1. ŚRODOWISKO

CoDeSys (aktualnie obowiązująca nazwa CODESYS) to zintegrowanie środowisko opracowanie przez niemiecką firmę 3S-Smart Software Solutions, wspomagające projektowanie różnych aspektów automatyki przemysłowej. Oprogramowanie ma charakter uniwersalny (nie jest dedykowane dla konkretnych sterowników/producentów) i zapewnia wsparcie dla większości 16 i 32-bitowych procesorów stosowanych w sterownikach automatyki przemysłowej (C166, TriCore, 80x86, ARM/Cortex, Power Architecture, SH, MIPS, BlackFin, itp.). Ta cecha powoduje, że zintegrowany kompilator może wygenerować kod binarny dla różnych układów sterowania na podstawie jednego uniwersalnego projektu.

CoDeSys jest zgodny z wytycznymi normy IEC 61131-3 i pozwala na tworzenie programów we wszystkich językach definiowanych w normie, tzn:

- *funkcjonalnych schematów blokowych FBD,*
- *schematów drabinkowych LD,*
- *tekstu strukturalnego ST,*
- *listy instrukcji IL,*
- *sekwencyjnego schematu funkcjonalnego SFC.*

Dodatkowo możliwe jest również programowanie w języku *Continuous Function Chart* (CFC) nie uwzględnionym w normie IEC.

Pakiet CoDeSys zawiera narzędzie ułatwiające testowanie napisanego programu – tzw. debager (*ang. debugger*), który pozwala na śledzenie przebiegu wykonywania programu ułatwiając lokalizację błędów. Kontrolowany debagerem program może być wykonywany na rzeczywistym sterowniku lub na symulatorze sterownika, gdy sterownik nie jest dostępny.

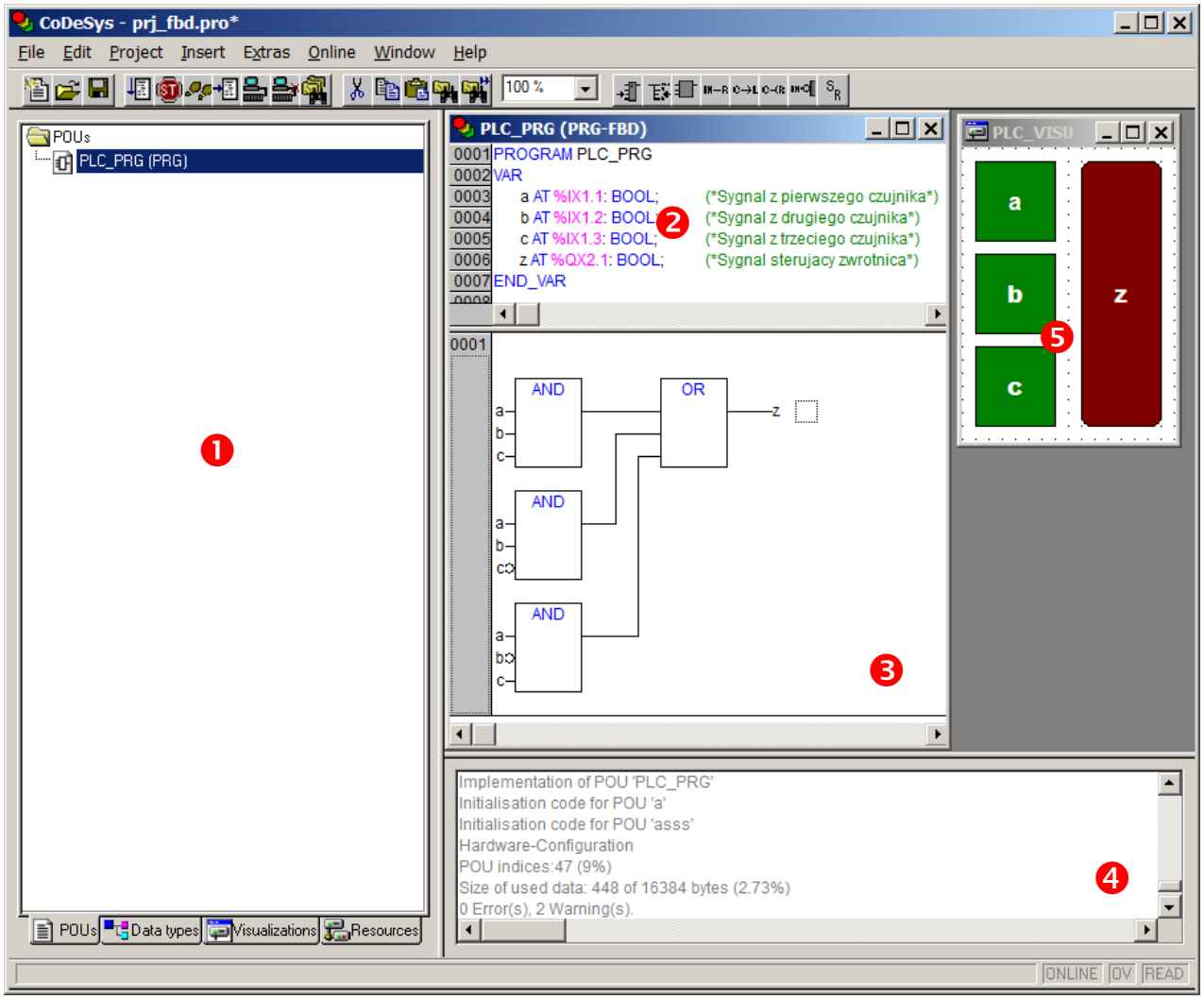
Pierwsza wersja programu została udostępniona w roku 1994. Obecnie rozwijana jest wersja 3.x, jednak nadal powszechnie używana jest starsza 2.3, dostarczana przez wielu producentów razem z ich sterownikami. CodeSys 2.3 pracuje z systemem Windows XP, 7, 8, 10 i ma znacznie mniejsze wymagania sprzętowe, wersja 3.x nie wspiera systemu XP. Poniższe opracowanie dotyczy wersji 2.3.

2. INSTALACJA

CoDeSys jest wolny od opłat licencyjnych, może być pobrany ze strony producenta (odpowiednie adresy zostały zamieszczone w sekcji **Linki** na stronie przedmiotu). Program jest bezpłatny, jednak wymaga rejestracji. W tym celu po przejściu na stronę https://store.codesys.com/engineering.html należy wybrać opcję *Login* (menu w górnej części ekranu) i utworzyć konto jako klient prywatny (*I am a private customer*). Istotne jest podanie poprawnego adresu e-mail, ponieważ serwis prześle na niego kod aktywujący konto (bez aktywacji nie można pobrać programu). Jeżeli kod aktywacyjny nie zostanie przesłany automatycznie należy zalogować się na swoje konto i wybrać opcję jego powtórnego wysłania (link dostępny obok informacji o nieaktywnym koncie).

Po zalogowaniu i aktywacji konta należy przejść do zakładki *Engineering* i pobrać plik instalacyjny. Program dostarczany jest jako pojedynczy plik „exe”, po kliknięciu rozpoczyna się standardowy proces instalacji. *Uwaga*: niektóre programy antywirusowe blokują działania programu instalacyjnego. Jeżeli w czasie instalacji pojawia się błąd (brak dostępu do folderów systemowych) należy wyłączyć antywirusa i powtórnie uruchomić instalację. Do prawidłowego działania programu w trybie symulacji wystarczające są elementy zaznaczone w instalatorze jako wymagane (ciemnoszare pola opcji) pozostałe składniki mogą być wyłączone (przyspiesza to instalację i zmniejsza obciążenie systemu w czasie działania programu). Wszystkie zainstalowane składniki będą umieszczony w menu Start grupie „3S SoFTWARE” (Windows XP, 7) , właściwa aplikacja w podfolderze „CODESYS V2.3”.

3. GŁÓWNE OKNO PROGRAMU



Rys. 3.1. Przykładowy projekt w środowisku CoDeSys

Rys. 3.1. przedstawia okno CoDeSys z przykładowym projektem prostego programu dla sterownika PLC. Panel boczny (nr 1), zależnie od wybranej zakładki zawiera wykaz jednostek programowych, zdefiniowanych struktur danych, wizualizacji lub zasobów sterownika. Okna nr 2 i 3 reprezentują program sterownika. Pierwsze z nich zawiera nagłówek i wykaz zdefiniowanych zmiennych, drugie właściwy kod programu (w tym przykładzie w języku FBD). Okno nr 4 służy do wyświetlania

komunikatów kompilatora, który tłumaczy kod programu na kod binarny, który będzie przesłany do sterownika. Okno nr 5 zawiera przykładową wizualizację, która na etapie symulacji ułatwia testowanie programu, a po uruchomieniu go na sterowniku pozwoli operatorowi kontrolować przebieg rzeczywistego procesu.

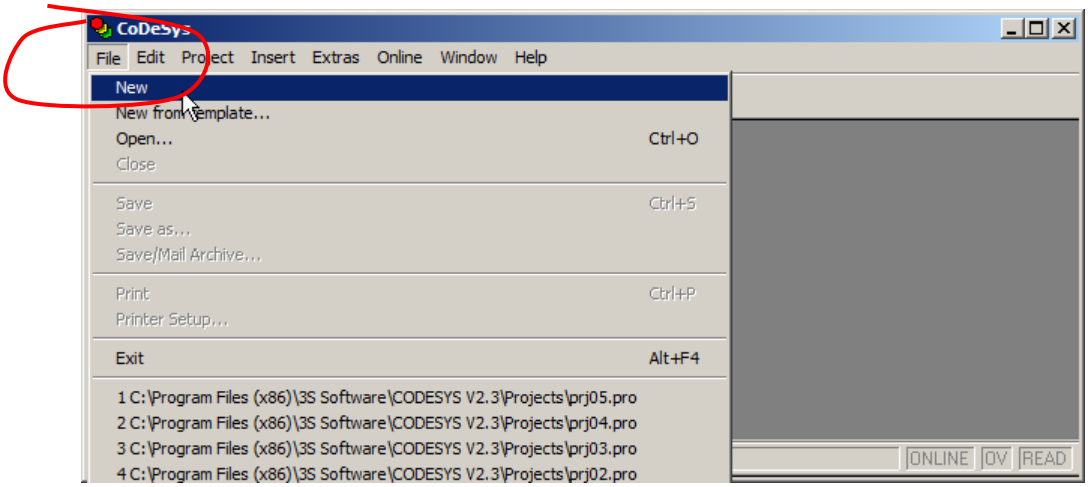
W następnych punktach zostaną przedstawione kolejne kroki prowadzące do stworzenia kompletnego programu:

- tworzenie nowego projektu,
- deklaracja zmiennych,
- edycja programu w językach FBD i LD,
- kompilacja, uruchomienie i testowanie programu,
- podstawy tworzenia wizualizacji.

4. TWORZENIE NOWEGO PROJEKTU

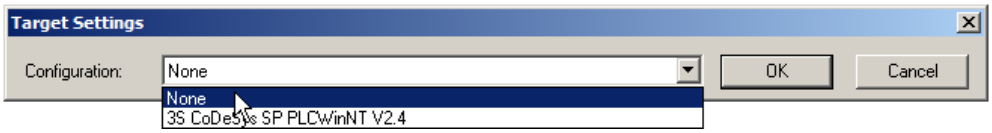
Przygotowanie dowolnego programu dla sterownika PLC wymaga stworzenia projektu w CoDeSys zawierającego główną jednostkę organizacyjną programu (POU – *Program Organization Unit*), która stanowi program główny realizowany przez sterownik. Jest to minimalna struktura projektu, która umożliwi uruchomienie programu w sterowniku PLC. W dalszym etapie prac taki projekt może być uzupełniony o kolejne elementy, takie jak podprogramy, bloki funkcyjne, lub wizualizacje (elementy interfejsu użytkownika, które pozwolą na interakcję operatora z wykonywanym programem).

