przeniesiony w miejsce błędu w oknie kodu programu. Błędy należy usuwać od pierwszego znajdującego się na liście, ponieważ jeden błąd może generować kolejne. Program będzie skompilowany jeśli nie będzie zawierał żadnego błędu (0 Errors), może jednak zawierać ostrzeżenia (Warnings). Poniżej przedstawiono podsumowanie kompilacji, oraz wybrane komunikaty błędów i ostrzeżeń.

<table>
<tr><td> 
</td><td>Poprawna kompilacja programu, 0 błędów, 0 ostrzeżeń</td></tr>
<tr><td> 
</td><td>Niepoprawna kompilacja programu, 1 błąd, 0 ostrzerzeń.</td></tr>
<tr><td colspan="2">Wybrane ostrzeżenia</td></tr>
<tr><td>Warning: failed to create symbol information file</td><td>Tworzony projekt nie został zapisany na dysku, kompilator nie mógł stworzyć pliku „*.tpy”. Plik z rozszerzeniem tpy tworzony jest w katalogu projektu i przechowuje informację o deklarowanych zmiennych</td></tr>
</table>

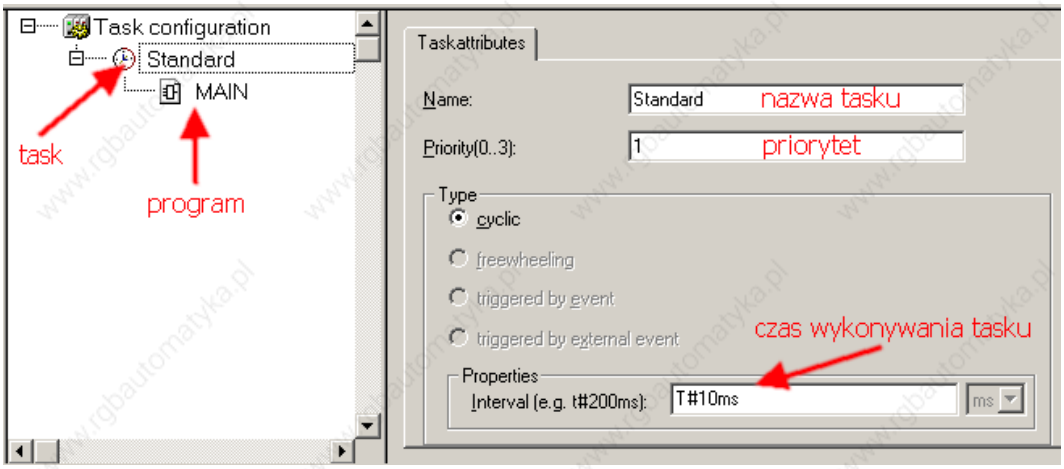
Warning 1990: No 'VAR_CONFIG' for 'MAIN.bInput'	Została zadeklarowana zmienna, która może łączyć się z fizycznym urządzeniem (np. D/A input/output lub port szeregowy), lecz zmienna ta nie została zlinkowana w System Managerze z żadnym urządzeniem.
Wybrane błędy	
Error 4266: MAIN (1): At least one statement is required	Wymagana chociaż jedna instrukcja wywołana w oknie programu napisanego w języku ST. Język ST wymaga użycia co najmniej jednej instrukcji, może to być instrukcja pusta zakończona średnikiem 
Error 4024: MAIN (5): Expecting ';' or ':=' before 'END_VAR' Error 3782: MAIN (5): Unexpected end	Błędna deklaracja zmiennej, brak średnika na końcu instrukcji lub błędna inicjalizacja początkowej wartości
Error 4001: MAIN (2): Identifier 'BZMIENNA' not defined	Brak deklaracji zmiennej o nazwie bZmienna.
Error 4024: MAIN (3): Expecting ';', ':=' or '(' before ''	Brak średnika kończącego instrukcję lub niepoprawne przypisanie.

3 Task(Wątek)

Aplikacje pisane w PLC Control na sterowniki z serii CX mogą mieć zdefiniowane maksymalnie 4 taski. Projekty przeznaczone dla sterowników z serii BX i BC mają zdefiniowany tylko jeden wątek. W każdym zdefiniowanym wątku należy wywołać przynajmniej jeden program. W jednym tasku można wywołać kilka programów. W tasku wywoływany może być tylko program. Task wykonywany jest cyklicznie, musi mieć on przypisany konkretny interwał czasowy oraz określony priorytet. Priorytety

przyjmują wartości od 0 do 4, przy czym wątek z priorytetem 0 jest najważniejszy i powinien mieć ustawiony najmniejszy czas cyklu. Wątki nie mogą mieć tego samego priorytetu. Każdy task może mieć przypisaną inną wartość interwału czasowego.

Domyślnie podczas tworzenia nowego projektu, pierwszy zdefiniowany program automatycznie jest przypisywany do pierwszego wątku. Domyślny czas cyklu w sterownikach CX to 10 ms a w sterownikach BC/BX to 20 ms. Wszystkie ustawienia związane z konfiguracją wątków znajdują się w PLC Control w zakładce ***Resources*** w pozycji ***Task Configuration.***



Dodając nowy wątek należy kliknąć PPM na drzewo Task coniguration i z menu kontekstowego wybrać spośród odstępnych opcji ***Appena Task***.

Analogicznie należy postąpić w przypadku przypisania programu do wątku, wybierając ***Append Program Call***. Standardowo wywołany jest już pierwszy task z dołączonym programem głównym. W przypadku usuwania programów z kolejki lub tasków – należy posłużyć się klawiszem ***delete*** lub klikając PPM na nazwę programu wybrać opcję ***Delete***.

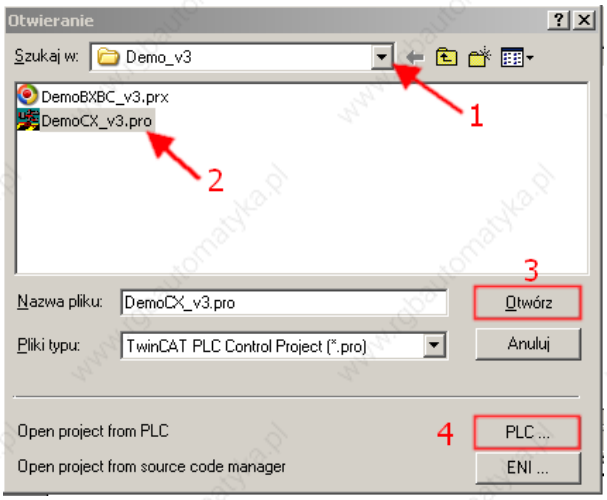
4 Edycja projektu

4.1 Tworzenie nowego projektu

Podczas uruchomienia TwinCAT PLC Control automatycznie otwierany jest ostatnio modyfikowany i zapisany projekt. W każdej chwili można stworzyć nowy projekt klikając w ikonę  lub wywołując polecenie z menu: ***File → New***

4.2 Otwieranie projektu

Chcąc otworzyć zapisany na dysku projekt należy wybrać z paska zadań ikonę folderu  lub wybrać z menu ***File → Open*** (Ctrl+O). Wybranie opcji spowoduje uruchomienie się okna pokazanego na rysunku poniżej.

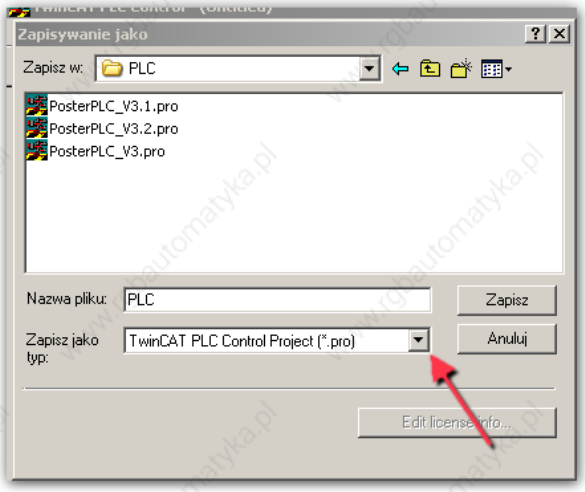


Wybierając ścieżkę do stworzonego projektu (nr 1) i odpowiedni plik z rozszerzeniem *.pro, *.prx lub *.pr6 (nr 2). Projekt zostanie otwarty po wybraniu przycisku ***Otwórz*** (nr 3) można otworzyć program zapisany na komputerze.

Wybierając opcję **Open project from PLC** – przycisk ***PLC*** (nr 4) można otworzyć program zapisany na sterowniku, jeżeli został on wcześniej tam wysłany (aby przesłać kod źródłowy na sterownik, po zalogowaniu się należy wybrać z menu ***Online → SourceCode download*** – jest to opisane w rozdziale **9.4 Menu online**).

