SIMATIC STEP7

Podstawy programowania w STEP7

Podręcznik - wydanie 3
Egzemplarz bezpłatny

SIMATIC STEP7

www.siemens.pl/S7-300

SIEMENS

SIEMENS

SIMATIC S7

Podstawy programowania w STEP 7

Podręcznik	Wydanie 3

SIEMENS

SIMATIC S7

Podstawy programowania w STEP 7

Wprowadzenie do STEP7 1

SIMATIC Manager 2

Programowanie symboliczne 3

Tworzenie programu w OB1 4

Tworzenie programu z wykorzystaniem bloków funkcyjnych i bloków danych 5

Konfiguracja jednostki podstawowej 6

Ładowanie i testowanie programu 7

Programowanie funkcji FC 8

Programowanie globalnego bloku danych 9

Programowanie wielokrotnego lokalnego bloku danych 10

Konfiguracja systemu rozproszonego 11

Warszawa 2010
Wydanie 3

Techniczne wskazówki bezpieczeństwa

Podręcznik ten zawiera wskazówki, których powinieneś przestrzegać dla własnego bezpieczeństwa oraz w celu uniknięcia uszkodzenia sprzętu. Wskazówki te oznaczono specjalnym trójkątem ostrzegawczym i w zależności od stopnia zagrożenia następująco opisano:

Niebezpieczeństwo

Oznacza zagrożenie śmiercią, poważnym uszkodzeniem ciała lub dużymi stratami, jeśli nie będzie przestrzegać się odpowiednich zasad bezpieczeństwa.

Ostrzeżenie

Oznacza **możliwość wystąpienia** zagrożenia śmiercią, poważnym uszkodzeniem ciała lub dużymi stratami, jeśli nie będzie przestrzegać się odpowiednich zasad bezpieczeństwa.

Uwaga

Oznacza możliwość wystąpienia lekkiego uszkodzenia ciała lub strat, jeśli nie będzie przestrzegać się odpowiednich zasad bezpieczeństwa.

Wskazówka

To ważna informacja o produkcie, jego użytkowaniu lub wskazanie szczególnie ważnej części dokumentacji.

Wykwalifikowany personel

Instalacja i użytkowanie sprzętu może przeprowadzać tylko **wykwalifikowany personel**. W rozumieniu tego podręcznika wykwalifikowany personel to osoby uprawnione do podłączania, uziemiania i oznaczania urządzeń, systemów i obiegów prądu zgodnie ze standardami bezpieczeństwa

Użytkowanie zgodnie z przeznaczeniem

Należy przestrzegać co następuje:

Ostrzeżenie

Urządzenie może być używane tylko w układach przewidzianych w katalogu i w opisie technicznym, tylko w połączeniu z urządzeniami zalecanymi lub dopuszczonymi przez firmę Siemens obcymi urządzeniami i komponentami.
Prawidłowa i bezpieczna praca urządzenia możliwa jest tylko przy założeniu, że nastąpił właściwy transport, właściwe magazynowanie, montaż i prawidłowe użytkowanie i konserwacja urządzenia.

Znaki towarowe

SIMATIC®, SIMATIC HMI® i SIMATIC NET® są zarejestrowanymi znakami towarowymi firmy Siemens. Pozostałe oznaczenia w tym tekście mogą być znakami towarowymi, a ich używanie przez osoby trzecie dla ich własnych celów może naruszać prawa właścicieli.

Witamy w STEP 7...

...podstawowym pakiecie programowym SIMATIC służącym do tworzenia programów dla sterowników SIMATIC S7-300/400,C7/WinAc w LAD, FDB lub STL.

Informacje zawarte w tej dokumentacji

Podręcznik ten pozwala Państwu zapoznać się z podstawami języka programowania STEP 7. Dodatkowo zostały przedstawione najważniejsze operacje wykonywane na ekranach edytora poparte przykładowymi ćwiczeniami. Dokumentacja podzielona jest tak, by możliwe było rozpoczęcie ćwiczeń w dowolnym rozdziale.

W każdym rozdziale znajduje się część opisowa i operacyjna.

Pomocne Państwu będą umiejętności pracy z myszką, ikonami, przewijalnym menu itp. Dodatkowym atutem będzie wiedza z zakresu sterowników programowalnych.

Na kursach szkoleniowych możecie Państwo pogłębić wiedzę z zakresu STEP 7, oraz projektowania i tworzenia systemów automatyki z wykorzystaniem sterowników SIMATIC S7.

Podstawowe wymagania pracy z dokumentacją

W celu przeprowadzenia praktycznych ćwiczeń STEP 7 z tej dokumentacji, konieczne jest:

- Programator PG lub komputer PC
- Pakiet programu STEP 7 i dyskietka autoryzacyjna
- Sterownik SIMATIC S7
  (rozdział 7 „Ładownie i testowanie programu”)

Dodatkowe dokumentacje do STEP 7

- STEP 7 Basic Information
- STEP 7 Reference Information

Na dysku CD programu znajduje się elektroniczny podręcznik, który także można zamówić w każdym biurze handlowym firmy Siemens. Wszelkie informacje z tego podręcznika są dostępne w STEP 7 i w „pomocy”.

Życzymy Państwu dobrej zabawy i powodzenia!

SPIS TREŚCI STRONA

1 WPROWADZENIE W STEP 7 6
1.1 CZEGO BĘDZIECIE SIĘ PAŃSTWO UCZYĆ 6
1.2 WSPÓŁPRACA OPROGRAMOWANIA I SPRZĘTU 8
1.3 PODSTAWOWE OPERACJE W STEP 7 9
1.4 INSTALACJA STEP 7 10

2 SIMATIC MANAGER 11
2.1 URUCHOMIENIE SIMATIC MANAGER I ZAKŁADANIE PROJEKTU 11
2.2 STRUKTURA PROJEKTU W SIMATIC MANAGER ORAZ WYWOŁANIE POMOCY DO STEP7 14

3 PROGRAMOWANIE SYMBOLICZNE 17
3.1 ADRESOWANIE ABSOLUTNE 17
3.2 PROGRAMOWANIE SYMBOLICZNE 18

4 TWORZENIE PROGRAMU W BLOKU OB1 20
4.1 OTWARCIE EDYTORA LAD/STL/FBD 21
4.2 PROGRAMOWANIE OB1 W SCHEMACIE DRABINKOWYM 23
4.3 PROGRAMOWANIE OB1 W LIŚCIE INSTRUKCJI (STL) 27
4.4 PROGRAMOWANIE OB1 W BLOKACH LOGICZNYCH (FBD) 30

5 TWORZENIE PROGRAMU Z WYKORZYSTANIEM BLOKÓW FUNKCYJNYCH I BLOKÓW DANYCH 34
5.1 DODAWANIE ORAZ OTWARCIE BLOKU FUNKCYJNEGO (FB) 34
5.2 PROGRAMOWANIE FB1 W SCHEMACIE DRABINKOWYM 36
5.3 PROGRAMOWANIE FB1 W LIŚCIE INSTRUKCJI 40
5.4 PROGRAMOWANIE FB1 W BLOKACH LOGICZNYCH 42
5.5 TWORZENIE LOKALNEGO BLOKU DANYCH ORAZ AKTUALIZACJA WARTOŚCI 45
5.6 PROGRAMOWANIE WYWOŁANIA BLOKU W SCHEMACIE DRABINKOWYM 47
5.7 PROGRAMOWANIE WYWOŁANIA BLOKU W LIŚCIE INSTRUKCJI 50
5.8 PROGRAMOWANIE WYWOŁANIA BLOKU W SCHEMACIE BLOKÓW LOGICZNYCH 52

6 KONFIGURACJA JEDNOSTKI PODSTAWOWEJ 55
6.1 KONFIGURACJA SPRZĘTOWA 55

7 ŁADOWANIE I TESTOWANIE PROGRAMU 57
7.1 TWORZENIE POŁĄCZEŃ ONLINE 57
7.2 ŁADOWANIE PROGRAMU DO STEROWNIKA 59
7.3 TESTOWANIE PROGRAMU – STATUS BLOKU 62
7.4 TESTOWANIE DZIAŁANIA PROGRAMU Z WYKORZYSTANIEM TABELI ZMIENNYCH 64
7.5 WYWOŁANIE BUFORA DIAGNOSTYCZNEGO 68

8 PROGRAMOWANIE FUNKCJI FC 70
8.1 ZAŁOŻENIE I OTWARCIE FUNKCJI 70
8.2 PROGRAMOWANIE FUNKCJI 72
8.3 WYWOŁANIE FUNKCJI W OB1 75

9 PROGRAMOWANIE GLOBALNEGO BLOKU DANYCH 78
9.1 ZAKŁADANIE I OTWIERANIE GLOBALNEGO BLOKU DANYCH 78

10 PROGRAMOWANIE WIELOKROTNEGO LOKALNEGO BLOKU DANYCH 82
10.1 ZAKŁADANIE I OTWIERANIE NADRZĘDNEGO BLOKU FUNKCYJNEGO 82
10.2 PROGRAMOWANIE FB10 84
10.3 TWORZENIE DB10 I WSTAWIANIE AKTUALNYCH WARTOŚCI 88
10.4 WYWOŁANIE FB10 W OB1 90

11 KONFIGURACJA SYSTEMU ROZPROSZONEGO 93
11.1 TWORZENIE SYSTEMU ZDECENTRALIZOWANEGO PRZY UŻYCIU SIECI PROFIBUS DP 93

1 Wprowadzenie STEP 7

1.1 Czego będziecie się Państwo uczyć

Za pomocą praktycznych ćwiczeń pokażemy, jak proste jest programowanie w LAD (Ladder Logic), FBD (Function Block Diagram) lub STL (Statement List) za pomocą STEP 7.

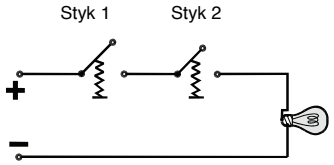
Zawarte w poszczególnych rozdziałach szczegółowe wskazówki pokazują krok po kroku szerokie możliwości zastosowania STEP 7.

Tworzenie programu z wykorzystaniem instrukcji binarnych

W rozdziałach 2 do 7 stworzymy program wykorzystując instrukcje binarne. Za pomocą instrukcji binarnych będziemy odpytać wejścia i wyjścia sterownika.

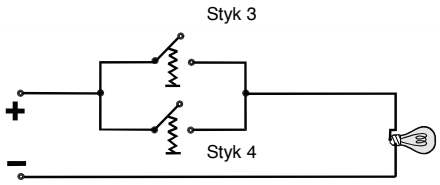
Przykłady programowania w tej dokumentacji bazują m.in. na trzech podstawowych instrukcjach binarnych.

Pierwsza funkcja binarna, którą będziemy później programować, to funkcja AND (iloczyn logiczny). Funkcję AND można zobrazować w obwodzie elektrycznym za pomocą dwóch styków.



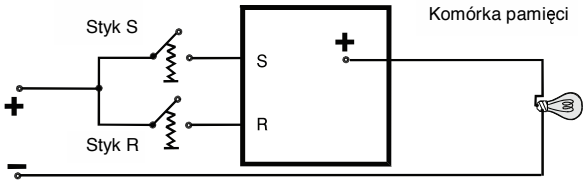
Przy zwartym styku 1 **i** 2 zapala się lampka

Druga funkcja binarna to OR (suma logiczna), która także może zostać przedstawiona w obwodzie elektrycznym.



Przy zwarciu styku 3 **lub** 4 zapala się lampka

Trzecia funkcja binarna to funkcja zapamiętywania. Funkcja SR reaguje w obwodzie elektrycznym na określone stany napięć i przekazuje je odpowiednio dalej.

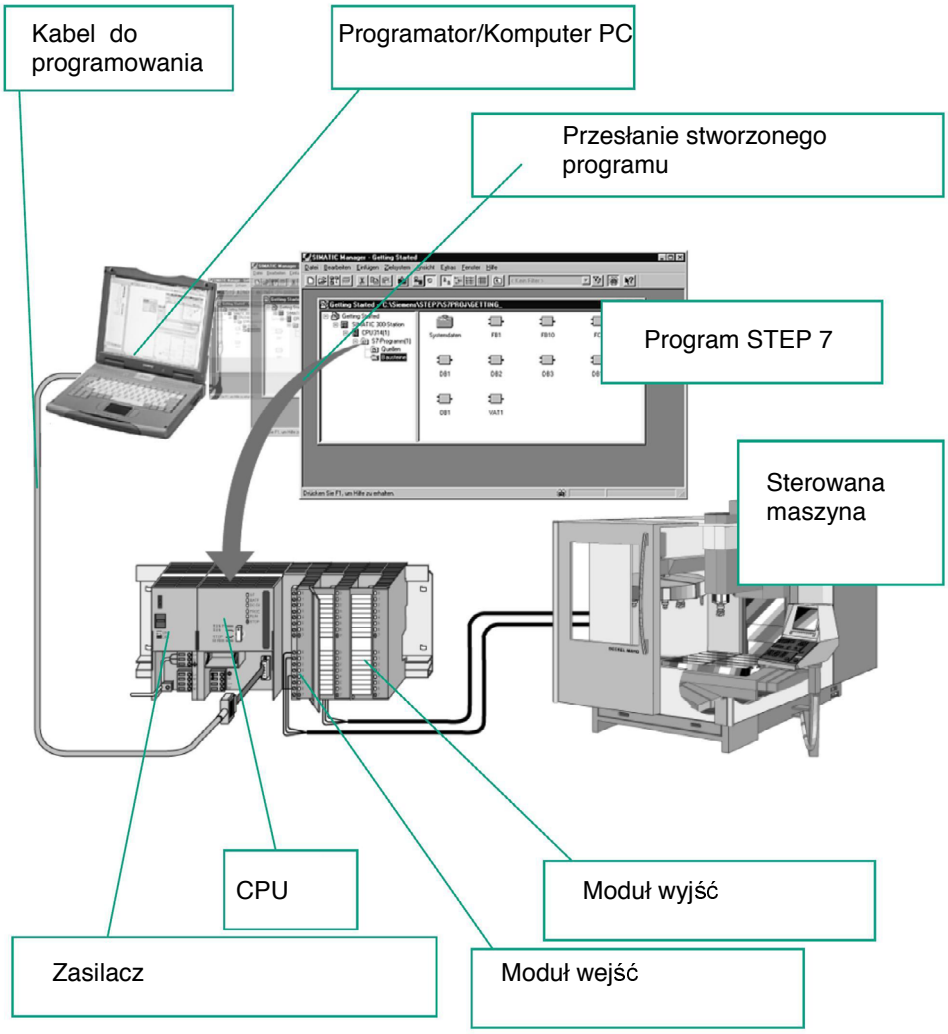


Po wciśnięciu przycisku S lampka pali się do momentu wciśnięcia przycisku R

1.2 Współpraca oprogramowania i sprzętu

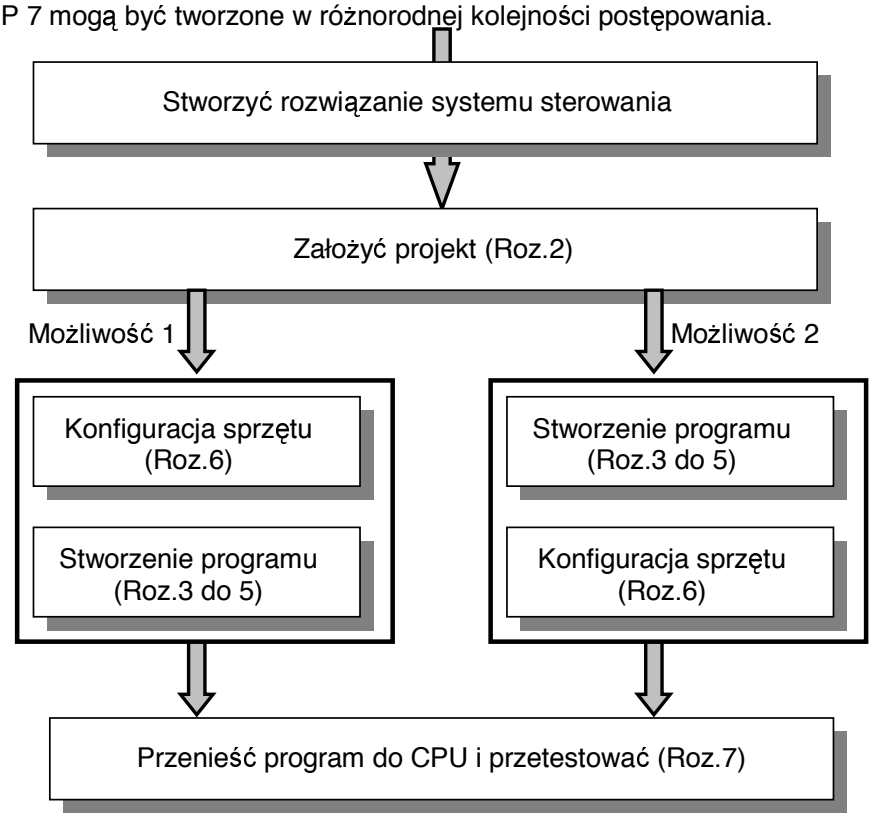
Za pomocą programu STEP 7 stworzymy w ramach jednego projektu własny program. Sterownik S7 składa się z zasilacza, CPU oraz modułów wejścia / wyjścia (moduły I/O).

Sterownik swobodnie programowalny (PLC) wykonując program nadzoruje i steruje pracą maszyny. W programie poszczególne wejścia i wyjścia modułów wywoływane są przy użyciu adresów.



1.3 Podstawowe operacje w STEP 7

Przed rozpoczęciem projektowania, powinni Państwo wiedzieć, iż projekty w STEP 7 mogą być tworzone w różnorodnej kolejności postępowania.



Przy tworzeniu skomplikowanych programów z dużą ilością wejść i wyjść polecamy stworzenie najpierw konfiguracji. Zaletą tego sposobu jest wskazywanie w edytorze konfiguracji możliwych do wykorzystania adresów wejść/wyjść.
W drugim sposobie należy ustalić sobie adresy sterownika w zależności od posiadanego sprzętu bez wykorzystania w tym celu konfiguracji w STEP 7.

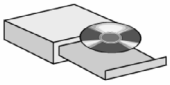
Tworząc konfigurację sprzętową ustalone zostają nie tylko adresy poszczególnych modułów, ale możemy także ustalić parametry i zachowanie się poszczególnych elementów sprzętu. Np. dla systemu sieciowego w którym mamy do czynienia z kilkoma sterownikami, możemy ustalić adresy w sieci MPI dla poszczególnych CPU.

Ponieważ w tej dokumentacji nie będziemy używali dużej ilości wejść i wyjść, pominięta zostanie konfiguracja sterownika i przejdziemy od razu do programowania.

1.4 Instalacja STEP 7

Niezależnie od tego, czy rozpoczynamy projektowanie od programowania, czy też konfiguracji sprzętu, nieodzownym jest zainstalowanie STEP7. W przypadku użycia programatora STEP7 jest już w nim zainstalowany.

Przed instalacją STEP7 na PG/PC bez zainstalowanej wersji należy zapoznać się ze wskazówkami dotyczącymi sprzętu i oprogramowania. Znajdują się one na CD STEP7 w zbiorze *Readme.wri.*



W celu zainstalowania STEP7 należy włożyć CD. Program instalacyjny uruchomi się automatycznie. Należy postępować wg wskazówek.

Jeśli program instalacyjny się nie uruchomi automatycznie, należy go znaleźć na CD pod
<dysk>:/STEP 7/Disk1/setup.exe

Po zakończeniu instalacji i restartowaniu komputera na Windows Desktop pojawi się ikonka SIMATIC Manager.

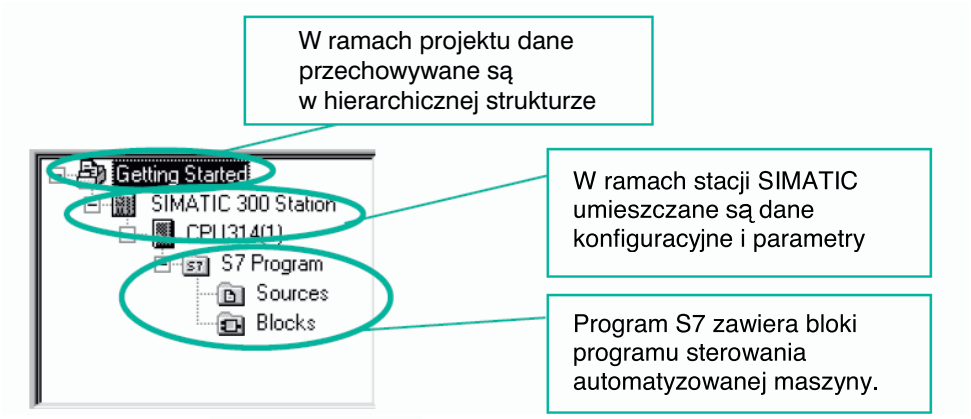
Podwójne kliknięcie na powstałą ikonkę SIMATIC Manager powoduje uruchomienie edytora STEP7.

Dalsze informacje dotyczące instalacji zawarte są w zbiorze *Readme.wri* znajdującym się na CD STEP7.

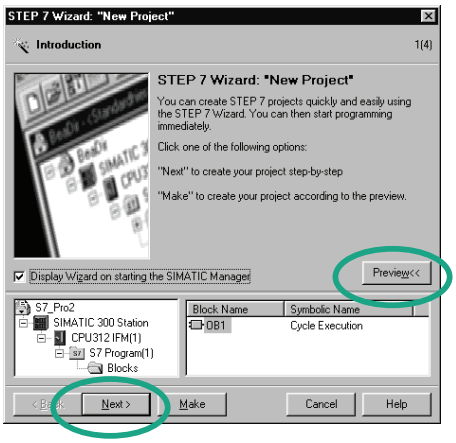
2 SIMATIC Manager

2.1 Uruchomienie SIMATIC Manager i zakładanie projektu

Po uruchomieniu STEP7, głównym aktywnym oknem jest SIMATIC Manager. Jednocześnie uruchamiany może być Wizard STEP7, który pomaga w założeniu nowego projektu. Projekt tworzony jest z zachowaniem odpowiedniej struktury.

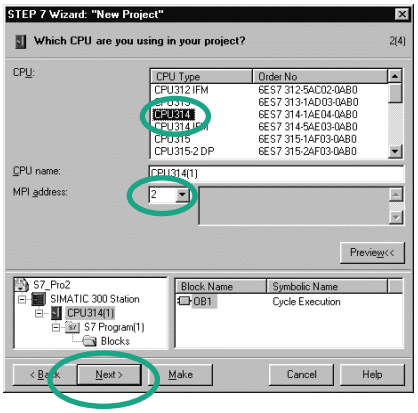


Dwukrotne kliknięcie na ikonkę **SIMATIC Manager** uaktywnia Wizard Step7.



Za pomocą przycisku **Preview** podejrzeć można strukturę zakładanego projektu.

By przejść na następne okienko dialogowe należy wcisnąć **Next**



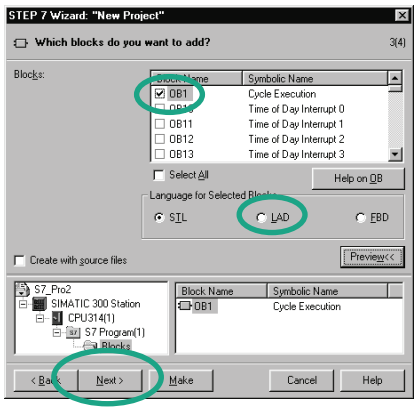
Wykonując projekt ćwiczeniowy wybierz CPU 314. Przykład jest tak wykonany, że można jednak wybrać aktualnie posiadany sterownik.

Wstępnie ustawiany jest adres MPI o numerze 2.

Przejście do następnego okna następuje po kliknięciu przycisku **Next.**

Każde CPU posiada odpowiednie parametry np. dotyczące rozbudowy pamięci czy też przestrzeń adresową. Dlatego też typ CPU należy wybrać przed rozpoczęciem programowania.

Ustawienie adresu MPI (Multi Point Interface) wymagane jest do komunikacji CPU z PG/PC



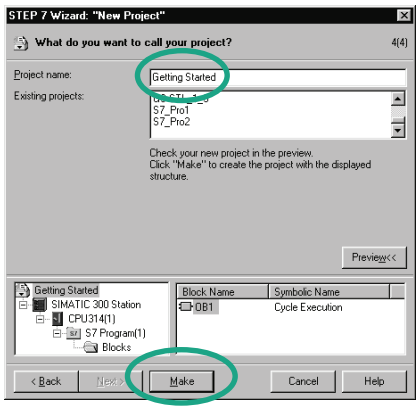
Wybierz blok OB1 jeśli nie jest jeszcze zaznaczony.

Wybierz sposób prezentacji programu: **(LAD)** Ladder Logic – schemat drabinkowy, **(STL)** Statement List – lista instrukcji, **(FBD)** Function Block Diagram – logiczne bloki funkcyjne.

Potwierdź ustawienia za pomocą klawisza **Next.**

OB1 jest głównym blokiem programowym, z którego wywoływane są inne bloki.

Zmiana sposobu prezentacji programu jest możliwa także później.



Kliknij dwukrotnie na nazwę projektu (pole „Project name”) i wprowadź nową nazwę np. „Getting Started”.

Kliknij na przycisk **Make** w celu wygenerowania projektu według dokonanych ustawień.