W celu utworzenia nowego projektu w CoDeSys należy w głównym oknie programu (rys.3.1.) wybrać opcję `File->New`:



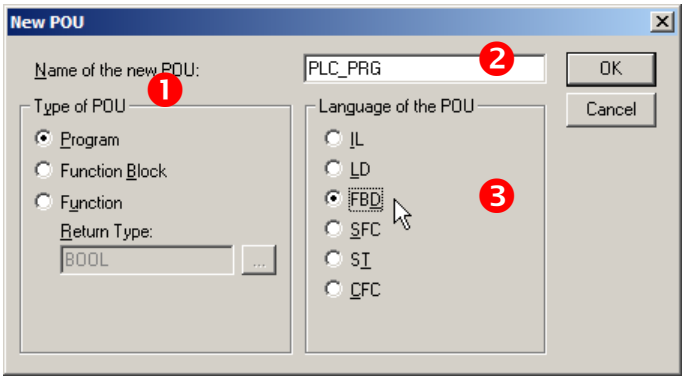
lub nacisnąć pierwszy przycisk paska narzędziowego:

W wyniku tej operacji pojawia się okno dialogowe `Target Settings`, które pozwala na wybór typu sterownika PLC, dla którego będzie tworzony program:



Różne sterowniki mogą mieć różną konfigurację, pracują pod kontrolą różnych procesorów, mają odmienne metody komunikacji z komputerem, więc CoDeSys musi znać parametry docelowego sterownika, żeby wygenerować właściwy kod programu. Konfiguracje sterowników są dostarczane przez ich producentów i mogą być dodawane do programu jako dodatkowe moduły. Domyślnie CoDeSys zawiera jeden sterownik (`3S CoDeSys SP PLCWinNT V2.4`), który jest programowym sterownikiem pracującym jako niezależna aplikacja dla Windows umożliwiająca przeprowadzenie symulacji całego procesu połączenia z urządzeniem, przesłania kodu i wykonania programu bez konieczności dostępu do rzeczywistego sterownika PLC. Uruchomienie i przetestowanie kodu programu nie wymaga wykorzystania tego elementu (CoDeSys pozwala na przeprowadzenie symulacji programu w swoim środowisku) więc ten sposób pracy nie będzie omówiony w tym opracowaniu. Jeżeli projekt jest tworzony wyłącznie dla celów symulacji należy w oknie `Target Settings` wybrać opcję `None`.

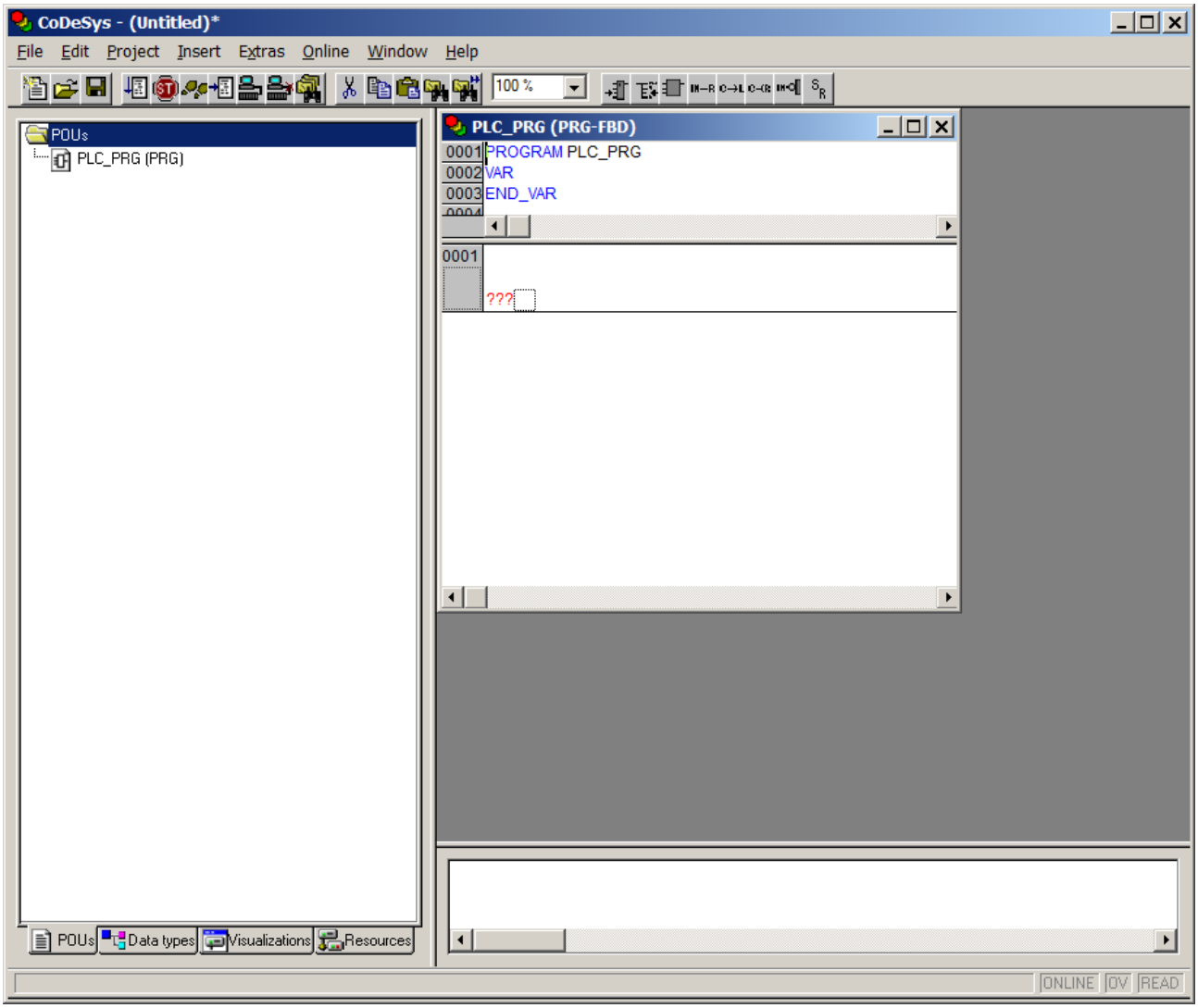
Po zaakceptowaniu typu sterownika pojawia się okno nowej jednostki organizacyjnej programu (`New POU`). W tym oknie należy wybrać: (1) typ, (2) nazwę i (3) język jednostki organizacyjnej:



W przypadku tworzenia programu głównego:

- w sekcji `Type of POU` należy pozostawić wybraną domyślnie opcję `Program`,
- pozostawić ustawioną domyślnie nazwę programu `PLC_PRG` (program główny musi nosić taką nazwę),
- w sekcji `Language of the POU` wybrać język programu głównego (na rysunku wybrany jest język FBD, jednak każdy program może być napisany w dowolnym języku).

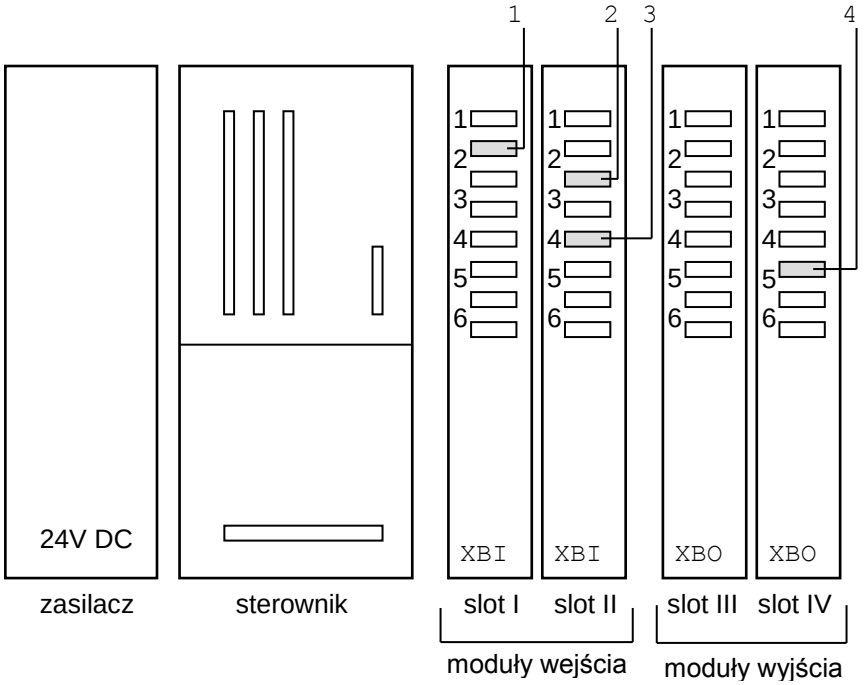
Po zaakceptowaniu ustawień w oknie New POU zostanie wygenerowany nowy projekt z pojedynczą jednostką organizacyjną (programem głównym):



Po utworzeniu nowy projekt powinien być zapisany przez wybranie opcji `File->Save`.

5. ZMIENNE

Każdy program wykorzystuje pewien zestaw zmiennych. Zmienne są używane przez jednostki organizacyjne programu do przechowywania i przetwarzania danych. Każda zmienna odpowiada pewnej informacji umieszczonej w strukturze wejść/wyjść (odpowiada sygnałowi który pojawia się na wejściu/wyjściu) lub w pamięci sterownika. W programach używa się najczęściej zmiennych prostych (inaczej skalarnych, jednoelementowych), czyli pojedynczych wartości należących do typu elementarnego (liczba, znacznik daty i czasu, itp.). Zmienne proste mogą być przedstawione symbolicznie (poprzez nazwę) lub reprezentowane bezpośrednio poprzez określenie ich położenia (odwołanie do wejścia/wyjścia lub komórki pamięci). Niezależnie od typu i konfiguracji sterownika PLC oraz języka jednostki organizacyjnej użycie zmiennych w kodzie programu wygląda tak samo. Poniższe przykłady będą korzystały ze sterownika pokazanego na rys. 5.1. Sterownik ten ma dwa moduły wejścia umieszczone w slotach (gniazdach) nr 1 i 2 oraz dwa moduły wyjścia umieszczone w slotach nr 3 i 4. Wszystkie moduły są typu binarnego z ośmioma kanałami, co oznacza, że na wejściu każdego modułu może wystąpić maksymalnie osiem sygnałów typu 1 lub 0.



Rys 5.1. Sterownik PLC z dwoma modułami wejścia i dwoma modułami wyjścia

5.1. Bezpośrednie adresowanie zmiennych

W celu bezpośredniego odwołania do zmiennej należy użyć notacji:

`%<położenie><rozmiar><adres>`

`<położenie>` określa obszar pamięci w którym jest umieszczona zmienna, dostępne wartości:

- I – wejście (zmienna związana z kanałem modułu wejściowego),
- Q – wyjście (zmienna związana z kanałem modułu wyjściowego),
- M – obszar danych (zmienna pomocnicza umieszczona w pamięci sterownika).

`<rozmiar>` określa liczbę bitów, które zmienna zajmuje (typ danych, które reprezentuje zmienna), dostępe wartości: X – bit (wartość 1 lub 0), B – bajt (8 bitów), W – słowo (16 bitów), D – podwójne słowo (32 bity), L – poczwórne słowo (64 bity).