4.3 Zapisywanie projektu

Po kliknięciu w ikonę (Ctrl+S) można zapisać wprowadzone zmiany w projekcie. Zapisanie projektu pod nową nazwą następuje po wybraniu polecenia z menu ***File → Save As***. Pojawia się okno dialogowe, w którym podajemy nazwę i nową lokalizację projektu:



Projekty zapisywane są w domyślnym formacie zależnym od sterownika, dla którego program jest przeznaczony (*.pro – PC i CX; *.pr6 – BC; *.prx – BC xx50 i BX). Odpowiedni typ pliku można wybrać z listy rozwijanej. Zmieniając rozszerzenie na *.lib projekt zapisany zostanie jako biblioteka. Biblioteka może podobnie jak i projekt zawierać typy zmiennych, funkcje i bloki funkcyjne definiowane w zapisywanym projekcie. Nie może zawierać tasków.

5 Deklaracja zmiennych

5.1 Wstęp

TwinCAT pozwala na tworzenie nazw zmiennych, które:

- Nie są słowami kluczowymi wykorzystywanymi przez kompilator np. VAR, END_VAR, TRUE, FUNCTION
- Nie są nazwami zmiennych, bloków funkcyjnych, np. REAL, TON
- Nie zawierają polskich znaków
- Nie zawierają znaków specjalnych jak: !, @, #, $, ^, &, *
- Nie zawierają pustych znaków, tj. spacja ‘ ‘
- Nie zaczynają się od cyfr
- Nie mogą różnić się jedynie wielkością znaków

Pole deklaracji zmienych

Deklaracje zmiennych umieszczane są pomiędzy słowami kluczowymi przedstawionymi poniżej:

VAR END_VAR	zmienne lokalne
VAR_INPUT END_VAR	zmienne wejściowe
VAR_OUTPUT END_VAR	zmienne wyjściowe
VAR_IN_OUT END_VAR	zmienne wejściowo – wyjściowe
VAR GLOBAL END_VAR	zmienne globalne

Komentarze

Komentarze zmiennych ograniczone są koloru zielonego i ograniczone znacznikami „(*” i „*)”, np. ***(* treść komentarza *)***. Komentarze można wstawiać w każde pole przyjmujące znaki. Komentarzy nie można stosować wewnątrz łańcucha znaków słów kluczowych, nazw zmiennych, nazw bloków funkcyjnych itp..

```
0029 PROGRAM MAIN
0030 VAR
0031 (*Wejścia cyfrowe*)
0032     bStart AT%IX0.0:BOOL;(*Start Maszyny*)
0033 (*Wejścia Analogowe*)
0034     wTErmopara AT%IW10(*Byte 10-11*):WORD;
0035 END_VAR
```

Poprawny komentarz

```
0029 PROGRAM MAIN
0030 VAR
0031     bStart AT%IX0.0:BOOL;
0032     wTErmopara AT%(*Byte 10-11*)IW10:WORD;
0033 END_(*koniec deklaracji *)VAR
```

Błędny komentarz

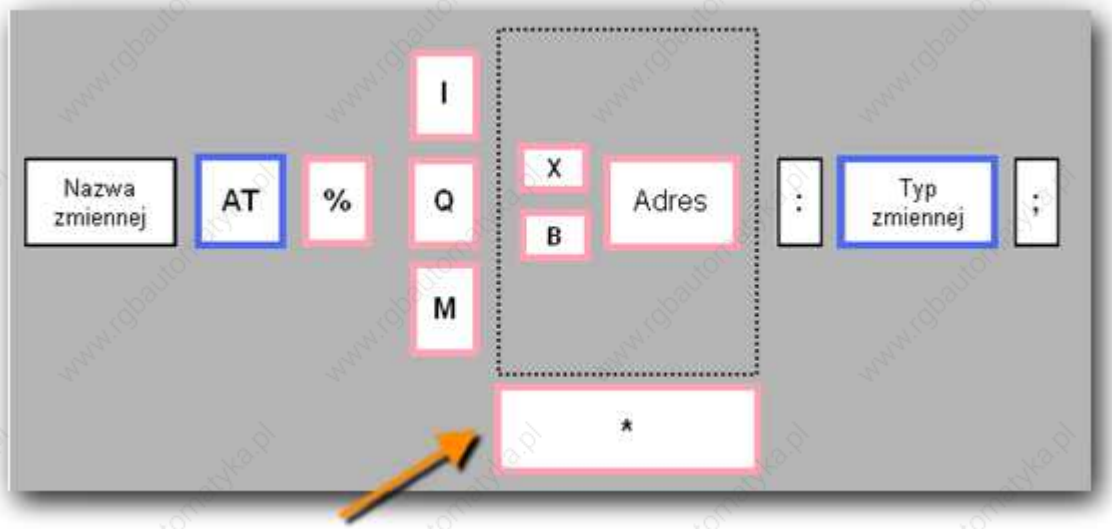
Prefix

TwinCAT pozwala na dowolność w tworzeniu nazw zmiennych. Dobrym nawykiem programistycznym jest stosowanie prefiksów. Poniżej przedstawiono przykłady tworzenia nazw zmiennych wykorzystujących Notację Węgierską.

Prefiks	Nazwa zmiennej
b – Boolean	bStart
r – Real	rAktualnaPozycja
s - String	sWiadomosc
ST_ - Definicja struktury	ST_Zawor (* Definicja *)
st – Deklaracja struktury	stZawrorPompy (* Deklaracja *)
FB_ - Definicja bloku funkcyjnego	FB_Skalowanie (* Definicja *)
fb	fbSkalTemp (* Deklaracja *)

5.2 Struktura Deklaracji

Ogólna struktura deklaracji zmiennych przedstawiona jest poniżej:



TwinCAT pozwala na zadeklarowanie dwóch grup zmiennych.

- **Zmienne wewnętrzne** – używane w obrębie sterownika. Ich deklaracja nie wymaga podawania adresu a tym samym nie są linkowane. Zmienne te nie mogą odwoływać się do urządzeń zewnętrznych. Deklaracja takiej zmiennej składa się z nazwy zmiennej i typu zmiennej. Przykładowa deklaracja:

  Nazwa_zmiennej : typ zmiennej;

  bStart : BOOL;

- **Zmienne zewnętrzne** – łączone są z fizycznymi urządzeniami w procesie linkowania np. zmienne odwołujące się do konkretnego fizycznego wejścia/wyjścia. Deklaracja takiej zmiennej składa się z dodatkowego słowa kluczowego **AT** oraz określenia rodzaju zmiennej i podania adresu zmiennej. Przedrostek zmiennej I, Q, M występujący po znaku **%** wskazuje obszar do jakiego zmienna będzie się odwoływać.
  - **I –** obszar zm. wejściowej (Input);
  - **Q –** obszar zm. wyjściowej (Output);
  - **M –** pamięć flag (Memory);

  Kolejnym elementem deklaracji jest określenie adresu zmiennej:
  - **X –** zmienna bitowa (logiczna) – wyrażona przez pojedynczy bit;
  - **B –** zmienna bajtowa (Byte) – wyrażona przez 8 bitów;
  - *** –** informuje kompilator, że adres będzie przydzielony podczas procesu linkowania

    Nazwa_zm1 AT %I* : BOOL;

    Nazwa_zm2 AT %MB0 : INT;

Zmienne zadeklarowane jako wejścia lub wyjścia powinny być zliknowane do odpowiadających im wejść i wyjść fizycznych w programie TwinCAT System Manager (program opisany w oddzielnym dokumencie dostępnym na ftp.beckhoff.com/poland/Pomoc/).

Podczas deklaracji zmiennej można przypisać jej **wartość początkową**. Wartość początkową można przypisać ręcznie poprzez dodania na końcu deklaracji:

bStart : BOOL := TRUE;

iOut AT %Q* :INT := 3300;

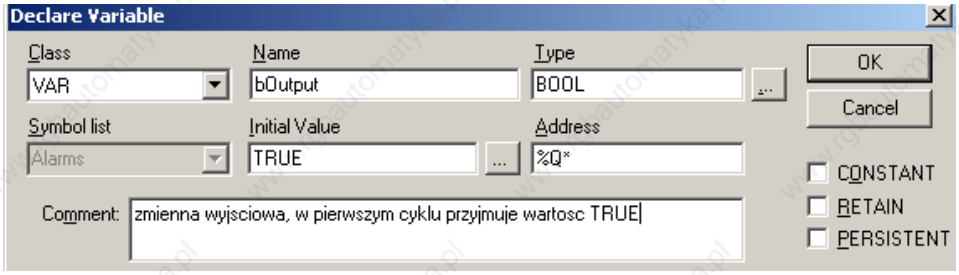
lub w polu Initial Value w oknie deklaracji zmiennych. Kreator deklaracji zmiennych opisany jest w rozdziale ***6.3 Kreator deklaracji zmiennej***.