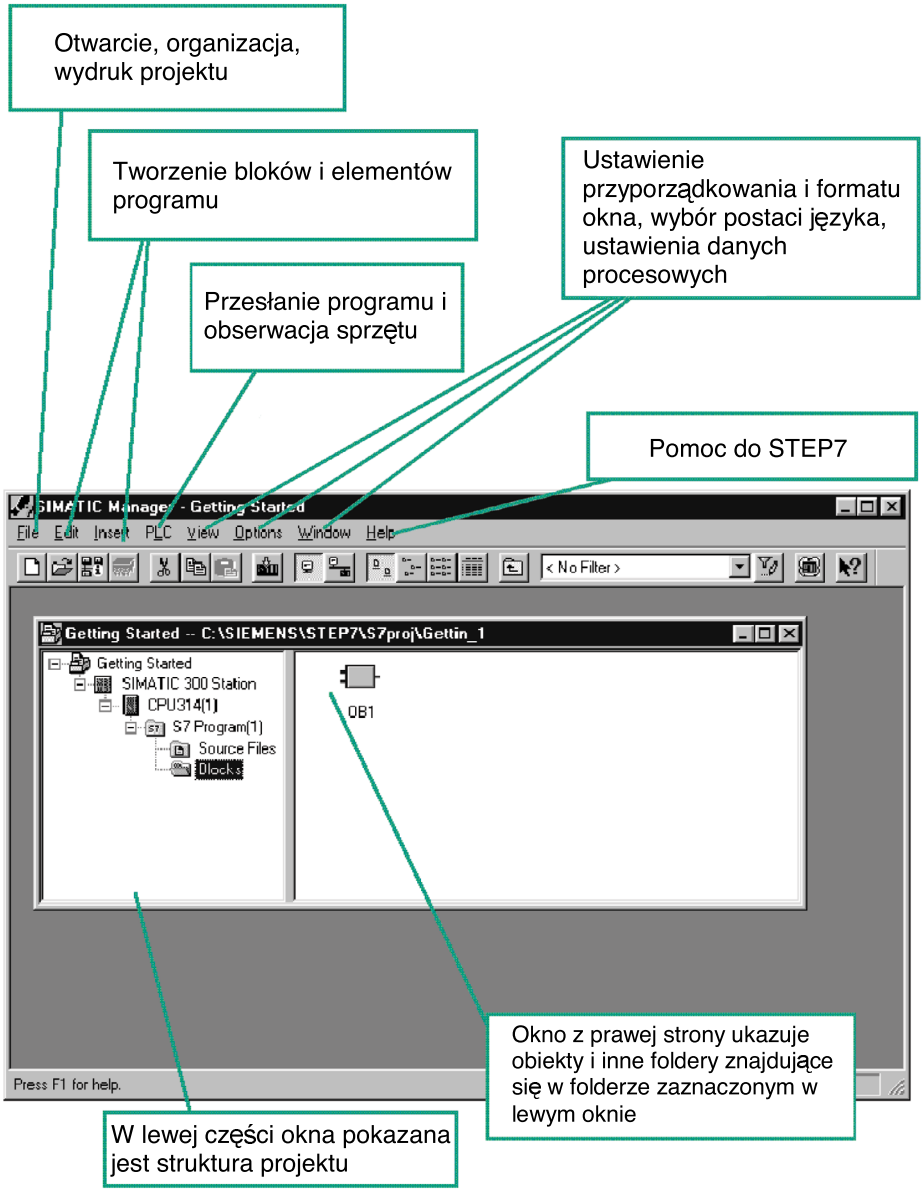
Po kliknięciu na przycisk **Make** w oknie SIMATIC Manager ukaże się projekt z nową nazwą (np. „Getting Started”). Na kolejnych stronach przedstawione zostanie znaczenie poszczególnych elementów tworzonej struktury projektu, oraz podamy efektywne metody ich wykorzystywania.

Wizard STEP7 będzie uruchamiany każdorazowo przy uruchomieniu programu. Tę własność można dezaktywować w pierwszym ukazującym się okienku. Jeśli chcemy tworzyć projekty bez wykorzystania Wizard, należy tworzyć projekt ustawiając każdy z parametrów projektu niezależnie.

Dalsze informacje znajdują się w **Help > Contents** "Setting Up and Editing the Project”.

2.2 Struktura projektu w SIMATIC Manager oraz wywołanie pomocy do STEP7

Z chwilą zakończenia tworzenia projektu przy pomocy Wizard w SIMATIC Manager otwarte zostaje okno projektu (np. „Getting Started”). Z tego poziomu możliwe jest wywoływanie wszystkich funkcji i okien edytora STEP7.



Wywoływanie pomocy do STEP 7

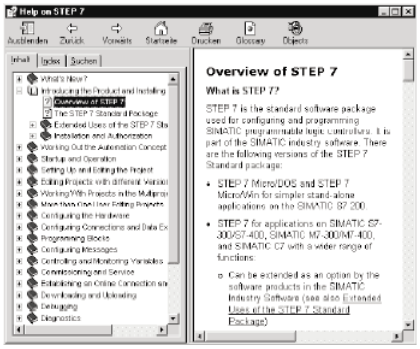
F1

Możliwość nr 1:

Zaznacz dowolne polecenie Menu i naciśnij przycisk F1. Otrzymasz fachową pomoc dotyczącą zaznaczonego polecenia Menu.

Możliwość nr 2:

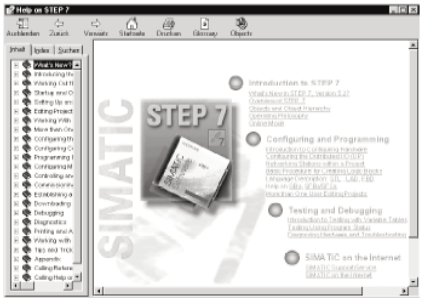
Poprzez Menu wchodzisz do pomocy do STEP 7.



W lewym polu okna pokazuje się spis treści zawierający różne tematy pomocy, a w prawym polu wybrany temat.

Poszukiwany temat znajduje się przyciskając **+** w polu **Contents.** W prawym polu okna pojawia się jednocześnie zawartość wybranego tematu.

Za pomocą **Index** i **Find** można podawać szukane określenia i w ten sposób odnajdywać żądane tematy pomocy.



Możliwość nr 3:

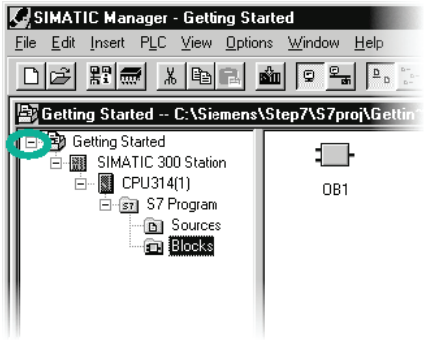
Kliknąć na ikonę ”Start page” w STEP 7 Online Help, która umożliwia wyświetlenie informacji pomocniczych według następujących rozdziałów:

- Getting Started with STEP 7;
- Configuring & Programming;
- Testing & Debugging;
- SIMATIC on the Internet

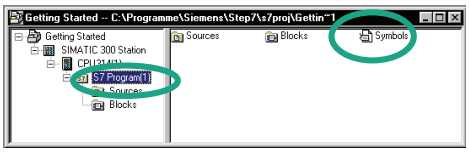
Możliwość nr 4:

Kliknij na ikonę pomocy. Kolejne kliknięcie na określony obiekt aktywuje pomoc.

Poruszanie się w strukturze projektu.



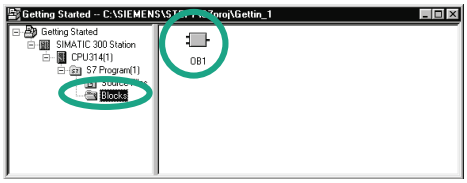
Założony właśnie projekt pokazywany jest z wybraną stacją S7 i CPU
Kliknij na **+** lub **–** w celu otwarcia lub zamknięcia poszczególnych plików.
W prawym polu ekranu wskazując poszczególne obiekty można później wywołać kolejne funkcje.



Kliknij na plik **S7-Program (1).** Zawiera on wszystkie konieczne części składowe programu.

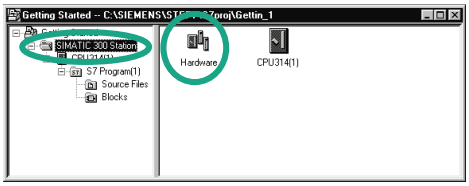
Za pomocą symboli nadaje się adresom symboliczne nazwy (patrz rozdz. 3).

Grupa Source File zawiera programy źródłowe. Programy źródłowe nie są omawiane w tej dokumentacji.



Kliknij na plik **Blocks**. Zawiera on już założony OB1. Tam też później znajdziemy kolejne bloki.

Wykorzystując bloki wprowadzany jest program sterownika w postaci LAD, STL lub FBD.



Kliknij na **SIMATIC 300 Station.** Tutaj znajdują się dane projektowe konfiguracji sprzętowej.

Ikona **Hardware** umożliwia wejście w ustawienia konfiguracji sprzętowej (patrz rozdział 6).

Pakiety opcjonalne takie jak np. PLC-SIM (symulacja pracy sterownika), czy też S7-GRAPH (graficzne programowanie), są po zainstalowaniu zintegrowane w STEP7. Dlatego też np. bloki funkcyjne stworzone w S7-GRAPH można wywoływać bezpośrednio w SIMATIC Manager.

Dalsze informacje znajdują się w **Help>Contents** w temacie „Working Out the Automation Concept” oraz ”Basics of Designing the Program Structure”.

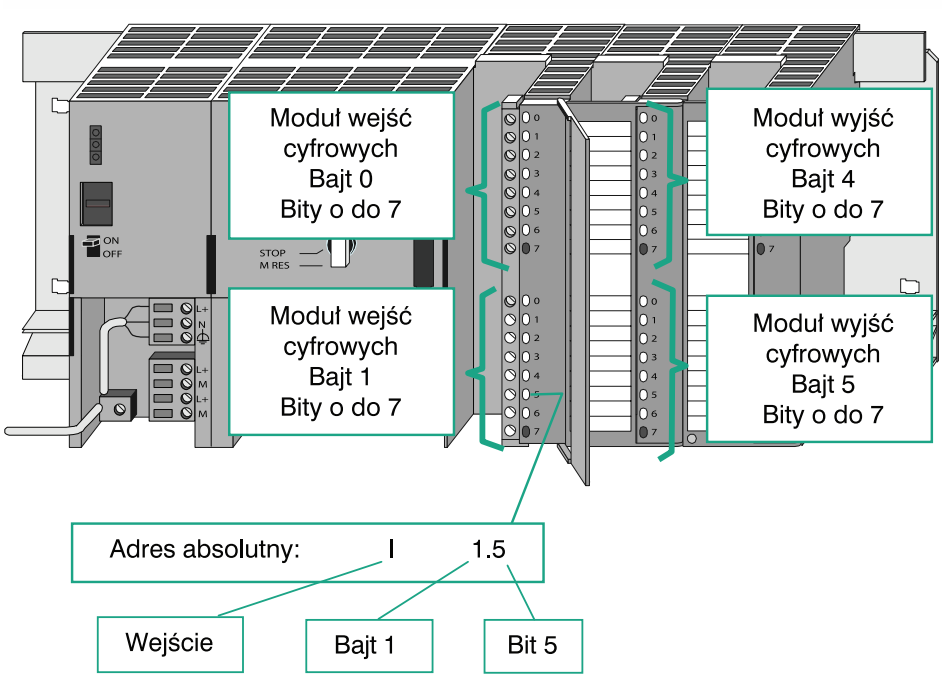
Więcej informacji na temat pakietów opcjonalnych znajdziecie Państwo w katalogu ST70 prezentującym rodzinę SIMATIC.

3 Programowanie symboliczne

3.1 Adresowanie absolutne

Każde wejście i wyjście posiada adres absolutny wynikający z konfiguracji sterownika. Adres ten jest określany bezpośrednio, tzn. absolutnie.

Adresy absolutne możemy zastąpić dowolnymi nazwami symbolicznymi.

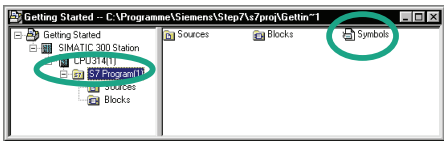


Programowania z użyciem adresów absolutnych powinno się używać tylko wtedy, gdy sterownik posiada niewiele wejść/wyjść.

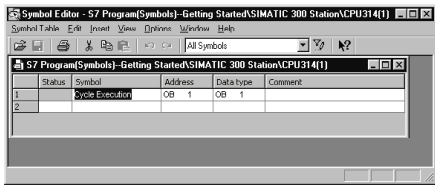
3.2 Programowanie symboliczne

W tabeli symbolicznej określa się nazwy symboliczne przypisując je adresom absolutnym elementów, które mają być używane w programie; np. Wejście I0.1 oraz symbol *„Przycisk 1”* . Nazwy obowiązują w całym programie i są określone jako zmienne globalne.

Praca w edytorze symboli



Należy w projekcie zaznaczyć poziom **S7-program(1)** i klikając dwukrotnie ikonkę **Symbols** otworzyć tabelę symboli.



Początkowo tabela symboli zawiera jedynie wstępnie zdefiniowaną nazwę bloku OB1.

	Status	Symbol	Address	Data type
1		Cycle Execution	OB 1	OB 1
2				

	Status	Symbol	Address	Data type
1		Main Program	OB 1	OB 1
2		Green Light	Q 4.0	BOOL

Comment

Kliknij *Cycle Execution* i zamień we własną nazwę programu np. *„Main Program”*.
Wprowadź w drugiej linii nazwę symboliczną np. *Green Light* i adres Q4.0. Typ danej zostanie dobrany automatycznie.
Klikając na kolumnę **Comment** możemy wprowadzić komentarze do symboli. Wprowadzanie nowego wiersza kończymy wciśnięciem przycisku **ENTER.**

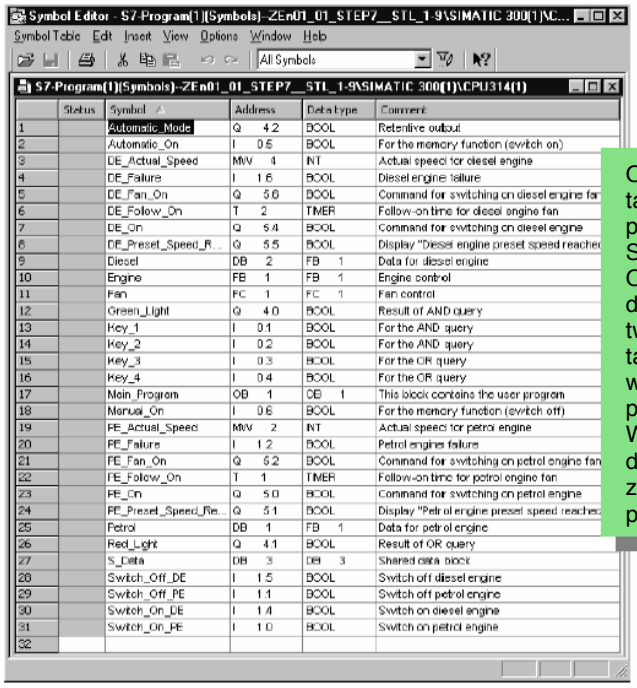
	Status	Symbol	Address	Data type
1		Main Program	OB 1	OB 1
2		Green Light	Q 4.0	BOOL
3		Red Light	Q 4.1	BOOL

Wprowadź w trzeciej linii nazwę *„Red light”* oraz adres Q4.1 i wciśnij ENTER w celu zakończenia operacji.

W ten sposób możemy przypisać symbole do wszystkich używanych w programie elementów.

Zabezpiecz wprowadzone symbole za pomocą ikonki **Save** i zamknij okno.

Istnieje możliwość kopiowania tabeli symbolicznych (np. z przygotowanego projektu).



	Status	Symbol	Address	Data type	Comment
1		Automatic_Mode	Q 4.2	BOOL	Retentive output
2		Automatic_On	I 0.6	BOOL	For the memory function (switch on)
3		DE_Actual_Speed	MW 4	INT	Actual speed for diesel engine
4		DE_Failure	I 1.6	BOOL	Diesel engine failure
5		DE_Fan_On	Q 5.6	BOOL	Command for switching on diesel engine fan
6		DE_Follow_On	T 2	TIMER	Follow-on time for diesel engine fan
7		DE_On	Q 5.4	BOOL	Command for switching on diesel engine
8		DE_Preset_Speed_R...	Q 5.5	BOOL	Display "Diesel engine preset speed reached
9		Diesel	DB 2	FB 1	Data for diesel engine
10		Engine	FB 1	FB 1	Engine control
11		Fan	FC 1	FC 1	Fan control
12		Green_Light	Q 4.0	BOOL	Result of AND query
13		Key_1	I 0.1	BOOL	For the AND query
14		Key_2	I 0.2	BOOL	For the AND query
15		Key_3	I 0.3	BOOL	For the OR query
16		Key_4	I 0.4	BOOL	For the OR query
17		Main_Program	OB 1	OB 1	This block contains the user program
18		Manual_On	I 0.5	BOOL	For the memory function (switch off)
19		PE_Actual_Speed	MW 2	INT	Actual speed for petrol engine
20		PE_Failure	I 1.2	BOOL	Petrol engine failure
21		PE_Fan_On	Q 5.2	BOOL	Command for switching on petrol engine fan
22		PE_Follow_On	T 1	TIMER	Follow-on time for petrol engine fan
23		PE_On	Q 5.0	BOOL	Command for switching on petrol engine
24		PE_Preset_Speed_Re...	Q 5.1	BOOL	Display "Petrol engine preset speed reached
25		Petrol	DB 1	FB 1	Data for petrol engine
26		Red_Light	Q 4.1	BOOL	Result of OR query
27		S_Data	DB 3	DB 3	Shared data block
28		Switch_Off_DE	I 1.5	BOOL	Switch off diesel engine
29		Switch_Off_PE	I 1.1	BOOL	Switch off petrol engine
30		Switch_On_DE	I 1.4	BOOL	Switch on diesel engine
31		Switch_On_PE	I 1.0	BOOL	Switch on petrol engine
32					

Obok przedstawiona jest tabela symboliczna dla programu „Getting Started”.
Ogólnie mówiąc dla danego programu tworzona jest tylko jedna tabela symboliczna bez względu na język programowania.
W tabeli symbolicznej dozwolone są wszystkie znaki specjalne (np. polskie fonty)

Format danej, który w tabeli symbolicznej jest nadawany automatycznie, obowiązuje także później w jednostce centralnej. STEP7 umożliwia nadanie między innymi następujących typów danych:

BOOL BYTE WORD DWORD	Dane tego typu są kombinacją binarną BOOL –1bit do DWORD- 32 bity
CHAR	Dane tego typu zawierają dokładnie jeden znak kodu ASCII
INT DINT REAL	Te typy umożliwiają prezentację wartości liczbowych (np. stosowanych w obliczeniach arytmetycznych)
S5TIME TIME DATE TIME_OF_DATE	Dane tego typu reprezentują różnego rodzaju zmienne daty i czasu charakterystyczne dla STEP7(np. ustawienie czasu zadanego Timera)

Więcej informacji znajdziesz w **Help->Contents** w rozdziale „Programming Blocks” oraz „Defining Symbols”.

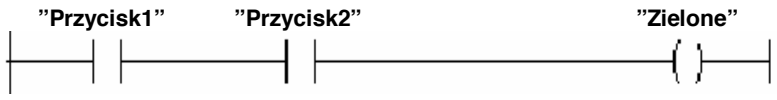
4 Tworzenie programu w bloku OB1

4.1 Otwarcie edytora LAD/STL/FBD

Wybór LAD Ladder Logic (zapisu drabinkowego), STL Statement List (listy instrukcji), lub FBD Function Vlock Diagram (bloków logicznych)

W STEP7 możemy tworzyć programy w trzech postaciach: LAD, STL i FBD. Praktycznie należy wybrać (także w tym rozdziale) którą postacią się posługujemy.

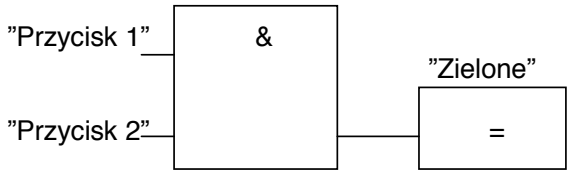
LAD (schemat drabinkowy)



STL (lista instrukcji)

```
A     "Przycisk 1"
A     "Przycisk 2"
=     "Zielone"
```

FBD (bloki logiczne)



Blok jest otwierany w postaci, która zostanie wcześniej ustalona podczas jego tworzenia. Później możemy dokonywać zmian ustawiając jedną z w.w postaci.

Kopiowanie tabeli symboli i otwieranie bloku OB1

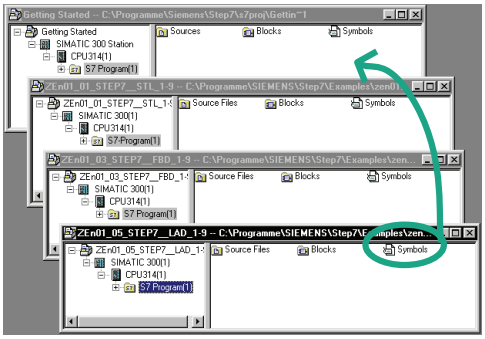
Jeśli to konieczne, otwórz projekt. W tym celu kliknij na przycisk **Open** na listwie narzędziowej, wybierz nazwę projektu i naciśnij **OK.**

W zależności od tego, jaki język został wybrany, otwórz jeden z projektów:

- **ZEn01_05_STEP7__LAD_1-9**
- **ZEn01_01_STEP7__STL_1-9**

or

- **ZEn01_03_STEP7__FDB_1-9**

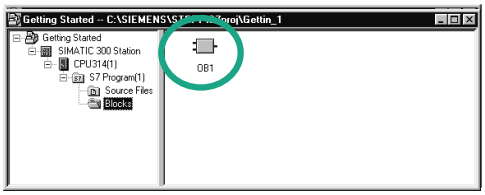


Tutaj możesz zobaczyć wszystkie trzy postaci przykładowego projektu.

Znajdź w przykładowych projektach tabelę symboliczną i skopiuj ją metodą „Drag and Drop" do swojego projektu.

Potem zamknij okno przykładowego projektu.

„Drag and Drop" oznacza, iż zaznaczamy interesujący nas obiekt i przenosimy go w miejsce docelowe trzymając cały czas klawisz myszy. Jeśli puścimy klawisz myszy, obiekt zostaje umieszczony we wskazanej pozycji.



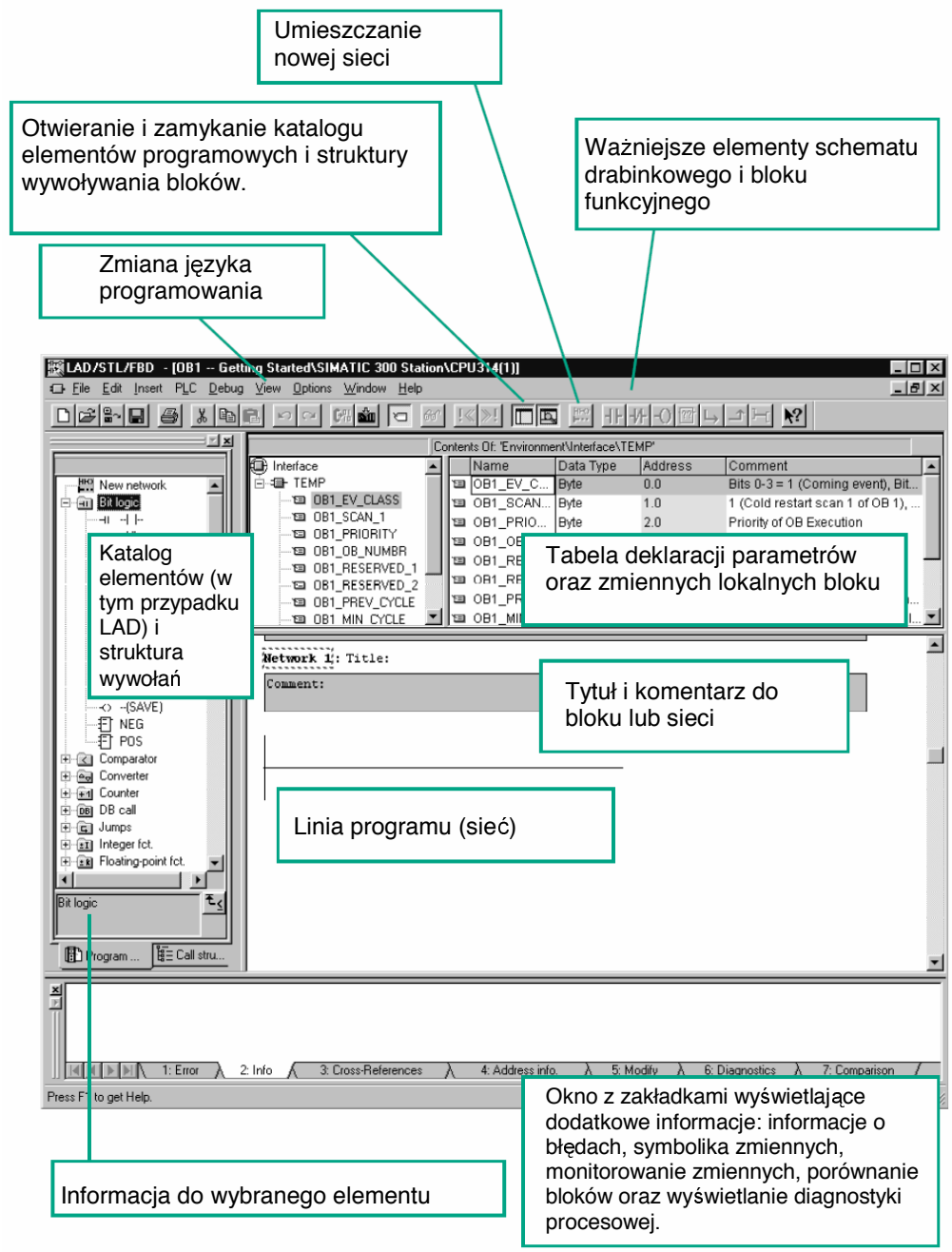
Kliknij podwójnie na blok OB1 w swoim projekcie. Zostaje w ten sposób otwarte okno edytora LAD/STL/FBD.