`<adres>` to ciąg liczb całkowitych rozdzielonych kropkami, które określają położenie zmiennej w danym obszarze pamięci sterownika, kolejne pola są interpretowane jako hierarchiczny sposób adresacji, pierwsze pole od lewej oznacza poziom najwyższy (adres zależy od urządzenia i jego konfiguracji).

Przykłady

1. Zmienna reprezentująca sygnał oznaczony numerem 1 na rys. 5.1. (drugi kanał binarnej karty wejściowej umieszczonej w pierwszym slocie):

```
%IX1.2
```

X – zmienna związana z kanałem modułu wejściowego, X – pojedynczy bit, 1.2 – drugi kanał z modułu umieszczonego w pierwszym slocie (pierwszy składnik oznacza najwyższy poziom adresowania – w tym przypadku numer slotu).

2. Zmienne reprezentujące sygnały oznaczone numerem 2 i 3 na rys.5.1:

```
%IX2.3, %IX2.5
```

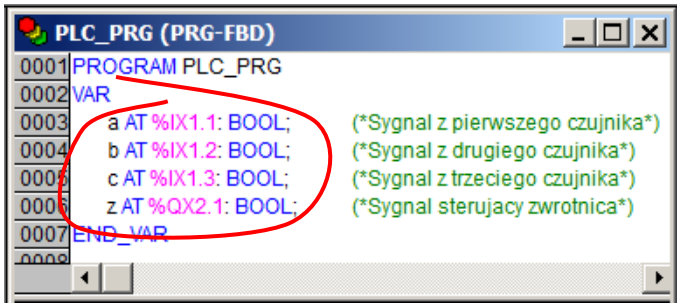
3. Zmienna reprezentująca sygnał oznaczony numerem 4 na rys.5.1. (moduł wyjściowy):

```
%QX4.6
```

5.2. Deklaracja zmiennych symbolicznych

Użycie zmiennych adresowanych bezpośrednio prowadzi do zmniejszenia czytelności programu, utrudnia jego modyfikacje i może powodować trudno wykrywalne błędy. Dowolna zmiana konfiguracji sterownika (np. przełożenie modułu z jednego slotu do innego lub przełączenie sygnału z czujnika do innego kanału) wymaga zmiany odwołań (adresowania) w całym kodzie programu. W przypadku dużych programów taka operacja jest czasochłonna, a pozostawienie pojedynczej nieprawidłowej wartości może powodować błędy, które ujawnią się tylko w szczególnych okolicznościach i będą trudne do wykrycia. Z tych powodów zalecane jest posługiwanie się zmiennymi symbolicznymi, które reprezentują każdą zmienną w postaci unikalnej nazwy (może być to słowo opisujące znaczenie zmiennej) i są powiązane z daną komórką pamięci w miejscu ich deklaracji. Dzięki takiemu rozwiązaniu zmiana konfiguracji sterownika nie wymaga modyfikacji całego kodu, a jedynie poprawienia deklaracji zmiennej (modyfikacja jednego wiersza programu).

Niezależnie od wybranego języka POU deklaracja zmiennych w CoDeSys wygląda tak samo. Wszystkie zmienne lokalne danej POU umieszczane są w górnej części okna edytora, poniżej nagłówka programu tak jak pokasuje to rys. 5.2.



Rys.5.2. Deklaracja zmiennych symbolicznych w oknie edytora POU

Schemat deklaracji zmiennej symbolicznej:

```
VAR
     <nazwa> AT<lokalizacja>: <typ> := <wartość>;
END_VAR
```

`<nazwa>` to unikalny ciąg znaków identyfikujący zmienną (dozwolone litery alfabetu angielskiego, cyfry, podkreślenie, nie może zaczynać się od cyfry),

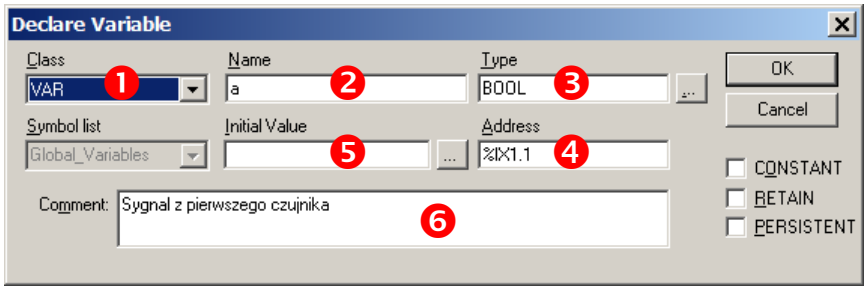
`<lokalizacja>` określa położenie zmiennej zgodnie z zasadami adresowania bezpośredniego (opisane w punkcie 5.1.),

`<typ>` określa rodzaj i zakres wartości, które może przyjmować zmienna: BOOL – zmienne boolowskie (binarne), BYTE, WORD, DWORD, LWORD – słowo 8, 16, 32 i 64-bitowe, INT – liczby całkowite, REAL – liczby rzeczywiste,

`<wartość>` określa początkową wartość zmiennej (opcjonalna).

Po właściwej deklaracji zmiennej można umieścić komentarz jako tekst pomiędzy znakami (*...*). Ten element nie ma znaczenia dla wykonania programu (jest ignorowany podczas kompilacji kodu), jednak konsekwentnie używany zwiększa czytelność kodu i ułatwia jego modyfikacje w przeszłości.

Deklaracja zmiennych może być wprowadzona ręcznie, można jednak wykorzystać do tego celu okno dialogowe `Declare Variable` pokazane na rys. 5.3.



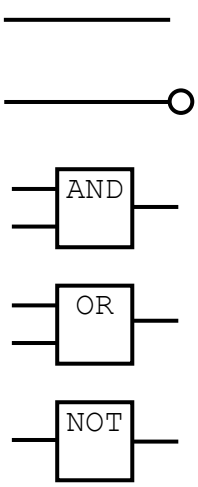
Rys. 5.3. Okno deklaracji zmiennej

Okno jest dostępne po wybraniu opcji `Edit->Auto Declare` lub naciśnięciu kombinacji `Shift+F2`. Deklarując zmienną należy określić: (1) klasę zmiennej (w przypadku zmiennych lokalnych POU należy pozostawić domyślną wartość VAR), (2) nazwę zmiennej, (3) typ zmiennej (lista typów jest dostępna po naciśnięciu przycisku z trzema kropkami) oraz (4) adres zmiennej. Dodatkowo jeżeli jest to potrzebne możliwe jest określenie (5) początkowej wartości zmiennej oraz dodanie opisu w formie komentarza (6). Po zaakceptowaniu okna wprowadzone wartości zostaną zamienione na odpowiedni zapis w oknie deklaracji zmiennych (rys. 5.2).

6. PROGRAMOWANIE W JĘZYKU FBD

Język FBD (*Function Block Diagram*) należy do grupy języków graficznych, jest wzorowany na opisie układów cyfrowych przy pomocy schematów logicznych. Program w języku FBD jest tworzony jako zbiór bloków funkcyjnych połączonych w obwody. Wykonanie programu polega na przepływie sygnału przez kolejne bloki analogicznie do przepływu prądu (gazu, cieczy) w układzie zbudowanym z elektronicznych (pneumatycznych, hydraulicznych) elementów logicznych.

Podstawowe elementy obwodów w języku FBD



połączenie

połączenie z negacją

koniunkcja (iloczyn logiczny) sygnałów wejściowych

alternatywa (suma logiczna sygnałów wejściowych

negacja sygnału wejściowego

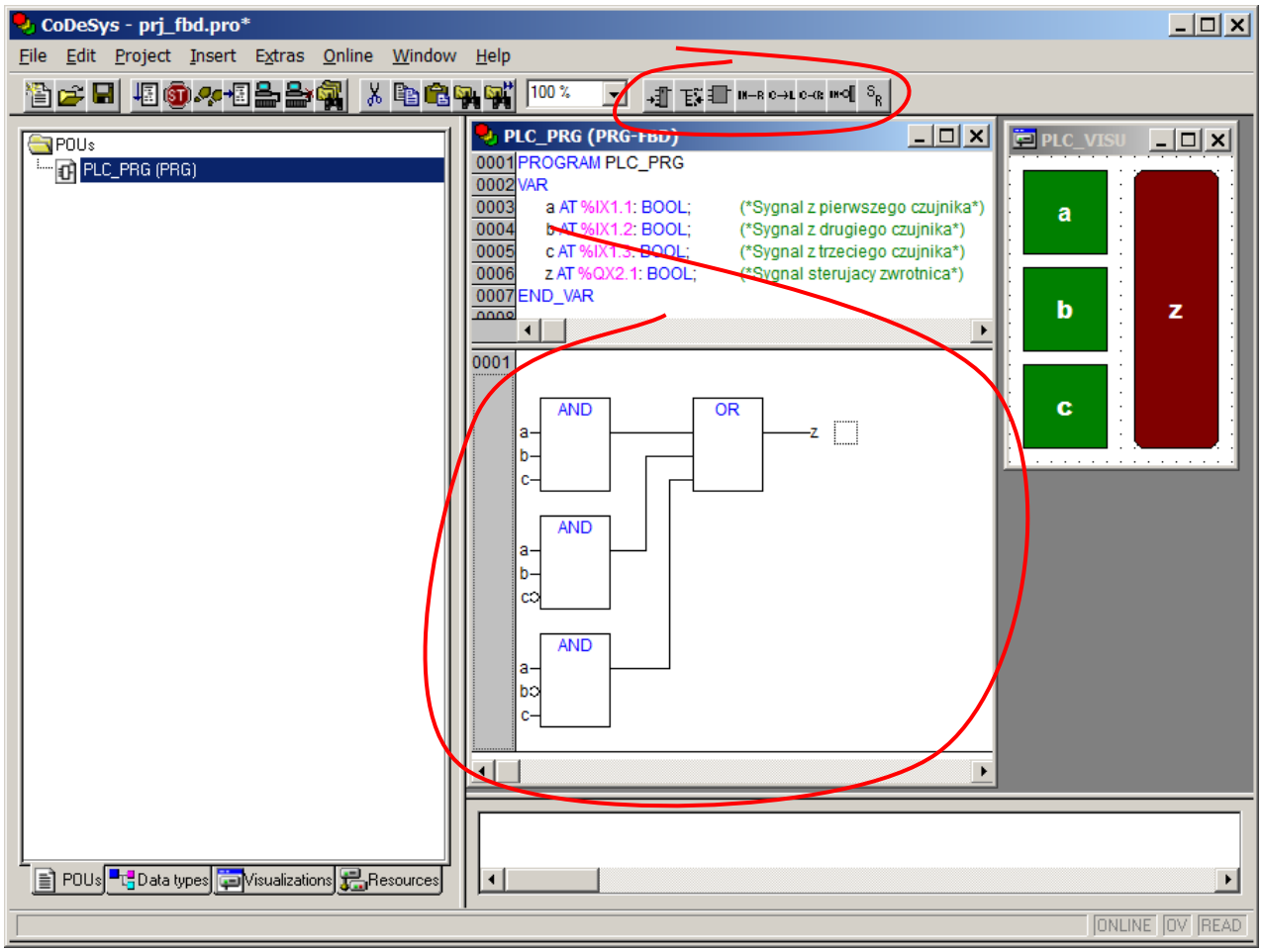
6.1. Edytor FBD

Do tworzenia/edycji POU w języku FBD służy specjalizowany edytor graficzny pokazany na rys. 6.2. W górnej części okna widoczny jest nagłówek programu i deklaracja zmiennych (punkt 5.), dodatkowo w głównym oknie CoDeSys wyświetlany jest pasek narzędziowy umożliwiający edycję kodu (rys. 6.1, 6.2).