Przykładowe deklaracje zmiennych:

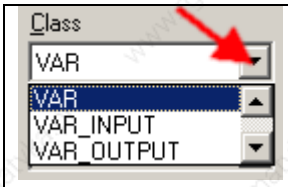
0002 VAR 0003 (*zmienne wewnętrzne *) 0004 bBool: BOOL; 0005 wWord : WORD:= 320; 0006 rReal: REAL; 0007 END_VAR	Zmienne lokalne wewnętrzne, których nie można łączyć z zewnętrznymi urządzeniami. Zmienna wWord zostanie zainicjalizowana wartością 320.
0001 VAR_GLOBAL 0002 (*Zmienne Globalne*) 0003 bBoolGlobalna :BOOL; 0004 bDigitalOutputGlobalna AT %Q*: BOOL := TRUE; 0005 END_VAR	Zmienne Globalne - deklarowane są w zakładce **Resources →Global Variables** Zmienne dostępne są we wszystkich POUs,

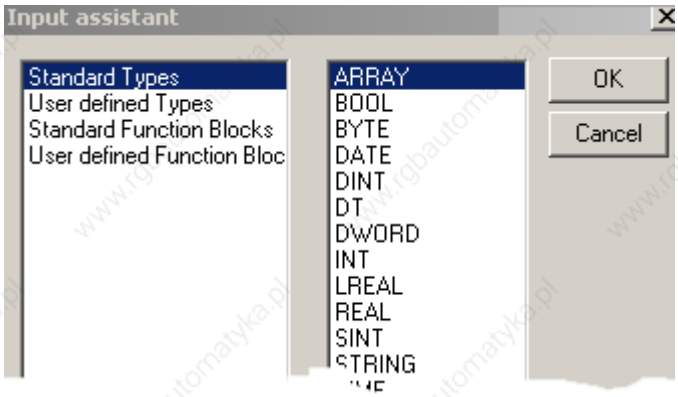
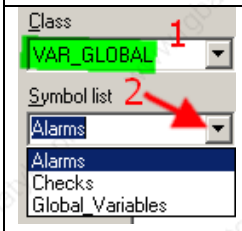
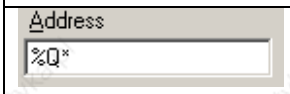
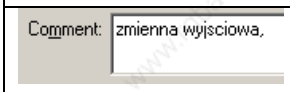
5.3 Kreator deklaracji zmiennej

Każda zmienna używana w programie musi być zadeklarowana. W przypadku użycia nowej nieznanej zmiennej w programie, edytor zapyta o jej deklarację. Automatycznie pojawia się okno deklaracji zmiennej ***Declare Variable***.



Poniżej przedstawione są dostępne w polu deklaracji opcje:

	Wybór klasy zmiennej: VAR – lokalna, VAR_INPUT – wejściowa, VAR_OUTPUT – wyjściowa, VAR _IN_OUT – wejściowo-wyjściowa, VAR_GLOBAL – globalna
	Nazwa zmiennej – nazwę należy wpisać ręcznie lub zatwierdzić istniejącą

	Typ zmiennej można wpisać ręcznie lub wybrać przycisk, który otworzy okno ***Input Assistant***. Pozwala ono na wybór typów standardowych lub własnych, zdefiniowany przez programistę. 
	Pole ***Symbol List*** (nr 2) pozwala wybrać grupę zmiennych, do której można dołączyć deklarowaną zmienną. Pole dostępne jest tylko przy deklaracji zmiennej globalnej wybrana klasa zmiennej VAR_GLOBAL (nr 1).
	W polu można nadawać wartość inicjalizacyjną – zmienna przyjmie ją w pierwszym cyklu programu
	Jeśli zmienna ma być linkowana z zewnętrznym wejściem/wyjściem to wpisujemy adres. Dla zmiennej wewnętrznej pole zostaje puste.
	Komentarz wyświetlany w linii deklaracji zmiennej,
	Deklaracja zmiennej stałej – nie można zmienić jej wartości w programie. Zmiennej można tylko incjalizować wartość.
	Deklaracja zmiennej nieulotnej typu RETAIN[1]
	Deklaracja zmiennej nieulotnej typu PERSISTENT[2]

W celu modyfikacji zmiennej istnieje możliwość samodzielnego wywołanie okna ***Declare Variable*** poprzez kliknięcie PPM w polu deklaracji zmiennej na jej nazwę i z rozwiniętego menu wybranie opcji **Auto Declare (Shift+F2).**

[1] Zmienne RETAIN opisane są w osobnej dokumentacji dostępnej na ftp.beckhoff.com\poland\pomoc
[2] Zmienne PERSISTENT opisane są w osobnej dokumentacji dostępnej na ftp.beckhoff.com\poland\pomoc

5.4 Typy Standardowe

W bibliotece standardowej zdefiniowane zostały podstawowe typy zmiennych.

Typ zmiennej	Wartość min.	Wartość max.	Wielkość	prefix
BOOL	0	1	8 bit	b
BYTE	0	255	8 bit	by
WORD	0	65535	16	w
DWORD	0	4294967295	32	dw
SINT	-128	127	8	si
USINT	0	255	8	usi
INT	-32768	32767	16	i
UINT	0	65535	16	ui
DINT	-2147483648	2147483647	32	di
UDINT	0	4294967295	32	udi
REAL	~ -3.402823 x 10^{38}	~ 3.402823 x 10^{38}	32	r
LREAL	~ -1.797693134862 31E308	~ 1.797693134862 32E308	64 Bit	lr
STRING	-	-	Domyślnie 81 Bajtów	s
TIME	T#0ms	T#71582m47s2 95ms	32 Bit	t
TIME_OF_DAY (TOD)	TOD#00:00	TOD#1193:02:4 7.295	32	tod
DATE	D#1970-01-01	D#2106-02-06	32	date
DATE_AND_TIME (DT)	DT#1970-01-01-00:00	DT#2106-02-06-06:28:15	32	dt

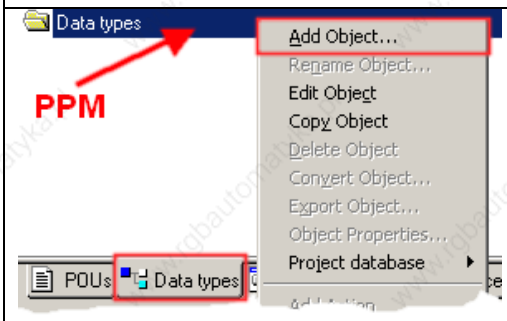
5.5 Typy Deklarowane przez użytkownika

W sytuacji, gdy standardowe typy są niewystarczające można zadeklarować własne typy. Użytkownik może tworzyć własne struktury, typy wybieralne (ENUM), definiować tablice typów.

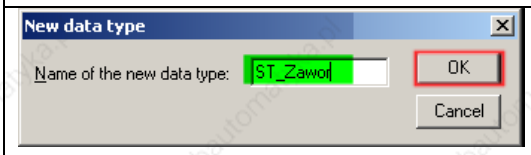
5.5.1 Deklaracja struktury

TwinCAT pozawala na tworzenie własnych struktur. Polami struktur mogą być zmienne typów standardowych lub stworzonych przez użytkownika. Struktura jest dobrym sposobem na grupowanie zmiennych odnoszących się do jednego obiektu np. zmienne służące do sterowania zaworu.

Tworzenie struktury



Tworzenie struktury odbywa się w Menagerze organizacji projektu w zakładce **Data Types** poprzez wybranie z menu kontekstowego opcji **Add Object…**



W oknie ***New data type*** należy wpisać nazwę struktury

Definicja struktury

```
0001 TYPE ST_Zawor :
0002 STRUCT
0003     sNazwa : STRING;
0004     iStopien_otwarcia : INT;
0005     bOtwarty : BOOL;
0006     bZamkniety : BOOL;
0007     bOtworz : BOOL;
0008     bZamknij : BOOL;
0009     bAwaria : BOOL;
0010 END_STRUCT
0011 END_TYPE
```

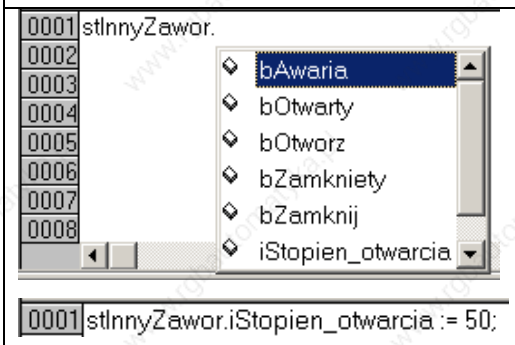
Po wpisaniu nazwy utworzy się okno definicji struktury. Pomiędzy słowa kluczowe STRUCT END_STRUCT należy wpisać wszystkie pola struktury.