W sterowniku blok OB1 jest wykonywany cyklicznie. CPU analizuje linię po linii wykonując poszczególne rozkazy programu. Gdy CPU powraca do pierwszej linii, zakończony zostaje jeden cykl. Czas przetwarzania takiego kawałka programu nazywa się czasem obiegu pętli, lub inaczej czasem cyklu.

W zależności od tego, jaka postać STEP7 została wybrana, kontynuuj pracę czytając rozdział 4.2 – LAD, 4.3- STL lub 4.4- FBD.

Okno programowania w LAD/STL/FBD

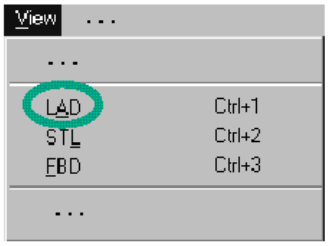
Wszystkie bloki programowe tworzone są w oknie LAD/STL/FBD. Poniżej widoczne jest okno w postaci LAD.



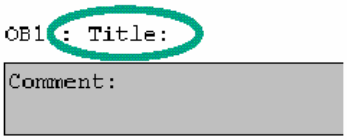
4.2 Programowanie OB1 w schemacie drabinkowym

W tym rozdziale przedstawione jest tworzenie w schemacie drabinkowym połączenia szeregowego, równoległego i funkcji SET/RESET.

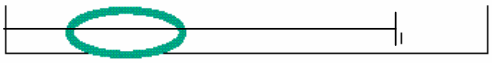
Programowanie połączenia szeregowego w LAD



Jeśli jest to konieczne, należy wybrać język **LAD** w menu **View.**



Kliknij na tytuł bloku (**Title:**) i wprowadź tekst np. *„Program główny wykonywany cyklicznie”.*



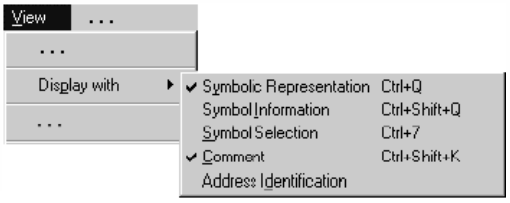
Zaznacz miejsce do umiejscowienia pierwszego styku.

Kliknij na ikonę styku na pasku narzędziowym.

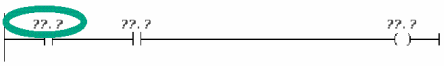
W ten sam sposób dodaj drugi styk.

Umieść cewkę po prawej stronie linii.

Aktualnie znajduje się kompletna linia logiczna, lecz brakuje adresów nad stykami oraz cewką.



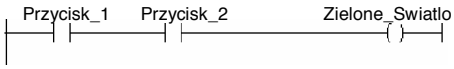
Sprawdź, czy aktywna jest firma wyświetlania adresów jako reprezentacja symboliczna.



Kliknij **??.?** i wprowadź nazwę symboliczną „Przycisk_1”. Zatwierdź Enterem.

Wprowadź dla drugiego styku nazwę „Przycisk_2”.

Dla cewki wprowadź nazwę „Zielone_Swiatlo”.



W tym momencie zostało zakończone tworzenie równania szeregowego.

Jeśli nie ma żadnych czerwonych symboli, zapisz blok.

Symbole będą zaznaczane na czerwono, jeśli np. nie będą zdefiniowane w liście symbolicznej, lub wystąpił błąd składni.

Można także wprowadzić bezpośrednio nazwę symboliczną. Kliknij na znak **??.?** a następnie wybierz w menu **Insert-> Symbol.** Znajdź w liście odpowiedni symbol. Nazwa dodawana jest automatycznie.

Programowanie połączenia równoległego w języku LAD

Wybierz sieć **Network 1**

Wprowadź nową sieć.

Zaznacz aktualną sieć.

Kliknij klawisz styku i cewki na listwie narzędziowej.

Wybierz linię prostopadłą do danej sieci.

Dodaj rozgałęzienie równoległe.

Dodaj drugi styk normalnie rozwarty do rozgałęzienia.

Zamknij rozgałęzienie (jeśli potrzebne, to wybierz dolną strzałkę)

Kompletna linia logiczna bez wypełnionych adresów.

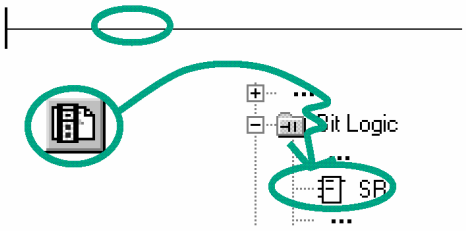
By dodać adresy symboliczne, należy powtórzyć procedurę z połączenia szeregowego.

Dodaj oznaczenia dla styków „Przycisk_3” i „Przycisk_4”, a dla cewki „czerwone_swiatlo”.

Zapisz blok.

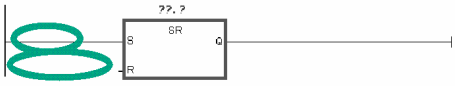
Programowanie przerzutnika SR

Wybierz **Network 2** i dodaj kolejną sieć

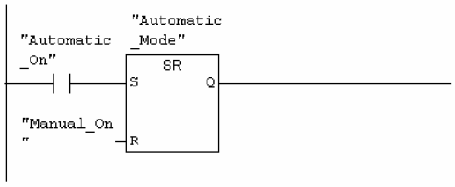


Zaznacz aktualną sieć..

Wyszukaj w katalogu w **Bit Logic** element **SR.**



Dodaj styki przed wejściem S i R.



Wprowadź nazwy symboliczne funkcji SR.

Zapisz blok.

Aby zobaczyć różnicę pomiędzy adresowaniem symbolicznym i absolutnym, należy wyłączyć reprezentację symboliczną w **View>Display>Symbolic Representation.**

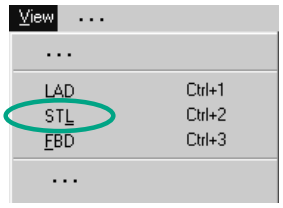
Możesz także zmienić szerokość styku wybierając w menu **Options>Customize** w pozycji „Width of adress field" w zakładce „LAD/FBD". Tutaj możesz ustawić szerokość 10 do 24 znaków.

Więcej informacji znajdziecie w **Help>Contents** w pozycjach „Programming Blocks"-programowanie bloków, „Creating Logic Blocks" - tworzenie bloków, i „Editing Ladder Instructions" – edycja instrukcji w schemacie drabinkowym.

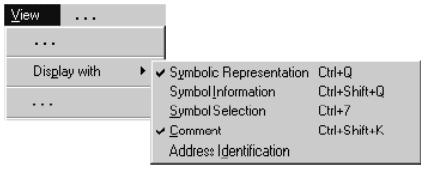
4.3 Programowanie OB1 w liście instrukcji (STL)

W tym rozdziale przedstawione jest tworzenie w liście instrukcji rozkazów AND, OR i instrukcji S – set, R – reset.

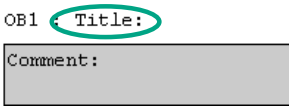
Programowanie instrukcji AND w postaci STL



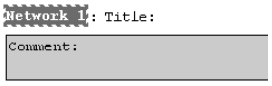
Jeśli jest to konieczne, należy wybrać postać **STL** w menu w pozycji **View.**



Sprawdź, czy jest aktywna reprezentacja symboliczna.



Kliknij na tytuł bloku (**Title:**) i wprowadź tekst *„Program główny wykonywany cyklicznie”.*



Zaznacz miejsce włożenia pierwszej instrukcji.

W pierwszej kolejności wpisz A (AND), następnie spację a potem nazwę symboliczną „Przycisk_1”.

Zakończ linię za pomocą przycisku **Enter.** Kursor przeskakuje do następnej linii

A „Przycisk_1”
A „Przycisk_2”
= „Zielone_swiatlo”

W ten sam sposób uzupełnij dalsze instrukcje.

Została stworzona kompletna funkcja AND. Zapisz blok, jeśli żaden z symboli nie jest już podświetlany na czerwono.

Symbole będą zaznaczane na czerwono, jeśli np. nie będą zdefiniowane w liście symbolicznej, lub wystąpi błąd składni.

Można także wprowadzić bezpośrednio nazwę symboliczną. Kliknij na znak **??.?** a następnie wybierz w menu **Insert-> Symbol.** Znajdź w liście odpowiedni symbol. Nazwa dodawana jest automatycznie.

Programowanie instrukcji OR w STL

Wybierz **Network 1.**

Wstaw nową sieć i przejdź do obszaru wprowadzania instrukcji.

O „Przycisk_3”

Wprowadź O (OR), spację, oraz nazwę symboliczną „Przycisk_3” (w taki sam sposób, jak w instrukcji AND)

O „Przycisk_3”
O „Przycisk_4”
= „Czerwone_swiatlo”

Uzupełnij instrukcje i zapisz blok.

Programowanie przerzutnika SR

Wybierz **Network 2** i wstaw nową sieć

A	„Automatyka_ON”	Wprowadź w pierwszej linii instrukcję A z oznaczeniem symbolicznym „Automatyka_ON”
A	„Automatyka_ON”	Uzupełnij pozostałe instrukcje. Zapisz blok.
S	„AUTO_ZAŁ”	
A	„Reka_ON”	
R	„AUTO_ZAŁ”	

Aby zobaczyć różnicę pomiędzy adresowaniem symbolicznym i absolutnym, należy wyłączyć reprezentację symboliczną w **View>Display>Symbolic Representation.**

Przykład: Adresowanie symboliczne w STL

```
A    „Przycisk_1”
A    „Przycisk_2”
=    „Zielone_Swiatlo”
```

Przykład: Adresowanie absolutne w STL

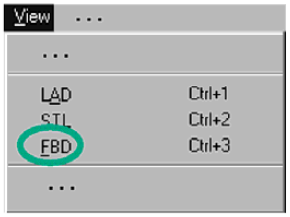
```
A    I    0.1
A    I    0.2
=    Q    4.0
```

Więcej informacji znajdziecie w **Help>Contents** w pozycjach „Programming Blocks”-programowanie bloków, „Creating Logic Blocks” - tworzenie bloków, i „Editing STLInstructions” – edycja instrukcji w liście instrukcji.

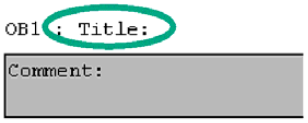
4.4 Programowanie OB1 w blokach logicznych (FBD)

W tym rozdziale przedstawione jest tworzenie w blokach logicznych rozkazów AND, OR i instrukcji S – set, R – reset.

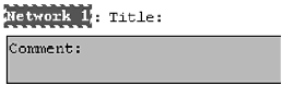
Programowanie funkcji AND w blokach logicznych



Jeśli jest to konieczne, należy wybrać postać **FBD** w menu w pozycji **View.**

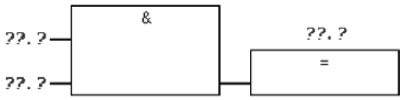


Kliknij na tytuł bloku (**Title:**) i wprowadź tekst *„Program główny wykonywany cyklicznie”.*

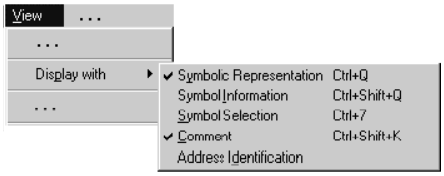


Zaznacz miejsce włożenia instrukcji AND.

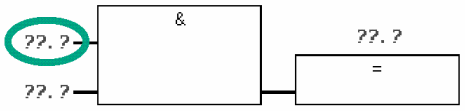
Wprowadź blok AND (&) i (=).



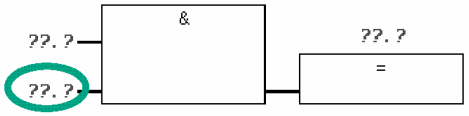
Adresy w stworzonej funkcji nadal pozostają nieokreślone.



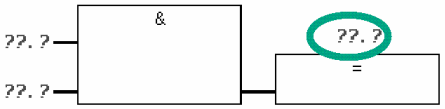
Sprawdź, czy aktywna jest reprezentacja symboliczna.



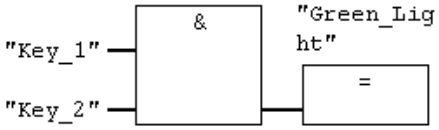
Kliknij **??.?** i wprowadź nazwę symboliczną „Przycisk_1”. Zatwierdź klawiszem **Enter.**



Wprowadź nazwę symboliczną „Przycisk_2” dla drugiego wejścia.



Wprowadź nazwę „Zielone_Swiatlo” dla instrukcji przypisania.
W ten sposób uzyskujesz kompletnie stworzoną funkcję AND.

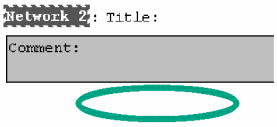


Jeśli nie ma żadnych symboli wskazywanych na czerwono, możesz zapisać blok.

Symbole będą zaznaczane na czerwono, jeśli np. nie będą zdefiniowane w liście symbolicznej, lub wystąpi błąd składni.

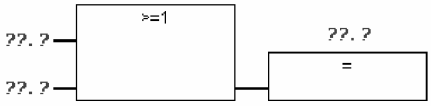
Programowanie funkcji OR w blokach logicznych

Wprowadź nową sieć

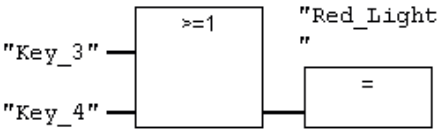


Zaznacz miejsce do wstawienia funkcji OR.

Kliknij ikonkę funkcji OR (≥1) i przepisanie (=).



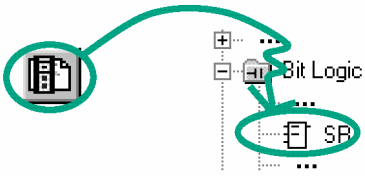
Dodaj oznaczenia dla styków „Przycisk_3” i „Przycisk_4”, a dla funkcji przypisania „czerwone_swiatlo”.



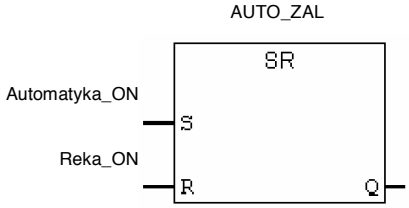
Zapisz blok.

Programowanie funkcji zapamiętywania w FBD

Wybierz **Network 2** i dodaj kolejną sieć. Zaznacz aktualną sieć.



Wyszukaj w katalogu w **Bit Logic** element **SR.** Kliknij na niego dwukrotnie.



Wprowadź nazwy symboliczne funkcji SR.

Zapisz blok.

Aby zobaczyć różnicę pomiędzy adresowaniem symbolicznym i absolutnym, należy wyłączyć reprezentację symboliczną w **View>Display>Symbolic Representation.**

Możesz także zmienić szerokość styku wybierając w menu **Options>Customize** w pozycji „Address Field Width” w zakładce „LAD/FBD”. Tutaj możesz ustawić szerokość 10 do 26 znaków.

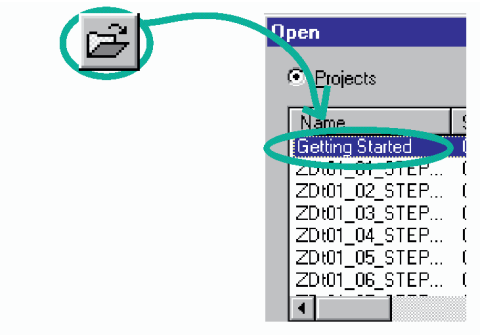
Więcej informacji znajdziecie w **Help>Contents** w pozycjach „Programming Blocks”-programowanie bloków, „Creating Logic Blocks” - tworzenie bloków, i „Editing FBD Statements” – edycja instrukcji w schemacie drabinkowym.

5 Tworzenie programu z wykorzystaniem bloków funkcyjnych i bloków danych

5.1 Dodawanie oraz otwarcie bloku funkcyjnego (FB)

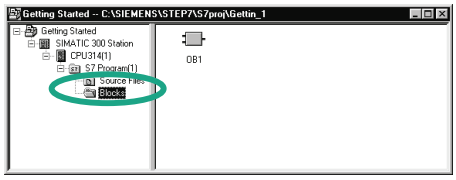
Blok funkcyjny FB hierarchicznie jest przyporządkowany do bloku organizacyjnego. Wszystkie parametry formalne oraz zmienne statyczne zapamiętywane są w specjalnym dołączonym bloku danych (DB).

W oknie edytora LAD/STL/FBD tworzymy blok funkcyjny (FB1 o nazwie symbolicznej np. „Engine"; patrz tabela symboliczna, rozdz.3). By to wykonać, należy posługiwać się informacjami zawartymi w rozdziale 4 (programowanie bloku OB1).



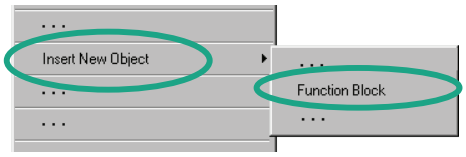
Istnieje możliwość skopiowania oryginalnej tabeli symbolicznej z projektu „Getting Started". Sposób wykonania tej operacji jest opisany w rozdz.4.

Jeśli jest to konieczne, otwórz projekt „Getting Started".

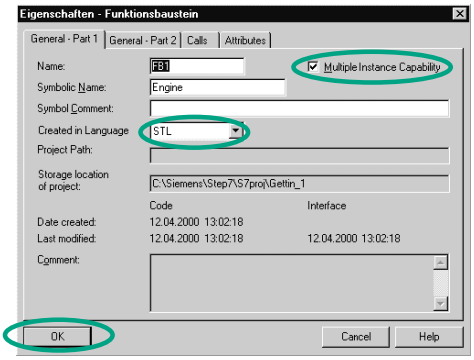


Znajdź i otwórz folder **Blocks** .

Kliknij na prawej części okna prawym klawiszem myszy.

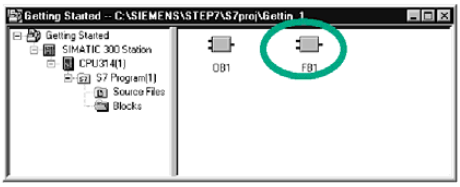


Pojawiło się kontekstowe menu zawierające ważniejsze operacje zawarte w listwie menu. Wstaw jako nowy obiekt **blok funkcyjny: „function block".**



Podwójne kliknięcie na blok FB1 powoduje otwarcie go w edytorze LAD/STL/FBD.

W okienku „Properties-Function Block” wybierz sposób prezentacji programu, w jakim chcesz edytować blok, oraz aktywuj opcję **„Multiple instance FB”.** Na końcu zatwierdź **OK.**



Blok **FB1** pojawi się w folderze bloków.

W zależności od tego, który język programowania wybrałeś, kontynuuj dalsze działania zgodnie z rozdziałem: 5.2 – schemat drabinkowy LAD, 5.3 – lista instrukcji STL lub 5.4 - bloki logiczne FBD.

Dalsze informacje znajdują się w **Help>Contents** pod hasłem „Programming Blocks” i „Creating Blocks and Libraries”.

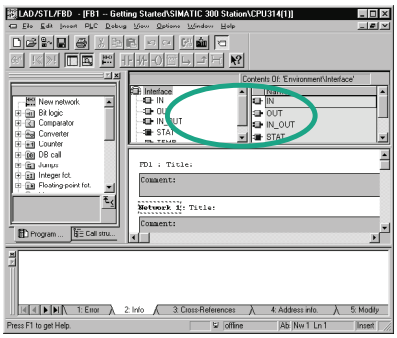
5.2 Programowanie FB1 w schemacie drabinkowym

Stworzymy blok funkcyjny sterowania i monitorowania stanu paliwa silnika dieslowskiego z użyciem dwóch różnych bloków danych.

Wszystkie sygnały charakterystyczne dla silnika przekazywane są jako parametry bloku funkcyjnego, dlatego też należy je zdefiniować w części deklaracyjnej bloku jako parametry wejściowe lub wyjściowe.

Powinieneś także do tej pory opanować wprowadzanie połączenia szeregowego, równoległego i funkcji pamiętania.

1. Wypełnianie tabeli deklaracji parametrów



Otwieramy edytor programu i w menu **View** wybieramy sposób prezentacji w schemacie drabinkowym **LAD.**

W nagłówku znajduje się oznaczenie bloku FB1, który został otwarty do edycji.

Obszar deklaracji zmiennych zawiera okno przeglądu zmiennych (lewa część) i okno dokładnych informacji (prawa część).

W oknie przeglądu zmiennych należy kolejno wybrać typ ”IN”, ”OUT” i ”STAT”, a następnie zdefiniować zmienne w oknie szczegółowym.
Zdefiniuj zmienne według rysunków umieszczonych poniżej. Typy zmiennych są definiowane za pomocą rozwijanej listy.

Contents Of: 'Environment\Interface\IN'

Name	Data Type	Address	Initial Value	Comment
Switch_On	Bool	0.0	FALSE	Switch on engine
Switch_Off	Bool	0.1	FALSE	Switch off engine
Failure	Bool	0.2	FALSE	Engine failure, causes the engine to switch off
Actual_Speed	Int	2.0	0	Actual engine speed

Contents Of: 'Environment\Interface\OUT'

Name	Data Type	Address	Initial Value	Comment
Engine_On	Bool	4.0	FALSE	Engine is switched on
Preset_Speed_Reached	Bool	4.1	FALSE	Preset speed reached

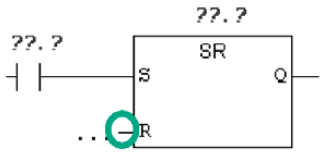
Contents Of: 'Environment\Interface\STAT'

Name	Data Type	Address	Initial Value	Comment
Preset_Speed	Int	6.0	1500	Requested engine speed

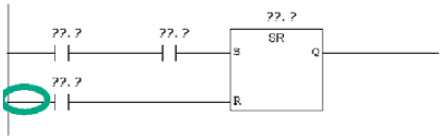
Nazwy zmiennych mogą składać się jedynie z liter, cyfr i podkreśleń.

2. Programowanie załączania i wyłączania silnika

Używając odpowiednich klawiszy z listwy narzędziowej, lub katalogu elementów programowych wprowadź do sieci 1 styk normalnie rozwarty, styk normalnie zwarty oraz element SR.

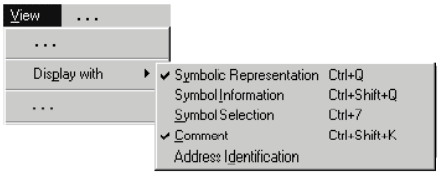


Następnie wybierz wejście R.



Wprowadź kolejny styk normalnie rozwarty na wejście R.

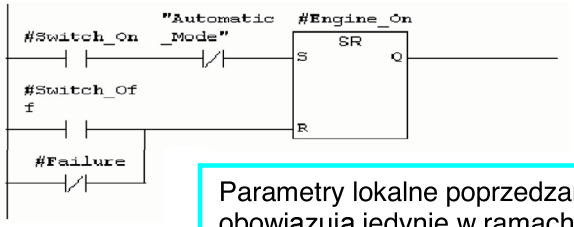
Następnie wprowadź równolegle styk normalnie zwarty używając wymienionych obok klawiszy.



Sprawdź, czy wybrana jest reprezentacja symboliczna.

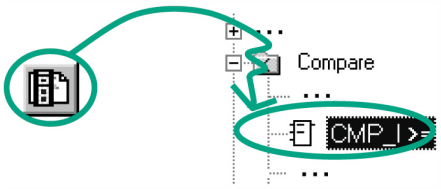
Wybierz pole ze znakami zapytania i wprowadź odpowiednią nazwę z tabeli deklaracji (znak # wprowadzany jest automatycznie).

Wprowadź nazwę „Tryb automatyczny” („Automatic Mode”) nad stykiem normalnie zwartym w połączeniu szeregowym.



Parametry lokalne poprzedzane są znakiem # i obowiązują jedynie w ramach danego bloku. Zmienne globalne umieszczone są w cudzysłowach. Nazwy te zdefiniowane są w liście symbolicznej i obowiązują w całym programie.
Stan sygnału „Tryb Automatyczny” („Automatic Mode”) ustalany jest w bloku OB1 (spójrz str.26), a teraz w FB1 tylko odpytywany.

3. Programowanie kontroli prędkości

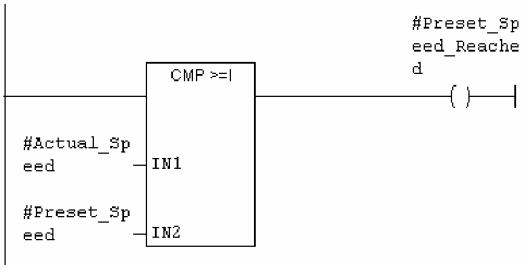


Wprowadź kolejną sieć i zaznacz bieżącą ścieżkę.