Rys. 6.1. Pasek narzędziowy FBD

Podstawowe funkcje paska FBD

1. Nowe wejście bloku, opcja w menu: `Insert->Input`, skrót: `Ctrl+U`
2. Nowy blok, opcja w menu: `Insert->Box`, skrót: `Ctrl+B`
3. Połączenie ze zmienną, opcja w menu: `Insert->Assign`, skrót: `Ctrl+A`
4. Negacja sygnału (dodaje/usuwa negację do połączenia), menu kontekstowe: `Negate`, skrót: `Ctrl+N`

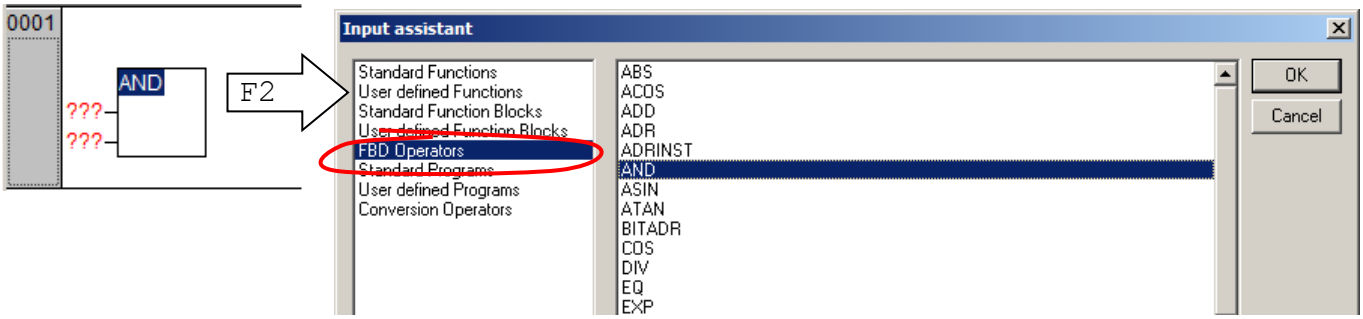


Rys. 6.2. Edytor programu w języku FBD

6.2. Podstawowe operacje w edytorze FBD

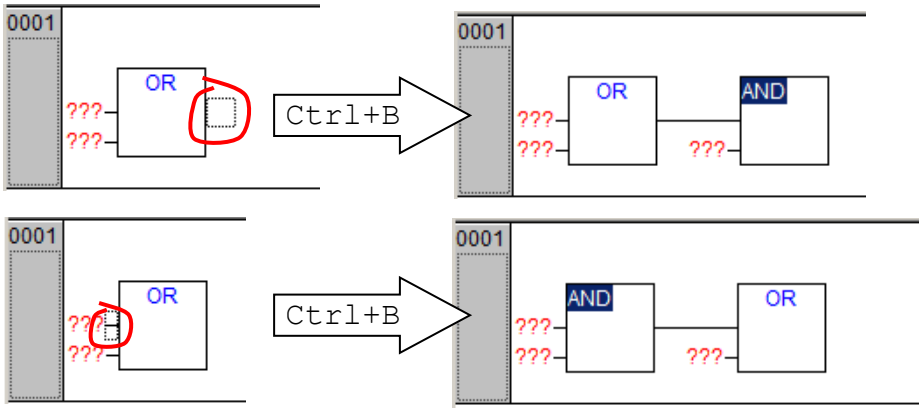
6.2.1. Dodane nowego bloku

Przycisk nr 1 na pasku FBD (rys. 6.2.), opcja menu `Insert->Box`, lub sekwencja `Ctrl+B`. Nowy blok jest funkcją AND z dwoma wejściami. W celu zmiany typu bloku należy kliknąć na aktualnej nazwie i wprowadzić nową nazwę z klawiatury lub skorzystać z asystenta (opcja menu `Edit->Input Assistant`, klawissz `F2`). Wykaz dostępnych bloków znajduje się w grupie `FDB Operators`. *Uwaga*: okno asystenta otwiera tylko gdy wybrana jest aktualna nazwa bloku.



6.2.2. Dołączenie bloku do bloku istniejącego

Należy zaznaczyć wejście lub wyjście, do którego ma być dołączony blok, a następnie wybrać funkcję wstawiania nowego bloku (punkt 6.2.1.). *Uwaga*: należy zaznaczyć połączenie, nie zmienną związaną z danym wejściem/wyjściem.

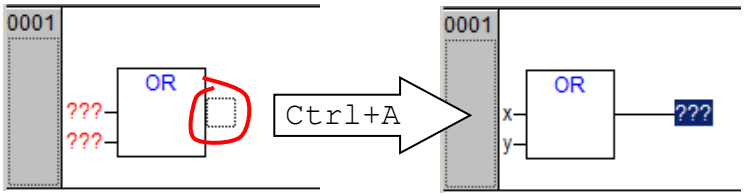


6.2.3. Przypisanie zmiennej do wejścia bloku

Należy zaznaczyć czerwony symbol „`???`” i wprowadzić nazwę zmiennej. *Uwaga*: jeżeli wprowadzona zmienna nie została wcześniej zadeklarowana CoDeSys wyświetli okno dialogowe `Declare Variable`, które umożliwi wprowadzenie wszystkich danych koniecznych do deklaracji zmiennej. Po zaakceptowaniu okna przyciskiem OK zmienna zostanie dodana do bloku deklaracji (punkt 5.2.).

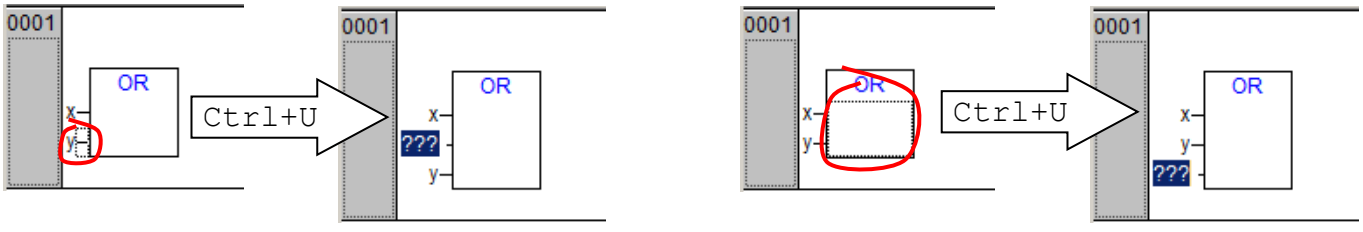
6.2.4. Przypisanie zmiennej do wyjścia bloku

Należy kliknąć w puste miejsce bezpośrednio za blokiem (pojawia się mały kwadrat zaznaczenia), następnie kliknąć przycisk nr 3 na pasku FDB, wybrać opcję `Insert->Assign`, lub sekwencję `Ctrl+A`. W miejscu zaznaczenia dodawana jest nowa linia połączenia zakończona symbolem pustej zmiennej (`???`). Przypisanie nazwy zmiennej odbywa się tak jak w przypadku wejścia (punkt 6.2.3)



6.2.5. Dodanie wejścia do bloku

Należy zaznaczyć wejście przed którym ma być dodane nowe, następnie kliknąć przycisk nr 1 na pasku FDB, wybrać opcję `Insert->Input`, lub sekwencję `Ctrl+U`, lub zaznaczyć blok (klikając poniżej nazwy) i wykonać jedną z operacji opisanych powyżej. W drugim przypadku wejście zostanie dodane jako ostatnie.

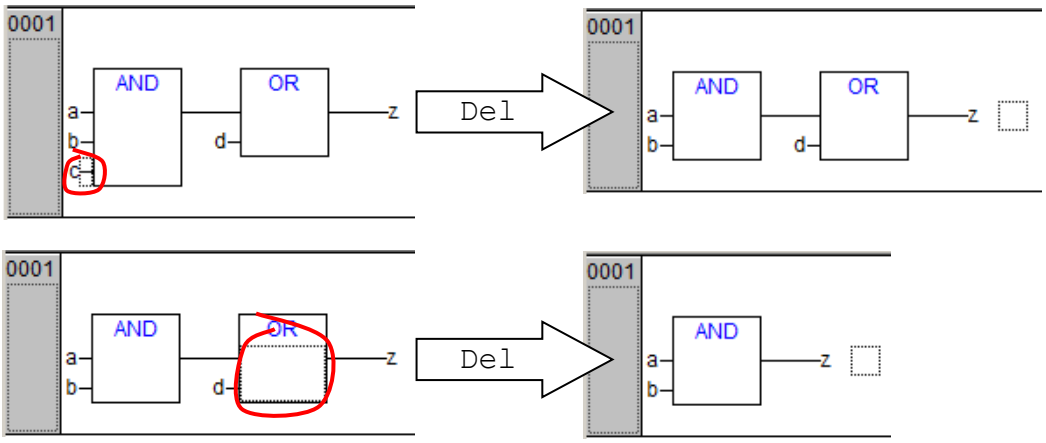


6.2.5. Wstawianie nowego obwodu

Należy wybrać obwód przed/za którym nowy obwód ma być wstawiony i wybrać opcję odpowiednio `Insert->Network before`, `Insert->Network after`, lub sekwencję `Ctrl+T` (wstawienie obwodu za bieżącym).

6.2.6. Usuwanie elementu

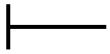
Należy zaznaczyć zbędny element (obwód, blok, wejście, połączenie), następnie wybrać opcję `Edit->Delete` lub klawisz `Del`. Uwaga: nie można usunąć elementów niezbędnych dla działania bloku (np. blok AND musi mieć minimum dwa wejścia), usunięcie wejścia z którym związany jest blok spowoduje usunięcie tego bloku.

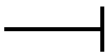


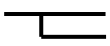
7. PROGRAMOWANIE W JĘZYKU LD

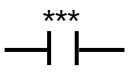
Język LD (*ang. Ladder Diagram – schemat drabinkowy*) należy do grupy języków graficznych, jest odpowiednikiem układów cyfrowych realizowanych w technologii stykowo-przekaźnikowej. Program w języku LD jest tworzony jako zbiór elementów połączonych w obwody. Wykonanie programu polega na przepływie sygnału przez kolejne obwody analogicznie do przepływu prądu w obwodzie zbudowanym z przekaźników.