Deklaracja struktury

```
0002 VAR
0003 (* deklaracja zmiennej typu ST_Zawor *)
0004     stZawor_1 :ST_Zawor;
0005     stInnyZawor :ST_Zawor :=  (sNazwa:='Z1',
0006                                iStopien_otwarcia:=50,
0007                                bAwaria:=FALSE);
0008 END_VAR
```

Deklaracja struktury odbywa się tak jak deklaracja zmiennej typu standardowego, czyli najpierw nazwa zmiennej a potem nazwa typu.

Strukturę można inicjalizować wartościami początkowymi wpisując w deklaracji w nawiasach nazwy i wartości pól oddzielone przecinkami. Pola pominięte będą inicjalizowane wartościami domyślnymi.

Odwołanie się do pola struktury	
	Do konkretnego pola struktury można się odwołać wpisując nazwę zmiennej kropkę i nazwę wybranego pola. Po wpisaniu kropki wyświetlana jest lista, z której można wybrać odpowiednie pole.

5.5.2 Deklaracja typu ENUM

Typ wyliczeniowy (Enumeration) służy do ograniczenia możliwości wyboru wartości. Zmienna ENUM ogranicza wybór do kilku/kilkunastu ściśle określonych przez programistę elementów oraz do odwoływania się do tych wartości za pomocą zdefiniowanej nazwy. Wewnątrz definicji typu wyliczeniowego umieszczona jest lista stałych zmiennych. Tylko te zmienne będzie można przechowywać w zdefiniowanym typie enum. Warto wiedzieć, że stałe, które wprowadzono w definicji typu wyliczeniowego reprezentują liczby całkowite. Każda nazwa posiada kolejną wartość. Domyślnie pierwszy element przyjmuje wartość zero a kolejne nazwy wartość o jedną większą. Programista może nadać również wartości nazwie, każda kolejna będzie miała wartość o jeden większą. Poniżej została przedstawiona definicja i deklaracja zmiennej typu enum.

5.5.3 Deklaracja tablicy

Tablica przechowuje wartości kilku zmiennych tego samego typu. Można zadeklarować tablice jedno-, dwu-, trójwymiarowe. Tablice mogą zawierać elementy dowolnego typu. Przykład deklaracji tablicy przedstawiono poniżej:

Przykładowe deklaracje tablic

(* tablica jednowymiarowa *) (*nazwa : słowo kluczowe [rozmiar: low.. max] of typ;*) aiTablica : ARRAY [1.. 10] OF INT;	deklaracja tablicy jednowymiarowej typu INT
(* tablica dwuwymiarowa *) abyTabDwuWym: ARRAY [1.. 10, 2.. 5] OF BYTE;	deklaracja tablicy dwuwymiarowej typu BYTE
(* tablica trójwymiarowa*) awTabTrzyWym: ARRAY [1.. 10, 2.. 5, 0.. 10] OF WORD;	deklaracja tablicy trójwymiarowej typu INT
(* tablica struktur *) astTabOfStruct: ARRAY [1.. 10] OF ST_Zawor;	deklaracja jednowymiarowej tablicy struktur typu ST_Zawor

Inicjalizacja wartości początkowych tablicy

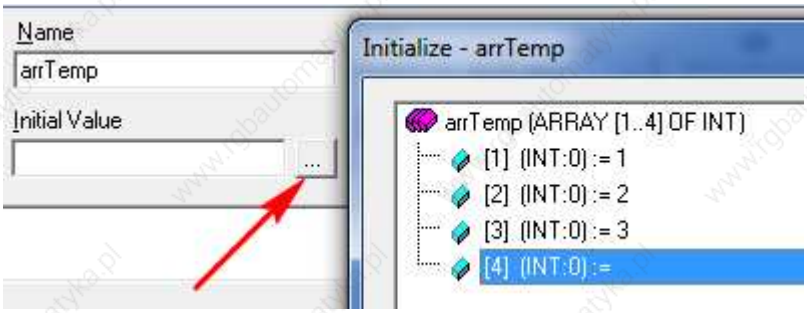
Wartości początkowe można wprowadzić:

- W polu deklaracji zmiennych – wpisując ręcznie wartości startowe:

(*inicjalizaja tablicy *) adwTab: ARRAY[0.. 3] OF DWORD:= 1,2,3,4;	inicjalizacja elementów tablicy wartościami 1,2, 3,4.

```
astZawory: ARRAY [1..10] OF ST_Zawor := (sNazwa:='Z1',iStopien_otwarcia:=50,bAwaria:=FALSE),
                                        (sNazwa:='Z2',iStopien_otwarcia:=34),
                                        (sNazwa:='Z3',iStopien_otwarcia:=100),
                                        (sNazwa:='Z4',iStopien_otwarcia:=0),
                                        (sNazwa:='Z5',iStopien_otwarcia:=56,bAwaria:=TRUE),
                                        (sNazwa:='Z6',iStopien_otwarcia:=63);
```

- W oknie deklaracji zmiennej – wciskając symbol „…” obok pola Initial Value i uzupełniając odpowiednie pola:



Odwołanie się do elementu tablicy	
Do tablicy można odwołać się podając za nazwą tablicy w nawiasach kwadratowych nr pola tablicy. Przykładowe odwołanie się do elementu tablicy przedstawiono poniżej	
001 aiTablica[1] : = 4;	wpisanie do pierwszego elementu tablicy jednowymiarowej wartości 4
002 byTemp := abyTabDwuWym[1,2];	przepisanie elementu [1,2] tablicy dwuwymiarowej do zmiennej byTemp
006 astZawory[3].sNazwa := 'Zawor 1';	odwołanie się do pola struktury umieszczonej w 3 elemencie tablicy
Przykład wykorzystania tablicy	
W tablicy umieszcza się wiele elementów tego samego typu. Tablice doskonale nadają się zarządzania wieloma elementami. **Wykorzystując pętle for można szybko i wygodnie zarządzać wszystkimi wartościami w tablicy**. Instrukcja często wykorzystywana jest do przeglądania, sortowania, wyszukiwania, zmieniania elementów tablicy. Poniżej pokazany jest sposób ustawiania wartości zmiennej we wszystkich elementach tablicy:	
008 FOR i:= 1 TO 10 DO 009 astZawory[i].bZamknij := TRUE; 010 END_FOR lub 008 FOR i:= 1 TO 10 DO 009 astZawory[i].bZamknij := bPolecenie; 010 END_FOR	Stworzony skrypt ma za zadnie sterować dziesięcioma zaworami. Każdy zawór reprezentowany przez strukturę ST_Zawor przechowującą wszystkie zmienne sterujące zaworami. Wszystkie struktury zgromadzone są w tablicy struktur astZawory. Jeden element tablicy dotyczy jednego zaworu. Zadaniem przedstawionego fragmentu kodu jest przesłać sygnał zamknięcia do wszystkich zaworów. Pętla przechodzi zatem po wszystkich elementach tablicy i ustawia wartość TRUE w polu struktury bZamknij. Zmienna „i” zwana jest iteratorem pętli FOR.

6 Uruchomienie programu, linkowanie zmiennych

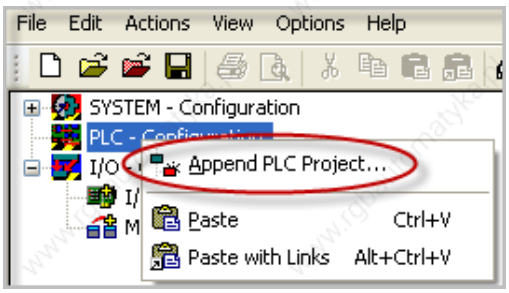
6.1 Kompilacja programu

Przed kompilacją programu, projekt należy zapisać na dysku. Najlepiej stworzyć sobie osobny katalog do każdego projektu, ponieważ w miejscu zapisu projektu kompilator utworzy pliki pomocnicze. Kompilacja rozpocznie się po wybraniu z menu opcji **Project → Rebuild**. Po kompilacji w oknie informacyjnym zostanie wyświetlone podsumowanie procesu kompilacji. Jeśli kompilator nie zgłasza żadnych błędów program może zostać wgrany na sterownik. Po poprawnej kompilacji automatycznie zostanie utworzony plik z rozszerzeniem ***.tpy**. Zostaje on zapisany w tej samej lokalizacji co projekt. Plik *.tpy zawiera różnorodne informacje, przechowuje również wszystkie szczegóły dotyczące zmiennych zewnętrznych (zmienne te w deklaracji zawierają słowo kluczowe AT). Chcąc połączyć zmienne zewnętrzne programu PLC z fizycznymi urządzeniami należy dodać plik *.tpy w TwinCAT System Manager a następnie zlinkować dodane zmienne[3].