Z katalogu elementów wybierz w grupie **„Compare”** instrukcję **CMP>=I.**

Do bieżącej sieci dodaj także cewkę.

Zaznacz ponownie znaki zapytania i wprowadź etykiety dla cewki oraz wejść funkcji porównania używając nazw z tabeli deklaracji.



Kiedy silnik jest załączany i wyłączany?
Jeśli zmienna #Switch On ma wartość 1 i sygnał „Automatic Mode” ma wartość 0, silnik jest włączany. Negacja sygnału „Automatic Mode” (styk normalnie zwarty) zapewnia takie działanie.
Jeśli sygnał #Switch Off jest ustawiony na 1 lub sygnał błędu #Failure jest 0, silnik jest wyłączany. Takie funkcjonowanie zapewnione jest również dzięki negacji. W normalnej sytuacji sygnał #Failure ma stan 1, natomiast w sytuacji awaryjnej przyjmuje stan 0.

Jak kontrolowane są obroty silnika poprzez funkcję porównania?
Porównywane są dwie wartości: aktualna #Actual_Speed, oraz zadana #Preset_Speed. Wynik porównania przekazywany jest do bitu potwierdzenia poprawności pracy #Preset_Speed_Reached (stan sygnału 1).

Dalsze informacje znajdują się w **Help>Contents** w temacie ”Programming Blocks”, „Creating Logic Blocks” oraz „Editing the Variable Declaration Table” lub też „Editing LAD Instruction”.

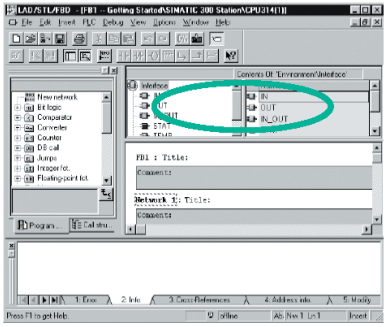
5.3 Programowanie FB1 w liście instrukcji

Stworzymy blok funkcyjny sterowania i monitorowania stanu paliwa silnika dieslowskiego z użyciem dwóch różnych bloków danych.

Wszystkie sygnały charakterystyczne dla silnika przekazywane są jako parametry bloku funkcyjnego, dlatego też należy je zdefiniować w części deklaracyjnej bloku jako parametry wejściowe lub wyjściowe.

Powinieneś wiedzieć, jak w STEP7 wprowadzić instrukcję AND, lub OR, czy też set/reset.

1. Wypełnianie tabeli deklaracji parametrów



Otwieramy edytor programu i w menu **View** wybieramy sposób prezentacji w liście instrukcji **STL.**

W nagłówku znajduje się oznaczenie bloku FB1, który został otwarty do edycji.

Wprowadź deklaracje w tabeli. Należy wzorując się na poniższych tabelach, w odpowiednie komórki wprowadzić nazwę parametru, jego format, a także komentarz do parametru. Typ parametru można wprowadzić wpisując jego nazwę, lub też uruchomić okienko typów prawym klawiszem myszy, którym możemy wybrać typy podstawowe: „**Elementary Types”.** Jeśli zadeklarujesz już wszystkie pola dotyczące zmiennej, zatwierdź to klawiszem **ENTER.**

Contents Of: 'Environment\Interface\IN'

Name	Data Type	Address	Initial Value	Comment
Switch_On	Bool	0.0	FALSE	Switch on engine
Switch_Off	Bool	0.1	FALSE	Switch off engine
Failure	Bool	0.2	FALSE	Engine failure, causes the engine to switch off
Actual_Speed	Int	2.0	0	Actual engine speed

Contents Of: 'Environment\Interface\OUT'

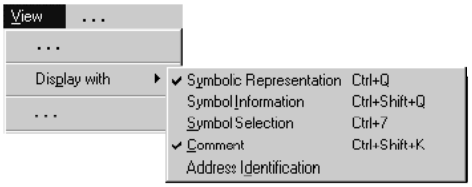
Name	Data Type	Address	Initial Value	Comment
Engine_On	Bool	4.0	FALSE	Engine is switched on
Preset_Speed_Reached	Bool	4.1	FALSE	Preset speed reached

Contents Of: 'Environment\Interface\STAT'

Name	Data Type	Address	Initial Value	Comment
Preset_Speed	Int	6.0	1500	Requested engine speed

Nazwy zmiennych mogą składać się jedynie z liter, cyfr i podkreśleń.

2. Programowanie załączania i wyłączania silnika



Sprawdź, czy wybrana jest reprezentacja symboliczna.

```
A     #Switch_On
AN    "Automatic_Mode"
S     #Engine_On
O     #Switch_Off
ON    #Failure
R     #Engine_On
```

Wprowadź instrukcje programu do sieci 1.

Parametry lokalne poprzedzane są znakiem # i obowiązują jedynie w ramach danego bloku.
Zmienne globalne umieszczone są w cudzysłowach. Nazwy te zdefiniowane są w liście symbolicznej i obowiązują w całym programie.
Stan sygnału „Tryb Automatyczny" („Automatic Mode") ustalany jest w bloku OB1 (patrz str.29), a teraz w FB1 tylko odpytywany.

3. Programowanie kontroli prędkości

```
L     #Actual_Speed
L     #Preset_Speed
>=I
=     #Preset_Speed_Reached
```

Wprowadź nową sieć oraz instrukcje programu. Następnie zapisz stworzony blok.

Kiedy silnik jest załączany i wyłączany?
Jeśli zmienna #Switch On ma wartość 1 i sygnał „Automatic Mode" ma wartość 0, silnik jest włączany. Negacja sygnału „Automatic Mode" (styk normalnie zwarty) zapewnia takie działanie.
Jeśli sygnał #Switch Off jest ustawiony na 1 lub sygnał błędu #Failure jest 0, silnik jest wyłączany. Takie funkcjonowanie zapewnione jest również dzięki negacji. W normalnej sytuacji sygnał #Failure na stan 1, natomiast w sytuacji awaryjnej przyjmuje stan 0.

Jak kontrolowane są obroty silnika poprzez funkcję porównania?
Porównywane są dwie wartości: aktualna #Actual_Speed, oraz zadana #Preset_Speed. Wynik porównania przekazywany jest do bitu potwierdzenia poprawności pracy #Preset_Speed_Reached (stan sygnału 1).

Dalsze informacje znajdują się w **Help>Contents** w temacie"Programming Blocks", „Creating Logic Blocks" oraz „Editing the Variable Declaration Table" lub też „Editing STL Instruction".

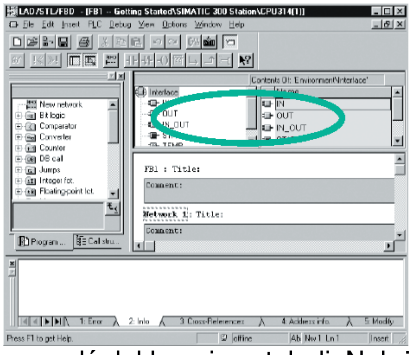
5.4 Programowanie FB1 w blokach logicznych

Stworzymy blok funkcyjny sterowania i monitorowania stanu paliwa silnika dieslowskiego z użyciem dwóch różnych bloków danych.

Wszystkie sygnały charakterystyczne dla silnika przekazywane są jako parametry bloku funkcyjnego, dlatego też należy je zdefiniować w części deklaracyjnej bloku jako parametry wejściowe lub wyjściowe.

Powinieneś wiedzieć, jak w STEP7 wprowadzić funkcję AND, lub OR, czy też funkcję pamiętania.

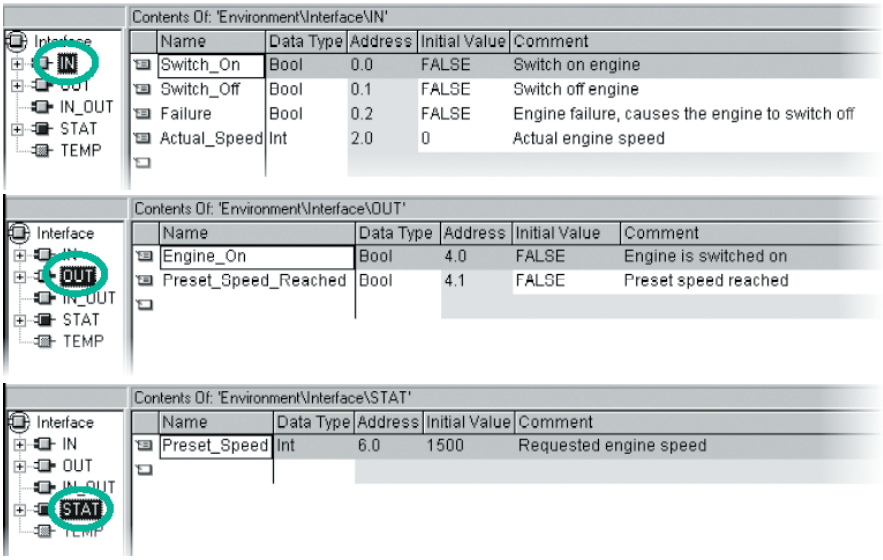
1. Wypełnianie tabeli deklaracji parametrów



Otwieramy edytor programu i w menu **View** wybieramy sposób prezentacji w liście instrukcji **FBD.**

W nagłówku znajduje się oznaczenie bloku FB1, który został otwarty do edycji.

Wprowadź deklaracje w tabeli. Należy wzorując się na poniższej tabeli, w odpowiednie komórki wprowadzić nazwę parametru, jego format, a także komentarz do parametru. Typ parametru można wprowadzić wpisując jego nazwę, lub też uruchomić okienko typów prawym klawiszem myszy, którym możemy wybrać typy podstawowe: „**Elementary Types”.** Jeśli zadeklarujesz już wszystkie pola dotyczące zmiennej, zatwierdź to klawiszem **ENTER.**



Contents Of: 'Environment\Interface\IN'

Name	Data Type	Address	Initial Value	Comment
Switch_On	Bool	0.0	FALSE	Switch on engine
Switch_Off	Bool	0.1	FALSE	Switch off engine
Failure	Bool	0.2	FALSE	Engine failure, causes the engine to switch off
Actual_Speed	Int	2.0	0	Actual engine speed

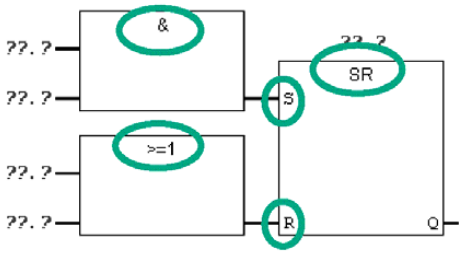
Contents Of: 'Environment\Interface\OUT'

Name	Data Type	Address	Initial Value	Comment
Engine_On	Bool	4.0	FALSE	Engine is switched on
Preset_Speed_Reached	Bool	4.1	FALSE	Preset speed reached

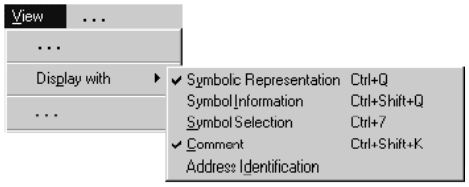
Contents Of: 'Environment\Interface\STAT'

Name	Data Type	Address	Initial Value	Comment
Preset_Speed	Int	6.0	1500	Requested engine speed

2. Programowanie załączania i wyłączania silnika



Wprowadź funkcje SR do sieci 1 używając katalogu funkcji (folder **Bit Logic**).
Dodaj funkcję AND na wejście S, oraz funkcję OR na wejście R.



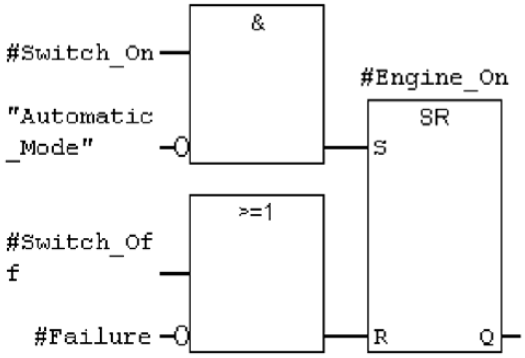
Sprawdź, czy wybrana jest reprezentacja symboliczna.

Wybierz pole ze znakami zapytania i wprowadź odpowiednią nazwę z tabeli deklaracji (znak # wprowadzany jest automatycznie).

Jedno z wejść funkcji AND należy zadeklarować jako „Automatic_Mode".

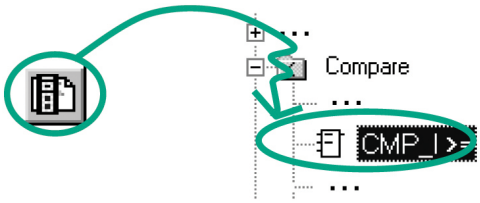
Wykorzystując odpowiedni klawisz listwy narzędziowej zaneguj wejścia „Automatic_Mode" oraz #Fault.

Następnie zabezpiecz blok.



Parametry lokalne poprzedzane są znakiem # i obowiązują jedynie w ramach danego bloku.
Zmienne globalne umieszczone są w cudzysłowach. Nazwy te zdefiniowane są w liście symbolicznej i obowiązują w całym programie.
Stan sygnału „Tryb Automatyczny" („Automatic Mode") ustalany jest w bloku OB1 (patrz str.33), a teraz w FB1 tylko odpytywany.

3. Programowanie kontroli prędkości

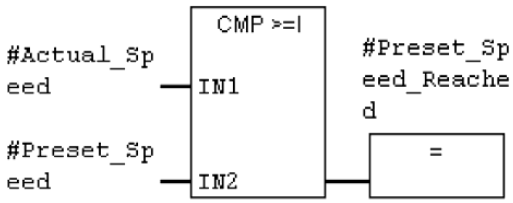


Wprowadź kolejną sieć i zaznacz bieżącą ścieżkę.

Z katalogu elementów wybierz w grupie **„Compare”** instrukcję **CMP>=I.**

Dodaj funkcję przepisania wyniku, a następnie oznacz symbolami wejścia i wyjście funkcji.

Następnie zapisz blok.



Kiedy silnik jest załączany i wyłączany?
Jeśli zmienna #Switch On ma wartość 1 i sygnał „Automatic Mode” ma wartość 0, silnik jest włączany. Negacja sygnału „Automatic Mode” (styk normalnie zwarty) zapewnia takie działanie.
Jeśli sygnał #Switch Off jest ustawiony na 1 lub sygnał błędu #Failure jest 0, silnik jest wyłączany. Takie funkcjonowanie zapewnione jest również dzięki negacji. W normalnej sytuacji sygnał #Failure na stan 1, natomiast w sytuacji awaryjnej przyjmuje stan 0.

Jak kontrolowane są obroty silnika poprzez funkcję porównania?
Porównywane są dwie wartości: aktualna #Actual_Speed, oraz zadana #Preset_Speed. Wynik porównania przekazywany jest do bitu potwierdzenia poprawności pracy #Preset_Speed_Reached (stan sygnału 1).

Dalsze informacje znajdują się w **Help>Contents** w temacie”Programming Blocks”, „Creating Logic Blocks” oraz „Editing the Variable Declaration Table” lub też „Editing LAD Instruction”.

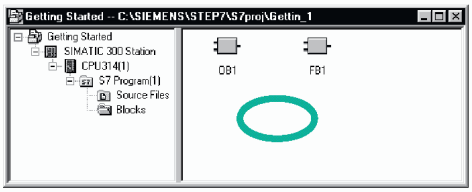
5.5 Tworzenie lokalnego bloku danych oraz aktualizacja wartości

Przed chwilą został zadeklarowany blok FB1. Zdefiniowaliśmy także w tabeli deklaracji jego parametry.

By móc potem wywołać ten zdefiniowany blok w bloku OB1, należy wcześniej stworzyć przypisany do niego blok danych. Tego typu blok nazywany będzie blokiem danych lokalnych i używany w powiązaniu z blokiem funkcyjnym, dla którego został zdefiniowany.

Blok funkcyjny steruje i nadzoruje pracą silnika. Różne wartości prędkości zadanej są zapamiętane w dwóch różnych blokach danych, w których można je aktualizować.

Dzięki temu, że blok funkcyjny programowany jest tylko jeden raz, oszczędzamy pamięć sterownika.

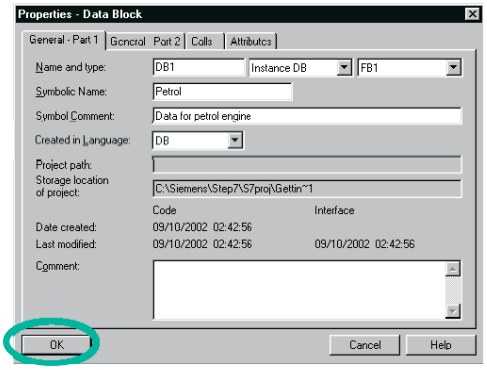


W SIMATIC Manager otwarty jest nasz projekt (np."Getting Started").

Należy wejść w folder **Blocks** i kliknąć prawym klawiszem myszy na prawej części okna.

Teraz można wprowadzić nowy blok wybierając z menu **Data Block.**

Wprowadzić DB1 oraz wybrać „Instance DB" do bloku funkcyjnego FB1

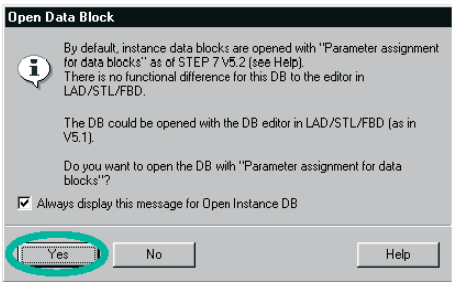


Należy zatwierdzić wszystkie ustawienia w okienku **Properties** za pomocą klawisza **OK.**

Blok danych DB1 dodawany jest do naszego projektu.

Kliknij podwójnie by otworzyć DB1,

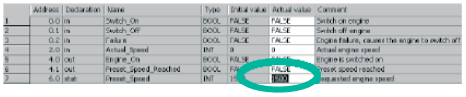
Należy zatwierdzić okno dialogowe przypisania parametrów lokalnego bloku danych.

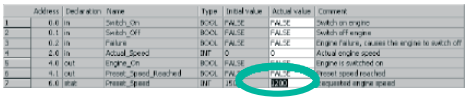


Następnie w kolumnie wartości aktualnych „Actual Value” w wierszu prędkości zadanej („Setpoint Speed”) wprowadźmy wartość 1500. W ten sposób zdefiniowaliśmy maksymalną prędkość tego silnika.

Teraz zapamiętujemy blok i zamykamy okno edytora.

W ten sam sposób definiujemy blok DB2 („Petrol”) wprowadzając jako wartość zadaną prędkości liczbę 1200.





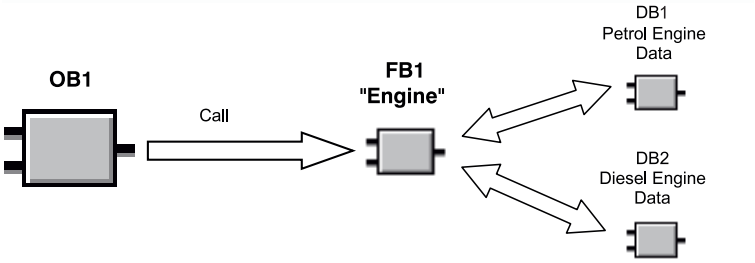
W tym momencie zakończyliśmy definiowanie bloków potrzebnych do sterowania dwoma silnikami za pomocą jednego bloku funkcyjnego. By dodać kolejne silniki wystarczy po prostu zdefiniować kolejne bloki danych.

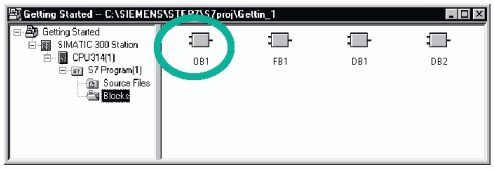
Następnym naszym krokiem będzie wywołanie bloku FB1 w bloku głównym OB1. By wykonać to w schemacie drabinkowym przeczytaj rozdział 5.6, w liście instrukcji rozdział 5.7, a w schemacie bloków logicznych rozdział 5.8.

Więcej infomacji znajduje się w **Help>Contents** w tematach „Programming Blocks” oraz „Creating Data Blocks”.

5.6 Programowanie wywołania bloku w schemacie drabinkowym

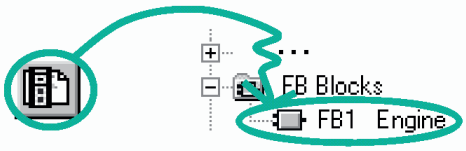
Wszystko, co do tej pory zostało stworzone, będzie bezużyteczne, dopóki nie wywołamy bloku FB1 w OB1. Blok FB1 wywoływany jest dwa razy, każdorazowo z innym blokiem danych.



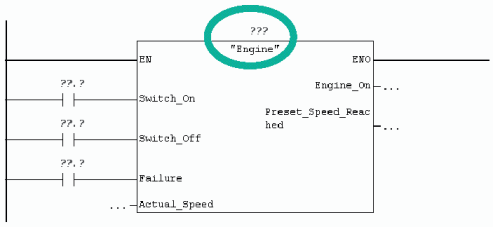


W SIMATIC Manager otwarty jest nasz projekt (np. „Getting Started”).

Należy klikając dwukrotnie otworzyć blok OB1.

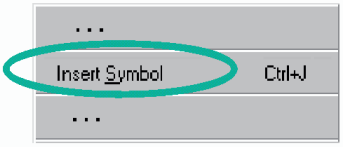


Należy stworzyć sieć 4, a następnie poruszając się po katalogu elementów w folderze **FB Blocks** znaleźć blok FB1 i dołożyć go do OB1.

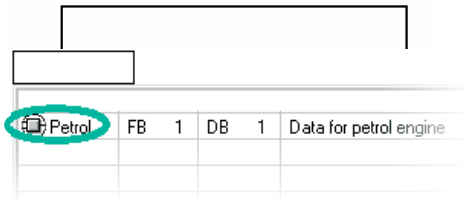


W miejsca wejść funkcji należy wprowadzić styki normalnie rozwarte.

Należy następnie zaznaczyć ??? nad blokiem, a następnie trzymając w tym samym miejscu kursor, nacisnąć prawy klawisz myszy.

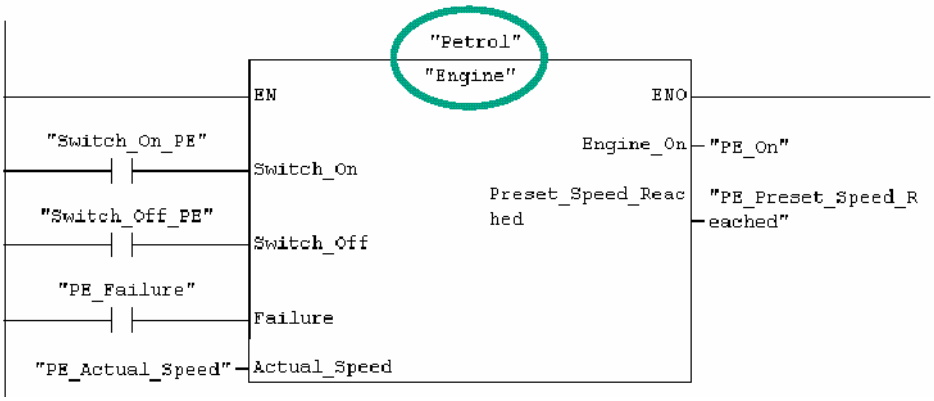


W menu kontekstowym należy wybrać **Insert Symbol**. Pojawi się lista elementów symbolicznych dostępnych w programie. Czynność ta wykonywana po raz pierwszy może być nieco czasochłonna.



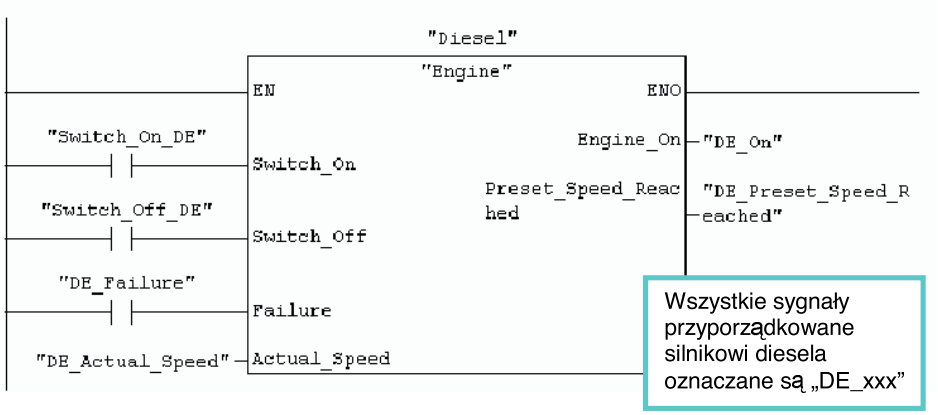
Wybierając blok „Petrol” spowodujemy automatycznie powiązanie tego wywołania bloku funkcyjnego FB1 z blokiem danych dla silnika benzynowego.

Wstaw adresy symboliczne dla wszystkich parametrów użytych dla tej funkcji poruszając się po dostępnej liście symbolicznej.



Wejścia i wyjścia funkcji wyświetlane są teraz jako jej parametry.
Zostaną im przypisane sygnały charakterystyczne dla silnika benzynowego „PE_xxx”.

Należy zaprogramować w kolejnej sieci wywołanie bloku FB1 z użyciem bloku danych „Diesel” DB2, oraz nadać adresy używając listy rozwijalnej.