Elementy obwodów w języku LD

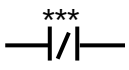

lewa szyna prądowa – ogranicza obwód z lewej strony


prawa szyna prądowa – ogranicza obwód z prawej strony (opcjonalna)

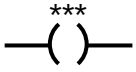

połączenie poziome i pionowe



styk zwierny (normalnie otwarty) – sygnał przepływa przez element gdy skojarzona z nim zmienna ma wartość 1



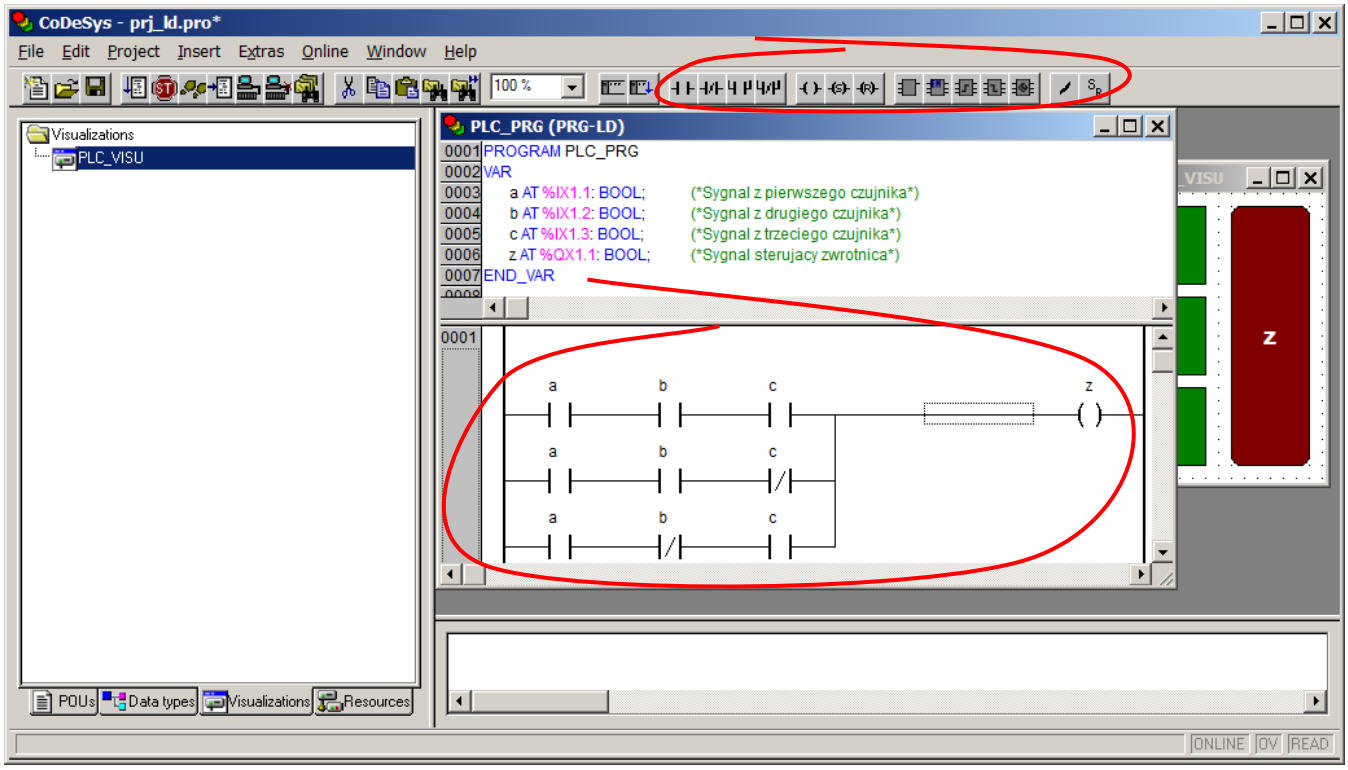
styk rozwierny (normalnie zamknięty) – sygnał przepływa przez element gdy skojarzona z nim zmienna ma wartość 0



cewka – stan połączenia z lewej strony cewki (0 lub 1) jest przenoszony na prawą stroną i zapamiętywany w skojarzonej zmiennej

7.1.. Edytor LD

Do tworzenia/edycji POU w języku FBD służy specjalizowany edytor graficzny pokazany na rys. 7.1.



Rys. 7.1. Edytor programu w języku LD

W górnej części okna widoczny jest nagłówek programu i deklaracja zmiennych (punkt 5.), dodatkowo w głównym oknie CoDeSys wyświetlany jest pasek narzędziowy umożliwiający edycję kodu (rys. 7.2).

Rys. 7.2. Pasek narzędziowy LD

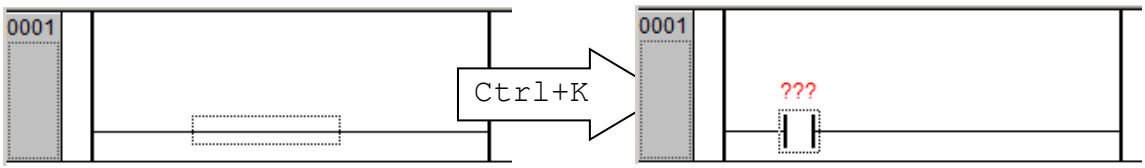
Podstawowe funkcje paska LD

1. Nowy obwód przed aktualnym, opcja w menu: `Insert->Network(before)`
2. Nowy obwód za aktualnym, opcja w menu: `Insert->Network(after)`, skrót: `Ctrl+T`
3. Nowy styk zwierny, opcja w menu: `Insert->Contact`, skrót: `Ctrl+K`
4. Nowy styk rozwierny, opcja w menu: `Insert->Contact(negated)`, skrót: `Ctrl+G`
5. Równoległy styk zwierny, opcja w menu: `Insert->Paralel Contact`, skrót: `Ctrl+R`
6. Równoległy styk rozwierny, opcja w menu: `Insert->Paralel Contact(negated)`, skrót: `Ctrl+R`
7. Nowa cewka, opcja w menu: `Insert->Coil`, skrót: `Ctrl+L`
8. Negacja styku (zamienia styk zwierny na rozwierny i odwrotnie), menu kontekstowe `Negate`, skrót: `Ctrl+N`

7.2. Podstawowe operacje w edytorze LD

7.2.1. Wstawienie styku

Przycisk nr 3/4 na pasku LD, opcja `Insert->Contact/Contact (negated)` lub odpowiedni skrót (zależnie od typu dodawanego styku).

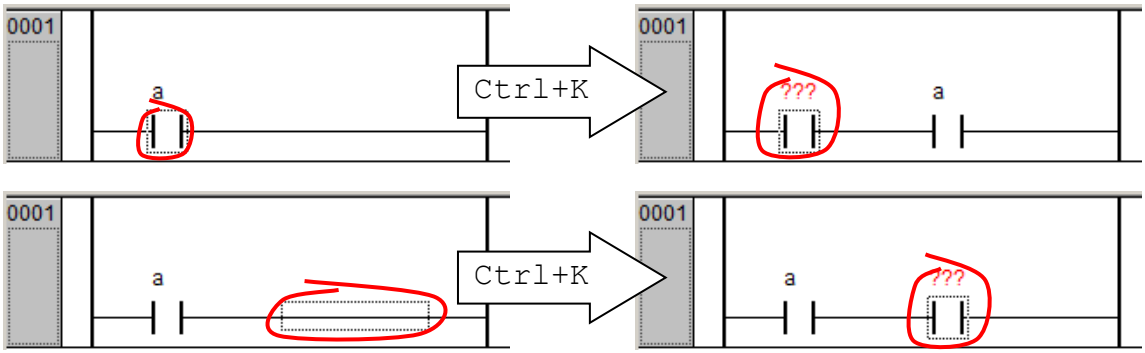


7.2.2. Przypisanie zmiennej do styku

Należy zaznaczyć czerwony symbol „`???`" i wprowadzić nazwę zmiennej. *Uwaga*: jeżeli wprowadzona zmienna nie została wcześniej zadeklarowana CoDeSys wyświetli okno dialogowe `Declare Variable`, które umożliwi wprowadzenie wszystkich danych koniecznych do deklaracji zmiennej. Po zaakceptowaniu okna przyciskiem OK zmienna zostanie dodana do bloku deklaracji (punkt 5.2.).

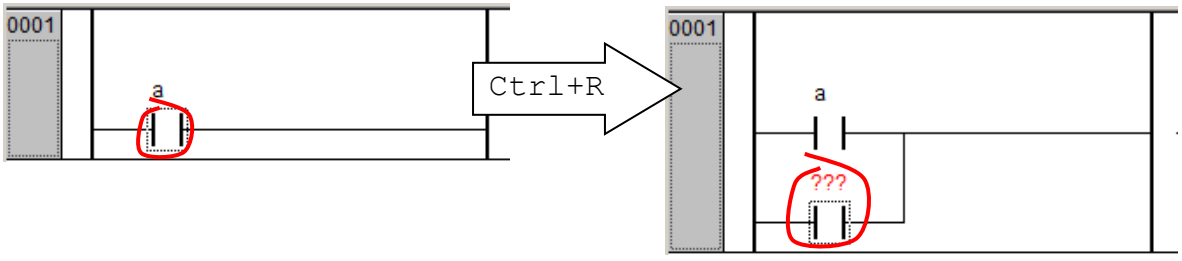
7.2.3. Wstawianie styku połączonego szeregowo

W celu dodania styku przed istniejącym należy ustawić zaznaczenie na styku przed którym nowy ma być dodany. W celu dodania styku na końcu obwodu należy ustawić zaznaczenie na pustym fragmencie obwodu za ostatnim istniejącym stykiem. Następnie wybrać odpowiedni przycisk na pasku narzędziowym (3/4), odpowiadającą mu opcję z menu lub skrót.



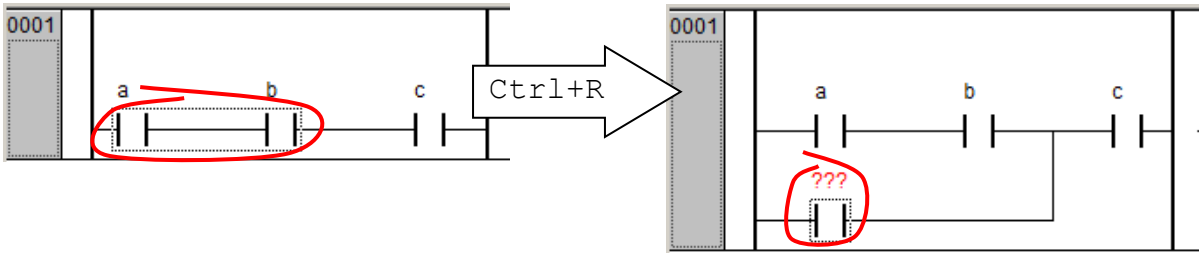
7.2.4. Wstawianie styku połączonego równolegle

Należy ustawić zaznaczenie na styku, z którym nowy styk ma być połączony równolegle, następnie wybrać odpowiedni przycisk na pasku narzędziowym (5/6), odpowiadającą mu opcję z menu lub skrót.



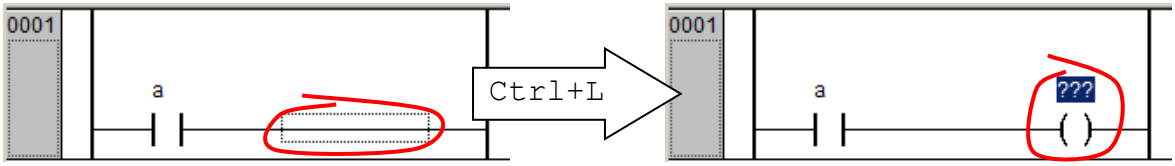
7.2.5. Tworzenie równoległej gałęzi obwodu

Należy zaznaczyć wszystkie styki, do których dodawana gałąź ma być równoległa, następnie wybrać odpowiedni przycisk na pasku narzędziowym (5/6), odpowiadającą mu opcję z menu lub skrót. Uwaga: kilka styków można zaznaczyć klikając na nich lewym przyciskiem myszy z wciśniętym klawiszem `Shift`.



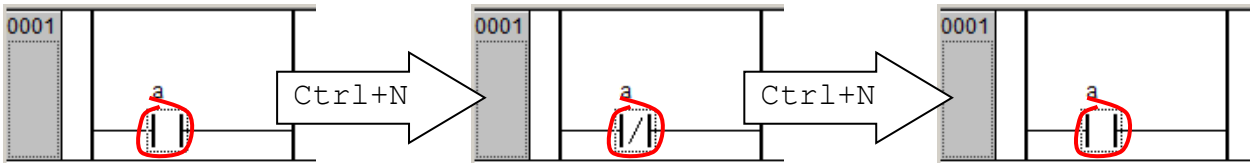
7.2.6. Wstawienie cewki

Należy ustawić zaznaczenie na pustym fragmencie obwodu za ostatnim istniejącym stykiem. Następnie wybrać przycisk nr 7, opcję `Insert->Coil`, lub skrót: `Ctrl+L`.