Do kompilacji służy również polecenie **Project →Build (Ctrl+F8)**. Polecenie **Build** kompiluje tylko nowo wprowadzone lub zmodyfikowane fragmenty programu. Dzięki temu kompilacja trwa krócej. Czasem po kompilacji nie wszystkie wprowadzone zmiany będą działały. Należy zatem co pewien czas przebudować cały projekt kompilując go poleceniem **Rebuild all**.

6.2 Linkowanie zmiennych[4]

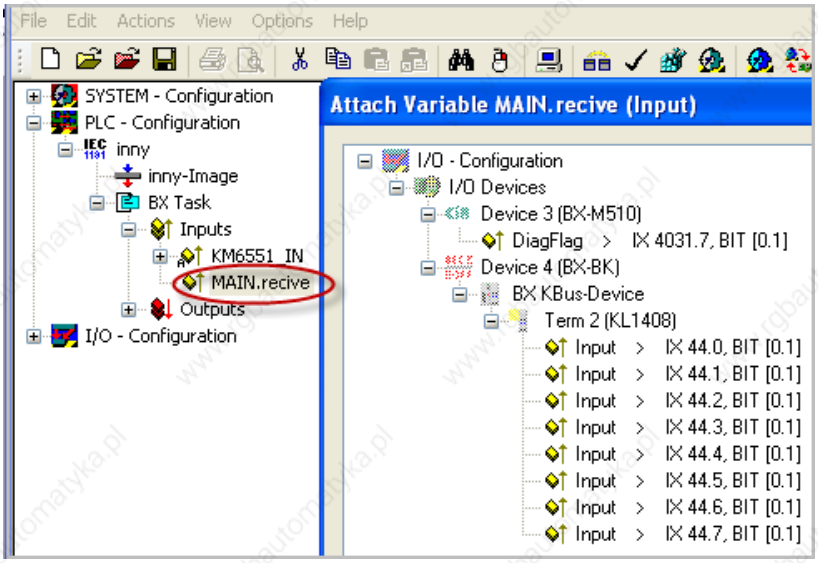
Plik *.tpy należy dodać w programie **TwinCAT System Manager** do przygotowanej wcześniej konfiguracji sprzętowej. Klikając PPM na zakładkę PLC – Configuration z menu kontekstowego należy wybrać opcję **Append PLC Project…**



[3] Proces linkowania zmiennych opisany jest szczegółowo w oddzielnej dokumentacji do programu TwinCAT System Manager dostępnej na ftp.beckhoff.com\poland\pomoc

[4] jw

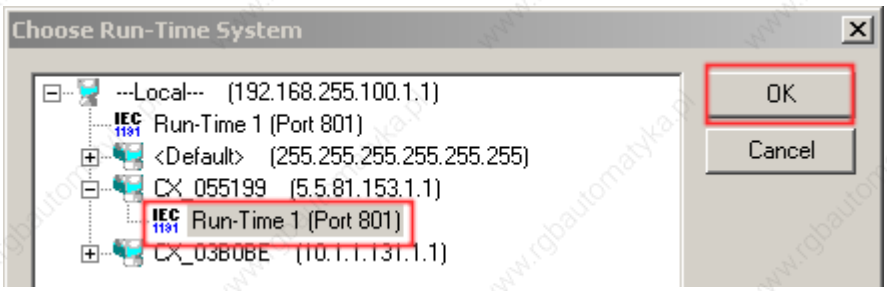
Po dodaniu pliku *.tpy oraz po rozwinięciu zakładki PLC Configuration można zobaczyć zmienne zadeklarowane w programie. Zmienne te należy **zlinkować** z fizycznymi wejściami i wyjściami. W tym celu wystarczy dwukrotnie kliknąć na daną zmienną i z okna ***Attach Variable*** wybrać dostępne urządzenie np. wejście/wyjście cyfrowe.



Na koniec należy ponownie wgrać konfigurację, wybierając opcję **Activate Configuration** .

6.3 Uruchamianie programu

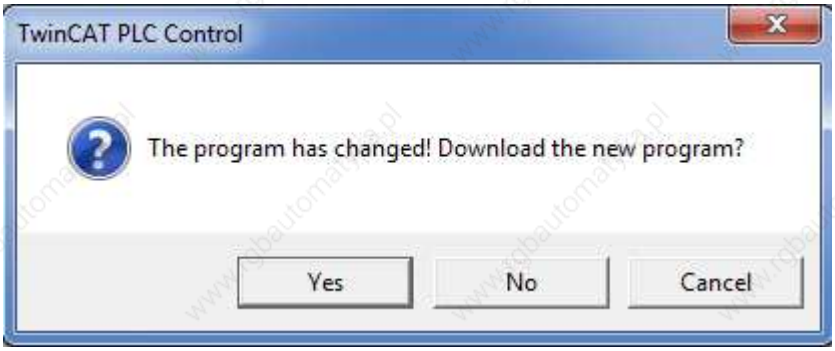
Gdy program jest poprawnie skompilowany, zmienne zlinkowane i wgrana konfiguracja można przystąpić do testowania programu. W pierwszej kolejności należy wybrać sterownik, na którym chcemy uruchomić program tzw. środowisko uruchomieniowe (run-time). W tym celu należy wybrać polecenie **Online → Choose Run-Time System...** Domyślnie wybrany jest Run-Time lokalny. Chcący wgrać program na sterownik należy wybrać Run-Time właściwego sterownika.



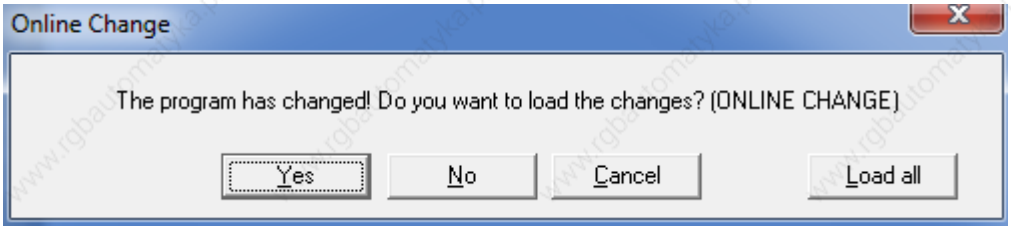
Lista sterowników wyświetlona w oknie **Choose Run-Time System** zawiera historię dotychczasowych połączeń ze sterownikami. Wybieramy właściwy Run-Time i

klikamy na nim dwukrotnie **LPM**. Od tej pory w prawym dolnym rogu okna głównego wyświetlana jest nazwa sterownika.

W dalszej kolejności za pomocą opcji **Online → Login (F11)** należy podjąć próbę zalogowania się na sterownik. W tym wypadku może wyświetlić się jeden z komunikatów:



Gdy nie ma programu na sterowniku lub gdy różni się on całkowicie od obecnego, pojawia się pytanie czy wgrać nowy program. Po potwierdzeniu program zostanie przesłany na sterownik automatycznie, ale nie zostanie on uruchomiony. Aby uruchomić program trzeba wykonać komendę **Online → Run (F5)**.



Podczas aktualizacji programu zostanie wyświetlone okno z propozycją zmiany online. W takim wypadku program znajdujący na sterowniku nie zatrzyma się. Możliwe jest wgranie programu od nowa za pomocą komendy **Load All** – zatrzyma to działanie programu. Zaleca się raz na kilka zmian online wykorzystywać rozkaz Load All.

Wgrany projekt jest przechowywany w pamięci ulotnej RAM. Jeżeli chcemy, aby był on projektem bootowalnym (uruchamiał się automatycznie po starcie sterownika) należy wybrać opcję **Online → Create Bootproject**.

7 Dodatek

7.1 Dodatkowa pomoc

ftp://ftp.beckhoff.com/poland/Pomoc/ - adres serwera ftp na którym umieszczane są pomoce, programy, materiały szkoleniowe oraz funkcje diagnostyczne.