Zapamiętaj program i zamknij blok.

Jeśli tworzymy program, w którym wykorzystywane są bloki organizacyjne, funkcyjne czy bloki danych, powinniśmy stworzyć wywołanie bloku podporządkowanego w bloku wyższym. Ta zasada obowiązuje we wszystkich tego typu przypadkach.

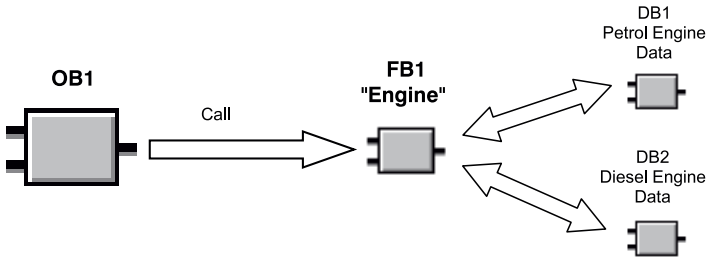
W tablicy symbolicznej nazywać możemy także poszczególne bloki programu.

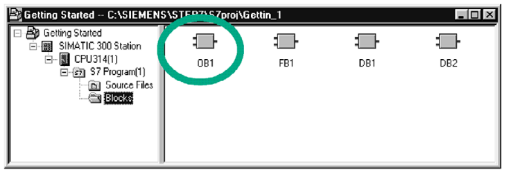
Jednocześnie możliwy jest wydruk i archiwizacja stworzonych bloków. Te możliwości znajdują się w menu **File>Print** lub **File>Archive.**

Więcej infomacji znajduje się w **Help>Contents** w tematach „Calling Reference Helps” oraz „Language Description: LAD” i „Program Control Instruction”.

5.7 Programowanie wywołania bloku w liście instrukcji

Wszystko, co do tej pory zostało stworzone, będzie bezużyteczne, dopóki nie wywołamy bloku FB1 w OB1. Blok FB1 wywoływany jest dwa razy, każdorazowo z innym blokiem danych.





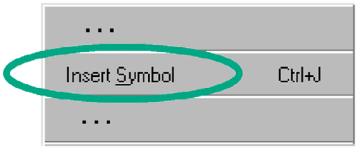
W SIMATIC Manager otwarty jest nasz projekt (np. „Getting Started”). Należy klikając dwukrotnie otworzyć blok OB1.

Należy stworzyć sieć 4.

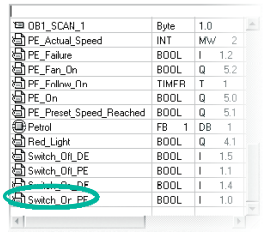
```
CALL  "Engine" , "Petrol"
  Switch_On            :=
  Switch_Off           :=
  Failure              :=
  Actual_Speed         :=
  Engine_On            :=
  Preset_Speed_Reached:=
```

Wprowadź instrukcję **CALL „Engine”, „Petrol”.**

Wyświetlają się wszystkie parametry użytego bloku funkcyjnego. Najedź na pozycję = parametru Switch_On, i naciśnij prawy klawisz myszy.



W dodatkowym oknie menu należy wybrać **Insert Symbol**. Pojawi się lista elementów symbolicznych dostępnych w programie. Czynność ta wykonywana po raz pierwszy może być nieco czasochłonna.



OB1_SCAN_1	Byte	1.0	
PE_Actual_Speed	INT	MW	2
PE_Failure	BOOL	I	1.2
PE_Fan_On	BOOL	Q	5.2
PE_Follow_On	TIMER	T	1
PE_On	BOOL	Q	5.0
PE_Preset_Speed_Reached	BOOL	Q	5.1
Petrol	FB 1	DB	1
Red_Light	BOOL	Q	4.1
Switch_Off_DE	BOOL	I	1.5
Switch_Off_PE	BOOL	I	1.1
[illegible]	BOOL	I	1.4
Switch_On_PE	BOOL	I	1.0

Wybierając blok „Petrol” spowodujemy automatycznie powiązanie tego wywołania bloku funkcyjnego FB1 z blokiem danych dla silnika benzynowego.

```
CALL  "Engine" , "Petrol"
 Switch_On            :="Switch_On_PE"
 Switch_Off           :="Switch_Off_PE"
 Failure              :="PE_Failure"
 Actual_Speed         :="PE_Actual_Speed"
 Engine_On            :="PE_On"
 Preset_Speed_Reached:="PE_Preset_Speed_Reached
```

Wstaw adresy symboliczne dla wszystkich parametrów użytych dla tej funkcji poruszając się po dostępnej liście symbolicznej.

Wszystkie sygnały przyporządkowane silnikowi diesla oznaczane są „DE_xxx”

```
CALL  "Engine" , "Diesel"
 Switch_On            :="Switch_On_DE"
 Switch_Off           :="Switch_Off_DE"
 Failure              :="DE_Failure"
 Actual_Speed         :="DE_Actual_Speed"
 Engine_On            :="DE_On"
 Preset_Speed_Reached:="DE_Preset_Speed_Reached"
```

Należy zaprogramować w kolejnej sieci wywołanie bloku FB1 z użyciem bloku danych „Diesel” DB2, oraz nadać adresy używając listy rozwijalnej.

Zapamiętaj program i zamknij blok.

Jeśli tworzymy program, w którym wykorzystywane są bloki organizacyjne, funkcyjne czy bloki danych, powinniśmy stworzyć wywołanie bloku podporządkowanego w bloku wyższym. Ta zasada obowiązuje we wszystkich tego typu przypadkach.

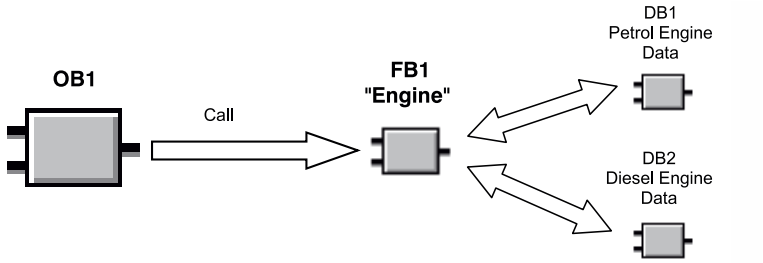
W tablicy symbolicznej nazywać możemy także poszczególne bloki programu.

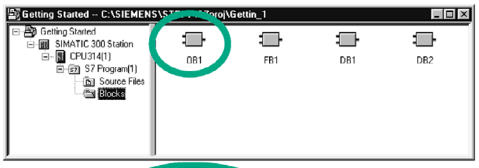
Jednocześnie możliwy jest wydruk i archiwizacja stworzonych bloków. Te możliwości znajdują się w menu **File>Print** lub **File>Archive.**

Więcej infomacji znajduje się w **Help>Contents** w tematach „Calling Reference Helps” oraz „Language Description: STL” i „Program Control Instruction”.

5.8 Programowanie wywołania bloku w schemacie bloków logicznych

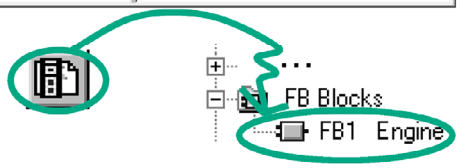
Wszystko, co do tej pory zostało stworzone, będzie bezużyteczne, dopóki nie wywołamy bloku FB1 w OB1. Blok FB1 wywoływany jest dwa razy, każdorazowo z innym blokiem danych.



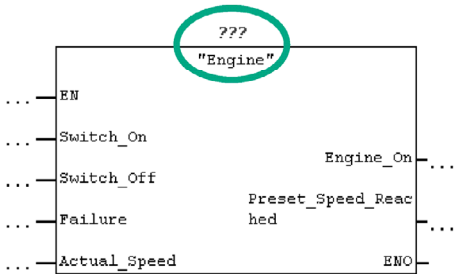


W SIMATIC Manager otwarty jest nasz projekt (np. „Getting Started”).

Należy klikając dwukrotnie otworzyć blok OB1.

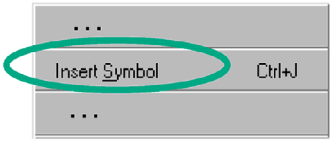


Należy stworzyć sieć 4, a następnie poruszając się po katalogu elementów w folderze **FB Blocks** znaleźć blok FB1 i dodać go do OB1.

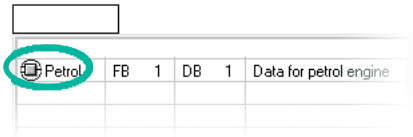


Pokazane są wszystkie parametry wejściowe i wyjściowe.

Należy następnie zaznaczyć ??? nad blokiem, a następnie trzymając w tym samym miejscu kursor, nacisnąć prawy klawisz myszy.

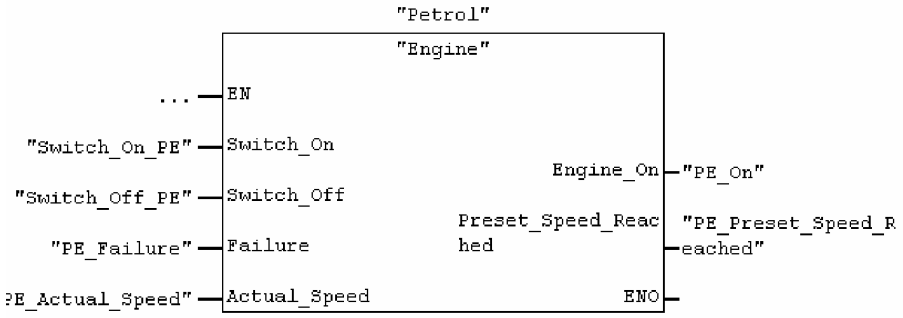


W dodatkowym oknie menu należy wybrać **Insert Symbol**. Pojawi się lista elementów symbolicznych dostępnych w programie. Czynność ta wykonywana po raz pierwszy może być nieco czasochłonna.



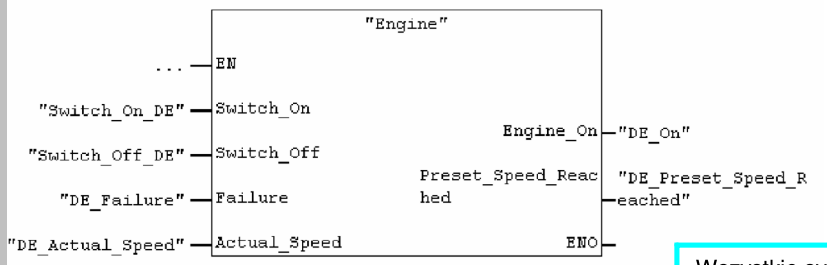
Wybierając blok „Petrol” spowodujemy automatycznie powiązanie tego wywołania bloku funkcyjnego FB1 z blokiem danych dla silnika benzynowego.

Wstaw adresy symboliczne dla wszystkich parametrów użytych dla tej funkcji poruszając się po dostępnej liście symbolicznej.



Wejścia i wyjścia funkcji wyświetlane są teraz jako jej parametry.
Zostaną im przypisane sygnały charakterystyczne dla silnika benzynowego „PE_xxx”.

Należy zaprogramować w kolejnej sieci wywołanie bloku FB1 z użyciem bloku danych „Diesel” DB2, oraz nadać adresy używając listy rozwijalnej.



Wszystkie sygnały przyporządkowane silnikowi diesla oznaczane są „DE_xxx”

Zapamiętaj program i zamknij blok.

Jeśli tworzymy program, w którym wykorzystywane są bloki organizacyjne, funkcyjne czy bloki danych, powinniśmy stworzyć wywołanie bloku podporządkowanego w bloku wyższym. Ta zasada obowiązuje we wszystkich tego typu przypadkach.

W tablicy symbolicznej nazywać możemy także poszczególne bloki programu.

Jednocześnie możliwy jest wydruk i archiwizacja stworzonych bloków. Te możliwości znajdują się w menu **File>Print** lub **File>Archive.**

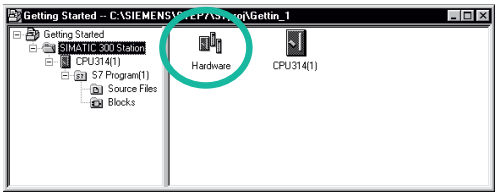
Więcej infomacji znajduje się w **Help>Contents** w tematach „Calling Reference Helps” oraz „Language Description: FBD” i „Program Control Instruction”.

6 Konfiguracja jednostki podstawowej

6.1 Konfiguracja sprzętowa

Przeprowadzenie konfiguracji sprzętowej jest możliwe wtedy, gdy założona została stacja SIMATIC. Struktura projektu została zdefiniowana w rozdziale 2.1.

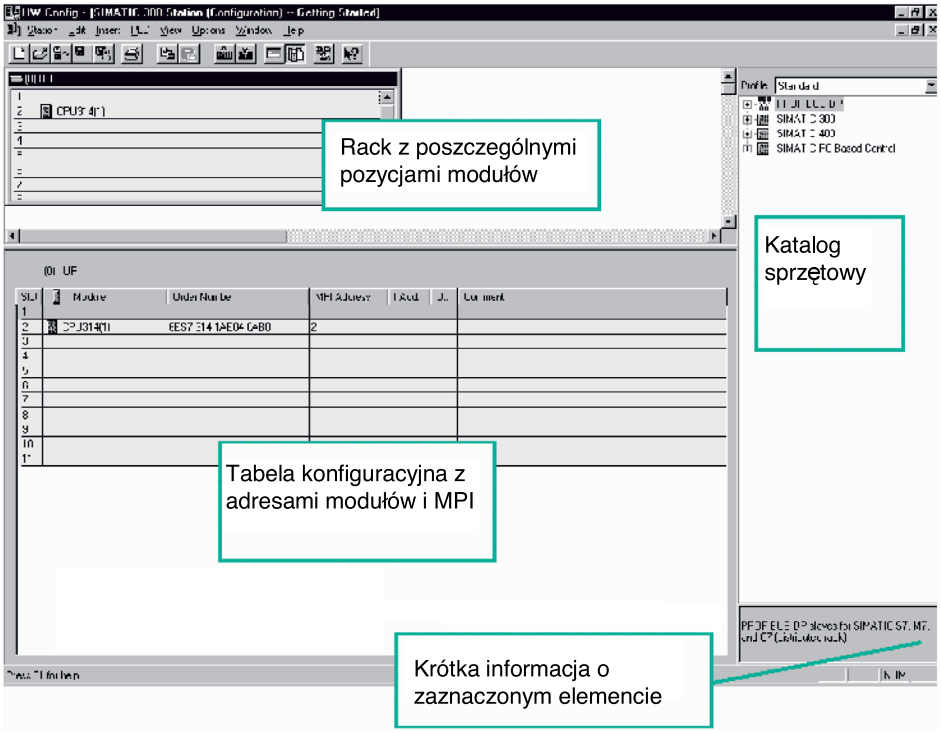
Konfiguracja sprzętowa tworzona jest w STEP7 i przenoszona do sterownika za pomocą funkcji „Download”. (patrz. Rozdz. 7).

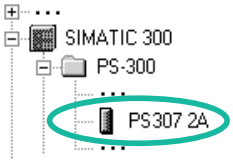


Punktem wyjścia jest otwarty SIMATIC Manager wraz z projektem „Getting Started”.

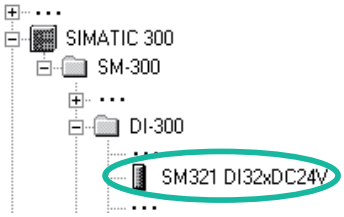
Otwórz katalog **SIMATIC 300-Station** i kliknij dwukrotnie na symbol **Hardware**.

Otworzy się okno ”HW Config”. Ukaże się jednostka centralna wybrana przy zakładaniu projektu. Dla projektu „Getting Started” jest to CPU 314.

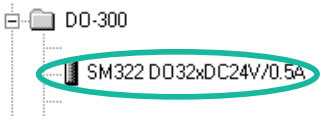




W pierwszej kolejności potrzebne jest zasilanie. Znajdź w katalogu **PS307 2A** i wprowadź metodą Drag and Drop w pozycję 1.



Wyszukaj moduł wejściowy (DI, Digital Input) **SM321 DI32xDC24V** i wprowadź go w pozycję 4. Pozycja 3 pozostaje wolna.



Analogicznie wprowadź na pozycję 5 moduł wyjściowy **SM322 DO32xDC24V/0.5A**.

Aby zmienić parametry modułu w ramach projektu, otwórz je klikając podwójnie na moduł. Parametry mogą być zmieniane jedynie wtedy, gdy jesteś całkowicie pewien, jaki wpływ będą miały te zmiany na sterownik.

Dla projektu „Getting Started” zmiany nie są konieczne.

Slot	Module	Order Number	MPI Address	I Add...	Q...	Comment
1	PS307 2A	6ES7 307-1BA00-0AA0				
2	CPU314(1)	6ES7 314-1AE04-0AB0	2			
3						
4	DI32xDC24V	6ES7 321-1BL00-0AA0		0...3		
5	DO32xDC24V/0.5A	6ES7 322-1BL00-0AA0			4...7	
6						
7						
8						
9						
10						
11						

Dane zapamiętane za pomocą **Save and Compile** przygotowane są do przekazania do CPU.

Po zakończeniu „HW Config” stworzona konfiguracja występuje jako System Data w folderze Blocks.

Dodatkowo możliwe jest sprawdzenie poprawności konfiguracji za pomocą rozkazu z Menu **Station >Consistency Check**. W przypadku wystąpienia błędów STEP 7 proponuje możliwe rozwiązania.

Więcej informacji uzyskasz w **Help > Contents** w „ Configuring the Hardware” i „Configuring Central Rack”.

7 Ładowanie i testowanie programu

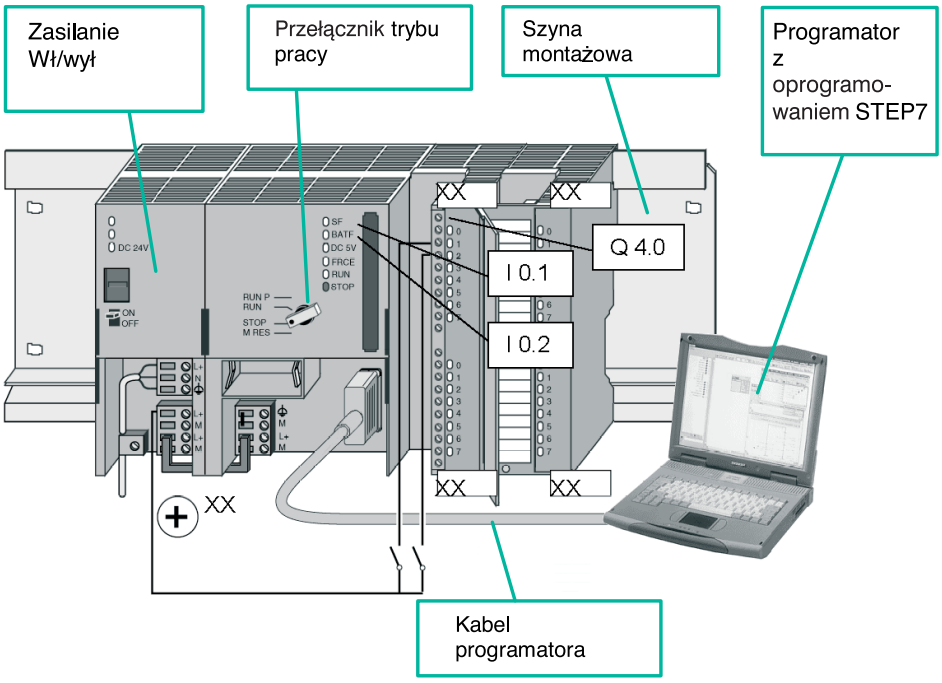
7.1 Ustanowienie połączenia ze sterownikiem

Za pomocą załączonego projektu „GS-LAD_Example” lub dotychczas utworzonego „Getting Started” pokażemy, w jaki sposób można przesłać program do sterownika, a następnie go testować.

Należy w tym celu:

- Posiadać skonfigurowany sprzęt w „Getting Started” (patrz. rozdz. 6)
- Zbudować stację sprzętową zgodnie z dokumentacją

Przykład iloczynu logicznego (Funkcja AND):
Wyjście Q 4.0 zapali się (na module wyjść cyfrowych zapala się dioda Q 4.0), jeśli zostaną włączone przełączniki I 0.1 i I 0.2. Proszę zbudować układ testowy za pomocą przewodów na własnym sterowniku



Montaż sterownika

Aby zamontować moduł na szynie, proszę postępować wg. wskazówek:

- Dołączyć moduł do łącznika szyny komunikacyjnej
- Zawiesić moduł i przechylić w dół
- Przymocować moduł dokręcając śrubę u spodu
- Zamontować pozostałe moduły
- Po zamontowaniu wszystkich modułów włożyć kluczyk do CPU.

Test można też przeprowadzić, jeśli używasz innej konfiguracji sprzętowej niż pokazana. Należy jednak uważać na prawidłowe adresowanie wejść i wyjść.

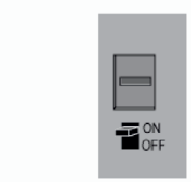
STEP 7 oferuje wiele możliwości testowych, np. test statusu programu lub test zmiennych.

Więcej informacji o konfiguracji jednostek podstawowych uzyskasz w podręczniku „S7-300, Hardware and Instalation/Module Specifikations” i „S7-400/M7-400 - Hardware”.

7.2 Ładowanie programu do sterownika

Ładowanie programu jest możliwe dopiero po utworzeniu połączenia On-Line.

Załączenie zasilania



Włącz napięcie wyłącznikiem ON/OFF. Zapali się dioda „DC 5V” na CPU.

Przekręć przełącznik trybu pracy na STOP (o ile nie jest już przełączony). LED „STOP” zapali się na pomarańczowo.

Kasowanie całkowite CPU i uruchomienie w tryb RUN

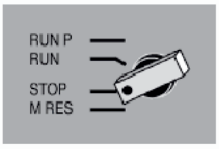
Ustaw przełącznik trybu pracy w pozycji **MRES** i przytrzymaj tak co najmniej 3 sek. dopóki LED „STOP” nie zacznie migać powoli na pomarańczowo.

Puść przełącznik trybu pracy i najpóźniej po 3 sek. znów przekręć w pozycję **MRES** . Jeśli LED „STOP” szybko miga, oznacza to, że CPU zostało całkowicie wykasowane.

Jeśli „STOP” nie miga w taki sposób, należy powtórzyć operację.

Kasowanie całkowite powoduje usunięcie wszystkich danych z CPU. CPU znajduje się w stanie podstawowym.

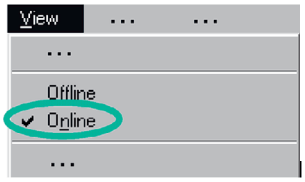
Załadowanie programu do CPU



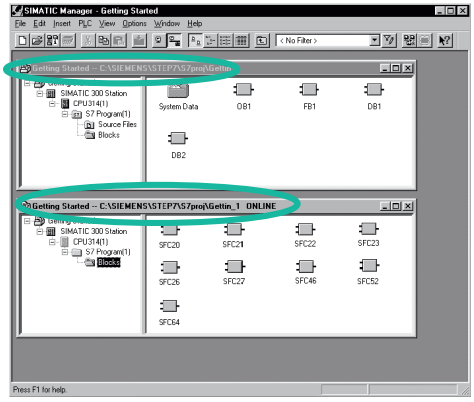
Aby załadować program ustaw przełącznik trybu pracy ponownie na „STOP”.

SIMATIC Manager

Uruchom SIMATIC Manager , otwórz w SIMATIC Managerze przez pole dialogowe „Open” projekt „Getting Started” (o ile nie jest już otwarty).



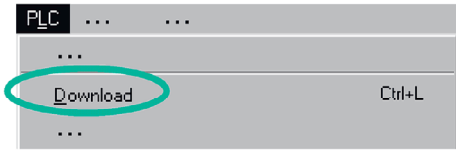
Wywołaj dodatkowo dla okna „Getting Started Offline” okno „Getting Started Online”. Status Offline/Online zaznaczony jest w nagłówku zmiennym kolorem.



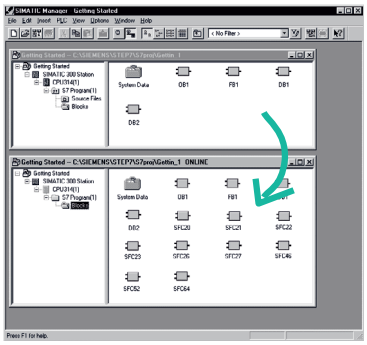
Przejdź w obu oknach do folderu **Blocks**.

Okno „Offline” pokazuje stan na PG, okno „Online” pokazuje stan na CPU.

Pomimo całkowitego kasowania w CPU znajdują się funkcje systemowe (SFCs). Nie trzeba ich ładować, nie można także ich skasować.



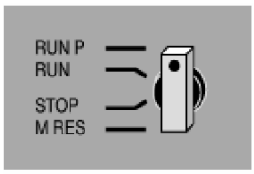
Zaznacz w oknie „Offline” folder **Blocks,** a następnie załaduj program do CPU wywołując **PLC>Download**.
Potwierdź polecenie przez **OK**.



W oknie „Online” po załadowaniu widoczne będą bloki programowe.

Polecenie Menu **PLC>Download** można wywołać przez odpowiedni klawisz listwy funkcyjnej lub z menu kontekstowego wywoływanego prawym klawiszem myszy.

Włączanie CPU i sprawdzanie trybu pracy.