7.2.7. Zmiana typu styku

Należy zaznaczyć styk, następnie wybrać przycisk nr 8 lub sekwencję `Ctrl+N`.



7.2.7. Wstawianie nowego obwodu

Należy wybrać obwód przed/za którym nowy obwód ma być wstawiony, następnie odpowiednio przycisk nr 1, opcję `Insert->Network before` (wstawienie obwodu przed bieżącym), lub przycisk nr 2, opcję `Insert->Network after`, sekwencję `Ctrl+T` (wstawienie obwodu za bieżącym).

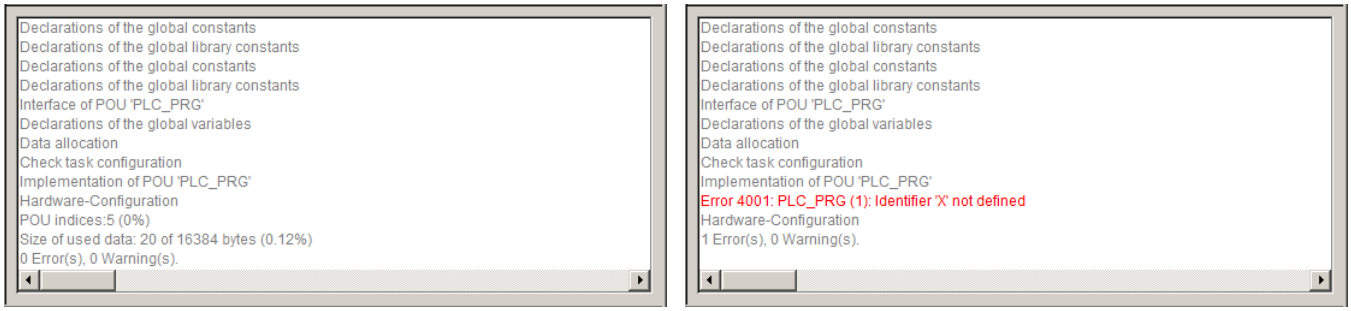
7.2.8. Usuwanie elementu

Należy zaznaczyć zbędny element (obwód, styk, cewkę), następnie wybrać opcję `Edit->Delete` lub klawisz `Del`.

8. URUCHOMIENIE PROGRAMU

8.1. Kompilacja

Niezależnie od języka, w którym został stworzony projekt przed jego załadowaniem do sterownika program musi zostać przetłumaczony na kod binarny. Podobna operacja musi być przeprowadzona przed wykonaniem symulacji programu w środowisku CoDeSys. Za wygenerowanie kodów binarnych odpowiada specjalne narzędzie nazywanie kompilatorem. Kompilator jest integralnym elementem środowiska CoDeSys i może być uruchomiony przy pomocy opcji `Project->Build` lub klawisza `F11`. Kompilacja jest poprzedzona sprawdzeniem błędów w programie. Kod binarny nie będzie wygenerowany jeżeli projekt zawiera błędy. Informacje o wynikach kompilacji wyświetlane są w oknie komunikatów (nr 4 na rys. 3.1.). Jeżeli zostaną znalezione błędy pojawia się czerwony komunikat z opisem. W przypadku prostych programów najczęstsze błędy to brak deklaracji zmiennej lub jej nieprawidłowe adresowanie. Na rys. 8.1. zostały pokazane przykłady okna komunikatów po kompilacji zakończonej prawidłowo (lewy rysunek – ostatni wiersz „0 Error(s), 0 Warning(s)), oraz kompilacji przerwanej w wyniku błędu (prawy rysunek – brak deklaracji zmiennej „X")



Rys. 8.1. Okno komunikatów

8.2. Załadowanie i uruchomienie programu

Skompilowany kod programu przed uruchomieniem musi zostać załadowany do sterownika. Operacja polega na ustaleniu połączenia za pomocą odpowiedniego protokołu komunikacyjnego (np. EtherNet, EtherCAT, itp.) i przesłaniu kodu binarnego do pamięci sterownika. Dopóki program CoDeSys utrzymuje otwarte połączenie można sterować programem i obserwować wyniki jego pracy na ekranie. Opisane operacje muszą być wykonane również w przypadku symulacji pracy pomimo tego, że CoDeSys nie będzie wykonywał połączenia z fizycznym urządzeniem.

Wszystkie operacje związane z załadowaniem i uruchomieniem programu dostępne są w menu `Online` oraz na pasku narzędziowym (rys. 8.2.).

Rys. 8.2. Pasek narzędziowy Uruchamianie

8.2.1. Uruchomienie programu

Należy wybrać opcję `Online->Login`, przycisk nr 3 na pasku narzędziowym, lub sekwencję `Alt+F8` (CoDeSys nawiąże połączenie ze sterownikiem i załadowuje kod binarny programu), następnie wybrać opcję `Online->Run`, przycisk nr 1 lub klawisz `F5` (program zostanie uruchomiony). Jeżeli kod nie był skompilowany wybór opcji Login automatycznie uruchomi proces kompilacji.

8.2.2. Przerwanie pracy programu

W celu przerwania pracy programu należy wybrać opcję `Online->Stop`, przycisk nr 2, lub sekwencję `Shift+F8`. Program zostanie zatrzymany (sterownik przestaje go wykonywać), ale połączenie ze nie jest przerywane. Zatrzymany program można wznowić wybierając opcję `Run`.

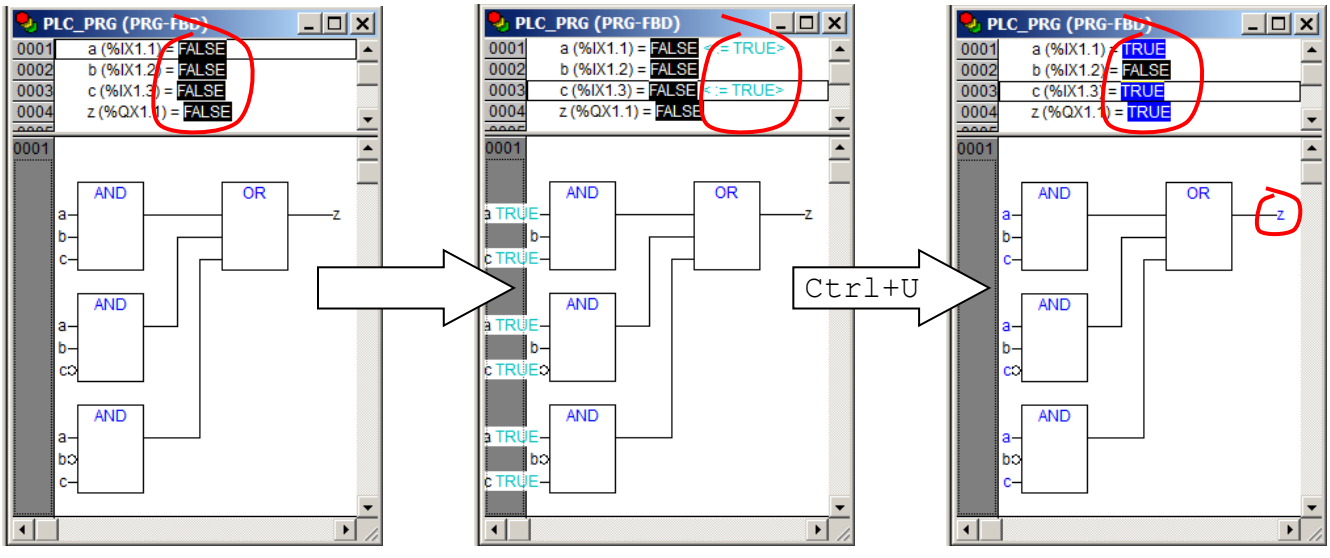
W celu przerwania programu i powrotu do projektu należy zakończyć połączenie ze sterownikiem wybierając opcję `Online->Logout`, przycisk nr 4, lub sekwencję `Ctrl+F8`.

8.2.3. Symulacja pracy programu

W przypadku symulacji programu bez połączenia ze sterownikiem należy zaznaczyć opcję `Online->Simulation Mode`. Proces uruchamiania i zatrzymywania programu przebiega tak jak zostało to opisane w punktach 8.2.1. i 8.2.2. Tryb symulacji jest wybrany domyślnie jeżeli podczas tworzenia projektu jako `Target` wybrano `none` (patrz punk 4.).

8.3. Testowanie programu

Załadowany i **uruchomiony** program można przetestować ustawiając wartości zmiennych wejściowych i przekazując je do sterownika. CoDeSys odczyta wartości wyjść i pokaże je na ekranie. W celu zmiany wartości zmiennej należy dwukrotnie kliknąć na zmiennej w oknie deklaracji lub w kodzie programu. Przekazanie wartości zmiennych następuje po wybraniu opcji `Online->Write Values` lub naciśnięciu sekwencji `Ctrl+F7`. Poniższe rysunki pokazują przykładowy testu programu napisanego w języku FBD. W przypadku języka LD wszystkie operacje wykonuje się tak samo.



Uwaga: Program musi być załadowany i uruchomiony, po załadowaniu jest możliwe ustawianie wartości zmiennych, jednak nie można wykonać kodu (sekwencja Ctrl+F7 nie działa).

9. WIZUALIZACJA

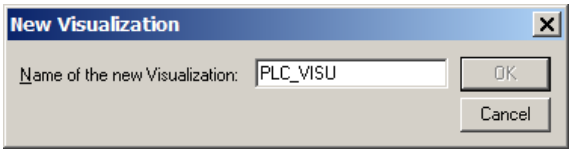
Wizualizacja jest graficznym interfejsem, który pozwala operatorowi na interakcję ze sterownikiem PLC. Wizualizacja może być wyświetlona na ekranie komputera lub w specjalizowanym panelu połączonym ze sterownikiem (tzw. panel HMI – Human Machine Interface). Za pomocą elementów wizualizacji operator może zarówno obserwować pracę sterowanego procesu jak i wpływać na jego przebieg.

CoDeSys posiada zintegrowany edytor umożliwiający tworzenie tzw. masek (inaczej okien lub formularzy) wizualizacji powiązanych bezpośrednio ze zmiennymi sterownika. Maski utworzone w systemie CoDeSys można bez żadnych modyfikacji wykorzystać w czterech wariantach przebiegu (również równolegle):

1. Bezpośrednio w systemie programowania – maska jest wyświetlana w systemie CoDeSys, programista otrzymuje obraz rzeczywistej wizualizacji taki jak inni użytkownicy (np. operator panelu HMI).
2. Wizualizacja dla Windows – maska jest wyświetlana na komputerze bez kompletnego interfejsu programistycznego (wymagane zainstalowanie aplikacji CoDeSys HMI).
3. Wizualizacja sieciowa – pliki wizualizacji (plik XML + aplet Java) są przechowywane w pamięci sterownika i wyświetlane w oknie przeglądarki internetowej.
4. Wizualizacja elementu docelowego – wizualizacja jest wyświetlana na specjalizowanych panelach zintegrowanych lub połączonych ze sterownikiem PLC (np. panele HMI).