7.2 Akcje (Action)

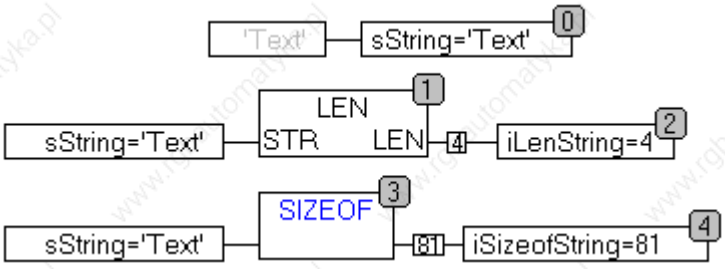
Akcja jest to wyodrębniony fragment kodu programu, który może być wywoływany w dowolnym miejscu. Akcja może być napisana w innym języku niż język danego POU'sa. Stosowanie akcji pozwala uporządkować kod, ułatwia przez to debugowanie programu i pozwala szybciej zrozumieć algorytm osobie, która korzysta z kodu napisanego przez innego programistę. Akcja nie posiada własnych zmiennych, korzysta ze zmiennych danego POU'sa. Nie można utworzyć akcji w akcji.

7.3 Operacje na Stringach

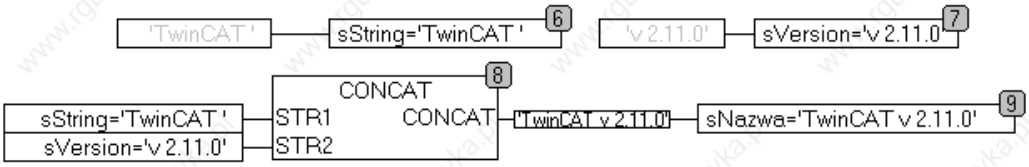
Zmienna typu STRING (łańcuch znaków) może mieć maksymalnie 255 znaków. Deklaracja może zawierać rozmiar zmiennej podany w nawiasach za typem, np. STRING(25), jeżeli rozmiar nie jest podany, to zmienna alokuje 80 bajtów na znaki. Dodatkowo jeden bajt jest zajmowany przez znak końca STRINGa.
W bibliotece standardowej znajdują się funkcje, które pozwalają na wykonywanie operacji na STRINGach.

- **LEN** - zwraca liczbę znaków występujących w stringu. Funkcja oblicza długość aktualnego łańcucha znaków, bez znaku końca STRINGa.
- **SIZEOF** - zwraca ilość pamięci, którą kompilator zarezerwował dla zmiennej typu STRING. Funkcja zawsze zwraca maksymalną liczbę znaków, która można umieścić w STRINGu + 1 (znak końca STRINGa).

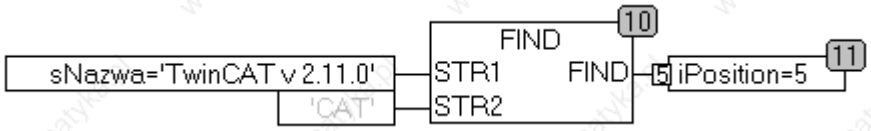
Rezultaty działania funkcji przedstawione są na rysunku poniżej.



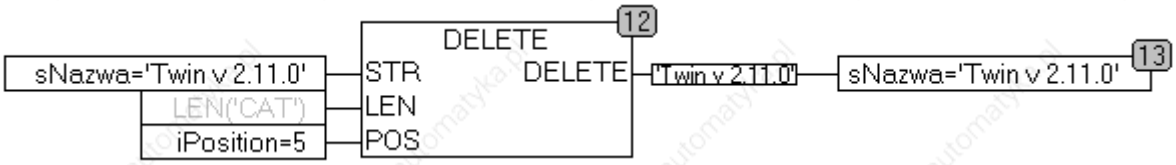
- **CONCAT** – służy do łączenia dwóch łańcuchów znaków



- **FIND** – znajduje jeden ciąg znaków w drugim ciągu znaków, zwraca pozycję pierwszego znaku znalezionego ciągu.



- **DELETE** – funkcja usuwa podaną liczbę znaków zaczynając od określonej pozycji



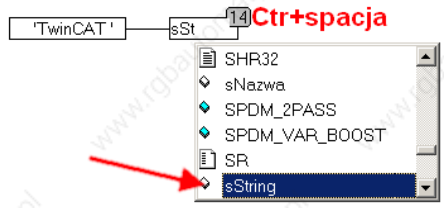
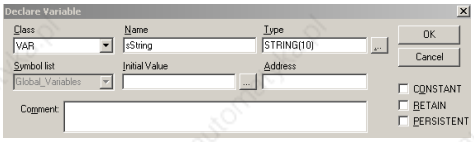
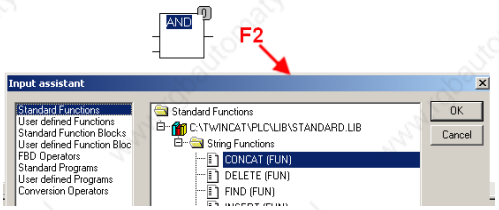
7.4 Znaki specjalne

Wprowadzanie znaków specjalnych (takich, które mają przypisane określone funkcję) odbywa się przez wpisanie znaku $ przed znakiem specjalnym. Poniżej przedstawiono przykłady użycia znaków specjalnych.

znak	opis
$$	znak dolara $
$’	apostrof
$L lub $l	nowy wiersz
$N lub $n	znak końca linii
$P lub $p	znak nowej strony
$R lub $r	znak przerwy w linii
$T lub $t	znak tabulatora

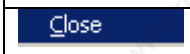
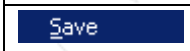
7.5 Skróty klawiszowe

Poniżej znajdują się wybrane skróty, które często wykorzystywane są podczas tworzenia programu:

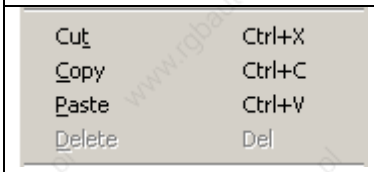
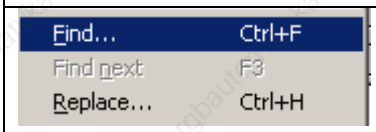
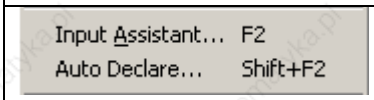
Ctrl + spacja 	Wywołuje menu kontekstowe, wyświetlające nazwy zmiennych, funkcji, bloków funkcyjnych, programu rozpoczynających się od danego ciągu znaków. Skrót szczególnie przydatny przy wywoływaniu zmiennych – zapobiega tzw. literówkom.
Shift + Esc	Otwieranie i zamykanie okna informacyjnego w TwinCAT PLC Control (na dole ekranu).
Shift + F2 	Wyświetlenie okna deklaracji zmiennych. Jeśli jakaś zadeklarowana zmienna została zaznaczona a skrót Shift +F2 został wywołany w oknie Declare Variable pola będą wypełnione – w ten sposób można edytować deklarację zmiennej
F2 	Wybrany przy zaznaczonej nazwie elementu wywołuje okno **Input assistant** wspomagającego wybór odpowiedniej nazwy bloku funkcyjnego, funkcji.
F4	Przenosi do następnego błędu wyświetlonego podczas kompilacji
Shift + F4	Przenosi do wcześniejszego błędu

8 Przegląd najważniejszych opcji programu TwinCAT PLC Control

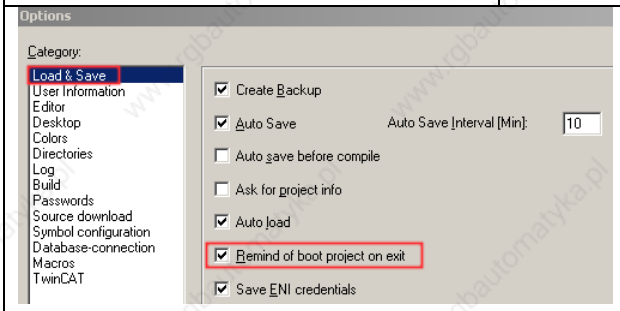
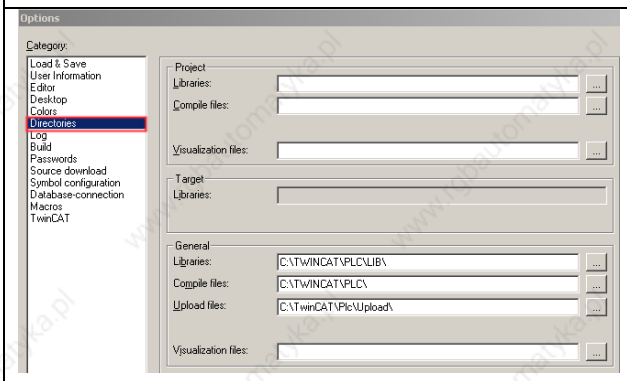
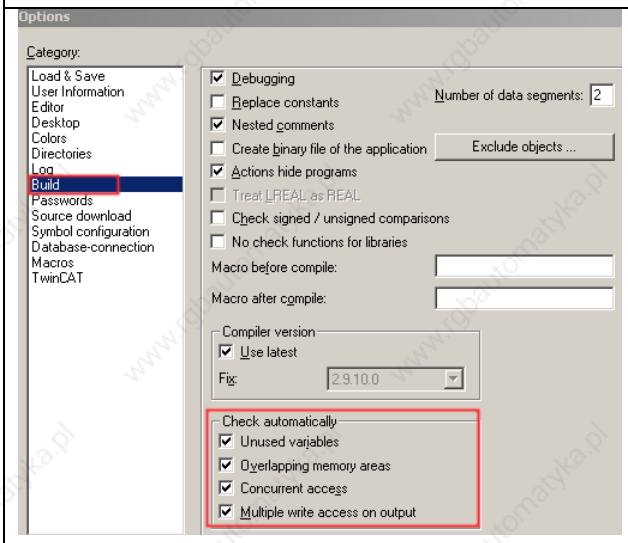
8.1 Menu File