Ustaw przełącznik trybu pracy w położenie **RUN-P**. Dioda „RUN” zapali się na zielono, a zgaśnie dioda „STOP”. CPU jest gotowe do pracy.

Jeśli pali się dioda RUN, można zaczynać test programu.

Jeśli pali się nadal dioda STOP, wystąpił błąd. Do zdiagnozowania błędu użyj bufora diagnostycznego.

Ładowanie poszczególnych bloków

Aby w praktyce móc szybko zareagować w przypadku wystąpienia błędów, można ładować bloki do CPU pojedynczo wykorzystując mechanizm Drag and Drop.

Przy ładowaniu bloków przełącznik trybu pracy na CPU musi być ustawiony na „RUN-P” lub „STOP”. Bloki załadowane przy trybie pracy „RUN-P” zostają natychmiast uruchomione. Należy przy tym pamiętać, że:

- Jeśli poprawne bloki zostaną nadpisane przez błędne, spowoduje to błędne funkcjonowanie urządzenia. Można tego uniknąć przeprowadzając przed ładowaniem test.
- Jeśli nie przestrzegano kolejności ładowania bloków – tzn: wpierw mniej ważne, potem główne – CPU przechodzi w tryb pracy „STOP”. Można tego uniknąć ładując cały program do CPU w odpowiedniej kolejności.

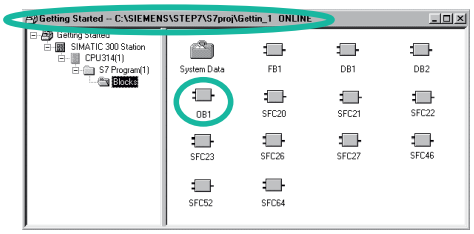
Programowanie Online

Dla celów testowych może zaistnieć w praktyce konieczność zmiany dopiero co załadowanych bloków do CPU. W tym celu kliknij dwukrotnie na żądany blok w oknie „Online”, aby otworzyć okno programu LAD/STL/FBD. Zaprogramuj blok na nowo. Należy pamiętać, że zaprogramowany i przepisany moduł zostaje natychmiast uaktywniony w CPU.

Dalsze informacje znajdują się w **Help > Contents** w rozdziałach "Establishing an Online Connection and Making CPU Settings" i "Downloading from the PG / PC to the Programmable Controller."

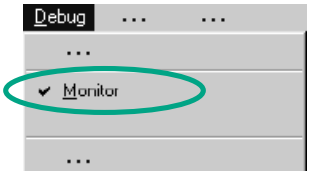
7.3 Testowanie programu – status bloku

Wykorzystując funkcję status bloku można podglądać działanie programu. Niezbędnym do realizacji jest istnienie połączenia Online z CPU. Jednostka centralna powinna znajdować się w trybie RUN lub RUN-P. Program powinien oczywiście zostać załadowany do CPU.



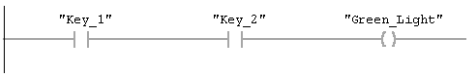
Otwórz **OB1** w oknie projektowym „Getting Started Online".

Otworzy się okno edytora programowania LAD/STL/FBD.



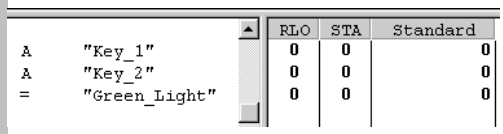
Uaktywnij funkcję **Debug> Monitor.**

Testowanie w LAD.



Przełącznik szeregowy w sieci 1 w LAD zostaje zaznaczony. Aż do styku 1 (I 0.1) zaznaczony jest przepływ prądu, tzn., że tu przyłożone jest napięcie.

Testowanie w STL



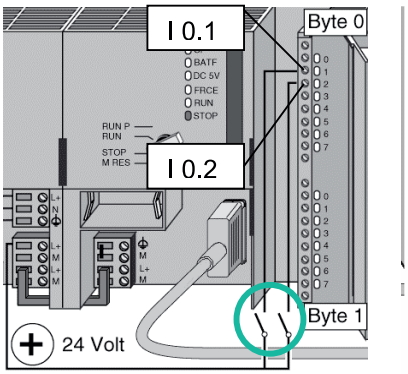
		RLO	STA	Standard
A	"Key_1"	0	0	0
A	"Key_2"	0	0	0
=	"Green_Light"	0	0	0

W STL w tabeli prezentowane są:
- Wynik operacji logicznej (RLO)
- bit statusu (STA)
- stan akumulatora 1 (STANDARD)

Testowanie w FBD

Stan sygnału oznaczony jest za pomocą „0" i „1". Linia przerywana oznacza, że nie jest spełniony warunek połączenia.

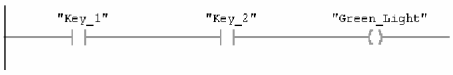
Przez **Option> Customize** można zmieniać sposób przedstawienia w danej postaci testowania.



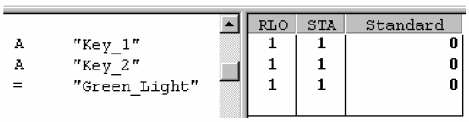
Załącz teraz na swoim modelu testowym oba przełączniki.

Na module wejściowym zapalą się diody wejść I 0.1 i I 0.2.

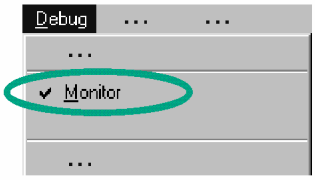
Na module wyjściowym zapali się dioda wyjścia Q 4.0.



W postaciach graficznych LAD/FBD możemy obserwować zmiany stanu stosując inne kolory dla różnych stanów.



W języku STL zmiany w układzie sterowania sygnalizowane są zmianami wartości w tabeli w kolumnach RLO i STA.



Zdeaktywuj **Debug>Monitor** i zamknij okno.

Następnie w SIMATIC Manager zamknij okno „Online”.

Nie zalecamy całkowitego ładowania obszernych programów do CPU, ponieważ utrudnia to zdiagnozowanie ewentualnego błędu z uwagi na zwiększoną ilość źródeł błędu. Zalecamy za to, dla lepszej przejrzystości, oddzielne ładowanie poszczególnych bloków i każdorazowe ich testowanie po załadowaniu.

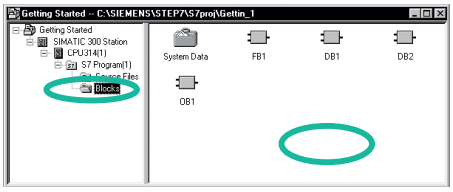
Dalsze informacje w **Help > Contents** w rozdziałach "Debugging" i "Testing with Program Status."

7.4 Testowanie działania programu z wykorzystaniem tabeli zmiennych

Testujesz poszczególne zmienne programu obserwując je i sterując nimi. Warunkiem działania jest istnienie połączenia Online z jednostką centralną będącą w trybie RUN-P, oraz wcześniejsze załadowanie programu.

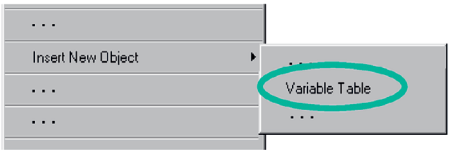
Podobnie jak przy obserwacji statusu programu możesz obserwować w tabeli zmiennych wejścia i wyjścia sieci 1 (przełącznik szeregowy tzn. funkcja AND). Dodatkowo możesz testować także aktualną prędkość obrotową silnika z funkcji porównania z bloku FB1.

Tworzenie tabeli zmiennych

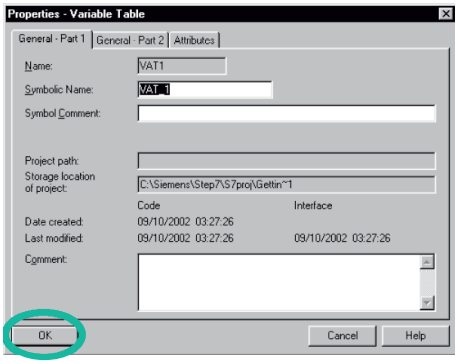


Punktem wyjścia jest znów SIMATIC Manager z otwartym projektem „Getting Started Offline”.

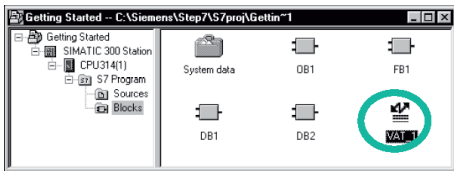
Przejdź do folderu **Blocks** i kliknij prawym przyciskiem myszy na prawą część okna.



Wprowadź za pomocą Menu kontekstowego **Variable Table**.



Przyjmij ustawienia wstępne zamykając pole dialogowe „Properties” klawiszem **OK**.

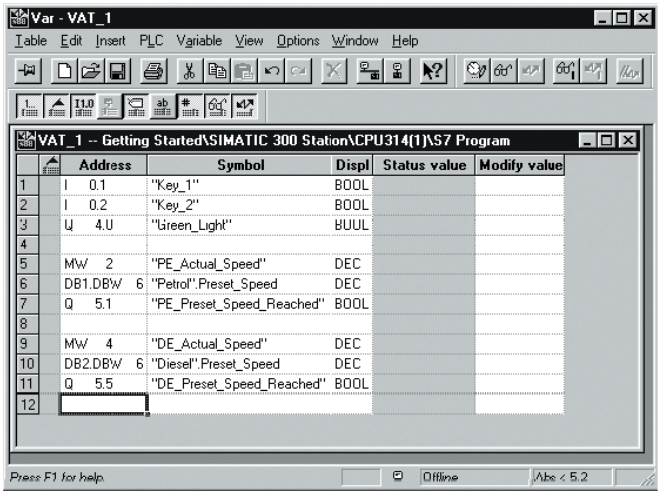


W katalogu bloków założona zostaje VAT1 (tabela zmiennych).

Otwórz **VAT1** podwójnym kliknięciem a znajdziesz się w oknie „Monitoring and Modifying Variable”.

Tabela zmiennych jest początkowo pusta. Wprowadź nazwy symboli lub adresy dla przykładu „Getting Started” zgodnie z rysunkiem. Pozostałe dane zostaną uzupełnione, jeśli zatwierdzisz wprowadzony element klawiszem Return.

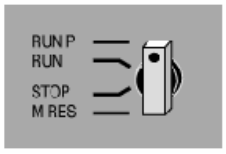
Zmień format statusu wszystkich wartości obrotów na format DEC. Kliknij na odpowiednią kratkę (strzałka myszy na kolumnie „Monitor Format”) i wybierz format DEC.



Zapisz swoją tabele zmiennych.

Załączenie tabeli zmiennych w tryb Online

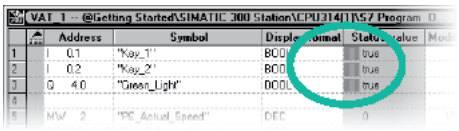
Kliknij przycisk **ON** w oknie "Monitoring and Modifying Variables".Zostanie stworzone połączenie Online. W listwie statusowej pojawi się napis ONLINE.



Ustaw kluczyk w pozycji **RUN-P** (o ile nie jest już tak ustawiony).

Obserwacja zmiennych

Kliknij na **Monitor Variable**. W listwie statusu widoczny jest tryb pracy CPU.

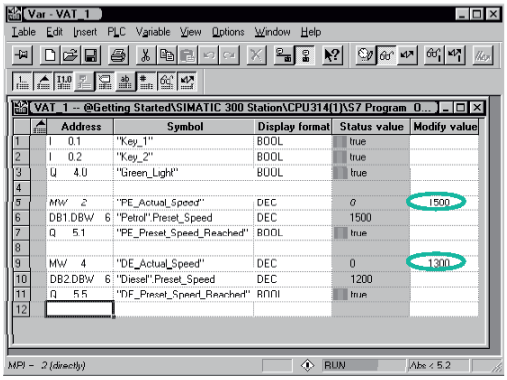


Włącz Przycisk 1 i 2 na wejściach sterownika i obserwuj wynik w tabeli zmiennych.

W tabeli zmiennych status wejść zmienia się z „false” na „true”.

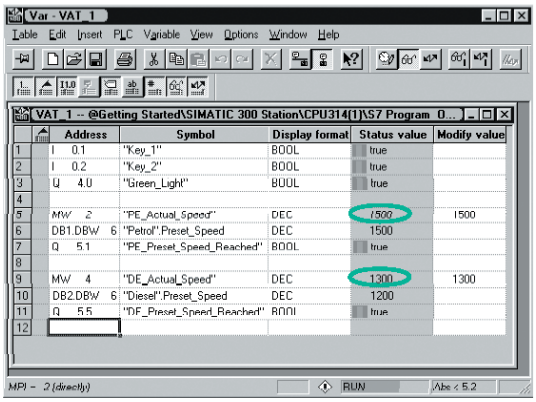
Modyfikacja zmiennych

Wprowadź wartość 1500 dla MW2 w kolumnie Modify Value, i dla MW4 wprowadź 1300.



Przenieś wprowadzone wartości do sterownika.

Po przesłaniu danych do CPU, można zaobserwować wynik porównania.



Często ze względu na ograniczenia ekranu bardzo obszerna tabela zmiennych nie może być pokazana w całości.

Jeśli posiadasz dużą tabelę zmiennych, zalecamy tworzenie za pomocą STEP 7 kilku tabel zmiennych do programu S7. Tabele te można dopasować dokładnie do twoich potrzeb testowych.

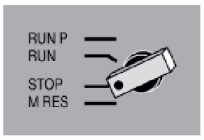
Analogicznie do bloków tabelom zmiennych można przyporządkowywać indywidualne nazwy symboliczne (np. zamiast VAT1 nazwę OB1_sieć1). Przyporządkowanie nazw odbywa się w tabeli symboli.

Dalsze informacje znajdują się w **Help > Contents** w rozdziałach "Debugging" i "Testing with the Variable Table."

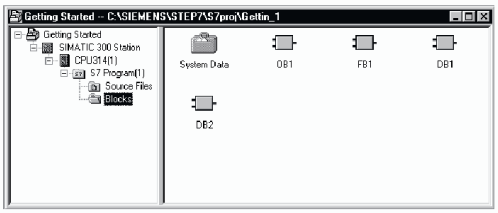
7.5 Wywołanie bufora diagnostycznego

W przypadku gdy podczas pracy programu CPU przejdzie w STOP, lub po załadowaniu programu nie da się przełączyć na RUN, można ustalić przyczynę takiego zachowania na podstawie informacji znajdujących się w buforze diagnostycznym.

Warunkiem jest istniejące połączenie Online oraz CPU w trybie STOP.

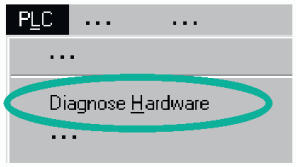


Ustaw przełącznik trybu pracy na „STOP”.

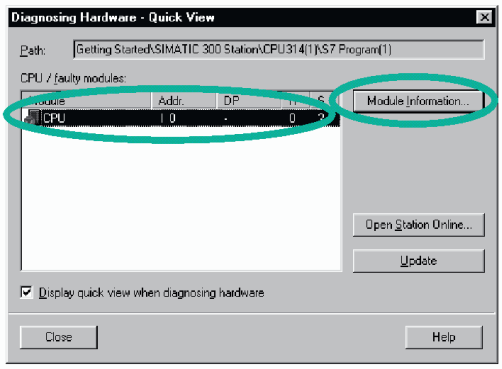


Punktem wyjścia jest znów SIMATIC Manager z otwartym projektem „Getting Started Offline”.

Zaznacz folder **Blocks**.



Jeśli twój projekt zawiera kilka CPU, ustal wpierw, które CPU ustawione są na STOP.



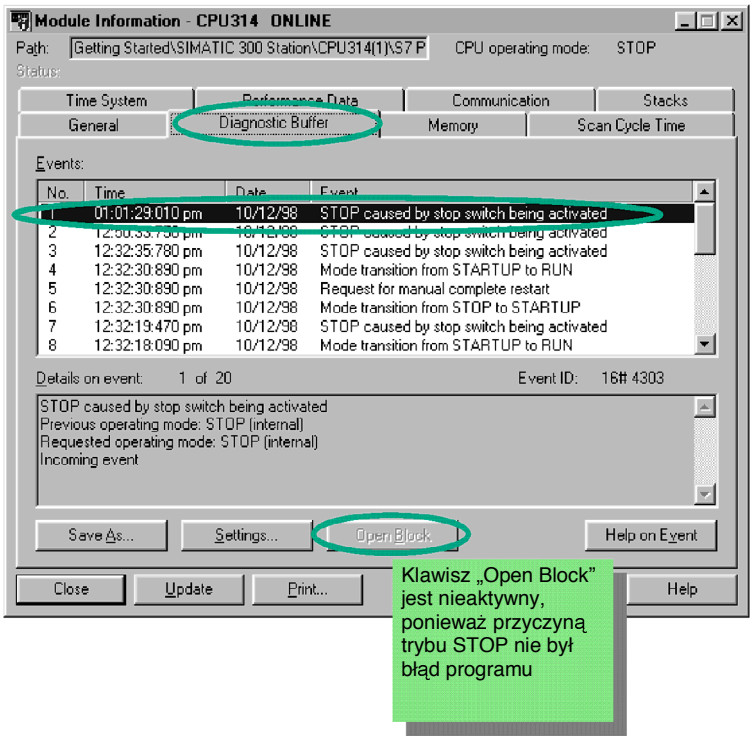
W polu dialogowym „Diagnosing Hardware” znajduje się lista wszystkich znalezionych CPU. Zaznaczony jest CPU w stanie STOP.

W projekcie „Getting Started” zdefiniowane jest tylko jedno CPU.

Kliknij na **Module Information**, aby zobaczyć bufor diagnostyczny tego CPU.

Jeśli jest podłączone tylko jedno CPU, możesz bezpośrednio zbadać stan modułów sterownika wykorzystując funkcję w menu **PLC > Module Information.**

W oknie „Module Information” możesz znaleźć informacje o cechach i parametrach twojego CPU. Aby sprawdzić przyczynę trybu pracy STOP wybierz zakładkę **Diagnostic Buffer.**



Ostatnie wydarzenie (nr 1) znajduje się na najwyższej pozycji. Pokazana jest przyczyna wystąpienia STOP. Zamknij wszystkie okna, aż do poziomu Managera SIMATIC.

Jeśli przyczyną STOP jest błąd programu, zaznacz wynik wydarzenia i kliknij na pole **Open** .

Blok zostanie otwarty w znanym już oknie edytora programu LAD/STL/FBD, a błędna sieć zostanie zaznaczona.

Ten rozdział kończy przedstawianie tworzenia projektu „Getting Started” od momentu jego założenia do testowania pracującego sterownika. W następnych rozdziałach pogłębisz swoją wiedzę wykonując przedstawione ćwiczenia.

Dalsze informacje znajdują się w **Help > Contents** w rozdziale "Calling the Module Information."

8 Programowanie funkcji FC

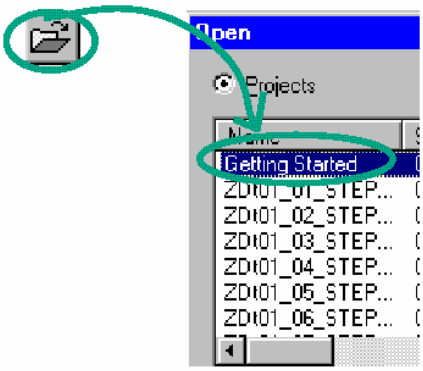
8.1 Założenie i otwarcie funkcji

Funkcja, podobnie jak blok funkcyjny, przyporządkowana jest blokowi organizacyjnemu. Zatem by funkcja mogła być wykonana, należy ją wywołać bezpośrednio w OB, lub też w innym bloku przyporządkowanym temu OB. Funkcja w przeciwieństwie do bloku funkcyjnego, nie ma przyporządkowanego bloku danych.

Podobnie jak w FB, funkcja może mieć także zadeklarowane parametry, jednakże nie posiada danych statycznych.

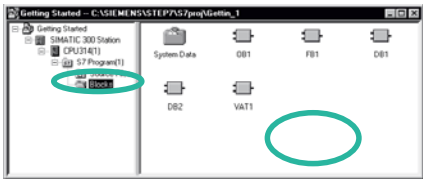
Analogicznie jak inne bloki, funkcję programuje się w edytorze LAD/STL/FBD.

Znamy już programowanie w postaci LAD, FBD lub STL (rozdz. 4) czy też programowanie symboliczne (rozdz. 3).



Jeśli zgodnie z rozdziałami 1-7 został stworzony projekt „Getting Started”, należy go otworzyć.

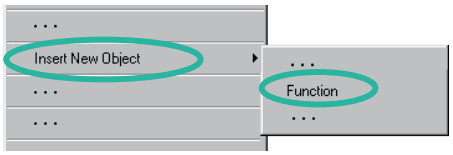
Jeśli nie, proszę założyć nowy projekt wykorzystując w SIMATIC Manager funkcję **File > "New Project" Wizard**. Postępując zgodnie ze wskazówkami w rozdziale 2.1 należy zmienić nazwę na „Getting Started Function”.



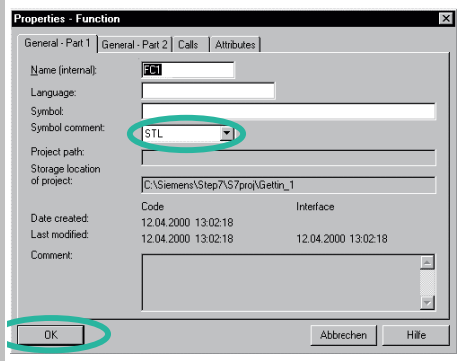
Dalej będziemy posługiwać się projektem „Getting Started”. Można jednak powtarzać kroki na projekcie stworzonym przez siebie.

Otwórz folder **Blocks.**

Kliknij prawym klawiszem myszy na prawej części okna.

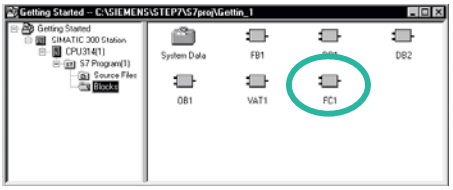


Wprowadź funkcję używając menu kontekstowego.



W oknie "Properties – Function" wybierz nazwę bloku FC1, oraz język w jakim chcesz tworzyć program.

Potwierdź ustawienia za pomocą klawisza **OK.**



Funkcja FC1 pojawi się w prawej części okna.

By otworzyć funkcję kliknij dwukrotnie na FC1.

Wewnątrz funkcji, w przeciwieństwie do bloku funkcyjnego, nie można definiować parametrów statycznych.

Dane statyczne definiowane w bloku funkcyjnym zachowują swoją wartość. Odpowiednikami danych statycznych będą także słowa merkerów (znaczników MW) np. dla wartości granicznej obrotów. (patrz rozdz. 5).

Podczas programowania funkcji można wykorzystywać nazwy symboliczne.

Dalsze informacje znajdują się w **Help > Contents** w rozdziale "Working Out the Automation Concept," "Basics of Designing a Program Structure," i "Blocks in the User Program."

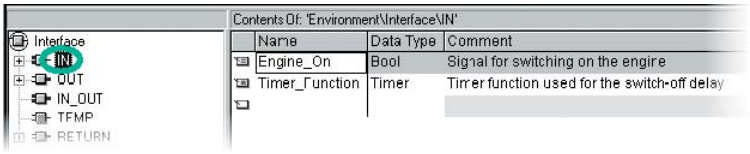
8.2 Programowanie funkcji

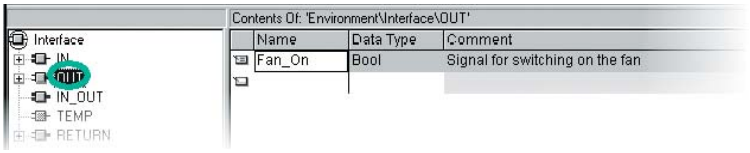
W naszym przykładzie zaprogramujemy funkcję czasową. Funkcja będzie działać w ten sposób, iż po załączeniu silnika (rozdz. 5) załączony zostanie wentylator, który po wyłączeniu silnika będzie pracować jeszcze przez cztery sekundy. (Opóźnienie wyłączenia).

Zgodnie z powyższym należy zadeklarować parametry wejściowe i wyjściowe funkcji.

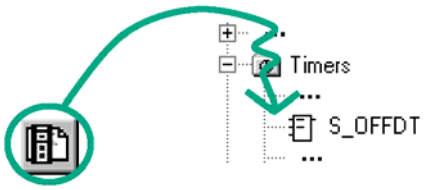
Otwórzmy edytor programu LAD/STL/FBD. Deklarację parametrów należy przeprowadzić podobnie, jak przy parametryzacji bloku funkcyjnego (rozdz. 5).

Wprowadź poniższe parametry:



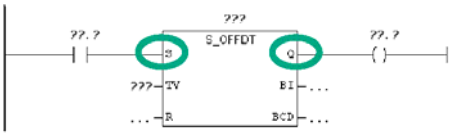


Programowanie funkcji czasowej w LAD



Zaznacz linię wprowadzania elementów LAD.

Znajdź w katalogu funkcji człon czasowy S_OFFDT, i wprowadź go do sieci.



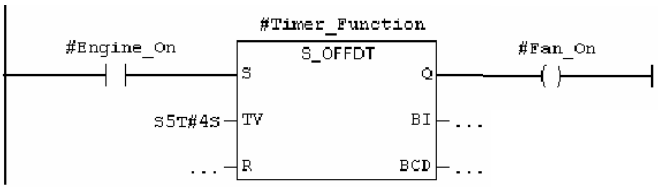
Wprowadź na wejście **S** styk normalnie rozwarty.