9.1. Tworzenie nowej wizualizacji

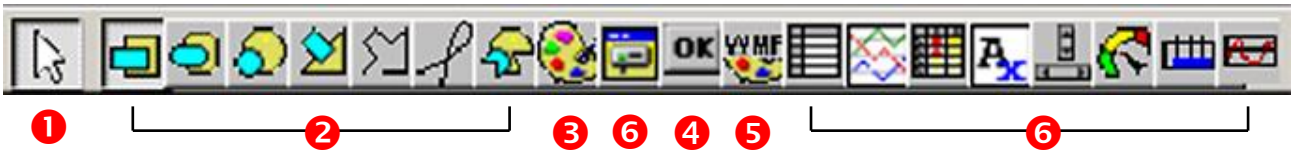
W celu utworzenia nowej wizualizacji w CoDeSys należy przejść na zakładkę `Visualizations` w bocznym panelu CoDeSys (nr. 1 na rys 3.1.) i wybrać opcję `Project->Object->Add` lub z menu kontekstowego panelu `Add Object`. W wyniku tej operacji zostanie wyświetlone okno dialogowe `New Visualization` (rys. 9.1.), w którym należy podać nazwę tworzonej wizualizacji. Nazwa może być dowolnym identyfikatorem, jednak wizualizacja startowa powinna mieć nazwę `PLC_VISU`.



Rys. 9.1. Okno nowej wizualizacji

9.2. Elementy wizualizacji

Wszystkie elementy z których można zbudować wizualizację dostępne są w menu `Insert` lub na pasku narzędziowym przedstawionym na rys. 9.2.



Rys. 9.2. Dostępne elementy wizualizacji

W tym opracowaniu zostaną omówione tylko podstawowe elementy oznaczone na rysunku numerami 2 do 6. Opis elementów oznaczonych wspólnie numerem 6 można znaleźć w dokumentacji programu.

Najprostsze wizualizacje można utworzyć wykorzystując podstawowe elementy graficzne. Na rysunku 9.2 zostały one oznaczone numerem 2. Są to kolejno: prostokąt (`Rectangle`), Zaokrąglony prostokąt (`Round Rectangle`), Elipsa (`Ellipse`), Wielokąt (`Poygon`), Łamana (`Polyline`), Krzywa (`Curve`), Wycinek koła (`Pie`). Każdy z tych elementów może reagować na działania operatora i ma podobne własności (np. kolor, grubość linii, tekst, itp), które mogą być wykorzystane do zasygnalizowania stanu procesu.

W przypadku sterowania przebiegiem procesu (np. ręczna zmiana sygnałów wejściowych) naturalnym elementem wizualizacji jest przycisk (element nr 4). Jego podstawowym zastosowaniem jest zarejestrowanie polecenia operatora. Przycisk może występować w dwóch stanach (wciśnięty-zwolniony) może więc służyć do włączania/wyłączania wybranej funkcji.

Wizualizacja może być uzupełniona o elementy graficzne. Służą do tego elementy nr 3 i 6. Pozwalają od wstawić odpowiednio grafikę bitmapową (format bmp, tif, jpg) oraz wektorową (format wmf). Grafika może poprawić czytelność wizualizacji, dodatkowo nie musi być elementem statycznym – wykorzystując odpowiednie własności tych elementów można zmieniać wyświetlany obraz zależnie od stanu procesu. Te zagadnienie nie będą jednak przedstawiane w tym opracowaniu.

W kolejnych punktach zostaną omówione wybrane własności podstawowych elementów graficznych (2) oraz przycisku (5) oraz ich wykorzystanie do budowy wizualizacji.

9.3. Wstawianie i edycja elementów wizualizacji

W celu **wstawienia nowego elementu** wizualizacji należy wskazać go w pasku narzędziowym lub wybrać odpowiednią opcję w menu `Insert` i zaznaczyć prostokątny obszar na masce wizualizacji. W przypadku wybranych elementów przed wstawieniem pojawia się dodatkowe okno dialogowe – np. z pytaniem o nazwę pliku dla elementów nr 3 i 5.

Edytor wizualizacji umożliwia edycję wstawionych elementów w sposób zbliżony do innych narzędzi tego typu. Podstawowe operacje:

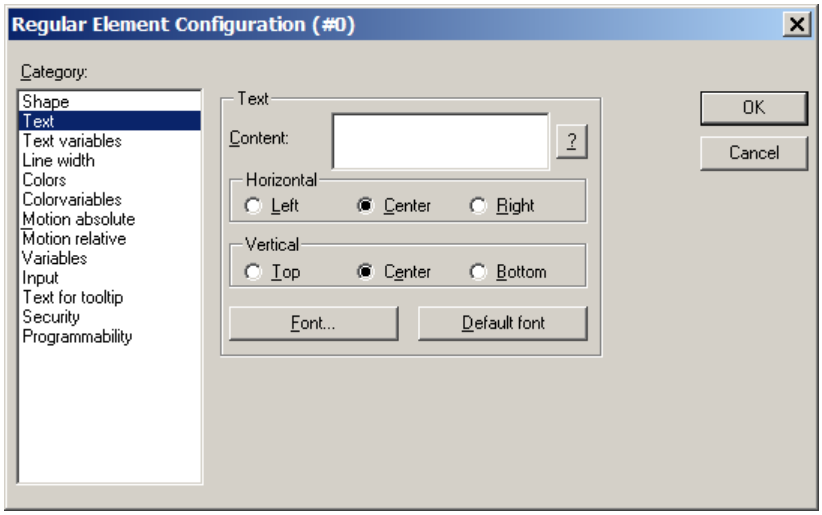
1. Zaznaczenie pojedynczego elementu: selektor obiektów (przycisk nr 1 na pasku narzędziowym) i kliknięcie na elemencie;
2. Zaznaczenie kilku elementów: `Shift+kliknięcie` na kolejnych elementach, lub zaznaczenie prostokątnego obszaru przy wciśniętym lewym przycisku myszy;
3. Przesuwanie: wciśnięcie lewego przycisku myszy gdy kursor jest wewnątrz elementu i przeciąganie;
4. Zmiana rozmiarów: chwyt za jeden z punktów wokół zaznaczonego elementu i przeciąganie;
5. Kopiowanie zaznaczonego elementu: opcja `Edit->Copy`, `Edit->Paste`;
6. Usuwanie: opcja `Edit->Del`, lub klawisz `Del`;
7. Przenoszenie do przodu/tyłu (istotne dla elementów, które przysłaniają się wzajemnie): `Extras->Bring to front`, `Extras->Send to back`;
8. Wyrównanie: opcja `Extras->Align (Left, Right, Top, …)`;
9. Grupowanie/rozgrupowanie: opcja `Extras->Group`, `Extras->Ungroup`;
10. Cofanie zmian: opcja `Edit->Undelete`.

Uwaga: zgrupowane elementy mają wspólne właściwości, każda operacja (przenoszenie zmiana rozmiarów) dotyczy całej grupy.

9.4. Własności elementów wizualizacji

Każdy element wizualizacji ma zestaw własności, które można modyfikować na etapie projektu przy pomocy okna dialogowego `Configuration` (rysunki w punktach 9.4.1-9.4.4.). W celu wyświetlenia własności danego elementu należy zaznaczyć go przy pomocy selektora elementów (przycisk nr 1 na rys. 9.2.) i wybrać opcję `Extras->Configure`. W oknie konfiguracji własności zostały podzielone na kategorie wyświetlane w liście z lewej strony. Po wskazaniu konkretnej kategorii wyświetlany jest odpowiedni zestaw własności, które można zmodyfikować. Poniżej zostaną omówione najważniejsze własności, dodatkowo w punkcie 9.5 będą przedstawione przykłady ich wykorzystania.

9.4.1. Text



Opis elementu (zawartość pola `Content`) sposób jego wyrównania i rodzaj czcionki. Poza tekstem statycznym przy pomocy własności teks można wyświetlać wartość zmiennej projektowej lub systemowej. Zmienna projektowa, której wartość będzie wyświetlana jest określana w kategorii Variables->Text display (patrz punkt 9.4.3.).

Sposób wyświetlenia zmiennej definiuje znacznik formatu:

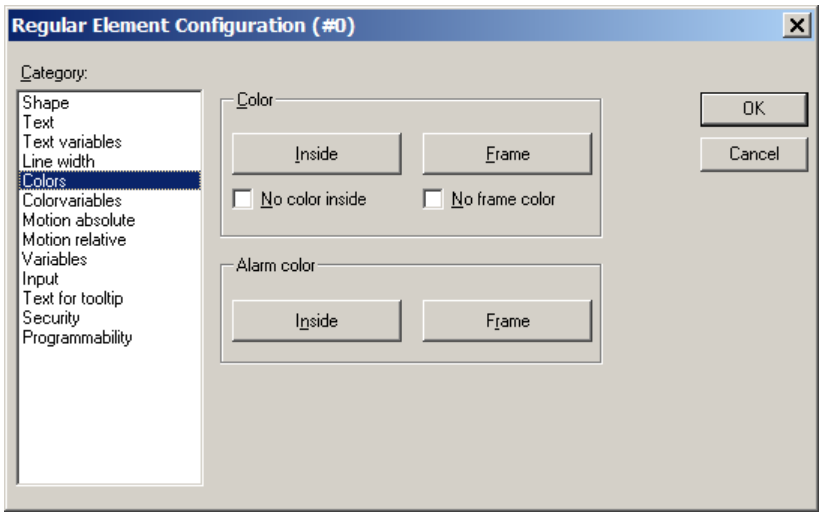
- %s – wartość wyświetlana jako ciąg znaków;
- %d, %o, %x – kolejno: liczba dziesiętna, ósemkowa, szesnastkowa;
- %f – liczba rzeczywista;
- %% – znak "%".

Symbol %t oznacza wyświetlanie **czasu systemowego**, dostępne znaczniki:

- %x, %X – kolejno data w formacie m/d/r i godzina w formacie g:m:s;
- %d, %m, %y, %Y – kolejno: dzień, miesiąc, rok w skrócie, rok pełny;
- %H, %I, %M, %S – kolejno: godzina (24), godzina (12), minuta, sekunda;
- %B, %b – kolejno: pełna i skrócona nazwa miesiąca;
- %A, %a – kolejno: pełna i skrócona nazwa dnia miesiąca.

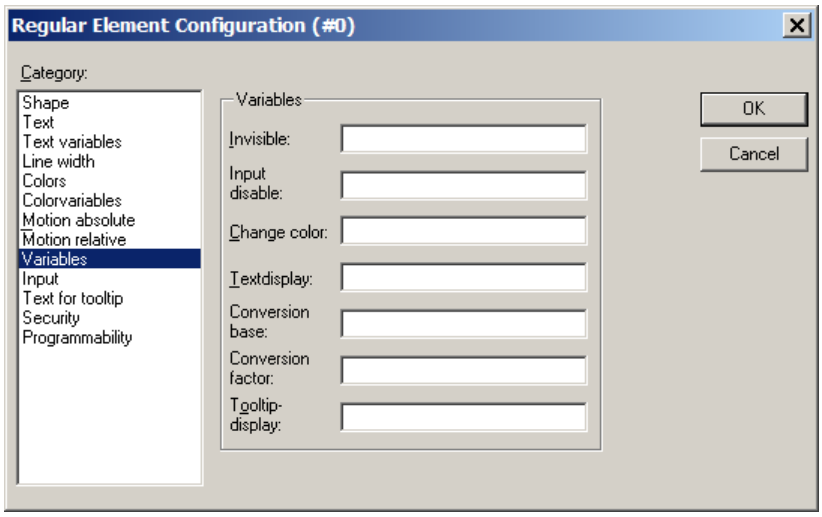
Kombinacja Ctrl+ENTER wprowadza znak przejścia do nowego wiersza.

9.4.2. Colors



Własność określa kolor elementu (dostępne dla podstawowych kształtów – nr 2 na rys 9.2). Możliwe jest przypisanie koloru normalnego i koloru alarmu. Element jest wyświetlany w kolorze normalnym, jeżeli skojarzona zmienna projektowa (patrz punkt 9.4.3.) ma wartość `False` i w kolorze alarmu jeżeli zmienna ma wartość `True`. Przycisk `Inside` określa kolor wypełnienia, `Frame` kolor obramowania.