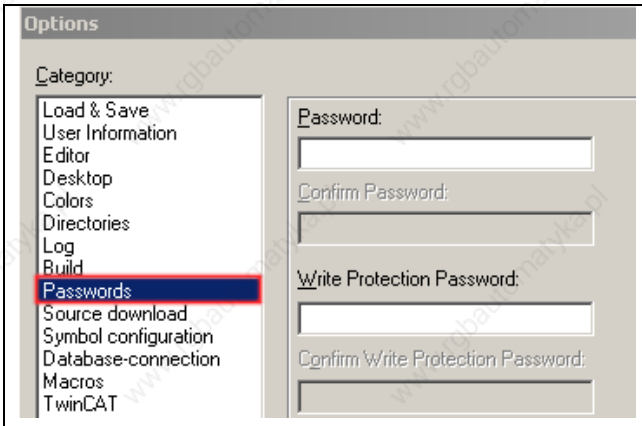
	tworzenie nowego projektu
	(Ctrl+O) otwieranie projektu
	zamykanie okna projektu
	(Ctrl+S) zapisywanie projektu
	zapisywanie projektu pod inną nazwą
	archiwizowanie całego projektu, możliwość wysyłania projektu za pomocą maila
	(Ctrl+P) drukowanie aktywnego okna programu
	Ustawienia drukowania
	(Alt+F4) Zamknięcie programu TwinCAT PLC Control
	Lista ostatnio otwartych projektów

8.2 Menu Edit

	Cofanie ostatniego kroku Ponowne wykonanie cofniętej funkcji
	Wycinanie, kopiowanie, wklejanie, usuwanie zaznaczonego fragmentu programu
	Szukanie w bieżącym oknie pierwszego pasującego ciągu znaków, szukanie kolejnego wystąpienia ciągu znaków, zastępowanie znalezionego ciągu znaków ciągiem podanym
	Wywołanie okna Input Assistant (asystent wyboru nazw i obiektów) Wywołanie okna Auto Declare (automatyczna deklaracja zmiennych.
	Przeniesienie w miejsce kolejnego błędu, wykrytego podczas kompilacji, Cofnięcie się do błędu wcześniejszego

8.3 Menu Project

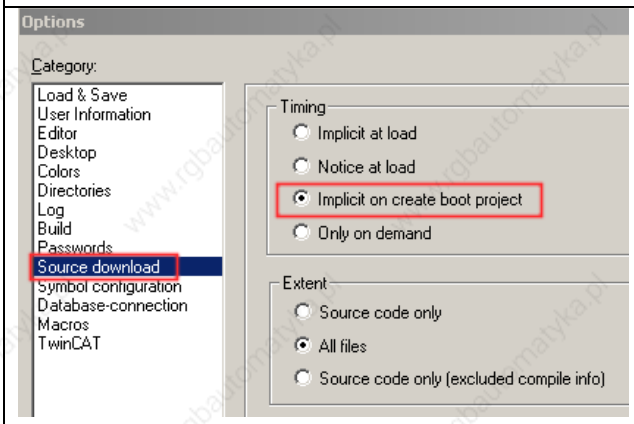
	**Build** – kompiluje tylko modyfikacje wprowadzone do projektu, natomiast **Rebuild all** kompiluje cały program od nowa
	Czyści pamięć i wszystkie pliki wykorzystywane do kompilacji oraz adresy przyznawane dynamicznie
	Opcje środowiska programistycznego
	Opcje dotyczących automatycznego zapisywania projektu oraz automatycznego wczytywania ostatniego edytowanego projektu po uruchomieniu TwinCAT. Zaleca się włączenie opcji Remind of boot Project on exit.
	Ustawienie ścieżek dostępu do katalogów przechowujących biblioteki. Domyślnie wybrane są katalogi instalacyjne TwinCAT'a. Domyślnie do tych katalogów instalowane są nowe biblioteki
	Ustawienia opcji kompilatora. Zaleca się włączenie opcji Check automatically: • Nieużywane zmienne • Nakładające się obszary pamięci • Jednoczesny dostęp • Nadpisywanie wyjść



Możliwość zabezpieczania projektu:

→ brak dostępu do projektu bez podania hasła

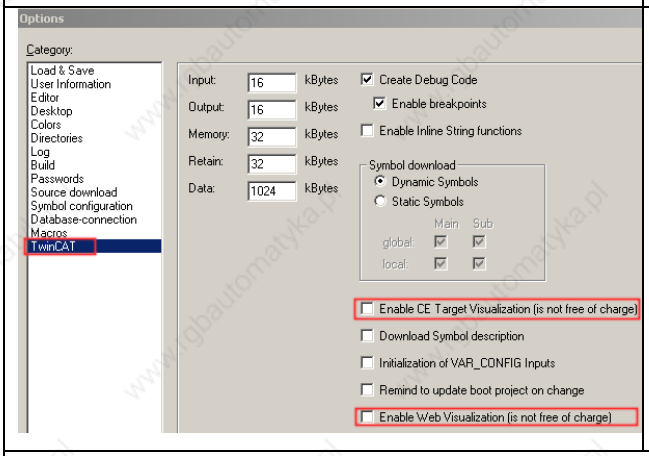
→ dostęp tylko do odczytu, możliwość dokonania modyfikacji po podaniu hasła



Ustawianie opcji wysyłania na sterownik całego projektu, wraz z kodem źródłowym i komentarzami.

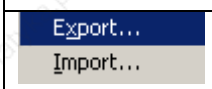
Domyślnie ustawiona jest opcja wgrywania na żądanie, czyli po wybraniu opcji **Online → Sourcecode download.**

Zaleca się ustawienie opcji **Implicite on create boot project** – wysyłania kodu programu podczas tworzenia projektu bootowalnego



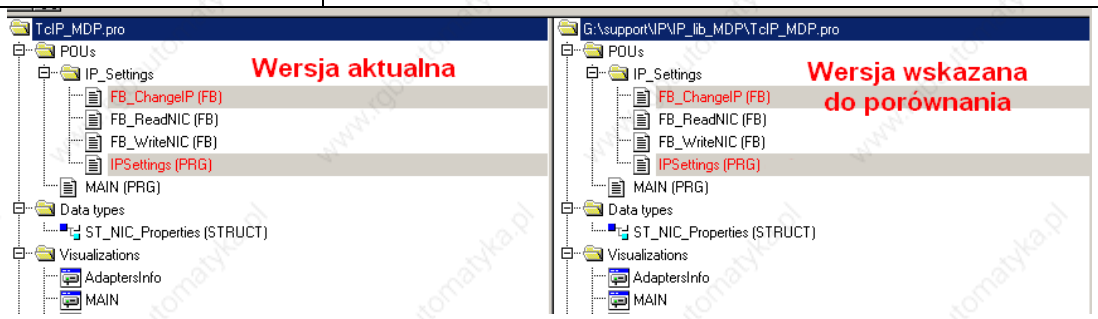
Zaznaczone opcje odpowiadają za wysyłanie wizualizacji do sterownika. Pierwsza opcja wspiera bibliotekę TargetVisu HMI – pozwala na wyświetlanie wizualizacji lokalnie. Druga opcja wspiera bibliotekę TargetVisu Web – pozwala wyświetlać wizualizację w przeglądarce internetowej.