Wprowadź cewkę na wyjście **Q**.

Zaznacz znaki zapytania i wprowadź w to miejsce nazwę parametru z tabeli deklaracyjnej (automatycznie zostanie poprzedzona znakiem #).

Na wejście TV wprowadź wartość czasu opóźnienia. Postać wprowadzana S5T#4s oznacza iż wprowadzana jest wartość stała w formacie S5TIME# (S5T#) o wielkości 4 sekund.

Zapamiętaj stworzoną funkcję i zamknij okno.



Parametr wejściowy „Engine_ON” powoduje uruchomienie #Timer_Function. Później trzeba będzie stworzoną funkcję FC wywołać dwukrotnie w bloku OB1, raz dla silnika benzynowego, drugi raz dla silnika Diesla. Podczas wywołania trzeba będzie określić parametry, które potem można oznaczyć symbolem w tabeli symbolicznej.

Programowanie funkcji czasowej w STL

```
A     #Engine_On
L     S5T#4S
SF    #Timer_Function
A     #Timer_Function
=     #Fan_On
```

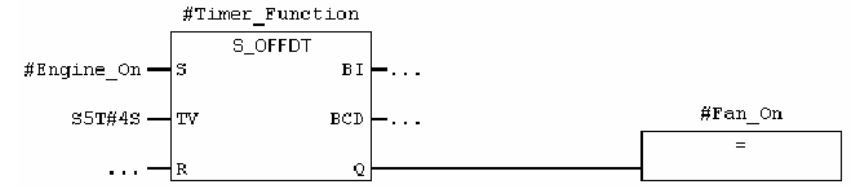
W przypadku programowania w postaci STL ustaw kursor na polu wprowadzania programu i wpisz obok przedstawione instrukcje.

Zapisz wprowadzoną funkcję i zamknij okno edytora.

Programowanie funkcji czasowej w FBD

W przypadku programowania w FBD ustaw kursor na polu wprowadzania funkcji i stwórz odpowiednią funkcję czasową.

Zapisz wprowadzoną funkcję i zamknij okno edytora.



By stworzona funkcja mogła być prawidłowo przetwarzana, należy ją wywołać w odpowiednio przyporządkowanym bloku nadrzędnym (np. w OB1).

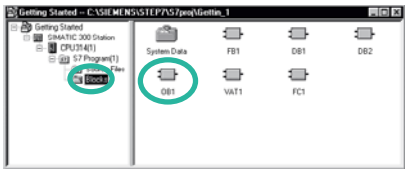
Dalsze informacje znajdują się w **Help >Contents** w rozdziale "Calling Reference Helps", "The STL, FBD, or LAD Language Description", i "Timer Instructions."

8.3 Wywołanie funkcji w OB1

Wywołanie funkcji FC1 w bloku OB1 wykonuje się tak samo jak we wcześniej stworzonym bloku funkcyjnym. Należy przy wywoływaniu funkcji określić wszystkie jej parametry.

Jeśli któryś z operandów przypisywanych do parametrów nie posiada nazwy symbolicznej, należy uzupełnić to używając tabeli symbolicznej.

Operant jest fragmentem instrukcji w STEP 7, który mówi z czym procesor powinien coś zrobić. Może on być w postaci absolutnej lub symbolicznej.



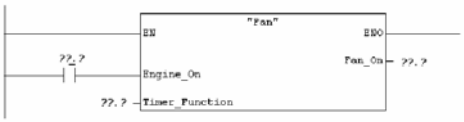
Otwarty jest aktualnie projekt „Getting Started”, lub też projekt stworzony samodzielnie.

Dojdź do folderu **Blocks** i otwórz blok **OB1.**

Otwiera się okno edytora LAD/STL/FBD.

Programowanie wywołania w LAD

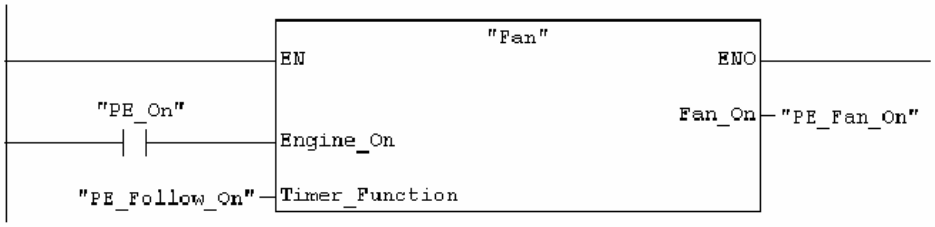
Znajdujemy się w edytorze LAD. Wprowadź nową sieć (6). Znajdź w katalogu elementów programowych folder FC Blocks, a w nim FC1, i wprowadź do programu.



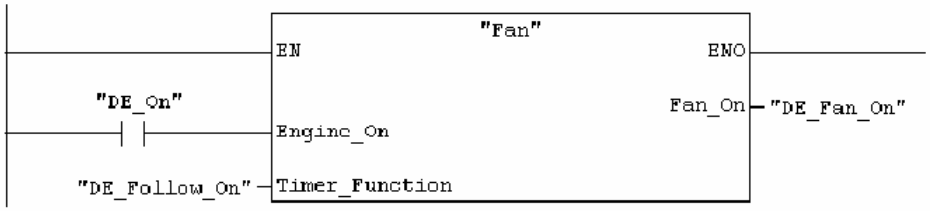
Wstaw styk normalnie rozwarty na wejście „Engine_On".

Używając View >Display >Symbolic Representation, możemy wybrać między symbolicznym i absolutnym sposobem adresowania.

Kliknij na znaki zapytania i wprowadź dalsze nazwy symboliczne.



W sieci 7 zaprogramuj wywołanie FC1 dla silnika Diesla. Sposób postępowania jest analogiczny, jak w ww. przypadku (w liście symbolicznej powinny znajdować się wcześniej przygotowane symbole dla potrzebnych operandów).



Zapisz blok i zamknij okno edytora.

Uaktywnij polecenie menu **View >Display > Symbol Information** by w każdej z sieci pojawiły się opisy symboli.

By zwiększyć ilość sieci prezentowanych na ekranie wyłącz **View > Display > Comment** oraz, w razie potrzeby **View > Display > Symbol Information**.

Używając polecenia menu **View > Zoom Factor**, można zmienić rozmiary wyświetlanych instrukcji programowych.

Programowanie wywołania w STL

```
Network 6 : Controlling the Fan for the Petrol Engine
    CALL  "Fan"
     Engine_On      :="PE_On"
     Timer_Function:="PE_Follow_On"
     Fan_On         :="PE_Fan_On"

Network 7 : Controlling the Fan for the Diesel Engine
    CALL  "Fan"
     Engine_On      :="DE_On"
     Timer_Function:="DE_Follow_On"
     Fan_On         :="DE_Fan_On"
```

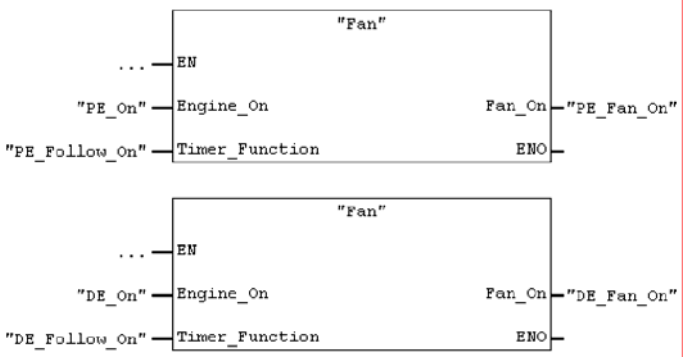
W przypadku programowania w STL, wejdź w obszar programowania i wprowadź instrukcje podane obok.

Zapisz wprowadzony program i zamknij okno edytora.

Programowanie wywołania w FBD

W przypadku programowania w FBD wejdź w obszar wprowadzania diagramu funkcji i operując w katalogu funkcji wprowadź poniżej przedstawione elementy.

Zapisz wprowadzony program i zamknij okno edytora.



W naszym przykładzie zastosowaliśmy wywołanie bezwarunkowe funkcji, tzn. każdorazowo funkcja będzie realizowana bez względu na wynik wcześniejszych operacji logicznych.

Można jednak w razie potrzeby realizację FC lub FB uzależnić od pewnych warunków np. od jakiegoś wejścia. By zrealizować wywołanie warunkowe należy ten warunek wprowadzić na wejście EN.

Dalsze informacje znajdują się w **Help > Contents** w rozdziałach "Calling Reference Helps," "The LAD, FBD, or STL Language Description," lub"Program Control Instructions."

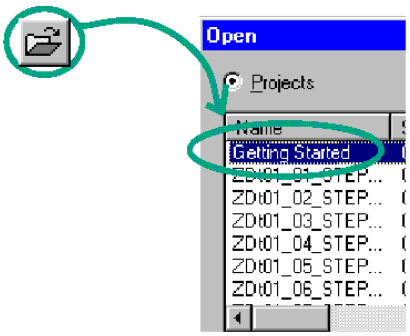
9 Programowanie globalnego bloku danych

9.1 Zakładanie i otwieranie globalnego bloku danych

Jeśli ilość wewnętrznych znaczników (merkerów) w CPU (komórki pamięci) nie jest wystarczająca do zapamiętania wszystkich danych, pewne wybrane dane mogą zostać zarchiwizowane w globalnym bloku danych.

Dane globalnego bloku danych są dostępne z każdego bloku programowego. Lokalny blok danych jest przyporządkowany określonemu blokowi funkcyjnemu, dane dostępne są tylko w tym bloku (por. rozdz. 5.5).

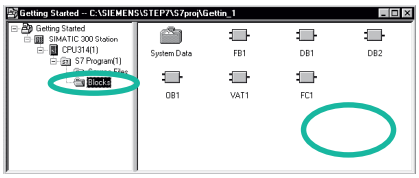
Zapoznałeś się już z programowaniem w LAD, STL i FBD (por. rozdz. 4 i 5) jak i programowaniem z wykorzystaniem symboli (por. rozdz. 3).



Jeśli przeprowadziłeś przykładowy projekt „Getting Started” z rozdziału 1-7, otwórz go teraz.

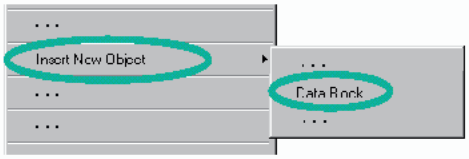
Jeśli nie, załóż nowy projekt w SIMATIC Manager za pomocą **File > „New Project” Wizard** . Analogicznie do rozdziału 2.1 nazwij projekt „Getting Started Global-DB”.

Punktem wyjścia w dalszej części będzie projekt „Getting Started”. Możesz jednakże wykonać każdy krok również w nowo założonym projekcie.

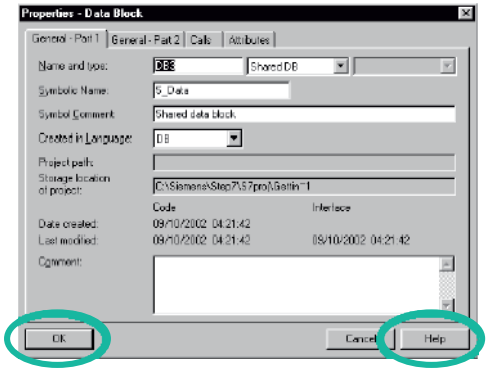


Przejdź do folderu **Blocks** i otwórz go.

Kliknij prawym przyciskiem myszy na prawe pole okna.



Dodaj **Data Block** z Menu kontekstowego



W polu dialogowym „Properties - Data Block” potwierdź wszystkie ustawienia klawiszem **OK**.

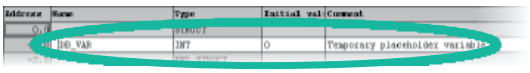
Dla uzyskania dalszych informacji użyj przycisku **HELP**.

Blok danych DB3 dodany zostanie do katalogu **Blocks**.

Podwójnym kliknięciem otwórz **DB3**.

Wskazówka: w rozdziale 5.5, utworzony jest lokalny blok danych aktywując opcję "Data block referencing a function block." Za pomocą "Data block" tworzony jest za to globalny blok danych .

Programowanie zmiennych w bloku danych



W kolumnie Name wpisz „PE_Actual_Speed”.

By ustalić typ wybierz prawym przyciskiem myszy z menu kontekstowego **Elementary Types > INT**.

Zdefiniowano przykładowo trzy dane globalne w DB3. Uzupełnij odpowiednie dane w tabeli deklaracyjnej zmiennych.

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	DB_VAR	INT	0	Temporary placeholder variable
=2.0		END_STRUCT		

Zmienne stanu obrotów w bloku danych „PE_Actual_Speed” i „DE_Actual_Speed” traktowane są tak samo jak słowa znaczników (merkerów) MW2 „PE_Actual_Speed” i MW4 „DE_Actual_Speed”. Pokażemy to w następnym rozdziale.

Zapisz globalny blok danych.

Przyporządkowanie symboli

Blokowi danych także możesz przyporządkować nazwę symboliczną.

Otwórz **Symbol Table** i wprowadź dla bloku danych DB3 symbol „S_Data”.

Jeśli w rozdziale 4 kopiowałeś tabelę symboli z przykładowego projektu (zEn01_02_STEP7__STL_1-10, zEn01_06_STEP7__LAD_1-10 lub zEn01_04_STEP7__FBD_1-10) do swojego projektu „Getting Started”, nie musisz teraz wprowadzać nowych symboli.

Symbol	Address	Data Type	Comment
...	...	...	...
S_Data	DB 3	DB 3	Shared data block

Zapisz tabelę symboli i zamknij okno „Symbol Editor”.

Zamknij też tabelę deklaracji zmiennych globalnego bloku danych.

Globalny DB w tabeli deklaracji zmiennych:
Używając **View > Data View** możesz zmieniać aktualne wartości danych typu INT w tabeli globalnego bloku danych (por. rozdz. 5.5).

Globalny DB w tabeli symboli:
W przeciwieństwie do lokalnego bloku danych typ danych globalnego DB w tabeli symboli jest zawsze adresem absolutnym. W naszym przykładzie typ danych to „DB3”. Dla lokalnego bloku danych na stałe podany jest jako typ danych przynależny FB.

Więcej informacji uzyskasz pod **Help > Contents** „Programming Blocks” i „Creating Data Blocks”.

10 Programowanie wielokrotnego lokalnego bloku danych

10.1 Zakładanie i otwieranie nadrzędnego bloku funkcyjnego

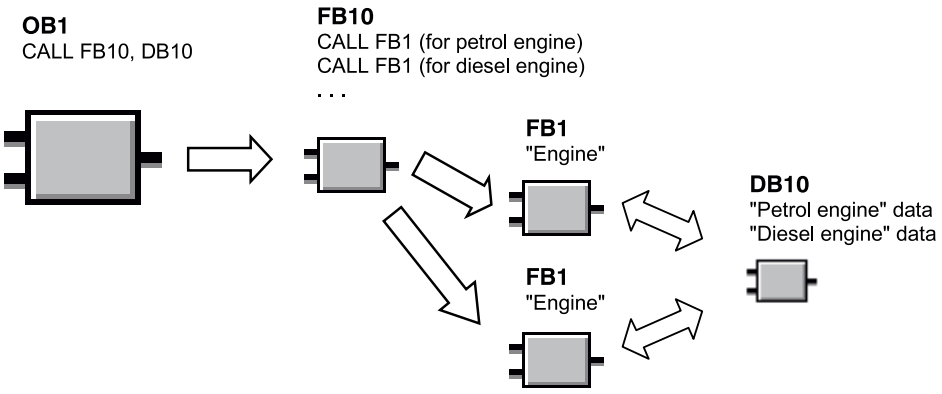
W rozdziale 5 został zaprogramowany blok funkcyjny FB1 „Engine”, który sterował działaniem silnika. Przy wywoływaniu tego bloku w OB1 wykorzystywane były bloki danych „Petrol” (DB1) i „Diesel” (DB2). Bloki danych zawierały parametry pracy danego silnika.

Co zrobić, jeśli w swoim programie potrzebujemy stworzyć układy sterowania większej ilości silników.

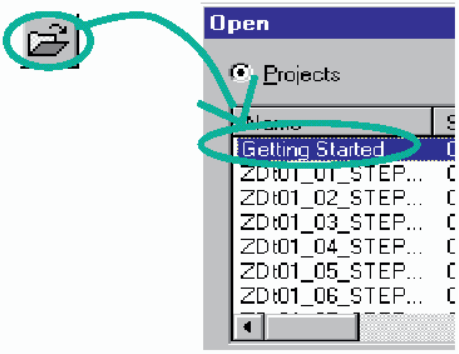
Zgodnie z wcześniej przedstawionym tokiem postępowania należałoby każdorazowo wywołać blok FB1 i stworzyć do niego kolejne lokalne bloki danych. Byłyby to np. DB3 i DB4.

Można uniknąć tworzenia większej ilości takich bloków danych stosując wielokrotny lokalny blok danych. Załóżmy zatem nowy blok funkcyjny (w naszym przykładzie FB10), i wywołajmy w nim niezmieniony blok FB1 jako „local instance”. Każdorazowo wywołanie FB1 powoduje dodanie danych w DB10 – lokalnym bloku danych przyporządkowanym FB10. Wszystkie wywołane FB będą pracować na jednym bloku (tutaj DB10).

W DB10 zintegrowane zostaną bloki DB1 i DB2. Niezbędnym do tego jest zadeklarowanie FB1 w zmiennych statycznych bloku FB10.

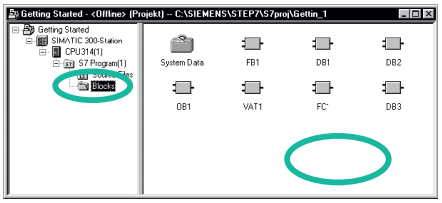


Znamy już programowanie w LAD, STL oraz FBD, a także reprezentację symboliczną.



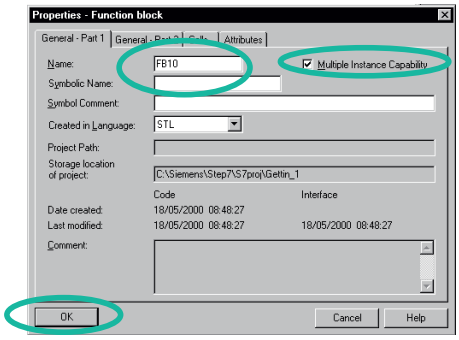
Jeśli zgodnie z rozdziałami 1-7 został stworzony projekt „Getting Started”, należy go otworzyć.

Jeśli nie, proszę otworzyć w Managerze SIMATIC jeden z projektów:
zEn01_06_STEP7__**LAD**_1-9 dla LAD,
zEn01_02_STEP7__**STL**_1-9 dla STL
zEn01_04_STEP7__**FBD**_1-9 dla FBD.



Otwórz folder **Blocks.**

Kliknij na prawej części okna prawym klawiszem myszy i za pomocą menu kontekstowego wprowadź blok funkcyjny.



Zmień nazwę bloku na FB10 i ustal dogodną postać programowania.

Jeśli jest to potrzebne, aktywuj opcję **Multiple instance**, i zatwierdź dokonane ustawienia przyciskiem **OK**.

Blok FB10 pojawi się w folderze **Blocks.** Otwórz go podwójnie klikając.

Wielokrotne bloki danych mogą być tworzone w połączeniu z dowolnymi blokami funkcyjnymi (np. z przeznaczeniem do sterowania zaworami). Jeśli chcemy pracować wykorzystując bloki wielokrotne, należy uważać na to, by było to możliwe zarówno dla bloku nadrzędnego, jak i podrzędnego.

Dalsze informacje znajdują się w **Help > Contents** w rozdziaach "Programming Blocks" i "Creating Blocks and Libraries."

10.2 Programowanie FB10

By wykorzystywać blok FB1 jako wielokrotny należy dla każdego jego wywołania stworzyć w części deklaracyjnej FB10 zmienne statyczne (każdorazowo opisane inną nazwą). Jako typ danych należy wybrać FB1.

Wypełnienie tabeli deklaracji

Otwarte jest okno edytora programowego. Zadeklaruj do wywołania FB1 następujące zmienne.

Contents Of: 'Environment\Interface\OUT'

Name	Data Type	Address	Initial Value	Comment
Preset_Speed_Reached	Bool	0.0	FALSE	Both engines have reached the preset speed

Contents Of: 'Environment\Interface\STAT'

Name	Data Type	Address	Initial Value	Comment
Petrol_Engine	Engine	2.0		First local instance of FB1 "Engine"
Diesel_Engine	Engine	10.0		Second local instance of FB1 "Engine"

Contents Of: 'Environment\Interface\TEMP'

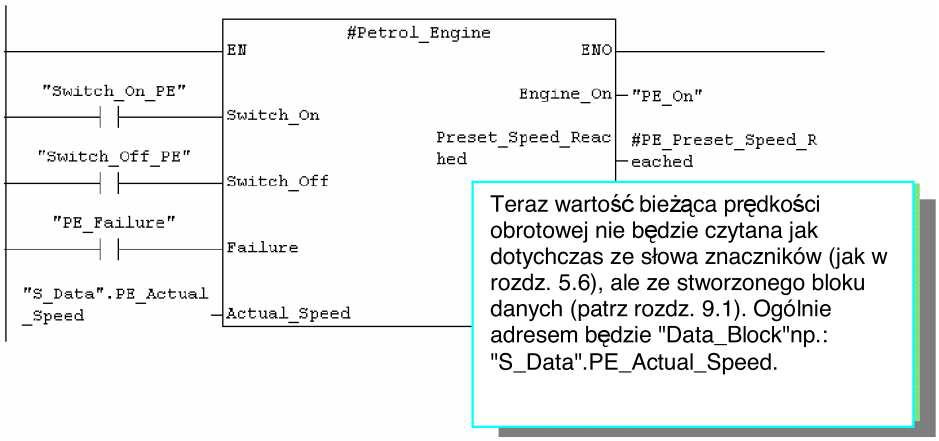
Name	Data Type	Address	Comment
PE_Preset_Speed_Reached	Bool	0.0	Preset speed reached (petrol engine)
DE_Preset_Speed_Reached	Bool	0.1	Preset speed reached (diesel engine)

Zadeklarowane bloki wielokrotne znajdą się w efekcie naszych działań w katalogu elementów programowych pod nazwą "Multiple Instances."

Programowanie FB10 w LAD

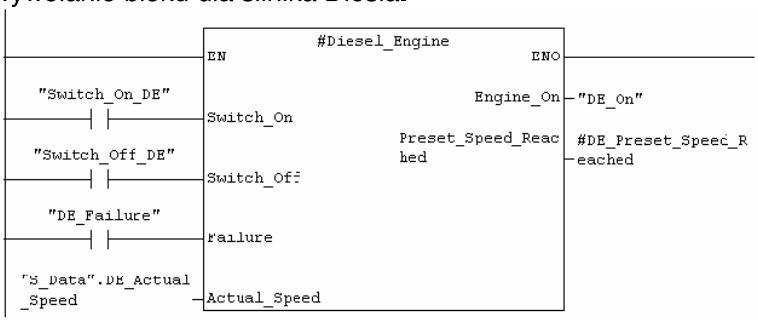
Wprowadź w sieci 1 wywołanie wielokrotnego bloku „Petrol_Engine”.

Uzupełnij wejścia odpowiednimi stykami oraz nadaj stykom adresy np. w postaci symbolicznej.



Teraz wartość bieżąca prędkości obrotowej nie będzie czytana jak dotychczas ze słowa znaczników (jak w rozdz. 5.6), ale ze stworzonego bloku danych (patrz rozdz. 9.1). Ogólnie adresem będzie "Data_Block"np.: "S_Data".PE_Actual_Speed.

Wprowadź nową sieć i postępując zgodnie z poprzednim przykładem dołóż wywołanie bloku dla silnika Diesla.



Wprowadź kolejną sieć i zaprogramuj połączenie szeregowe odpowiednio je adresując. Następnie zapisz program i wyjdź z bloku programowego.

```
#PE_Preset_Speed_R  #DE_Preset_Speed_R  #Preset_Speed_Reac
eached              eached              hed
──────┤ ├──────────────────┤ ├──────────────────( )──────┤
```

Zmienne lokalne ("PE_Setpoint_Reached" i "DE_Setpoint_Reached") przypisane zostaną do parametru wyjściowego "Setpoint_Reached," który będzie mógł być wykorzystany w OB1.

Programowanie FB10 w STL

```
CALL  #Petrol_Engine
 Switch_On            :="Switch_On_PE"
 Switch_Off           :="Switch_Off_PE"
 Failure              :="PE_Failure"
 Actual_Speed         :="S_Data".PE_Actual_Speed
 Engine_On            :="PE_On"
 Preset_Speed_Reached:=#PE_Preset_Speed_Reached

CALL  #Diesel_Engine
 Switch_On            :="Switch_On_DE"
 Switch_Off           :="Switch_Off_DE"
 Failure              :="DE_Failure"
 Actual_Speed         :="S_Data".DE_Actual_Speed
 Engine_On            :="DE_On"
 Preset_Speed_Reached:=#DE_Preset_Speed_Reached

A     #PE_Preset_Speed_Reached
A     #DE_Preset_Speed_Reached
=     #Preset_Speed_Reached
```

W przypadku programowania w postaci STL wstaw nowy segment, ustaw kursor na polu wprowadzania programu, i wpisz obok przedstawione instrukcje.

Zapisz wprowadzony program i zamknij blok.

Programowanie FB10 w FBD

W przypadku programowania w FBD, ustaw kursor na polu wprowadzania programu w nowej sieci i wprowadź z katalogu odpowiedni diagram funkcyjny.