9.4.3. Variables

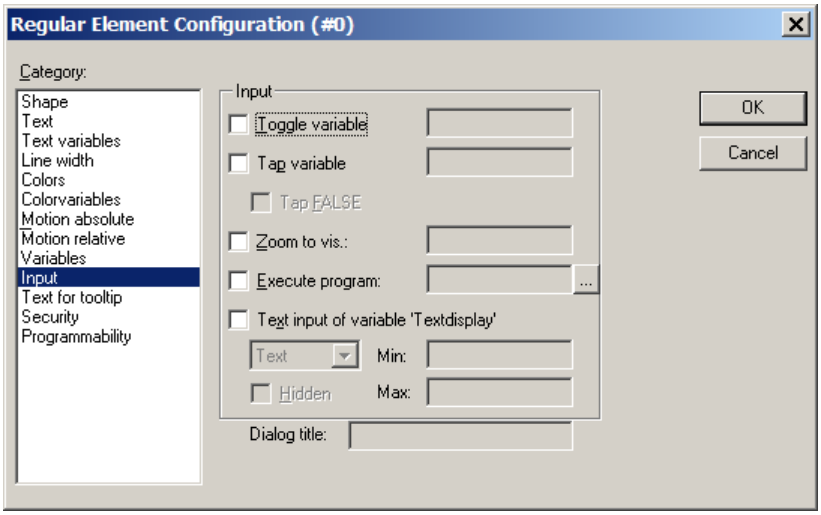


Własność określa skojarzone z elementem zmienne, które wpływają na jego stan:

- `Invisible` – określa czy element jest widoczny, jeżeli skojarzona zmienna projektowa ma wartość `True` element jest niewidoczny, jeżeli wartość `False` element jest wyświetlany na masce wizualizacji;
- `Change color` – określa zmianę koloru z normalnego na alarmowy (punkt 9.4.2), jeżeli skojarzona zmienna projektowe ma wartość `True` element jest wuświetlany w kolorze normalnym, jeżeli `False` w kolorze alarmowym;
- `Text display` – określa zmienną, której wartość będzie wyświetlana w etykiecie elementu (punkt 9.4.1).

Uwaga: Odwołania do zmiennych projektowych mają postać: `nazwa_POU.Nazwa_zmiennej` (np. `PLC_PRG.a`). Wybór zmiennej ułatwia okno `Input assistant` dostępne po naciśnięciu klawisza `F2`.

9.4.4. Input

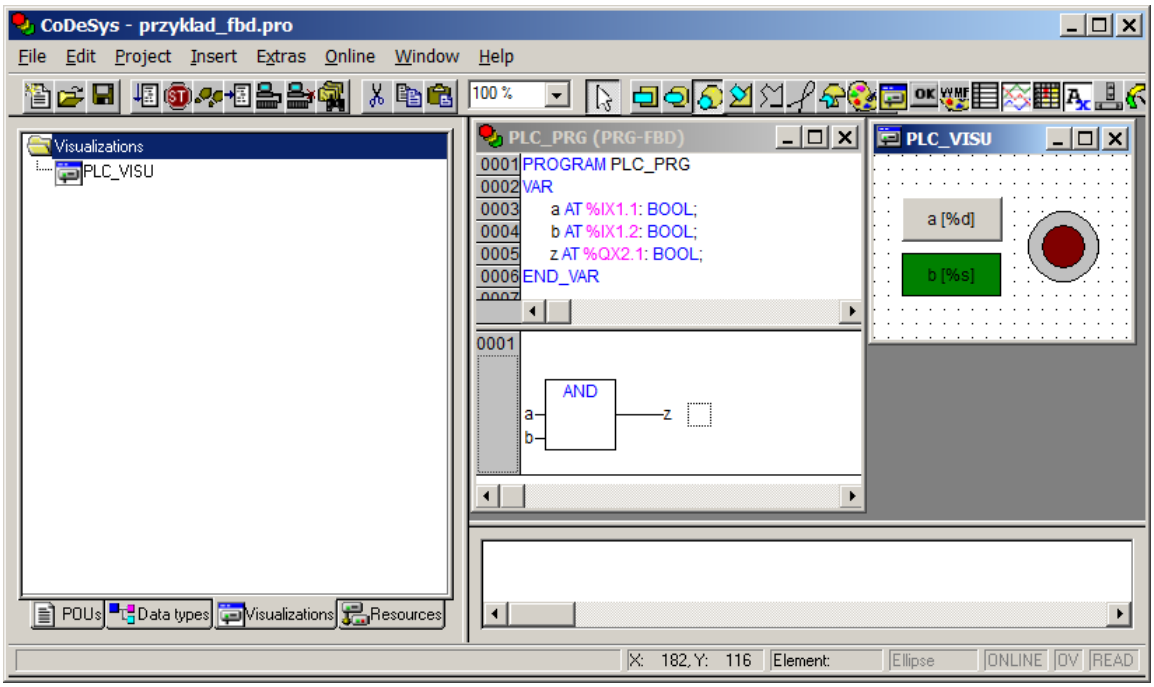


Własność określa reakcję elementu na działania operatora (kliknięcie myszą, dotknięcie panelu, itp.):

- `Toggle variable` – kliknięcie na elemencie na trwale przełącza wartość skojarzonej zmiennej binarnej, ponowne kliknięcie przywraca wartość pierwotną (`True->False`, `False->True`, itd.);
- `Tap variable` – wciśnięcie przycisku myszy na elemencie zmienia wartość skojarzonej zmiennej binarnej z `False` na `True`, zwolnienie przycisku myszy przywraca wartość `False`. W przypadku włączenia opcji `Tap FALSE` działanie zostaje odwrócone;
- `Zoom to vis` – zamyka bieżącą wizualizację i otwiera drugą, której nazwa została określona w polu edycyjnym.

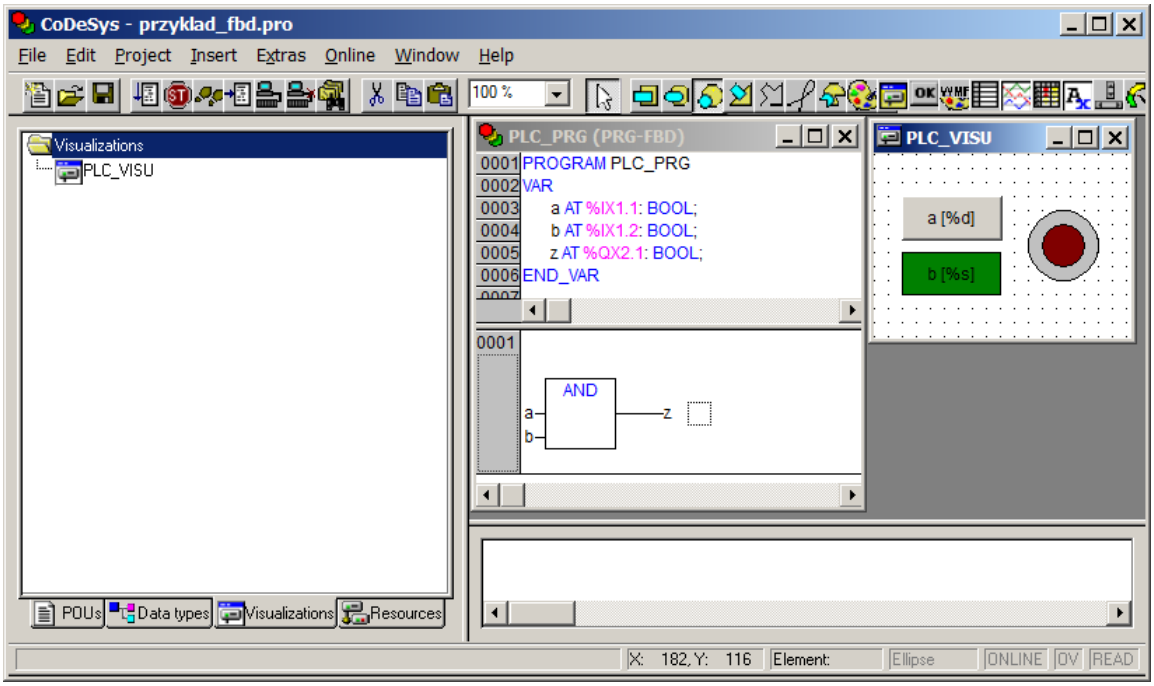
Uwaga: Odwołania do zmiennych projektowych mają postać: `nazwa_POU.Nazwa_zmiennej` (np. `PLC_PRG.a`). Wybór zmiennej ułatwia okno `Input assistant` dostępne po naciśnięciu klawisza `F2`.

9.5. Przykład



Rys. 9.3. Przykładowy projekt

Przykład prezentuje wykorzystanie omówionych elementów (prostokąt, koło i przycisk) do wizualizacji działania programu realizującego funkcję AND. Kliknięcie na przycisk i kwadrat zmienia wartość zmiennych wejściowych (`a` i `b`), wartość zmiennej wejściowej (`z`) jest sygnalizowana przez zmianę koloru koła. Elementy związane z wejściem wyświetlają aktualne wartości skojarzonych zmiennych (liczbowa w przypadku przycisku i tekstowo w przypadku prostokąta). Omawiany projekt został pokazany na rysunku 9.3. z programem zrealizowanym w FBD, jednak wybór języka nie wpływa na sposób wykonania wizualizacji (projekt w wersji FBD i LD jest dostępny jako spakowane archiwum).



Konfiguracja elementów wizualizacji:

1. Przycisk (sygnał wejściowy `a`):
   - `Text->Content: a [%d]`
   - `Variables->Textdisplay: PLC_PRG.a`
   - `Input->Toggle variable: PLC_PRG.a`
2. Prostokąt (sygnał wejściowy `b`):
   - `Text->Content: b [%s]`
   - `Colors->Color->Inside: ciemnozielony`
   - `Colors->Alarm color->Inside: jasnozielony`
   - `Variables->Change color: PLC_PRG.b`
   - `Variables->Textdisplay: PLC_PRG.b`
   - `Input->Toggle variable: PLC_PRG.b`
3. Koło (sygnał wyjściowy `z`), narysowany jako dwa koła, konfiguracja dotyczy elementu na górze:
   - `Colors->Color->Inside: ciemnoczerwony`,
   - `Colors->Alarm color->Inside: czerwony`,
   - `Variables->Change color: PLC_PRG.z`

Umieszczenie znaczników formatu `%d` i `%s` we własności `Text->Content` oraz ustawienie zmiennej `Variables->Textdisplay` pozwala wyświetlić w etykietach elementów skojarzonych z wejściem aktualnych wartości zmiennych wejściowych (odpowiednio w postaci liczby i tekstu). Odpowiednie ustawienie kolorów (własność `Colors`) w elementach związanych ze zmiennymi `b` i `z` oraz skojarzenie odpowiedniej zmiennej z własnością `Variables->Change color` powoduje, że zmiana wartości

zmiennej jest sygnalizowana jaśniejszym kolorem elementu. Własność `Input->Toggle variable` skojarzona z odpowiednią zmienną wejściową pozwala przełączać wartości wejść w czasie działania programu przez kliknięcie na przycisk lub prostokąt. *Uwaga*: ustawienie własności `Input->Toggle variable` dla elementu związanego ze zmienną wyjściową nie byłoby właściwe – wartość sygnału wyjściowego zależy tylko od sygnałów wejściowych, więc operator nie może zmienić tej wartości bezpośrednio, może tylko obserwować reakcje wyjścia na zmianę wejścia.