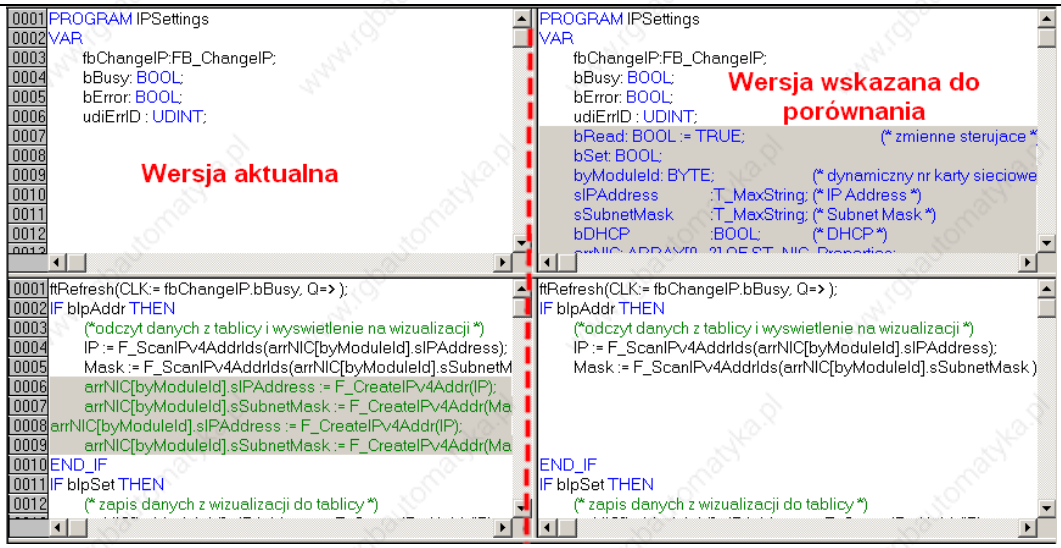
Tworzenie dokumentacji dotyczącej programu. Opcja pozwala na wydrukowanie wybranych elementów projektu tj. kody programów, zmienne globalne, wizualizacje itp.



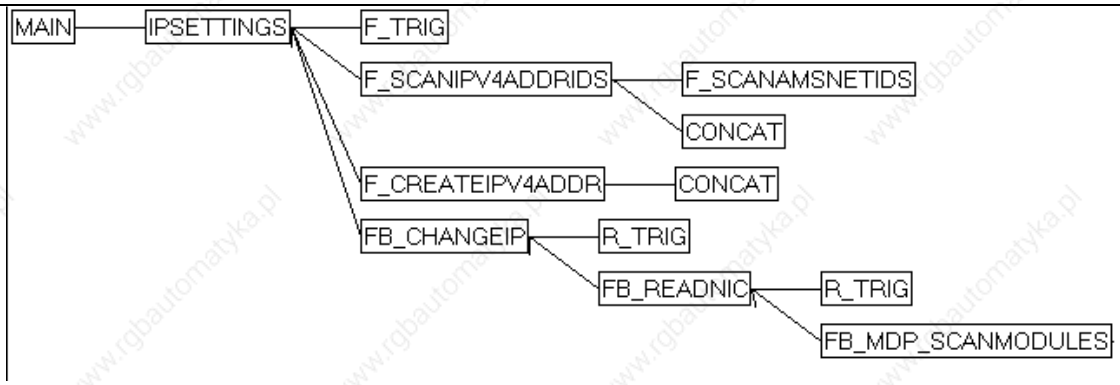
Exportowanie służy do zapisania wybranych części

	projektów do jednego pliku. Importowanie jest operacją odwrotną, pozwala na wczytanie wyexportowanego pliku.
Merge...	Służy do łączenia dwóch projektów. Po wybraniu opcji należy wskazać projekt, który będzie dołączony do aktualnego.
Compare...	Polecenie porównuje dwa projekty, obecny ze wskazanym poprzez podanie ścieżki. Opcja zaznacza kolorami różniące się części projektu. Poniżej pokazano porównanie dwóch projektów

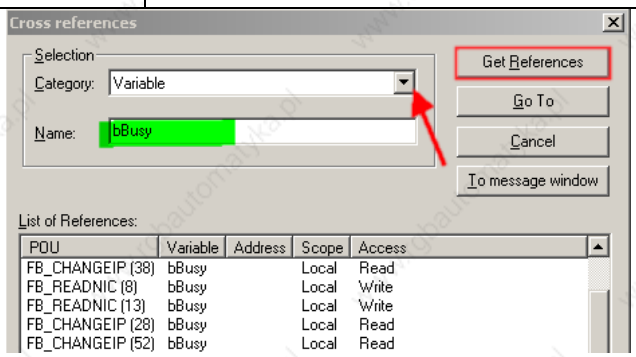


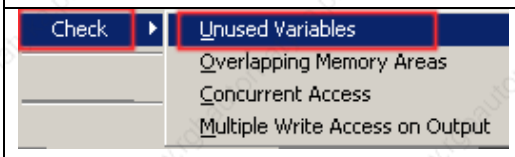
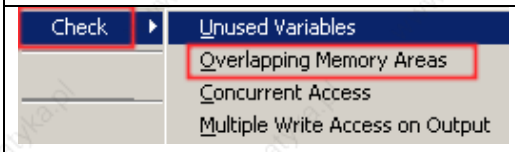
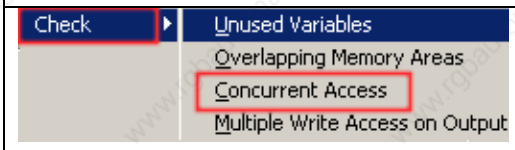
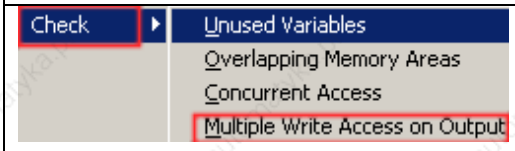


Global Search... Ctrl+Alt+S Global Replace...	Wyszukiwanie ciągów znaków w całym projekcie np. zmiennych,
Show Call Tree	Call Tree przedstawia strukturę programu. Opcja staje się aktywna dopiero po poprawnym skompilowaniu programu i wybraniu programu.



Show Cross Reference...	Pokazuje miejsca wystąpienia oraz prawo dostępu podanej zmiennej, adresu, POUs'a. Cross references pozwala na przeniesienie się do miejsca występowania zmiennej.



	wyszukuje nieużywane zmienne
	wyszukuje nakładające się obszary pamięci
	sprawdza bieżący dostęp do zmiennej
	wykrywa nadpisywanie wyjść
Add Action	dodaje akcję do wybranego POUs'a

8.4 Menu Online

Login	(**F11**) logowanie się do wybranego sterownika
Logout	(**F12**) wylogowanie się ze sterownika
Run	(**F5**) uruchomienie programu na sterowniku
Stop	(**Shift + F8**) zatrzymanie działania programu
Reset	przywracanie programu do ustawień inicjalizacyjnych, zachowanie zmiennych nieulotnych
Reset All	przywracanie programu do ustawień inicjalizacyjnych, zerowanie również zmiennych nieulotnych
Toggle Breakpoint	(**F9**) ustawienie punktu zatrzymania programu, włączenie trybu krokowego wykonywania programu
Breakpoint Dialog	wywołanie okna zarządzania pracą debbugera
Step over	(**F10**) praca krokowa, przejście do następnej wykonywanej instrukcji programu, nie wchodzimy do wnętrza elementów.
Step in	(**F8**) praca krokowa, przejście do następnej wykonywanej instrukcji programu, wchodzimy do wnętrza elementów
Single Cycle	(**Ctrl + F5**) wykonanie pojedynczego cyklu programu
Write Values	(**Ctrl+ F7**) wpisanie wartości do wybranej zmiennej, wartość zmiennej może być zmieniona w kolejnym cyklu
Force Values	(**F7**) forsowanie wartości wybranej zmiennej tzn. wpisanie wartości do zmiennej w każdym cyklu programu
Release Force	(**Shift + F7**) usuwanie wszystkich forsowań zmiennych
Show Call Stack...	wyświetlanie informacji o miejscu zatrzymania programu
Display Flow Control	(**Shift + F11**) podświetlenie linii wykonywanych w danej chwili
Simulation Mode	Tryb symulatora, opcja aktywna dla sterowników serii BC, BX
Sourcecode download	wgranie kodu źródłowego programu do wybranego sterownika
Choose Run-Time System...	wybór sterownika, do którego będzie można się zalogować
Create Bootproject	Tworzenie bootprojectu - po włączeniu zasilania sterownika, zostanie uruchomiony program wgrany na PLC
Delete Bootproject	Usunięcie bootprojectu, po włączeniu sterownika program wgrany do jego pamięci nie wystartuje automatycznie