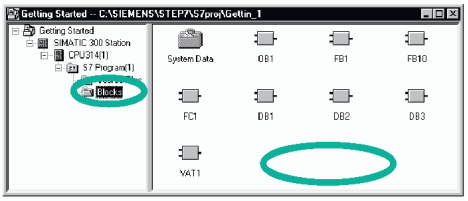
Zabezpiecz wprowadzony program i zamknij obrabiany blok.

By było możliwe wywołanie wielokrotne bloku FB1, blok nadrzędny FB10 musi być sam wywołany np. w OB1.
Bloki wielokrotne można programować z przeznaczeniem dla FB. Nie można zaprojektować bloku wielokrotnego dla funkcji.

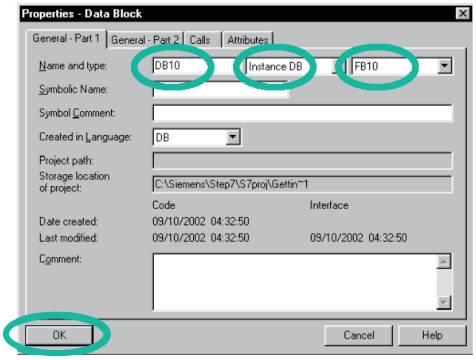
Dalsze informacje znajdują się w **Help > Contents** w rozdziale "Programming Blocks," "Creating Logic Blocks," i "Multiple Instances in the Variable Declaration Table."

10.3 Tworzenie DB10 i wstawianie aktualnych wartości

Nowotworzony blok DB10 będzie zastępować bloki DB1 i DB2. W nim zostaną zapamiętane dane dla silnika benzynowego i Diesla, które potem będą wykorzystywane przy wywołaniu bloku FB10 w bloku OB1. (patrz też wywołanie FB1 w OB1.)

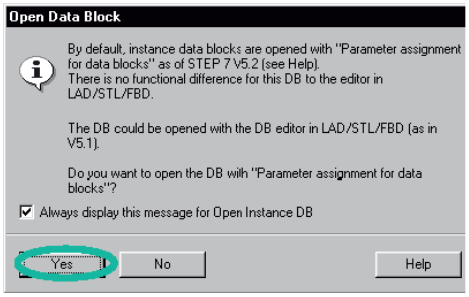


Uzupełnij nasz projekt wprowadzając do foldera **Blocks** blok DB10 (wykorzystując menu kontekstowe uruchamiane prawym klawiszem myszy).
Zmień w tym celu numer bloku na DB10, oraz zatwierdź ustawienia klawiszem OK.



Podwójnie kliknij na blok DB 10.

W następnym oknie wciśnij przycisk YES. Blok DB 10 zostaje otwarty.



Postać wyświetlania danych umożliwia wyświetlenie szczegółowych wartości, także takich, które wynikają z wywołania bloku FB1. Postać deklaracyjna prezentuje dane w formacie zadeklarowanych parametrów w FB10.

Zmień aktualną wartość dla silnika Diesla na 1300, zapisz zmiany i zamknij blok.

	Address	Declaration	Name	Type	Initial value	Actual value	Comment
1	0.0	out	Preset_Speed_Reached	BOOL	FALSE	FALSE	Both engines have reached the preset speed
2	2.0	stat:in	Petrol_Engine.Switch_On	BOOL	FALSE	FALSE	Switch on engine
3	2.1	stat:in	Petrol_Engine.Switch_Off	BOOL	FALSE	FALSE	Switch off engine
4	2.2	stat:in	Petrol_Engine.Failure	BOOL	FALSE	FALSE	Engine failure, causes the engine to switch off
5	4.0	stat:in	Petrol_Engine.Actual_Speed	INT	0	0	Actual engine speed
6	6.0	stat:out	Petrol_Engine.Engine_On	BOOL	FALSE	FALSE	Engine is switched on
7	6.1	stat:out	Petrol_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Preset speed reached
8	8.0	stat	Petrol_Engine.Preset_Speed	INT	1500	1500	Requested engine speed
9	10.0	stat:in	Diesel_Engine.Switch_On	BOOL	FALSE	FALSE	Switch on engine
10	10.1	stat:in	Diesel_Engine.Switch_Off	BOOL	FALSE	FALSE	Switch off engine
11	10.2	stat:in	Diesel_Engine.Failure	BOOL	FALSE	FALSE	Engine failure, causes the engine to switch off
12	12.0	stat:in	Diesel_Engine.Actual_Speed	INT	0	0	Actual engine speed
13	14.0	stat:out	Diesel_Engine.Engine_On	BOOL	FALSE	FALSE	Engine is switched on
14	14.1	stat:out	Diesel_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Preset speed reached
15	16.0	stat	Diesel_Engine.Preset_Speed	INT	1500	1300	Requested engine speed

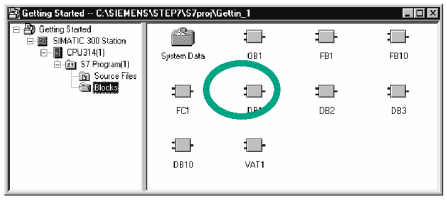
W postaci deklaracyjnej tabeli widoczne są tylko ogólne dane dla poszczególnych silników.

Zmienne wewnętrzne bloku FB1 posiadają swoje nazwy symboliczne np. „Eingine_On”. Dostęp do nich jest możliwy dzięki podaniu pełnej nazwy np.: "Petrol_Engine.Switch_On."

Dalsze informacje znajdują się w **Help > Contents** w rozdziale "Programming Blocks" i "Creating Data Blocks."

10.4 Wywołanie FB10 w OB1

W naszym przykładzie blok FB10 zostanie wywołany w bloku OB1. Zasada budowy i funkcjonowania tego wywołania jest identyczna, jak przy zrobionym już wywołaniu bloku FB1 (patrz rozdz. 5.6). W rozdziale 5.6 do zaprogramowania użyliśmy wywołań w sieciach 4 i 5.



Otwórz blok OB1 w projekcie, w którym właśnie zaprogramowałeś blok FB10.

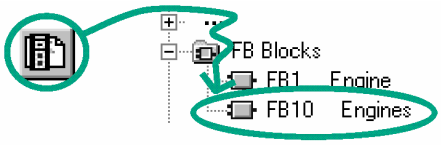
Jeśli w rozdziale 4 skopiowałeś przykładową tabelę symboli do Twojego projektu, nie trzeba ich teraz ponownie definiować.

Definicja nazw symbolicznych

Otwarte jest okno edytora programu. Otwórz tabelę symboli używając w menu **Options > Symbol Table** i wprowadź oznaczenia symboliczne dla bloków FB10 i DB10.
Zamknij tabelę i okno symboli.

Symbol	Address	Data Type	Comment
...	...	...	...
Engines	FB 10	FB 10	Example of multiple instances
Engine_Data	DB 10	FB 10	Instance data block for FB10 10
...	...	...	...

Stworzenie wywołania bloku

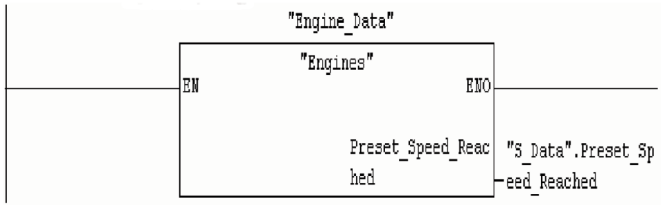


Wprowadź na końcu bloku nową sieć, i uzupełnij go o wywołanie bloku FB10 („Engines”).

Uzupełnij wolne pola wywołanego bloku odpowiednimi nazwami symbolicznymi.

Usuń w sieci 4 i 5 dotychczasowe wywołania bloku FB1. Teraz blok ten wywoływany jest tylko w FB10.

Zapamiętaj program i zamknij blok.



Sygnał wyjściowy bloku: „Preset_Speed_Reached" przekazywany jest do globalnego bloku danych.

Wywołanie w STL

Jeśli posługujemy się postacią STL, w nowo stworzonej sieci wejdź na pole wprowadzania programu, i wpisz poniżej podane instrukcje. Użyj w tym celu element katalogu **FB Blocks>FB10 Engines.**

Usuń w sieci 4 i 5 dotychczasowe wywołania bloku FB1. Teraz blok ten wywoływany jest tylko w FB10.

Zapamiętaj program i zamknij blok.

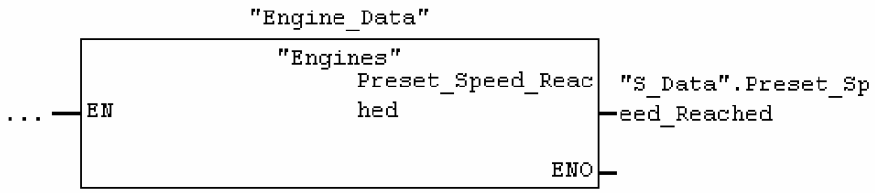
```
CALL „Engines", „Engine_Data"
 Preset_Speed_Reached:="S_Data".Preset_Speed_Reached
```

Wywołanie w FBD

Jeśli posługujemy się postacią FBD, w nowo stworzonej sieci wejdź na pole wprowadzania programu i wpisz poniżej podane instrukcje. Użyj w tym celu element katalogu **FB Blocks>FB10 Engines.**

Usuń w sieci 4 i 5 dotychczasowe wywołania bloku FB1. Teraz blok ten wywoływany jest tylko w FB10.

Zapamiętaj program i zamknij blok.



Jeśli potrzebne byłoby sterowanie kolejnych silników, postępowanie byłoby identyczne, też w tym celu moglibyśmy użyć bloku FB10.
Należałoby zadeklarować w tabeli deklaracji bloku FB10 dalsze silniki i wywołać w nim kolejne bloki FB1. W tabeli symboli należałoby dodać adresację i oznaczenia dla nowych silników.

Dalsze informacje znajdują się w **Help > Contents** w rozdziale "Calling References Helps," "The STL, FBD, or LAD Language Description," i "Program Control Instructions."

11 Konfiguracja systemu rozproszonego

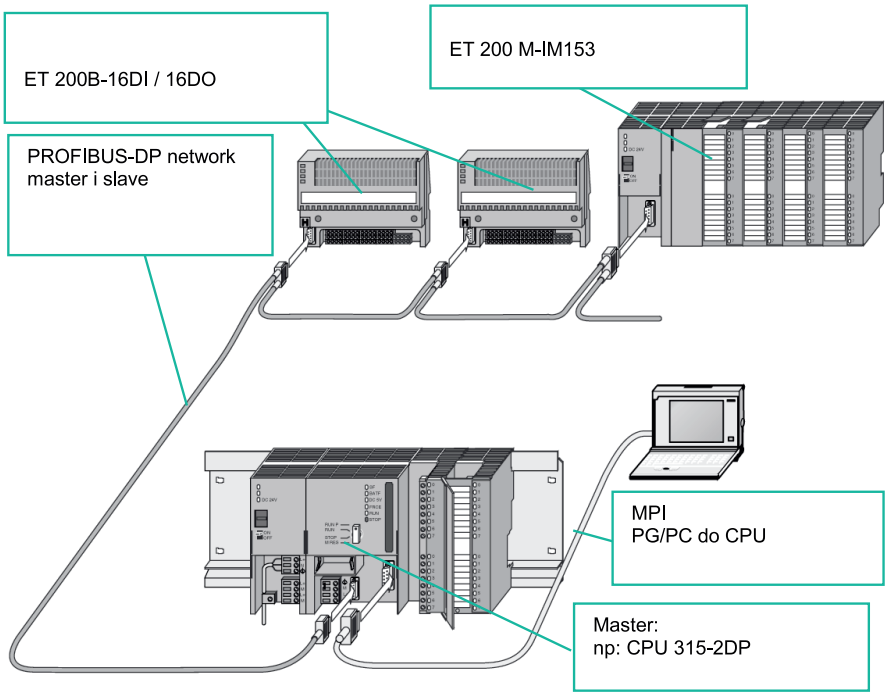
11.1 Tworzenie systemu zdecentralizowanego przy użyciu sieci PROFIBUS DP

Standardowo w centralnych systemach sterowania wszelkie sygnały z czujników oraz do elementów wykonawczych przekazywane są z modułów sterownika centralnego. Przy większych odległościach pociąga to za sobą wysokie koszty okablowania.

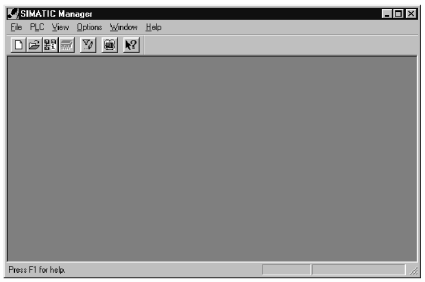
Stosując system rozproszony, umieszczając moduły wejść i wyjść w pobliżu czujników i elementów wykonawczych, ogranicza się nakład pracy i koszty okablowania. Połączenie pomiędzy modułami i jednostką centralną następuje poprzez sieć PROFIBUS-DP.

Dotychczas poznaliśmy programowanie sterownika w systemie centralnym. Programowanie tego typu systemów i systemu zdecentralizowanego jest identyczne. Podobnie również odbywa się konfiguracja.

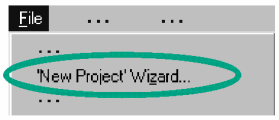
Pomocna byłaby w dalszej pracy znajomość metod konfiguracji układów centralnych (patrz rozdz. 2.1 i 6).



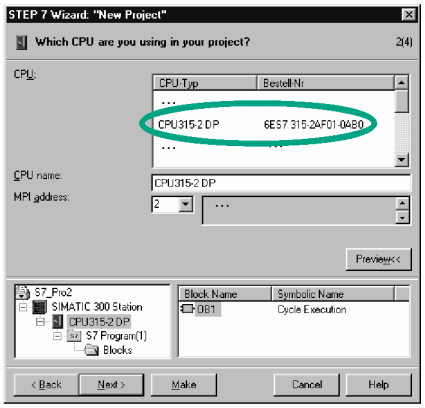
Założenie nowego projektu



Zamknij wszelkie dotychczas otwarte okna i otwórz tylko SIMATIC Manager.



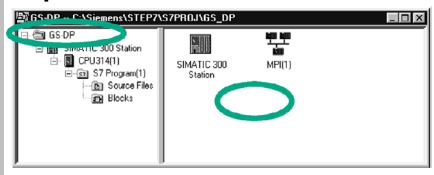
Załóż nowy projekt.



Załóż w odpowiednim oknie jednostkę centralną typu 315-2DP (CPU z siecią PROFIBUS-DP).

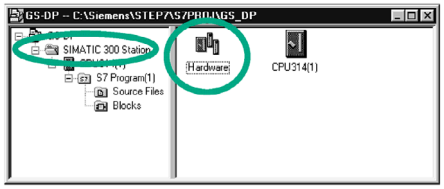
Postępując zgodnie z opisanymi krokami w rozdz. 2.1 nadaj nową nazwę projektowi.

Wprowadzenie sieci PROFIBUS

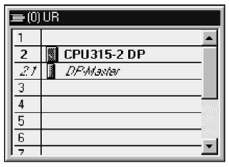


Zaznacz folder z nazwą projektu i wprowadź sieć PROFIBUS używając menu kontekstowego wywołanego prawym klawiszem myszy.

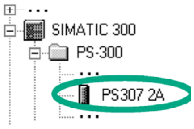
Konfiguracja stacji



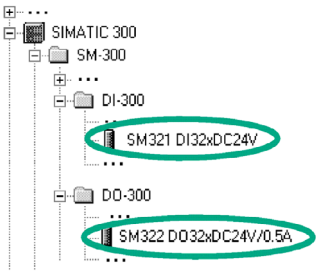
Zaznacz folder **SIMATIC 300-Station** i kliknij dwukrotnie **Hardware.** Otwiera się edytor konfiguracji.



CPU 315-2 DP jest widoczne w raku. Jeśli trzeba, uaktywnij katalog elementów sterownika klikając ikonkę na listwie narzędziowej, lub wybierając w menu **View > Hardware Catalog.**

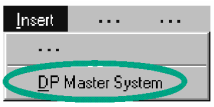


Wprowadź na pozycję 1 zasilacz **PS307 2A.**



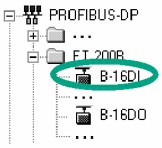
Na pozycję 4 wprowadź moduł wejść **DI32xDC24V,** na pozycję 5 moduł wyjść **DO32xDC24V/0,5A.**

Konfiguracja DP-Master

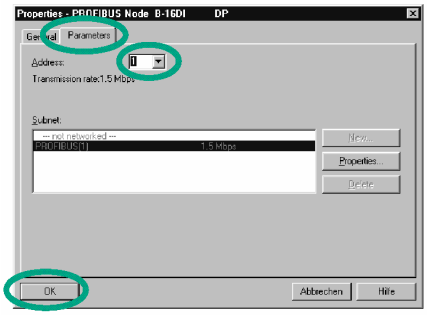


Zaznacz na poz. 2.1 DP-Master i wprowadź **DP-master system.**

Wszystkie obiekty skonfigurowane jako uczestnicy sieci DP można dowolnie przemieszczać w ramach tej sieci.

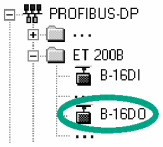


Znajdź w katalogu moduł **B-16DI** i wprowadź go do sieci (puszczając na przerywaną linię).



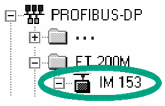
W oknie „Properties” można zadeklarować nową sieć oraz adres wprowadzanego modułu.

Zatwierdź wprowadzony adres 1.



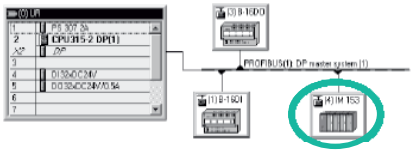
Wykonaj te same operacje dla modułu **B-16DO.**

Tym razem adres zostanie zaproponowany automatycznie. Zatwierdź ustawienia przyciskiem OK.

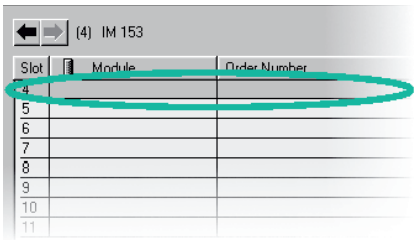


Wprowadź do sieci moduł interfejsu IM153 (moduł ET200M) i zatwierdź zaproponowany adres klawiszem OK.

W naszym przykładzie przyjęte zostaną adresy zaproponowane przez system. Można je jednak w razie potrzeby zmienić.

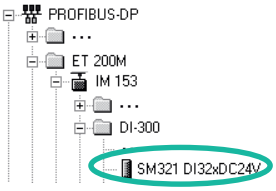


Kliknij na sieci ET200M. W dolnej części okna ukaże się tabela konfiguracyjna modułu ET200M. Wybierz pozycję 4.

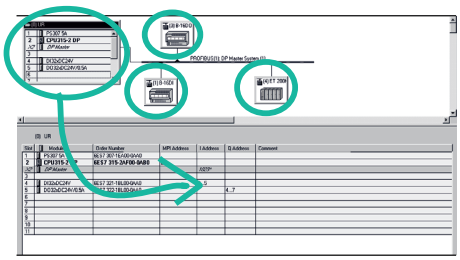


Do ET200M można dołożyć dalsze moduły. Weźmy np. moduł **DI32xDC24V** i wprowadźmy na pozycję 4.

Należy zwracać uwagę z jakiego foldera dokładane są moduły. Np. dla systemu ET200M należy korzystać z folderu ET200M.

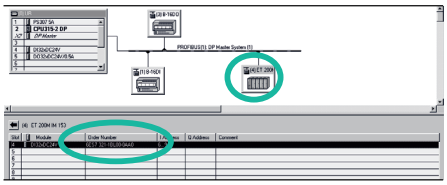


Zmiana adresów uczestników sieci



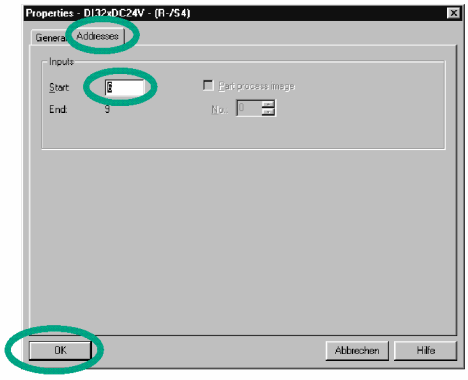
W naszym przykładzie zmiana adresu nie jest konieczna. W praktyce jednak może być czasem przydatna.

Zaznaczając po kolei wszystkich uczestników sieci sprawdź ich adresy. System zapewnił ich niepowtarzalność.

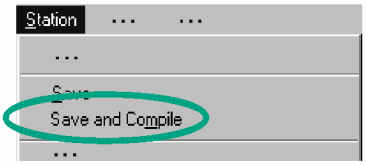


Załóżmy, że chcemy zmienić adresy w ET200M:

Zaznacz moduł na pozycji 4 **DO32xDC24V/0,5A.**



Zmień w oknie „Properties” ustawienie adresu z 6 na 12, i zatwierdź ustawienia klawiszem OK.

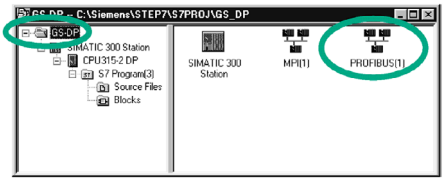


Na zakończenie zapisz i skompiluj konfigurację **(save and compile).**

Zamknij okno.

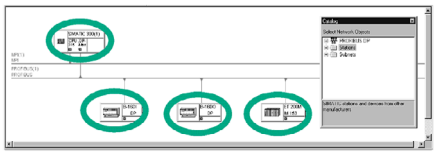
Przeprowadzenie funkcji „save and compile” powoduje jednocześnie sprawdzenie rozkładu modułów. W przypadku bezbłędnej konfiguracji stworzone zostaną dane systemowe i będzie można przesłać konfigurację do sterownika.
W przypadku wykorzystania jedynie funkcji zabezpieczenia (save), może zostać zapamiętana także błędna konfiguracja.

Opcjonalnie: Projektowanie sieci



Konfigurację systemu rozproszonego można także przeprowadzić wykonując projekt sieci.

Kliknij dwukrotnie na sieć PROFIBUS w SIMATIC Manager.



Otwiera się okno NETPRO.

Możemy teraz dodawać z katalogu dalszych uczestników sieci (DP-Slave).

Kliknij dwukrotnie na dowolny element. Otwarte zostaje okno konfiguracji sprzętowej.

Za pomocą funkcji **Station > Consistency Check** (w oknie "Configuring Hardware") lub **Network > Consistency Check** (w oknie "Configuring Networks") można sprawdzić poprawność wykonania konfiguracji sieciowej przed jej zabezpieczeniem. W przypadku wykrycia błędów zostaną podane ich miejsca oraz poprawne rozwiązania.

Dalsze informacje znajdują się w **Help > Contents** w rozdziale "Configuring the Hardware" i "Configuring the Distributed I/O."

Gratulacje!

Znasz już sposób tworzenia projektu w systemie STEP7. Znasz sposób zakładania i konfigurowania stacji, oraz rodzaje i sposoby tworzenia bloków programowych, a także inne funkcje programu np. postać symboliczną czy też monitorowanie działania programu.
W przypadku wątpliwości, jakie mogą powstać przy tworzeniu kolejnych projektów polecamy korzystanie z bogatego systemu pomocy.
Jeśli chcesz pogłębić swoją wiedzę na temat STEP7 zalecamy wzięcie udziału w prowadzonych szkoleniach, a także rozszerzanie wiedzy o szczegóły zawarte w dokumentacji do systemu SIMATIC S7.

Życzymy powodzenia w tworzeniu dalszych projektów.

Notatki

Notatki

Promocyjny zestaw startowy SIMATIC S7-300 + STEP7

Numer katalogowy: 6ES7 313-5BF03-4YB0

Promocyjny zestaw zawiera:

- oprogramowanie STEP7 v 5.4
- kompaktowy sterownik S7-300 (CPU 313C, 24 DI / 16 DO / 4 AI / 2 AO, 64kB pamięci work)
- kartę pamięci SIMATIC MicroMemory Card 64kB
- PC Adapter USB z kablem USB (5m)
- szynę DIN 160mm
- front connector 40 PIN (2 szt.)
- zestaw manuali (En, Ge, It, Sp, Fr)

Produkt do kupienia u autoryzowanych dystrybutorów: www.siemens.pl/simatic/dystrybutorzy
i w Regionalnych Biurach Sprzedaży

Biura sprzedaży:

Siemens Sp. z o.o.
Sektor Industry IA AS
03-821 Warszawa
ul. Żupnicza 11
tel.: 22-870 90 22
fax: 22-870 98 68

Regionalne biura sprzedaży:

80-309 Gdańsk	40-527 Katowice	31-476 Kraków	60-164 Poznań	87-100 Toruń	53-611 Wrocław
Al. Grunwaldzka 413	ul. Gawronów 22	ul. Lublańska 38	ul. Ziębicka 35	ul. Włocławska 169	ul. Strzegomska 46b
tel.: 58-764 60 92	tel.: 32-208 41 34	tel.: 12-299 89 11	tel.: 61-664 98 61	tel.: 56-651 46 49	tel.: 71-777 50 60
fax: 58-764 60 99	fax: 32-208 41 39	fax: 12-299 89 20	fax: 61-664 98 64	fax: 56-651 46 50	fax: 71-777 50 50

www.siemens.pl/S7-300 **email: simatic.pl@siemens.com**